

IN THE
UNITED STATES COURT OF APPEALS
FOR THE FEDERAL CIRCUIT

AKAMAI TECHNOLOGIES, INC.,

Plaintiff-Appellant,

and

THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY,

Plaintiff-Appellant,

v.

LIMELIGHT NETWORKS, INC.,

Defendant-Cross-Appellant.

**Appeals from the United States District Court for the District of
Massachusetts in case nos. 06-CV-11109 and 06-CV-11585,
Judge Rya W. Zobel.**

**CORRECTED BRIEF OF PLAINTIFFS-APPELLANTS
AKAMAI TECHNOLOGIES, INC. AND
THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

ROBERT S. FRANK, JR.
CHOATE, HALL & STEWART LLP
Two International Place
Boston, MA 02110

*Attorneys for Plaintiff-Appellant
The Massachusetts Institute of Technology*

March 8, 2010

DONALD R. DUNNER
KARA F. STOLL
ELIZABETH D. FERRILL
FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, LLP
901 New York Avenue, NW
Washington, DC 20001-4413
(202) 408-4000

*Attorneys for Plaintiff-Appellant
Akamai Technologies, Inc.*

CERTIFICATE OF INTEREST

Counsel for Plaintiff-Appellant Akamai certifies the following:

1. The full name of every party or amicus represented by us is:
Akamai Technologies, Inc.

2. The name of the real party in interest (if the party named in the caption is not the real party in interest) represented by us is:
Akamai Technologies, Inc.

3. All parent corporations and any publicly held companies that own 10% or more of the stock of any party represented by us are:
None

4. The names of all law firms and the partners or associates that appeared for the parties now represented by us in the trial court or are expected to appear in this court are:

FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
Donald R. Dunner, Kara F. Stoll, Elizabeth D. Ferrill

CHOATE, HALL & STEWART
Robert S. Frank, Jr., Carlos J. Perez-Albuerne, G. Mark Edgerton,
Richard C. Abati

McDERMOTT, WILL & EMERY LLP
Sarah Chapin Columbia

CERTIFICATE OF INTEREST

Counsel for Plaintiff-Appellant Massachusetts Institute of Technology certifies the following:

1. The full name of every party or amicus represented by us is:
Massachusetts Institute of Technology

2. The name of the real party in interest (if the party named in the caption is not the real party in interest) represented by us is:
Massachusetts Institute of Technology

3. All parent corporations and any publicly held companies that own 10% or more of the stock of any party represented by us are:
None

4. The names of all law firms and the partners or associates that appeared for the parties now represented by us in the trial court or are expected to appear in this court are:

CHOATE, HALL & STEWART LLP
Robert S. Frank, Jr., Carlos J. Perez-Albuerne, G. Mark Edgerton,
Richard C. Abati

McDERMOTT, WILL & EMERY LLP
Sarah Chapin Columbia

TABLE OF CONTENTS

STATEMENT OF JURISDICTION.....	1
I. STATEMENT OF THE ISSUES	2
II. STATEMENT OF THE CASE.....	4
A. Course of Proceedings and Disposition Below	4
B. Preliminary Statement	5
III. STATEMENT OF FACTS	6
A. The Parties and Their Products.....	6
A. The Patents-in-suit	7
1. Technical Background.....	7
2. Problems in the Prior Art	9
3. The Patent Specification.....	11
a. Various Methods for “Tagging” the URL Associated with an Embedded Object	13
b. Various Levels of the Intelligent DNS Hierarchy.....	14
4. The ’703 Patent Claims	15
5. The ’645 Patent Claim.....	17
6. The ’413 Patent Claims	19
B. Akamai’s Content Delivery Service and Business.....	21
C. Limelight’s Relationship with Content Providers	22
1. Limelight’s Contract Requires Content Providers to Perform the Claim Steps in Order to Use Limelight’s Service	23

2.	Limelight Provides Specific Instructions to Content Providers to Perform the Claim Steps Via the “Then Current Company Process”	24
D.	District Court Proceedings.....	28
1.	The Court’s Claim Construction	28
a.	“A Given Object . . . Is Associated With an Alphanumeric String” in Claim 1 of the ’645 Patent.....	28
b.	“The Given Name Server That Receives the DNS Query Being Close to the Client Local Name Server as Determined by Given Location Information” in Claim 1 of the ’645 Patent	30
c.	“Selecting a Given One of the Name Servers in the Content Delivery Network” in Claims 8, 18, and 20 of the ’413 Patent	32
2.	The Jury Verdict.....	33
3.	The District Court’s Entry of JMOL.....	33
IV.	SUMMARY OF ARGUMENT	36
V.	ARGUMENT	39
A.	The Court Should Reverse the District Court’s JMOL of No Joint Infringement.....	39
1.	Substantial Evidence Supports the Jury Verdict of Joint Infringement	39
2.	The District Court Erred in Its Analysis of <i>Muniauction</i>	44
a.	The District Court Erred in Finding “No Material Difference Between Limelight’s Interaction with Its Customers and That of Thomson in <i>Muniauction</i> ”	44

b.	<i>Muniauction</i> Did Not Establish that “Direction or Control” Can Never Be Shown by a “Contractual Agreement to Pay for a Defendant’s Services”	46
c.	<i>Muniauction</i> Did Not Eliminate the <i>BMC Resources</i> Possibility of Satisfying the Control or Direction Test by Providing Instructions or Directions	48
B.	The District Court’s Construction of the ’645 and ’413 Patent Claims Should Be Reversed	50
1.	“Associated With an Alphanumeric String” in Claim 1 of the ’645 Patent Should Not Be Construed to Include the Content Provider’s URL.....	50
a.	The Court’s Original URL Interpretation Is Inconsistent with the Claim Language, Specification, Prosecution History, and Stipulated Definition.....	51
b.	The Court’s Original URL Interpretation Is Inconsistent with Other Limitations in the Claim, Which Show that the “Alphanumeric String” is a Hostname, not a URL	54
c.	The District Court’s Original URL Construction Creates a Requirement Found Nowhere (and in No Embodiment) in the Specification.....	56
d.	Other Claims Confirm that the “Alphanumeric String” Is Not a URL	56
e.	The District Court’s Original URL Construction Is Premised on a Fundamental Misunderstanding of the Requirements of the Invention and Scope of the Claims	57

2.	The Court Should Not Read “Selecting by the Alternative Domain Name System” into Claim 1 of the ’645 Patent.....	59
3.	The Court Should Not Read “the Content Delivery Network’s Domain Name System Selects” into Claims 8, 18, and 20 of the ’413 Patent	62
VI.	CONCLUSION.....	63

TABLE OF AUTHORITIES

FEDERAL CASES	PAGE(S)
<i>Agfa Corp. v. Creo Prods. Inc.</i> , 451 F.3d 1366 (Fed. Cir. 2006).....	39, 60
<i>Agilent Techs., Inc. v. Affymetrix, Inc.</i> , 567 F.3d 1366 (Fed. Cir. 2009).....	60
<i>Arrowhead Indus. Water, Inc. v. Ecolochem, Inc.</i> , 846 F.2d 731 (Fed. Cir. 1988).....	49
<i>BJ Serv. Co. v. Halliburton Energy Serv., Inc.</i> , 338 F.3d 1368 (Fed. Cir. 2003).....	61
<i>BMC Resources, Inc. v. Paymentech, L.P.</i> , 498 F.3d 1373 (Fed. Cir. 2007).....	passim
<i>Borges Colon v. Roman-Abreu</i> , 438 F.3d 1 (1st Cir. 2006).....	40, 43
<i>Cybor Corp. v. FAS Techs., Inc.</i> , 138 F.3d 1448 (Fed. Cir. 1998) (en banc).....	50
<i>DSW, Inc. v. Shoe Pavilion, Inc.</i> , 537 F.3d 1342 (Fed. Cir. 2008).....	58, 59, 62
<i>Liebel-Flarsheim Co. v. Medrad, Inc.</i> , 358 F.3d 898 (Fed. Cir. 2004).....	61
<i>Linear Tech. Corp. v. Int’l Trade Comm’n</i> , 566 F.3d 1049 (Fed. Cir. 2009).....	53
<i>Masco Corp. v. U.S.</i> , 303 F.3d 1316 (Fed. Cir. 2002).....	53
<i>Muniauction, Inc. v. Thomson Corp.</i> , 532 F.3d 1318 (Fed. Cir. 2008).....	passim

<i>NTP, Inc. v. Research In Motion, Ltd.</i> , 418 F.3d 1282 (Fed. Cir. 2005).....	56
<i>Oatey Co. v. IPS Corp.</i> , 514 F.3d 1271 (Fed. Cir. 2008).....	61
<i>Phillips v. AWH Corp.</i> , 415 F.3d 1303 (Fed. Cir. 2005) (<i>en banc</i>)	52
<i>RCA/Ariola Int’l, Inc. v Thomas & Grayston Co.</i> , 845 F.2d 773 (8th Cir. 1988).....	50
<i>Rexnord Corp. v. Laitram Corp.</i> , 274 F.3d 1336 (Fed. Cir. 2001).....	38, 55
<i>SanDisk Corp. v. STMicroelectronics, Inc.</i> , 480 F.3d 1372 (Fed. Cir. 2007).....	49
<i>Verizon Servs. Corp. v. Vonage Holdings Corp.</i> , 503 F.3d 1295 (Fed. Cir. 2007).....	51

FEDERAL STATUTES

28 U.S.C. § 1295(a).....	1
28 U.S.C. § 1338(a).....	1

RULES

Fed. R. App. P. 4(a)(4)(A)	1
----------------------------------	---

STATEMENT OF RELATED CASES

No appeal in or from this Civil Action Case Nos. 06-CV-11109 and 06-CV-11585 was previously before this or any other appellate court. There are no cases known to counsel that may be directly affected by this Court's decision.

STATEMENT OF JURISDICTION

The U.S. District Court for the District of Massachusetts (Judge Rya W. Zobel) had jurisdiction over this patent infringement action giving rise to this appeal pursuant to 28 U.S.C. § 1338(a).

The U.S. Court of Appeals for the Federal Circuit has jurisdiction over this appeal pursuant to 28 U.S.C. § 1295(a).

The notice of appeal from the Final Judgment entered on May 27, 2009, was timely filed on May 26, 2009, in accordance with Fed. R. App. P. 4(a)(4)(A). Because the notice of appeal was filed prior to entry of the Final Judgment, a supplement to the notice of appeal was filed on June 19, 2009.

I. STATEMENT OF THE ISSUES

This appeal presents an issue of joint infringement and three claim construction issues:

1. Whether the district court erred in vacating the jury's verdict of infringement of the '703 patent based on *Muniauction, Inc. v. Thomson Corp.*, 532 F.3d 1318 (Fed. Cir. 2008), where:

(1) *Muniauction*, properly read, only applies the "control or direction" standard of *BMC Resources, Inc. v. Paymentech, L.P.*, 498 F.3d 1373, 1381 (Fed. Cir. 2007), to a different set of distinguishable facts;

(2) the jury was properly instructed on the "control or direction" standard; and

(3) the jury heard substantial evidence to support its verdict of joint infringement, including that when the invention is used, Limelight contractually obligates content providers to perform, on its behalf, the claim steps Limelight itself does not perform, and Limelight provides explicit technical instruction on exactly how to perform those claim steps.

2. Whether the district court erred in interpreting "a given object of a participating content provider is associated with an alphanumeric string" in the preamble of claim 1 of the '645 patent to require that the alphanumeric string

“include[] the URL used to identify the object in the absence of a content delivery network,” where:

(1) the district court and parties agreed that “associated” and “alphanumeric string” should have their ordinary and customary meaning; and

(2) the district court’s construction contradicts the remaining claim language and prosecution history, which both show that an “alphanumeric string” does not include a URL.

3. Whether the district court erred in interpreting “the given name server that receives the DNS query being close to the client local name server as determined by given location information” in ’645 patent claim 1 to require that the given name server be “selected by the alternative domain name system,” where:

(1) the claim language says nothing of selecting, let alone what structure performs the selecting; and

(2) there is no basis in the specification or prosecution history for reading this limitation into the claim.

4. Whether the district court similarly erred in interpreting “selecting a given one of the name servers in the content delivery network” in claims 8, 18, and 20 of the ’413 patent as “the content delivery network’s domain name system selects a particular name server,” where:

(1) the claim language does not specify the structure that performs the selecting; and

(2) there is no basis in the specification or prosecution history for reading this limitation into the claim.

II. STATEMENT OF THE CASE

A. Course of Proceedings and Disposition Below

This is an appeal from a judgment as a matter of law that Limelight does not infringe claims 19-21 and 34 of U.S. Patent No. 6,108,703 (the '703 patent). The theory of infringement is joint infringement. After a three-week trial, the jury returned a verdict of infringement. By its verdict, the jury necessarily found that Limelight and a content provider perform every step of the asserted claims and that Limelight controls or directs the content provider in performing the steps that Limelight alone does not perform. The district court, however, entered JMOL of no infringement, holding that under this Court's decision in *Muniauction*, it was impossible for Limelight to infringe the claims as a matter of law.

This is also an appeal from the district court's construction of claim 1 in U.S. Patent No. 7,103,645 (the '645 patent) and claims 8, 18, and 20 in U.S. Patent No. 6,553,413 (the '413 patent). Below, Akamai stipulated that it could not prove infringement of the '645 patent under the district court's construction but reserved its right to appeal. (A1-2.) Akamai now challenges the court's construction on

appeal, seeking a remand for a determination of infringement under the proper claim construction. Regarding the '413 patent, the district court entered summary judgment of noninfringement of claims 8, 18, and 20 on the ground that the evidence did not support infringement of the claims as construed. On appeal, Akamai challenges the court's construction of claims 8, 18, and 20, and seeks a remand for a determination of infringement under the proper claim construction.

B. Preliminary Statement

This Court has held that “a party cannot avoid infringement . . . simply by contracting out steps of a patented process to another entity.” *BMC Resources*, 498 F.3d at 1381. If these words have any significance, they should apply here, where Limelight alone performs almost every step of the asserted claims and contracts out the steps it does not alone perform to its content-provider customers. Indeed, Limelight's contract expressly obligates a content provider to perform the claim steps to enable its web content to be delivered by Limelight. Moreover, Limelight directs content providers to perform the claim steps by providing Installation Guidelines and other documents specifically instructing content providers to perform the steps, creates and assigns a unique “hostname” that must be used in performing the steps, and assigns a Limelight employee to help, if needed, in performing the steps. Finally, Limelight's documents make it abundantly clear that the entire content delivery process—including the steps contracted out to content

providers—is Limelight’s “service” and Limelight’s “implementation.” Contrary to the district court’s analysis, this level of control or direction far surpasses that found in *BMC Resources* and *Muniauction*, and constitutes more than substantial evidence to support the jury verdict on this fact-intensive issue.

This Court should also reverse the district court’s construction of claim 1 of the ’645 patent and claims 8, 18, and 20 of the ’413 patent. In each claim construction ruling on appeal, the district court imported unnecessary limitations into the claims, ignored conflicting claim language, and misread the patent specification. Further, the district court failed to appreciate that the specification does not disclaim other embodiments or in any other way limit the invention to a single embodiment. To the contrary, the specification discloses “alternative,” “illustrative,” “preferred,” and “general” embodiments, and it expressly states that such embodiments are not meant to be limiting. Because the district court’s constructions are based on a misunderstanding of the specification, the claims, and claim construction principles, this Court should reverse and remand for a determination of infringement under the proper constructions.

III. STATEMENT OF FACTS

A. The Parties and Their Products

Akamai and Limelight are direct competitors. (A582-83:103-104.) They provide content delivery services to businesses that offer content on the Internet.

As described to the jury, content providers (*e.g.*, CNN or Yahoo!) maintain web sites that allow persons who use the Internet (*i.e.*, “users”) to obtain content. (A333:14-16; A338-39:35-40.) Content may include any information delivered over the Internet, including web pages (which typically include other content such as images and graphics, videos, sound clips, and even software downloads). (A350:85.) Companies like Akamai and Limelight offer “content delivery” services to content providers over the “content delivery networks” (sometimes referred to as “CDNs”) that they operate. A CDN (content delivery network) typically includes a number of physical locations around the Internet at which computers, called “content servers,” are positioned. (A338:36; A367:58; A558-59:13-14.) These content servers are shared among many different websites that use the content delivery service. (A338-339:37-38.)

A. The Patents-in-suit

1. Technical Background

The patents-in-suit relate to delivering content over the Internet. (A267, 1:10-12.) Thus, a basic discussion of how content is delivered over the Internet may be helpful.

A typical web page includes a “base document,” which serves as an outline for the web page, and “embedded objects,” such as images, audio, and video, filling the outline specified in the base document. (A269, 5:23-27.) To access a

web page, a user enters a Uniform Resource Locator (“URL”) into a web browser or selects a link to that page. (A268-69, 4:63-5:2.) URLs may identify base documents or embedded objects. (A269, 5:23-41.) Typically, the URL is a string of characters (*e.g.*, <http://www.cnn.com/world/picture.jpg>) consisting of a protocol (*e.g.*, <http://>), a hostname (*e.g.*, www.cnn.com), a path (*e.g.*, [/world/](http://www.cnn.com/world/)), and an object name (*e.g.*, [picture.jpg](http://www.cnn.com/world/picture.jpg)). (A340:44-45; A369:67.) A slash (“/”) separates the hostname, the path, and the object name. (A17896-97.)

The hostname portion of the URL corresponds to a set of numbers, referred to as an Internet Protocol (“IP”) address (*e.g.*, 157.166.226.25), which identifies one or more content servers. (A340:45; A378:104-105.) Thus, when a user enters a URL, the user’s computer passes the hostname portion of the URL—not the entire URL, the path, or the object name—to a “local” name server, which then attempts to find the “address” of that hostname, just as someone might use a name to look up a telephone number in a telephone book. (A340:44-45.) The Internet’s equivalent of a telephone book is called the Domain Name Service (“DNS”), which is a system of machines and processes that translates hostnames into IP addresses—*e.g.*, www.cnn.com to 157.166.226.25. (A378-79:105-107.) This process is referred to as “resolving.” (A271, 9:22-25.)

A name server can only resolve hostnames; it cannot resolve a full URL. (A17440, A369:66-68.) It never receives the path or object name. (A369:67-68.)

After the resolving process, DNS returns the IP address corresponding to the hostname to the user's local name server, which passes it back to the user's computer. (A271, 10:58-60.) Then, the user's computer can send a request for the web page directly to the content server associated with that IP address. (A272, 12:26-28.) The content server responds by sending the requested base document to the user's computer. (A272, 12:32-34.) The base document typically contains one or more URLs corresponding to the embedded objects associated with the web page. (A267, 1:21-25.)

The user's computer and the local name server may then repeat the resolution process for each embedded object URL. That is, the user's computer sends the hostname associated with the embedded object to DNS for resolution, receives the IP address for the content server, requests the embedded object directly from that content server, and receives the embedded object sent by the content server. (A267, 1:30-34.)

2. Problems in the Prior Art

Conventionally, Internet content providers had limited choices for delivering their web content. A content provider could host all its content on a single content server that would respond to all users' requests, delivering the base document as well as each embedded object. (A17241.) Alternatively, a content provider could outsource the entire process to an Internet Service Provider that would deliver the

base document and the embedded objects. (A274, 15:33-45, 16:37-69.) But Internet congestion problems quickly surfaced when a single content server received multiple, simultaneous requests for the same web page. (A274, 15:15-16; A17496.) Such requests, deemed “flash crowds,” may be anticipated, such as football fans visiting ESPN.com on Super Bowl Sunday, or unanticipated, such as the large amount of traffic received at CNN.com on September 11th. (A334-36:21-29.) Moreover, deliveries could be hampered when the Internet was congested in the server’s vicinity or the server was located far away from the user’s computer. (A17857; A336:26.)

To mitigate these problems, some content providers employed techniques such as “mirroring.” (A267, 1:35-41.) Specifically, they distributed identical (*i.e.*, mirrored) copies of web content over multiple servers at different locations. (*Id.*, A17496) To further improve service, content providers employed software and hardware solutions to deliver the requested content from a mirrored server based on the user’s location (A17860-64), and to balance the load across these servers (A17497-98).

But mirroring solutions had scalability problems. (A267, 1:51-59.) “Operational difficulties” occurred in that a content provider would “not only lease and manage physical space in distant locations, but [also would need to] buy and maintain the software or hardware that synchronizes and load balances the

[mirrored] sites.” (A267, 1:60-65; A273, 14:36-43; A17497.) Furthermore, as explained in the specification,¹ mirroring did not adequately address the problem of flash crowds. (A274, 15:15-25.)

3. The Patent Specification

In response to the problems with conventional methods for delivering content, the inventors of the patents-in-suit sought to provide a scalable solution that could effectively and efficiently deliver large amounts of web content and handle flash crowds.

To this end, the specification discloses delivering (or “serving”) embedded objects in a web page from a “domain other than the content provider’s domain.” Specifically, embedded objects are delivered from a “content delivery network” or “CDN” belonging to a “content delivery provider.” (A268, 3:8-14; A269, 5:33-37.) To make this process work, the inventors had to develop a way for Internet users to receive content from the CDN. (A339:40). To this end, the specification discloses that “the embedded object URL is first modified” (or, more broadly, “tagged”) to include a virtual server hostname to “condition the URL to be served by” the CDN. (A269, 6:41-46.) The virtual server hostname has several special

¹ The ’703, ’645, and ’413 patents are related and share the same written description.

features—it points to the CDN (unlike the original hostname, which points to the content provider), and it is “virtual” in the sense that it points to a “continually changing set or group of computers” in the CDN. (A268, 4:4-5; A269, 6:35-45; A342-44:53-60.)

Later, when a user’s computer seeks to retrieve an embedded object whose URL is modified with the virtual server hostname, the user’s local name server requests resolution of the virtual server hostname. In response, an “intelligent DNS” in the CDN (referred to as an “alternative DNS” in the ’645 claims) resolves the virtual server hostname into one or more IP addresses for optimal content servers in the CDN. (A270-71, 8:66-9:1; 9:22-25.) In particular, the intelligent DNS uses information in the virtual server hostname to select content servers that are preferably close to the end user, not overloaded, and likely to have the content requested. (A268, 3:10-14; A269, 5:37-41.)

This intelligent DNS can resolve only virtual server hostnames that point to the CDN. (A256, 9:13-15, 9:38-41; A17440.) It does not resolve entire URLs, paths, or object names. Further, it cannot resolve the hostname in an embedded object’s original URL because that hostname points to the content provider, not the CDN. (A343:54-57; A369:66-68.)

a. Various Methods for “Tagging” the URL Associated with an Embedded Object

When tagging the embedded object URL, the virtual server hostname may be substituted for the original hostname in the URL of an embedded object in various ways. As the specification explains, the key idea is to change the hostname in the URL so that the embedded object will be served from a content server in the CDN. (A269, 6:35-68.)

In a preferred embodiment, the URL is tagged by “prepending” (*i.e.*, inserting at the beginning) the virtual server hostname into an embedded object’s URL. (A270, 8:5-12.) But the specification makes clear that there are other ways to substitute the virtual server hostname. Indeed, the specification first discloses modifying the URL generally, stating: “*According to the invention*, the embedded object URL is first modified, preferably in an off-line process, to condition the URL to be served by the global hosting servers.” (A254, 6:54-57 (emphasis added).) After this general teaching, the specification discloses the “*preferred method* for modifying the URL,” in which “a virtual server hostname is *prepended* into the URL for the given embedded object” (A254, 6:57-58, 6:62-64 (emphasis added)).

The specification provides an example of prepending for the embedded image object “space.story.gif,” having an original URL of:

<http://www.provider.com/TECH/images/space.story.gif>

In the example, the virtual server hostname “ghost467.ghosting.akamai.com” is prepended into the original URL resulting in the following modified URL:

<http://ghost467.ghosting.akamai.com/www.provider.com/TECH/images/space.story.gif>

(A270, 8:4-13 (emphasis added).) The virtual server hostname (*i.e.*, ghost467.ghosting.akamai.com) points to the CDN; thus, requests for the embedded object resolve to the CDN instead of the content provider. Note that the original URL follows the first single slash and is no longer part of the hostname that gets resolved by DNS. (A369:67.) It is now part of the path and object name, which are not sent to DNS. (*Id.*, A369:67-68.)

The jury heard evidence that one of ordinary skill would have known of other conventional techniques for replacing a hostname that points to a content provider with a hostname that points to a CDN, including modifying a “CNAME record” within DNS. (A420-21:127-128; A442:38-40; A460:106-108; A571:63-65; A575:78-79.)

b. Various Levels of the Intelligent DNS Hierarchy

One issue in this appeal is whether the specification expressly limits the invention to selection of a name server in the CDN by a DNS server that is also in the CDN—*i.e.*, limiting the invention to a “two-tier DNS.” Thus, it is helpful to discuss this specific portion of the specification.

The specification discloses that the intelligent DNS may contain “several levels of processing” in which CDN DNS servers select lower-level DNS servers, which select lower-level DNS servers, and so on, down a “hierarchy” of DNS servers. (A271, 9:48-54.) In a “preferabl[e]” two-level embodiment, “there are two types of DNS servers,” referred to as “top-level and low-level,” respectively. (A271, 9:31-33.) In this embodiment, the top-level DNS servers select a low-level DNS server that is close to the user (and the low-level DNS server then selects content servers). (A271, 9:48-49; A268, 3:37-40; A269, 5:55-57.) But the specification makes clear that the invention is not limited to either (1) a two-level DNS or (2) selection of the close DNS server by a top-level DNS server. To the contrary, the specification discloses that “there may be additional levels in the DNS hierarchy” or, “[a]lternatively, there may be a single DNS level that combines the functionality of the top level and low-level servers.” (A269, 5:54-57.) Further, the specification discloses that the selection of the close DNS server may be accomplished by “other techniques.” (A271, 9:55-67.)

4. The '703 Patent Claims

At trial, Akamai asserted independent claims 19 and 34 and dependent claims 20-21 of the '703 patent. By its verdict, the jury determined that Limelight either performed (or directed or controlled the performance of) all elements of claims 19-21 and 34.

Claim 34 recites (with the step performed on behalf of Limelight in italics):

34. A content delivery method, comprising:

distributing a set of page objects across a network of content servers managed by a domain other than a content provider domain, wherein the network of content servers are organized into a set of regions;

for a given page normally served from the content provider domain, *tagging at least some of the embedded objects of the page* so that requests for the objects resolve to the domain instead of the content provider domain;

in response to a client request for an embedded object of the page:

resolving the client request as a function of a location of the client machine making the request and current Internet traffic conditions to identify a given region; and

returning to the client an IP address of a given one of the content servers within the given region that is likely to host the embedded object and that is not overloaded.

(A276 (emphasis added).)

Based on an earlier construction by Judge Zobel in another case, the parties agreed that “*tagging*” means “providing a ‘pointer’ or ‘hook’ so that the object resolves to a domain other than the content provider domain.” (A17874.) At trial, Akamai presented evidence that, in Limelight’s delivery service, Limelight creates unique tags (*i.e.*, virtual server hostnames) and provides them to content providers, along with detailed instructions to use the tags to tag embedded objects of a page.

Asserted claim 19 also requires “tagging,” and additionally recites the step of “serving [*i.e.*, delivering] the given page from the content provider domain,”

which Akamai argued (and the jury necessarily found) content providers perform under Limelight's direction or control.

5. The '645 Patent Claim

The '645 patent has one independent claim, claim 1, which recites (with the two disputed claim elements in bold and italics, and other limitations relevant to claim construction in italics alone):

1. In a wide area network in which an Internet domain name system (DNS) is useable to resolve DNS queries directed to participating content provider content that is available from participating content provider sites, a method of content delivery wherein participating content providers identify content to be delivered by a service provider from a set of content servers that are distinct from the participating content provider sites and associated with the service provider, wherein ***a given object of a participating content provider is associated with an alphanumeric string***, the method comprising:

having the service provider establish an alternative domain name system (DNS), distinct from the Internet domain name system and any client local name server, and *having authority to resolve the alphanumeric strings associated with the objects* identified by the participating content providers so that the objects identified by the participating content providers are available to be served from the service provider's content servers, *the service provider's alternative domain name system having one or more DNS levels*, wherein at least one DNS level comprises a set of one or more name servers;

for each of one or more participating content providers, delivering a given object on behalf of the participating content provider, wherein the given object is delivered by the following steps:

responsive to a DNS query to the given object's associated alphanumeric string, the DNS query originating from a client local name server, receiving the DNS query at a given name server of a lowest level of the one or more DNS levels in the service provider's alternative domain name system, ***the given name server***

that receives the DNS query being close to the client local name server as determined by given location information;

having the given name server that receives the DNS query *resolve the alphanumeric string into an IP address* that the given name server then returns to the client local name server, *wherein the alphanumeric string is resolved without reference to a filename for the given object*, wherein the IP address returned as a result of the resolution is associated with a content server within a given subset of the set of content servers, the subset of the set of content servers being associated with the given name server, the content server associated with the IP address returned by the given name server being selected according to a load sharing algorithm enforced across the subset of the set of content servers associated with the given name server;

at the content server associated with the IP address, receiving a request for the given object, the request having the filename associated therewith;

if the given object is available for delivery from the content server associated with the IP address, serving the given object from the content server.

(A260 (emphasis added).)

The preamble of claim 1 requires that “a given object of a participating content provider is associated with an alphanumeric string.” The district court held that this limitation means that the “alphanumeric string” must include “the URL used to identify the object in the absence of a content delivery network”—*i.e.*, the original URL. (A68.)

During prosecution, Akamai explained that “alphanumeric string” is “a known term of art for any character string up to 24 characters drawn from the

alphabet (a-z), digits (0-9), minus sign (-), and period(.).” (A16780.) Further, to overcome a written description rejection, Akamai cited examples of alphanumeric strings, each one being a hostname: a123.akamai.com; a1234.g.akamaitech.net; a123.g.g.akamaitech.net; and ghost1467.ghosting.akamai.com. (A16780; appearing in the patent at A256, 10:3, 10:18, 10:1, and 9:41, respectively.) Notably, none of the examples is a full URL or includes the URL used to identify the object “in the absence of a content delivery network” as the district court’s construction requires. (A68.)

The parties disputed another limitation in method claim 1: “the given name server that receives the DNS query being close to the client local name server as determined by given location information.” The district court interpreted this limitation, which does says nothing about selecting, as requiring that the name server be “selected by the alternative domain name system [*i.e.*, DNS].” (A72.)

6. The ’413 Patent Claims

Akamai asserted claims 8, 18, and 20 of the ’413 patent. Representative claim 8 recites (with the one disputed claim element in bold and italics, and other limitations relevant to construction of the disputed claim element in italics alone):

8. A method of content delivery wherein participating content providers identify content to be delivered by a content delivery network service provider from a set of content servers associated with the content delivery network service provider, wherein a given object of a participating content provider is associated with a Uniform Resource Locator (URL) that includes, in addition to a filename, an alphanumeric string, comprising:

having the content delivery network service provider establish a domain name system (DNS) having authority to resolve the alphanumeric strings in the URLs of the objects identified by the participating content providers, the content delivery network service provider's *domain name system having one or more DNS levels*, wherein at least one DNS level comprises a set of one or more name servers;

for each of one or more participating content providers, delivering a given object on behalf of the participating content provider, wherein the given object is delivered by the following steps:

responsive to a DNS query, selecting a given one of the name servers in the content delivery network service provider's domain name system;

at the given one of the name servers, resolving the alphanumeric string to an IP address, wherein the alphanumeric string is resolved without reference to the filename for the given object;

at a server associated with the IP address, the server being one of the set of content servers, receiving a request for the given object, the request having the filename associated therewith;

from the server, serving the given object; and

caching the given object at the server so that the given object is available for delivery from the server for a given time period in the event that a new DNS query to resolve the alphanumeric string is received at the domain name system and is resolved to the IP address of the server.

(A291 (emphasis added).)

Similar to its interpretation of claim 1 of the '645 patent (*supra*, at 19), the district court interpreted the limitation “responsive to a DNS query, selecting a given one of the name servers in the content delivery network” in claims 8, 18, and

20 as requiring that the selecting step be performed by specific *structure*—“the content delivery network’s domain name system [*i.e.*, DNS]” (A78)—even though the claims are method claims.

B. Akamai’s Content Delivery Service and Business

The inventions that led to the patents-in-suit were conceived by Tom Leighton (then a professor at MIT) and Daniel Lewin (one of Leighton’s graduate students). (A337:30-31; A346-47:68-70.) In August 1988, they left MIT and founded Akamai to commercialize the technology in the patents-in-suit. (A347:70-71.) Although the business model of content delivery services existed prior to Akamai’s formation in 1998, Leighton and Lewin revolutionized the business by implementing a CDN that took full advantage of the virtual server hostname and intelligent DNS concepts in the patents-in-suit. (A369:66.)

While they believed their invention had significant promise and could solve problems associated with conventional Internet content delivery, Leighton and Lewin faced significant skepticism about their solution from investors and potential Internet service providers. (A346:66-68.) They began to write software and build a prototype system. (A347:72-73.) In Akamai’s early days, Leighton and Lewin used their own credit cards to buy the computer servers and pay for rent. (A348:74.) After completing their prototype system, the inventors

demonstrated it to content providers such as Yahoo!, Disney, and CNN. (A348:74-75.) The reaction was mixed; there was both interest and skepticism. (A348:75.)

In 1998, Leighton and Lewin built a commercial-scale system. (A348:75-76.) Their “if we build it, they will come” strategy, however, was not successful—no content provider was willing to even test the system in early 1999. (A348-49:77-78.) Finally, Disney agreed to use a virtual server hostname for a single object on one of its web pages for 90 days. (*Id.*) In Spring 1999, Akamai’s fortunes began to look brighter, as the invention was used to deliver content for two significant Internet events: ESPN’s March Madness and a Star Wars movie trailer. (A349-50:78-83.) After CNN wrote a positive news story about Akamai, the CEO of Apple Computer, Steve Jobs, even offered to buy the company. (A350:83.)

While Akamai’s prospects were improving, it was not profitable in 1999. (A354:9.) Indeed, it was not until 2004 that the Company earned a profit on its use of the invention. (A354:9.) By 2007, its annual revenues using the invention grew to over \$600 million. (A354:9.) Today, its business is a substantial success. (A355:11.)

C. Limelight’s Relationship with Content Providers

By its verdict, the jury found that Limelight’s content delivery service includes each of the steps of claims 19-21 and 34 of the ’703 patent. Accordingly,

it is not necessary to address the specific way in which all steps are performed. Rather, the relevant issue is whether Limelight controls or directs content providers to perform the tagging and serving steps that Limelight alone does not perform. Thus, a discussion of the relationship between Limelight and content providers will be helpful.

1. Limelight’s Contract Requires Content Providers to Perform the Claim Steps in Order to Use Limelight’s Service

At trial, Akamai presented evidence that Limelight’s standard contract obligates content providers to tag the web content that they want delivered by Limelight so that requests for that content resolve to the Limelight CDN instead of the content provider. (A17803, A587:121.) In turn, Limelight’s contractual obligation to provide its content delivery service is contingent on the content provider performing the tagging step. Specifically, the contract states:

Customer [*i.e.*, content provider] shall be responsible for identifying via the then current Company process all uniform resource locators (“URLs”) of the Customer Content to enable such Customer Content to be delivered by the Company Network [*i.e.*, Limelight’s Network].

(A17807.) As the jury heard, this step of “identifying” all URLs corresponds to the claimed tagging step. (A587:121-122.) Limelight’s contract permits no other way for content providers to use Limelight’s content delivery service. The contract further provides:

Customer [*i.e.*, content provider] shall provide Company with all cooperation and information reasonably necessary for [Limelight] to implement the [Limelight Content Delivery Service].

(A17807.)

The contract further contemplates that the content provider must deliver (*i.e.*, “serve”) the web pages containing the tags when requested by the user. (A441-42:37-38.) Otherwise, Limelight’s network will not “see” the user’s request for content and will be unable to perform the contracted service for which the content provider has paid. (A587:120-122.) Section 3.1 of the Limelight standard contract, titled “Content Delivery Service Availability,” confirms that Limelight expects and requires content providers to serve the page, stating:

Service Interruptions caused by . . . failure of [content provider] origin server (equipment down, *not serving content* [*e.g.*, pages], broken links or similar issues that would prevent the [Limelight] Service from working successfully, . . .) are ineligible for [Limelight’s] availability guarantee.

(A17807 (emphasis added).) In other words, the contract holds the content provider responsible for serving the web page.

2. Limelight Provides Specific Instructions to Content Providers to Perform the Claim Steps Via the “Then Current Company Process”

Akamai also presented evidence at trial that, after the content provider signs the contract, Limelight begins its installation process, which includes the tagging step. During that time, Limelight promises to “provide a seamless and smooth integration of *our* services into” the content provider’s business and that “*our*

implementation should minimize the impact of [the content provider's] transition to Limelight.” (A17789 (emphasis added).)

To this end, Limelight provides a virtual hostname to the content server for tagging embedded objects so that requests for those objects resolve to the Limelight CDN. The jury heard that Limelight sends the content provider a Welcome Letter containing a Limelight-assigned hostname (*e.g.*, xyz.vo.llnwd.net) that the content provider must “integrate” into its web pages through tagging. (A17790, A17237.) The “llnwd.net” portion of the Limelight-assigned hostname points to the Limelight CDN. (A583:106.) The “vo” portion tells the Limelight DNS to return the IP address of a particular type of content server in Limelight’s content delivery network. (A583-84:106-108.) The “xyz” portion of the assigned hostname is “a label that is unique to the [content provider].” (A583:107.)

The content provider must use this exact assigned hostname for the Limelight content delivery service to deliver its content, because “[t]his unique identifier is used by [Limelight’s] ContentEdge to identify a particular content provider.” (A17220, A587:122, A17583.) If the content provider fails to use the exact assigned hostname (*e.g.*, xyz.vo.llnwd.net), Limelight’s network will not see the request for the content provider’s content and cannot respond. (A587:122.)

Akamai also presented evidence that Limelight sends Installation Guidelines to content providers. The Installation Guidelines “outline[] the steps and

information needed” to install Limelight’s service, including two methods for identifying (*i.e.*, tagging) the content provider’s web content to be delivered by Limelight’s ContentEdge system:

(1) Prepending (referred to as “Limelight Origin (Prepend)”) (A17220; A570:58-59); and

(2) CNAMEing (referred to as “CNAME/Customer Origin”) (A17789; A17791; A584:109-110; A587:121-122).

The Installation Guidelines specify the steps the content provider must perform to complete either method. (A17791.)

Whether prepending or CNAMEing is used, Limelight engineers initially perform extensive installation and quality assurance testing. (*Id.*) As Akamai explained at trial, when the content provider uses the prepending method, Limelight directs the content provider to update its links to use what Limelight calls a “Prepend URL.” (A17792; A584:110-111.) Specifically, Limelight engineers generate a custom Prepend URL for each content provider using the Limelight-assigned hostname. (A17235.) The jury heard that Limelight engineers are responsible for testing the custom Prepend URL to make sure it is correct. (A17235.) Then Limelight directs the content provider to “modify their HTML source code to insert the [Limelight] Prepend URL for any objects that they wish to have cached [*i.e.*, stored] on [Limelight’s content delivery network].” (A17235,

A17792.) Limelight’s engineers work with the content provider to resolve any issues or problems with this process. (A17235.)

Alternatively, when the CNAMEing method is used, Limelight directs the content provider to modify its CNAME record to redirect users’ requests, allowing Limelight to act as the origin for the content. (A17220, A17263, A570:60-61.) To complete this process, the content provider need only complete two additional steps. (A17791.) First, Limelight directs the content provider to update the CNAME table with the Limelight-assigned hostname (*e.g.*, xyz.vo.llnwd.net). (A17791, A17220.) Second, Limelight directs the content provider to use the CNAME for those embedded objects on the content provider’s web site that the customer wants delivered over Limelight’s network. (A17220, A17791.)

Limelight engineers are available to assist throughout installation. (A17794-95.) Indeed, Akamai presented evidence that Limelight takes the lead in the installation process by dedicating a Technical Account Manager who “coordinate[s] between your [the content provider’s] technical team and [Limelight] engineers to ensure a quick and complete implementation of [Limelight content delivery] services.” (A17790.) Limelight’s Technical Account Managers “track the progress” of each step of the installation process. (*Id.*, A17235.)

Finally, the jury heard testimony that once the content provider has inserted the Limelight-assigned hostname into its web pages or modified the CNAME record, a user's request for content is redirected to the Limelight content delivery network. (A628:123-124.) Upon request from a user, the content provider delivers (or serves) the web page with the tagged URLs. If the content provider does not deliver the pages with the tagged URLs as instructed by Limelight, the Limelight system will not deliver the content. (A586:118-119; A587:121-122; A441-442:37-38.)

D. District Court Proceedings

1. The Court's Claim Construction

a. "A Given Object . . . Is Associated With an Alphanumeric String" in Claim 1 of the '645 Patent

The parties disputed the meaning of the limitation "a given object of a participating content provider is associated with an alphanumeric string" in the '645 claim 1 preamble. Akamai asserted that the claim "language at issue need not be construed at all, as the type of 'association' called for by Claim 1 is made clear by the claim itself." (A16847.)

Limelight contended that the limitation means "a particular object is associated with an alphanumeric string *by combining it with the domain name [i.e., hostname] conventionally used by the content provider to identify the content.*" (A16895-97, A16856 (emphasis added).) Limelight argued that "the specification

describes only one method to associate a given object with an alphanumeric string”—*i.e.*, by “combin[ing] the string with the domain name conventionally used by the content provider to identify the object”—and, thus, the claims must be so limited. (A16856, A15137.)

In response, Akamai argued that there is no basis for Limelight’s narrow construction, noting the claim language “simply requires that there be an association, the characteristics of which are spelled out [later] in the claim itself.” (A16848.) Akamai further argued that, although the specification discloses a preferred embodiment in which the content provider’s hostname is retained in the tagged URL, there is no basis for limiting the claims to that embodiment. (A16849, A16851.)

While it did not adopt Limelight’s proposed construction verbatim, the district court held that the term “a given object of a participating content provider is associated with an alphanumeric string” means “a *particular* object of a participating content provider is associated with an alphanumeric string *that includes the URL used to identify the object in the absence of a content delivery network.*” (A68 (emphasis added).) In so doing, the court simply repeated the words of the claim limitation, substituting the term “particular” for the term “given,” and added the further requirement that the alphanumeric string “include[]

the URL used to identify the object in the absence of a content delivery network.”

(*Id.*)

The district court reasoned that “the specification describes the invention as associating a particular object of a content provider with an alphanumeric string consisting of a virtual server hostname prepended onto the URL for the object” (A69 (emphasis in original)), ignoring parts of the specification describing the prepending embodiment as simply “preferred” (A254, 6:51-58.) The court also reasoned that “[t]he URL of the object *is necessary* to the inventive global framework in order to retrieve the object” (A69 (emphasis added)), even though the specification explains that the object may be retrieved in other ways.

b. “The Given Name Server That Receives the DNS Query Being Close to the Client Local Name Server as Determined by Given Location Information” in Claim 1 of the ’645 Patent

The parties also disputed the meaning of “the given name server that receives the DNS query being close to the client local name server as determined by given location information” in claim 1 of the ’645 Patent. For its part, Limelight argued that the court should read in the requirement that “the given name server that receives the DNS query” must be “selected by the alternative domain name system” because “[t]he patent discloses only one way of determining that a ‘given name server’ will receive DNS queries.” (A15146-47.)

Akamai responded that the claim language “only requires that a DNS query be received by the name server in the content delivery network” and does “not require that the DNS be selected by anything or by any method.” (A15200.) Akamai further explained that, by requiring that the “particular name server that receives the DNS query” be “selected” by “the alternative domain server system,” Limelight’s construction improperly limits claim 1 of the ’645 patent to a two-level DNS system. (*Id.*) Akamai explained that, while the ’645 patent discloses a preferred embodiment in which a top-level domain name server selects a low-level domain name server (*i.e.*, a two-level DNS), the specification discloses that one, two, or three DNS levels may be used and claim 1 expressly states that the “service provider’s alternative domain name system ha[s] *one* or more DNS levels.” (A15200.)

The district court nonetheless agreed with Limelight and interpreted the term as “the particular name server that receives the DNS query *is selected by the alternative domain name system.*” (A72 (emphasis added).) Referring to its prior discussion of “alphanumeric string,” the court again emphasized what it deemed to be limiting language in the specification. The district court interpreted the specification narrowly, describing “the present invention” as “manipulat[ing] the DNS system so the name is resolved to one of the [content servers] that is near the client” (A74 (emphasis in original)), notwithstanding that the specification uses

non-limiting language like “preferable” and “general” in describing selection of the name server.

c. “Selecting a Given One of the Name Servers in the Content Delivery Network” in Claims 8, 18, and 20 of the ’413 Patent

Finally, the parties disputed the meaning of “responsive to a DNS query, selecting a given one of the name servers in the content delivery network” in claims 8, 18, and 20 of the ’413 patent. As with the “given name server that receives the DNS query” limitation in ’645 patent claim 1, the disagreement focused on whether the method claims should be limited to selection of the name server by “*the content delivery network’s domain name system.*”

The district court again adopted Limelight’s claim construction, interpreting “responsive to a DNS query, selecting a given one of the name servers in the content delivery network” as “in response to a DNS query, *the content delivery network’s domain name system selects* a particular name server.” (A78 (emphasis added).) The court rejected Akamai’s argument that this construction would require a two-level DNS system. (A79.) To this end, the district court opined—without any basis in the patent specification or otherwise—that a single-level DNS system could “contact a content delivery provider’s top-level name server,” which would then “directly communicate with a particular local name server, based on

the user's location, to resolve the server's IP address and return it to the user, rather than requir[ing] the user to conduct a second lookup." (A79-80.)

2. The Jury Verdict

After the close of evidence, the jury was properly instructed on the "control or direction" test. (A49-51.) Based on that instruction, the jury found that Limelight infringed claims 19, 20, 21, and 34 of the '703 patent. (A93-94.) The also jury found that the claims were not invalid under §§ 102, 103, or 112 and awarded lost profits of \$40,102,000 and reasonable royalties of \$1,424,946, with prejudgment interest for both, and price erosion of \$4,000,000. (A94-99.)

3. The District Court's Entry of JMOL

Following the verdict, Limelight moved for JMOL of noninfringement, on the ground that substantial evidence did not support the verdict that Limelight directs or controls the tagging step (in claims 19-21 and 34) and the serving step (in claims 19-21). (A51.) Initially, the district court denied the motion "because, unlike in *BMC Resources*, here there was evidence that, not only was there a contractual relationship between Limelight and its customers, but that Limelight provided those customers with instructions explaining how to utilize its content delivery service." (A51.)

Subsequently, this Court issued its decision in *Muniauction*, and Limelight moved for reconsideration on the ground that this Court's intervening decision in

Muniauction “established that the joint infringement theory relied on by Akamai in the present case requires a showing that the accused direct infringer is vicariously liable for acts committed by any others.” (A17878.) Alternatively, Limelight argued that *Muniauction* “held as a matter of law that an accused infringer’s control over access to an Internet-based system, coupled with instructions to customers on how to use that system, is insufficient to establish direct infringement.” (*Id.*)

Akamai responded that *Muniauction* did not articulate new law, but rather applied the *BMC Resources* standard of direction or control to a different set of facts. (A17884.) Further, Akamai argued that there was sufficient evidence for the jury to find infringement under that standard and that the facts of this case were “grossly different” from *Muniauction*. Akamai noted that here, unlike *Muniauction*, the standard contract between Limelight and a content provider requires the content provider to perform the very step in the claim in order to use Limelight’s delivery service. (A17886.)

On reconsideration, the district court granted JMOL of noninfringement to Limelight. At the outset, the court held that *Muniauction* did not hold that vicarious liability is “a necessary condition to satisfy *BMC Resources*’ control or direction standard.” Rather, it is “merely a condition sufficient to find infringement within the spectrum of possible interactions ranging from an arms-

length agreement to ‘contracting out steps of a patented process to another entity.’” (A53.)

Notwithstanding that *Muniauction* relies on the same “control or direction” standard as *BMC Resources*, the district court agreed with Limelight that *Muniauction* did change the requirements for joint infringement, making it impossible for Akamai to prove control or direction under the facts presented. The court recognized that “Limelight’s customers . . . follow[] Limelight’s instructions . . . [to] modify the embedded objects of their web pages or alter their DNS records so that requests for the objects resolve to the content delivery service domain.” (A58.) The court also recognized that, under Limelight’s contract, a “customer must perform one or more steps of the patented process in order to receive the benefits of” Limelight’s service (A57) and that, in contrast, the *Muniauction* decision makes no mention of a contract other than to “quote[] the lower court’s finding that the defendant charges the bidders a fee for its services” (A55.) Notwithstanding these significant differences, the court held that there is “no material difference between Limelight’s interaction with its customers and that of Thomson in *Muniauction*.” (A58.)

To this end, the court characterized Limelight’s contract primarily as “the provision of a service in exchange for payment.” (A56.) With this

characterization in mind, the court concluded that “the existence in the instant case of a contract for services does not give rise to direction or control.” (A57.)

IV. SUMMARY OF ARGUMENT

The district court’s JMOL of noninfringement should be reversed because substantial evidence supports the jury’s verdict that Limelight controlled or directed content providers in performing the claim steps it alone does not perform. First, when the invention is used, Limelight’s form contract expressly requires content providers to perform the tagging and serving claim steps that Limelight does not alone perform. In the words of this Court in *BMC Resources*, 498 F.3d 1381, Limelight “contracts out steps of a patented process to another entity.”

Moreover, Limelight participates in, controls, and directs the claimed *tagging* step by creating and assigning the *tag*—*i.e.*, the virtual server hostname. Limelight not only creates the tag, but it then directs content providers to use that tag and serve the tagged page so that requests for the object resolve to Limelight’s CDN. Further, as the jury heard, Limelight’s content delivery service will not work if content providers do not use the Limelight-supplied hostname or follow Limelight’s specific directives. And, if that were not enough, Limelight offers technical assistance to help in the claimed process.

Furthermore, Limelight’s own documents make it clear that the entire claimed process—including tagging and serving the tagged page—is Limelight’s

developed process, not the content provider's. None of the cases cited by the district court involved anywhere near this level of control or direction. Given these facts and the standard of review, this Court should reverse the district court's JMOL of noninfringement.

The district court also erred in interpreting "a given object of a participating content provider is associated with an alphanumeric string" in the preamble of claim 1 to require that the alphanumeric string include the object's original URL. This interpretation is not supported by the claim language, specification, prosecution history, or stipulated construction of "alphanumeric string." In fact, it is non-sensical in view of this evidence.

First, "alphanumeric string" was expressly defined in the prosecution history as "a character string up to 24 characters drawn from the alphabet (a-z), digits (0-9), minus signs (-), and periods (.)." The parties even stipulated to this definition. The district court's construction, which expressly states that the alphanumeric string "includes" the original URL, contradicts this construction.

Moreover, the district court's construction makes no sense in light of the remaining claim limitations. The term "alphanumeric string" appears four times in the body of claim 1 and these other claim limitations make clear that the "alphanumeric string" is not a URL, but rather a hostname that is resolved by DNS. Because a hostname cannot include a URL (and a full URL is not resolved),

the district court's construction contradicts the remaining claim limitations. *Cf. Rexnord Corp. v. Laitram Corp.*, 274 F.3d 1336, 1342 (Fed. Cir. 2001) (reversing because district court's construction was inconsistent with later claim language).

The prosecution history further supports that the “alphanumeric string” is a hostname and does not include the original URL. Akamai cited several examples of “alphanumeric strings” during prosecution and, notably, they are all hostnames. None is a URL, and none includes “the URL used to identify the object in the absence of a content delivery network” as required by the court's construction.

Building on this point, because the “alphanumeric string” is a hostname, the district court's construction creates a requirement found nowhere in the specification. Indeed, there is no support (in any embodiment) for including the original URL as part of the virtual server hostname. For all of these reasons, this court should reverse the district court's construction.

Likewise, this Court should reverse the district court's construction of claim 1 of the '645 patent and claims 8, 18, and 20 of the '413 patent as requiring selection by DNS. Initially, each claim is directed to a *method* of content delivery, not to the *structure* of a content delivery system. Thus, this Court should reject the district court's construction because it improperly incorporates a structural limitation—a DNS—into a method claim.

Moreover, Limelight’s argument that the claims should be so limited because the specification allegedly “discloses only one way” of performing the claim steps must be rejected. First, as this Court has repeatedly held, depiction of a single preferred embodiment in a patent does not necessarily limit the claims to that depicted scope. *Agfa Corp. v. Creo Prods. Inc.*, 451 F.3d 1366, 1376 (Fed. Cir. 2006). Second, the specification suggests that other structures may be used.

Finally, under the district court’s construction, if the content delivery network’s DNS was the only structure that could select a name server, the content delivery network’s DNS would, by definition, be at least a two-level DNS (with the name server performing the selecting as one level and the name server being selected as the second level). But construing the claim to exclude a one-level DNS is inconsistent with the claim language, specification, and prosecution history—all of which describe the invention as including a one-level DNS.

V. ARGUMENT

A. The Court Should Reverse the District Court’s JMOL of No Joint Infringement

1. Substantial Evidence Supports the Jury Verdict of Joint Infringement

“[W]here the actions of multiple parties combine to perform every step of a claimed method, the claim is directly infringed only if one party exercises ‘control or direction’ over the entire process such that every step is attributable to the controlling party, *i.e.*, ‘the mastermind.’” *Muniauction*, 532 F.3d at 1329.

Whether the accused infringer exercises “control or direction” over the entire claimed process is a factual question that this Court reviews for substantial evidence. *BMC Resources*, 498 F.3d at 1381. The jury’s verdict of infringement “must be upheld unless the facts and inferences, viewed in the light most favorable to the verdict, point so strongly and overwhelmingly in favor of [Limelight] that a reasonable jury could not have returned the verdict.” *Borges Colon v. Roman-Abreu*, 438 F.3d 1, 14 (1st Cir. 2006) (internal quotation marks and citations omitted).

Here, substantial evidence supports the jury’s determination that Limelight exercises control or direction over the entire claim process. Limelight alone performs every step of claim 34 except tagging and every step of claims 19-21 except tagging and serving. For those steps that Limelight does not perform itself, Limelight both controls (through contractual requirements) and directs (through explicit detailed technical instructions) content providers to perform those steps.

First, Limelight contracts out to content providers the claim steps that it alone does not perform. Limelight’s standard form contract obligates content providers to perform the claim steps of tagging the embedded objects and serving the tagged page so that requests for the embedded objects resolve to Limelight’s network instead of the content provider’s. (A17807.) The contract states: “Customer [*i.e.*, content provider] **shall** be responsible for identifying via the then

current [Limelight] process all uniform resource locators (“URLs”) of the Customer Content to enable such Customer Content to be delivered by [Limelight].” (A17807.) The contract further states: “Customer *shall* provide [Limelight] with all cooperation and information necessary for [Limelight] to implement the [Content Delivery Service].” (*Id.*)

Thus, the jury received unequivocal evidence that, when the invention is used, Limelight obligates the content provider to perform the claim steps that it does not perform. As this Court has explained, “[a] party cannot avoid infringement [] simply by contracting out steps of a patented process to another entity.” *BMC Resources*, 498 F.3d at 1381. “It would be unfair indeed for the mastermind in such situations to escape liability.” *Id.*

In addition, the jury heard undisputed testimony that Limelight further participates in, controls, and directs the tagging process by creating and assigning a unique hostname for the content provider. As the jury heard, “tagging” means “providing a ‘pointer’ or ‘hook’ so that the object request resolves to a domain other than the content provider domain.” (A396:35-36, A17874.) The Limelight-assigned hostname is the “pointer” or “hook.” Thus, its creation is integral to the process of tagging. (A587:122-23.) Indeed, the jury heard that Limelight’s content delivery service will not work if content providers do not use the Limelight-supplied hostname to identify the objects. (A587:122-23.) Limelight’s

provision of the very “pointer” or “hook” further supports the jury’s finding of control or direction.

Further, by including direction, the “control *or* direction” test must include something other than control, and this case presents the ultimate in direction. As presented to the jury, Limelight provides explicit step-by-step instructions to perform the missing claim steps. (*Supra*, at 24-28.) Pages 5-6 of Limelight’s Installation Guidelines outline the steps required to tag the embedded objects. (A17791-92.) Among other things, they state that the content servers “*will need to do* the following to complete the transition . . . (ii) update web site links to use the new [Limelight] URL.” (A17792 (emphasis added).) And, in the CNAME method, they state that content servers “*will need to do* the following to complete the transition . . . (i) [u]pdate customer DNS with new CNAME(s)” and (ii) “update web site links to use new [Limelight] published CNAME(s).” (A17791 (emphasis added).) Further, as the jury heard, content providers must follow the directives provided by Limelight because Limelight’s content delivery process will not work if content providers do not tag the objects or serve the tagged page as specifically instructed by Limelight. (A587:122-23.)

And, if that were not enough, Limelight offers technical assistance to help in the process. (A17790.) As Limelight’s Installation Guidelines explain, a Limelight employee “coordinate[s] between the customer’s technical team and

Limelight engineers,” “tracks the progress of the order,” and “ensure[s] a quick and complete implementation” of Limelight’s service. (*Id.*) In other words, Limelight tells content providers what to do, manages the process, and follows up to ensure that the steps are performed correctly.

Furthermore, Limelight’s own documents (including the Installation Guidelines) make it very clear that the entire claimed process—including tagging and serving the tagged page—is Limelight’s developed process, not the content server’s. The Installation Guidelines refer to Limelight’s “goal,” Limelight’s “service,” and Limelight’s “implementation,” while specifying that Limelight’s “service” and “implementation” include the steps of tagging and serving. (A17789-91.) Similarly, Limelight’s form contract requires customers to follow “the then current [Limelight] process” for tagging. (A17807.)

For all of these reasons, a reasonable jury could have easily found that Limelight is the “mastermind” behind the process, including the steps performed by content providers. The facts and inferences, viewed in the light most favorable to the verdict, do not “point so strongly and overwhelmingly in favor of [Limelight] that a reasonable jury could not have returned the verdict.” *Borges*, 438 F.3d at 14.

2. The District Court Erred in Its Analysis of *Muniauction*

a. The District Court Erred in Finding “No Material Difference Between Limelight’s Interaction with Its Customers and That of Thomson in *Muniauction*”

The district court’s opinion is premised on its incorrect finding that there is “no material difference between Limelight’s interaction with its customers and that of Thomson (the accused infringer) in *Muniauction*.” Comparison of the facts in both cases reveals the court’s error.

First, the evidence of direction or control in *Muniauction* only tangentially related to the claimed process. In *Muniauction*, the claim step performed by bidders required “inputting data associated with at least one bid . . . into said bidder’s computer.” *Muniauction*, 532 F.3d at 1330. But the evidence of direction or control related to “control[ing] access to [Thomson’s] system,” not directing users on inputting bid information. *Id.* In contrast, Limelight’s contract expressly requires customers to perform the “tagging” and “serving” steps when the claimed process is used, and Limelight provides explicit instructions to perform those steps.

Second, the *Muniauction* opinion says nothing about the contract between Thomson and the bidders and whether bidders were obligated to perform the claim step. This is in stark contrast with this case, where Limelight contracted out the performance of the claim step or steps to content providers. Indeed, when the ’703

patented invention is used, content providers are contractually obligated to perform the claim steps that Limelight does not perform.

Finally, the level of instruction or direction is materially different. In Thomson's bidding system, Thomson did not tell bidders what bid to enter. Here, on the other hand, Limelight's level of control or direction is so significant that Limelight tells content providers not only how to tag, but also what to tag with. Contrary to the district court's suggestion (A57), the Limelight-supplied hostname is not "similar" to the bidder ID and password that Thomson supplied in *Muniauction*. Here, the hostname is the tag itself and, thus, is integral to the performance of the claimed steps of tagging and serving the tagged page. And Limelight's contract requires that the tag be used every time the infringing process is used. (A17807.) The bidder ID and password in *Muniauction*, on the other hand, merely provided access to Thomson's bidding system; it was merely incidental to the claimed inputting step.

Given the material differences between this case and *Muniauction*, the district court erred in holding that, on the continuum between an arms-length relationship and vicarious liability, this case, like *Muniauction*, falls on the arms-length relationship side of the spectrum. To the contrary, the properly-instructed jury heard substantial evidence to support its conclusion that this case satisfies the control or direction standard.

b. *Muniauction* Did Not Establish that “Direction or Control” Can Never Be Shown by a “Contractual Agreement to Pay for a Defendant’s Services”

The district court also mistakenly held that “*Muniauction* establishes that direction or control requires something more than merely a contractual agreement to pay for a defendant’s services and instructions or directions on how to utilize those services.” (A55.) According to the court, “Limelight does not ‘contract[] out steps of a patented process to another entity’ because ‘the fundamental agreement between Limelight and its customers is the provision of a service in exchange for payment.’” (A56.) Instead of focusing on the obligations imposed by the contract, the court focused on the supplier-customer relationship between Limelight and the content providers.

But neither *Muniauction* nor *BMC Resources* precludes a finding of “direction or control” simply because Limelight provides a service and content providers pay for that service. While direction or control might more frequently exist between an employer and a contract employee, for example, there is no *per se* rule prohibiting a finding of direction or control between a service provider and a customer, and the district court erred in holding otherwise.

Moreover, that a content provider could choose not to use Limelight’s service is not dispositive of the control inquiry. The critical issue here is whether Limelight controls or directs the performance of the claimed process when the

process is used. The district court erred by focusing on what happens when the claimed process is *not* used, as opposed to when it is used. And a reasonable jury could have found that, when the invention is used, content providers are obligated by the contract to perform the missing claim steps.

The district court also erred in reasoning that direction or control cannot be established “by the existence of a contractual relationship in which one entity *incidentally* has to perform a step of a patented process to receive the benefits of the contract” given this Court’s holding that “mere ‘arms-length cooperation’ will not give rise to direct infringement by any party.” Even if this premise were true as a general matter, it does not apply to the facts in this case. Indeed, while Limelight and content providers may be related as independent contractors, a content provider’s performance of the missing claim steps is hardly *incidental*. To the contrary, Limelight’s contract expressly obligates content providers to perform those steps when the invention is used, and Limelight’s other documents tell content providers exactly how to do so.

The district court also erred in finding that, while “Limelight’s customers follow[] Limelight’s instructions” to tag, they do so “not because they are contractually obligated to do so; rather, they do so because they wish to avail themselves of Limelight’s service.” (A58-59.) The court’s conclusion in this regard demonstrates a fundamental misunderstanding of the “direction or control”

test. The relevant inquiry is not whether customers could have used another content delivery service, but rather whether, when the claimed process is performed, Limelight “exercises ‘control or direction’ over the entire process such that every step is attributable to the controlling party, *i.e.*, the ‘mastermind.’” *Muniauction*, 532 F.3d at 1329. This Court’s precedent does not require plenary control over the content providers. Rather, Akamai need only show control or direction over the performance of the claim steps when the claimed process is performed. For the reasons discussed above, a reasonable jury could have found Akamai met this burden.

Finally, regarding independent claim 19, the district court erred in holding that “serving the initial web page from the content provider’s domain, is performed by the content provider whether it subscribes to Limelight’s services or not.” (A58.) This is not correct. The claims call for serving the “given page”—*i.e.*, the one that was “tagged,” not just any ordinary page. Moreover, as described *supra* at 24, Limelight directs the content provider to serve the tagged page.

c. *Muniauction* Did Not Eliminate the *BMC Resources* Possibility of Satisfying the Control or Direction Test by Providing Instructions or Directions

The district court also erred by holding that *Muniauction* eliminated the *BMC Resources* possibility of satisfying the control or direction test by, among other things, providing instructions or directions. (A54-55.) While the Court in

Muniauction did describe as not relevant a jury instruction asking jurors if “there [was] one party teaching, instructing, or facilitating the other party’s participation in the electronic auction process,” 532 F.3d at 1329, the Court also affirmed the “control or direction” test. The Court’s inclusion of the “direction” language in the standard makes clear that instructions—particularly mandatory ones—are relevant.

Moreover, the Court’s statement must be understood in the context of the “teaching, instructing, or facilitating” evidence in *Muniauction*. “Most, if not all, ‘tests’ employed in the art of judging arise in a particular factual milieu. Hence they must be read, applied, and perhaps modified in light of the facts of subsequent cases.” *Arrowhead Indus. Water, Inc. v. Ecolochem, Inc.*, 846 F.2d 731, 736 (Fed. Cir. 1988), abrogated on other grounds, *SanDisk Corp. v. STMicroelectronics, Inc.*, 480 F.3d 1372 (Fed. Cir. 2007).

In *Muniauction*, Thomson provided little to no direction to bidders on how to perform the claim step. Here, in contrast, Limelight provides Installation Guidelines with step-by-step instructions to perform the claim steps, as well as a customized hostname for use in performing the claim steps. Further, Limelight employees provide assistance in performing the claim steps and Limelight’s contract and other documents make clear that Limelight designs, directs, and requires performance of the entire claim process whenever the Limelight content delivery service is used. As the district court recognized, “Limelight’s customers

follow[] Limelight’s instructions [to] modify the embedded objects of their web pages or alter their DNS records so that requests for the objects resolve to the content delivery service domain.” (A58.) These facts support the jury verdict and should not have been considered irrelevant.

Finally, in analogous contexts involving control or direction standards or vicarious liability, other courts have found instructions relevant. *See, e.g., RCA/Ariola Int’l, Inc. v Thomas & Grayston Co.* 845 F.2d 773 (8th Cir. 1988) (manufacturer of cassette duplicating machines held vicariously liable for copyright infringement because manufacturer had retained control over retailers’ use of machines by providing instructions for use). For all of the above reasons, the district court’s JMOL should be reversed and the jury’s verdict of infringement reinstated.

B. The District Court’s Construction of the ’645 and ’413 Patent Claims Should Be Reversed

This court reviews claim construction *de novo* on appeal. *Cybor Corp. v. FAS Techs., Inc.*, 138 F.3d 1448, 1456 (Fed. Cir. 1998) (en banc).

1. “Associated With an Alphanumeric String” in Claim 1 of the ’645 Patent Should Not Be Construed to Include the Content Provider’s URL

The district court erred in interpreting “a given object of a participating content provider is associated with an alphanumeric string” in the preamble of claim 1 to require that the alphanumeric string must “include[] the URL used to

identify the object in the absence of a content delivery network”—*i.e.*, the original URL. (A70.) As explained below, there is no basis in the claims, specification, or prosecution history for reading in this requirement.

a. The Court’s Original URL Interpretation Is Inconsistent with the Claim Language, Specification, Prosecution History, and Stipulated Definition

First, nothing in the claim language supports the court’s construction. “Alphanumeric string” does not appear in the specification and, thus, it should be given its ordinary meaning. *Verizon Servs. Corp. v. Vonage Holdings Corp.*, 503 F.3d 1295, 1304 (Fed. Cir. 2007) (“Since the specification does not define the term ‘server,’ we look to its ordinary meaning to a person of ordinary skill in the art.”). To one of ordinary skill in the art, the ordinary meaning of “alphanumeric string” is a series of letters and numbers. (*See e.g.*, A16780 (citing Internet Standards).) Indeed, the parties stipulated that “alphanumeric string” is “a character string up to 24 characters drawn from the alphabet (a-z), digits (0-9), minus signs (-), and periods (.)” (A17875.) This is the very same definition Akamai proffered during prosecution of the ’645 patent. (A16780.) Thus, there is no basis for reading the original URL into the term “alphanumeric string.”

Likewise, the specification and prosecution history do not define “associated” as having a meaning other than its ordinary meaning. (A254, 6:48-49.) Indeed, during the claim construction hearing, the district court and the

parties all agreed that “associated” should have its ordinary and customary meaning. (A16897-98.) Accordingly, there is no basis for reading the original URL into the term “associated.”

Limelight nonetheless argues that the claims should be limited because “the specification describes only one method to associate a given object with an alphanumeric string”—*i.e.*, by prepending the virtual server hostname to the original URL used to identify the object in the absence of a content delivery network. But this Court has repeatedly held that it is improper to read a preferred embodiment into a claim. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1323 (Fed. Cir. 2005) (*en banc*) (“[A]lthough the specification often describes very specific embodiments of the invention, we have repeatedly warned against confining the claims to those embodiments.”)

Further, contrary to the district court’s conclusion (A69), the specification does *not* describe “the invention as associating a particular object of a content provider with an alphanumeric string consisting of a virtual server hostname prepended onto the URL for the object.” Rather, the specification very clearly indicates that tagging by prepending a virtual server hostname onto the object’s URL is merely a “preferred method for modifying the object URL.” (A256, 6:57-58.) Thus, one of ordinary skill in the art would understand that other tagging

methods may be used to, in the language of the claims, “associate an alphanumeric string” with the object.

The district court’s construction also contradicts the prosecution history. During prosecution, Akamai cited examples of alphanumeric strings from the specification. (A16780.) As discussed *supra* at 18-19, every single cited example is a hostname. One of the cited examples, “a1234.g.akamaitech.net,” is referred to in the specification as a “representative hostname.” (A255, 7:14-15; *see also* A256, 9:38-41.)

None of the examples cited by Akamai during prosecution is a full URL (*e.g.*, <http://www.cnn.com/world/picture.jpg>). None includes an object or file name (*e.g.*, “space.story.gif,” (A255, 8:66); “frontpage.jpg,” (A256, 9:25-26)). And, most importantly, none includes “the URL used to identify the object in the absence of a content delivery network” as required by the court’s construction. Thus, the court’s construction also contradicts the prosecution history and, for this additional reason, should be rejected. *Masco Corp. v. U.S.*, 303 F.3d 1316, 1325-26 (Fed. Cir. 2002) (rejecting proposed claim construction as inconsistent with the prosecution history); *see also Linear Tech. Corp. v. Int’l Trade Comm’n*, 566 F.3d 1049, 1057-58 (Fed. Cir. 2009) (prosecution history evinced no clear and unmistakable disavowal of claim scope).

b. The Court’s Original URL Interpretation Is Inconsistent with Other Limitations in the Claim, Which Show that the “Alphanumeric String” is a Hostname, not a URL

In addition, the district court’s original URL interpretation makes no sense in light of the remaining claim limitations. “Alphanumeric string” appears four times in the body of claim 1 and the other claim limitations make clear that the “alphanumeric string” is not a URL, but rather the virtual server hostname.

In particular, the first limitation in claim 1 requires that the “service provider” (*i.e.*, the content delivery network provider) “establish an alternate domain name system . . . having authority to resolve the alphanumeric strings associated with the objects identified by the participating content providers.” As explained *supra* at 8, domain name servers (DNS) resolve hostnames, not URLs. Thus, “alphanumeric string” must be a hostname. Under the court’s construction, an “alphanumeric string that includes the URL” would not be resolved, because a URL includes more than just a hostname. (*Supra*, at 8, 29-30.)

Further, even if the district court’s “alphanumeric string that includes a URL” could be resolved, and it cannot, the DNS of the content delivery provider would not be able to resolve the URL “used to identify the object in the absence of the content delivery network” because that URL does not point to a content server in the content delivery network. Rather, that URL points to a server in the content

provider's network. Thus, the court's claim construction should be rejected as non-sensical on this ground alone.

Similarly, the next two references to "alphanumeric string" in the body of claim 1 clearly contemplate that "alphanumeric string" refers only to a resolvable hostname and, for this additional reason, the district court's "alphanumeric string" cannot include the original content provider's URL. More specifically, the claim requires that a given name server receive a DNS query "to the given object's associated alphanumeric string" and that the name server "resolve the alphanumeric string into an IP address." The name server can only resolve a hostname, not a full URL. (*Supra*, at 8.) Thus, for the name server to "resolve the alphanumeric string into an IP address," the alphanumeric string must be a hostname (and not a hostname plus something else).

Finally, the claim states that the "alphanumeric string is resolved without reference to a file name for the given object," lending even more support that "alphanumeric string" and the object's file name (which is included in the URL, *supra* at 8) are distinct.

Accordingly, because only hostnames can be resolved into IP addresses and the language in claim 1 clearly contemplates that the "alphanumeric string" must be resolved, the district court's construction must be reversed. *Cf. REXNORD CORP.*,

274 F.3d at 1342 (reversing because district court’s construction was inconsistent with later claim language).

c. The District Court’s Original URL Construction Creates a Requirement Found Nowhere (and in No Embodiment) in the Specification

The district court’s construction is based on a fundamental misunderstanding of the technology. The court did not appreciate the basic undisputed fact that DNS can only resolve hostnames, not URLs. Thus, the court’s construction, requiring the claimed alphanumeric string—which is a hostname—to include the original URL, creates a requirement found nowhere (and in no embodiment) in the specification. Indeed, as explained *supra* at 13-14, even in the preferred prepending embodiment, the specification discloses replacing the original hostname in the URL with the virtual server hostname. It is the virtual server hostname (not the object’s URL) that is resolved into an IP address.

d. Other Claims Confirm that the “Alphanumeric String” Is Not a URL

The district court’s construction of “alphanumeric string” in the ’645 patent is also inconsistent with the usage of that same term in claim 18 of the related ’413 patent. Because the patents share the same written description and “derive from the same parent application,” like terms “must [be] interpret[ed] consistently.” *NTP, Inc. v. Research In Motion, Ltd.*, 418 F.3d 1282, 1293 (Fed. Cir. 2005). The preamble of claim 18 states, in part, “wherein a given object of a participating

content provider is associated with a Uniform Resource Locator (URL) that includes, in addition to a filename, an alphanumeric string.” If, as the district court held, an “alphanumeric string” (of the ’645 patent claim 1) includes the original URL, this preamble makes little sense because it requires a URL to include the alphanumeric string, not the other way around. Either a URL includes an alphanumeric string (as positively recited in ’413 patent claim 8) or it does not; the reverse situation (the district court’s claim construction in the ’645 patent) cannot also be true at the same time. The district court’s construction cannot be reconciled against the wording of the ’413 patent claim 8.

e. The District Court’s Original URL Construction Is Premised on a Fundamental Misunderstanding of the Requirements of the Invention and Scope of the Claims

The district court’s incorrect claim construction seemingly stems from a fundamental misunderstanding of the requirements of the invention, as well as the scope of the claims. In its claim construction order, the district court emphasized that “[t]he URL of the object *is necessary* to the inventive global framework in order to retrieve the object from the content provider’s server if no copy exists on a ghost [*i.e.*, content] server.” (A69 (emphasis added).) The specification indicates otherwise.

In particular, the specification describes retrieving the missing content from either the content provider’s original server *or* another content server in the content

delivery network. (A257, 12:54-56 (“If, however, no copy of the data on the ghost exists, a copy is retrieved from the original server *or another ghost server.*”) (emphasis added).) In other words, the district court erred in concluding that the only way the content delivery network worked was by including the “URL used to identify the object in the absence of a content delivery network” to facilitate retrieving missing data from the content provider’s original server. *DSW, Inc. v. Shoe Pavilion, Inc.*, 537 F.3d 1342, 1348 (Fed. Cir. 2008) (refusing to confine the claims to the preferred embodiment to the exclusion of other embodiments.)

Furthermore, although the specification discloses that the content provider’s original URL may be helpful in the event an object is *not* available on its network (A257, 12:54-60), this is not the circumstance *actually claimed* in method claim 1, which is expressly directed to the circumstance in which the “object *is* available for delivery from the content server.” (*Supra*, at 18.)

For all of these reasons, the phrase “a given object of a participating content provider is associated with an alphanumeric string” should not be encumbered by requiring the alphanumeric string to include the original URL. The district court’s construction ignores the language of the claim, the parties’ stipulation on the meaning of “alphanumeric string,” the prosecution history, and the specification.

2. The Court Should Not Read “Selecting by the Alternative Domain Name System” into Claim 1 of the ‘645 Patent

The district court erred in interpreting “the given name server that receives the DNS query being close to the client local name server as determined by given location information” in method claim 1 of the ‘645 patent as requiring that the given name server be “selected by the alternative [DNS].” (A72-75.) First, claim 1 is directed to a method of content delivery, not to the structure of a content delivery system. This Court should thus reject the district court’s construction because it incorporates a structural limitation—DNS—into a method claim. *DSW, Inc.*, 537 F.3d at 1348 (rejecting claim construction that incorporated structure from preferred embodiment into a method claim).

Moreover, to compound the court’s error, claim 1 of the ‘645 patent says nothing about selecting in the first place. Rather, this claim limitation requires only that the content delivery network’s DNS server that receives a DNS query be close to the client (*i.e.*, the user’s) local name server. (*Supra*, at 17-18.) This “closeness” requirement provides no basis for reading in selecting, let alone selecting by DNS.

Limelight nonetheless argues that the claims should be limited to selecting by DNS because “the patent discloses only one way of determining that a ‘given name server’ will receive DNS queries.” (A015146.) Even if this were true, this Court “has repeatedly rejected the contention that depiction of a single [preferred]

embodiment in a patent necessarily limits the claims to that depicted scope.” *Agfa*, 451 F.3d at 1376. Without any indication in the specification or prosecution history “to suggest limiting the invention to [a] single embodiment, the broader language of the claims cannot carry [an] unexpressed and unintended . . . limitation.” *Id.* at 1377.

Moreover, under the district court’s construction, if the content delivery network’s DNS were the *only* structure that could select a name server, the content delivery network’s DNS would, by definition, be at least a two-level DNS (with the name server doing the selecting as one level and the name server being selected as the second level). But construing the claim to exclude a one-level DNS is inconsistent with the claim language, specification, and prosecution history.

First, claim 1 specifically recites that “the [content delivery network] service provider’s . . . domain name system ha[s] *one or more* DNS levels.” (*Supra*, at 19.) Thus, the district court’s construction, which necessitates a two or more level DNS, is improper because it is “contrary to the plain language of the claim.” *Agilent Techs., Inc. v. Affymetrix, Inc.*, 567 F.3d 1366, 1378 (Fed. Cir. 2009).

Second, although a preferred embodiment using a two-level DNS is explained, the specification expressly discloses that a one-level DNS may be used. (A285, 5:62-6:29; A287, 9:27-29, 64-65.) In a one-level DNS, the selection of the single name server could have been implemented using (as the patent states) “other

techniques,” including Anycasting, (A17429-30), or by a router that is part of, and under the control of, the content delivery network. *BJ Serv. Co. v. Halliburton Energy Serv., Inc.*, 338 F.3d 1368, 1372 (Fed. Cir. 2003) (holding the specification encompassed techniques known in the prior art). Accordingly, because the district court’s construction “excludes embodiments disclosed in the specification” (one-level DNS), the court’s construction should be reversed. *Oatey Co. v. IPS Corp.*, 514 F.3d 1271, 1276 (Fed. Cir. 2008).

In addition, nowhere in the ’645 patent prosecution history did Akamai limit itself to a two or more level DNS. On the contrary, the examiner allowed claim 1 after Akamai amended it to include the “has one or more DNS levels” language. (A16807-13, A16776-77.) There was no “clear and unmistakable” disavowal of claim scope. *See Liebel-Flarsheim Co. v. Medrad, Inc.*, 358 F.3d 898, 909 (Fed. Cir. 2004) (rejecting narrow construction of claims where, as here, patentee broadened claim language and did not limit claims to described embodiments during prosecution). Accordingly, the claim language, specification, and prosecution history contradict the district court’s limiting construction, and the construction should be reversed.

3. The Court Should Not Read “the Content Delivery Network’s Domain Name System Selects” into Claims 8, 18, and 20 of the ’413 Patent

For the same reasons, the district court erred in interpreting “selecting a given one of the name servers in the content delivery network” in method claims 8, 18, and 20 of the ’413 patent as “the content delivery network’s domain name system [‘DNS’] selects a particular name server.” (A77-80.) Nothing in the claim language, specification, or prosecution history supports the court’s requirement that a particular structure—the content delivery network’s DNS—perform the claim step. Thus, the district court improperly imported structure to perform a method step. *See DSW*, 537 F.3d at 1348.

Moreover, as with claim 1 of the ’645 patent, the court read into the claims the requirement that the close DNS server must be selected by the content delivery network’s DNS; in other words, a two or more level DNS. But, once again, claims 8, 18, and 20 expressly allow a one-tier DNS. (*Supra*, at 21.) And, while a two-level DNS is a preferred embodiment, the specification also explicitly discloses a one-level DNS. (A254, 6:2-4, 10:2.) In addition, the examiner allowed claims 8, 18, and 20 without objection after Akamai added the “has one or more DNS levels” language to the claim. (A16511, A16474, A16490.) Thus, the district court’s construction is inconsistent with the prosecution history and plain claim language.

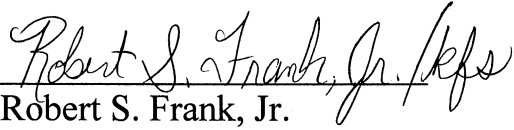
In an attempt to resolve this inconsistency, the district court created—out of whole cloth—an exemplary “one-level” DNS that purportedly satisfied the court’s construction of this term. (A79-80.) But this Court should reject the court’s concocted “one-level” DNS example. First, the district court’s example does not find support in the specification, the prosecution history, or any other evidence. Moreover, the district court’s example appears to conflate the function of the “local name server” with that of the CDN name server, which makes the example sound more like a two-level DNS (because there are multiple name servers) rather than a one-level DNS. In the end, however, the example is inapposite; it is not what the claims, the specification, and the prosecution history describe.

For all of these reasons, the limitation “selecting a given one of the name servers in the content delivery network” in method claims 8, 18, and 20 of the ’413 patent should not be construed as requiring that DNS selects a particular name server. Such a construction improperly reads in limitations from a preferred embodiment and ignores the language of the claim.

VI. CONCLUSION

For the above reasons, Akamai requests that this Court reverse the district court’s JMOL of no infringement of the ’703 patent, reverse the district court’s construction of the ’413 and ’645 patents, and remand for further proceedings under the proper construction of the ’413 and ’645 patents.

Respectfully submitted,




Robert S. Frank, Jr.
CHOATE, HALL & STEWART
LLP

Two International Place
Boston, MA 02110
(617) 248-5000

Attorneys for Plaintiff-Appellant
The Massachusetts Institute of
Technology

March 8, 2010



Donald R. Dunner
Kara F. Stoll
Elizabeth D. Ferrill
FINNEGAN, HENDERSON,
FARABOW, GARRETT & DUNNER,
LLP

901 New York Ave., N.W.
Washington, D.C. 20001-4413
(202) 408-4000

Attorneys for Plaintiff-Appellant
Akamai Technologies, Inc.

**IN THE
UNITED STATES COURT OF APPEALS
FOR THE FEDERAL CIRCUIT**

AKAMAI TECHNOLOGIES, INC.,
Plaintiff-Appellant,

and

THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY,
Plaintiff-Appellant,

v.

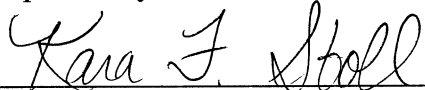
LIMELIGHT NETWORKS, INC.,
Defendant-Cross Appellant.

DECLARATION OF AUTHORITY

In accordance with Federal Circuit Rule 47.3(d) and pursuant to 28 U.S.C. § 1746, I, Kara F. Stoll, hereby declare under penalty of perjury that Robert S. Frank, Jr. has authorized me to sign the foregoing CORRECTED BRIEF OF PLAINTIFFS-APPELLANTS AKAMAI TECHNOLOGIES, INC. AND THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY on his behalf.

March 8, 2010

Respectfully submitted,



Kara F. Stoll

FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, L.L.P.
901 New York Avenue, NW
Washington, DC 20001
*Attorney for Plaintiff-Appellant
Akamai Technologies, Inc.*

ADDENDUM

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF MASSACHUSETTS

AKAMAI TECHNOLOGIES, INC., and)	
MASSACHUSETTS INSTITUTE OF)	
TECHNOLOGY,)	
)	
Plaintiffs,)	
)	Civil Action No. 06 CV 11109 RWZ
)	Civil Action No. 06 CV 11585 RWZ
vs.)	
)	
LIMELIGHT NETWORKS, INC.,)	
)	
Defendant.)	

RWZ ~~PROPOSED~~ FINAL JUDGMENT

Pursuant to Federal Rule of Civil Procedure 58, the Court hereby enters Final Judgment in this action as follows:

- (1) For the reasons stated in the Court's April 24, 2009 Memorandum and Order, Limelight does not infringe claims 19, 20, 21 and 34 of U.S. Patent No. 6,108,703 (the "703 patent").
- (2) For the reasons stated by the Court when granting Limelight's Motion for Summary Judgment, and in view of the Court's construction of the claims of U.S. Patent No. 6,553,413 (the "413 patent"), Limelight does not infringe claims 8-12, 18 and 19 of the '413 patent.
- (3) By reason of Akamai's stipulation, in view of the Court's construction of the claims of U.S. Patent No. 7,103,645 (the "645 patent"), that it cannot prove infringement of the '645 patent, Limelight does not infringe the '645 patent. When so stipulating, Akamai expressly

reserved all rights of appeal, including its right to challenge the Court's claim constructions and Limelight agreed that in the event that the Court's construction of the claims of the '645 patent were to be altered or overturned on appeal, Akamai would be free to reassert its claim of infringement of the '645 patent.

(4) By reason of the jury's verdict and the Court's rulings on Limelight Networks, Inc.'s Motion For Judgment As A Matter Of Law and its Motion For Judgment Of Obviousness and in view of the Court's construction of the claims of the '703 patent, claims 19, 20, 21 and 34 of the '703 patent are not invalid on any of the grounds stated in 35 U.S.C. §§101, 102, 103 or 112; and, accordingly, Limelight's Counterclaim seeking a declaration of invalidity is dismissed with prejudice.

(5) For the reasons stated in the Court's April 24, 2009 Memorandum and Order, Limelight's Counterclaim seeking a declaration of unenforceability of the '703 patent is dismissed with prejudice.

(6) Except as provided above, Limelight's Counterclaims regarding the '645 patent, the '413 patent and the claims of the '703 patent other than claims 19, 20, 21 and 34 of the '703 patent are dismissed without prejudice.

Because there are no further pending claims remaining in this action, judgment is entered in favor of Limelight, and this action is hereby dismissed. The dismissal is with prejudice, except as provided above in paragraph (6). ~~[Akamai proposal: Costs are awarded to neither party.] [Limelight proposal: Limelight shall recover of Akamai its cost of this action.]~~

RWZ

**AKAMAI TECHNOLOGIES, INC., and
MASSACHUSETTS INSTITUTE OF
TECHNOLOGY,**

LIMELIGHT NETWORKS, INC.,

By their attorneys:

By its attorneys:

/s/ Robert S. Frank, Jr.
Robert S. Frank, Jr. (BBO #177240)
Carlos Perez-Albuerna (BBO #640446)
G. Mark Edgerton (BBO #657593)
Choate, Hall & Stewart LLP
Two International Place
Boston, Massachusetts 02110
Telephone: (617) 248-5000
Facsimile: (617) 248-4000
medgarton@choate.com

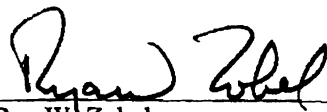
/s/ Alexander F. MacKinnon
Robert G. Krupka (pro hac vice)
Alexander F. MacKinnon (pro hac vice)
Kirkland & Ellis LLP
777 South Figueroa Street
Los Angeles, California 90017
Telephone: (213) 680-8400
Facsimile: (213) 680-8500

Gael Mahony (BBO #315180)
Thomas M. Johnston (BBO #644689)
Holland & Knight LLP
10 St. James Avenue, 11th Floor
Boston, Massachusetts, 02116
Telephone: (617) 523-2700
Facsimile: (617) 523-6850
thomas.johnston@hklaw.com

Dated: May 14, 2009

IT IS SO ORDERED.

Dated: May 22, 2009.



Hon. Rya W. Zobel
United States District Judge

UNITED STATES DISTRICT COURT
DISTRICT OF MASSACHUSETTS

CIVIL ACTION NO. 06-11109-RWZ

AKAMAI TECHNOLOGIES, INC., et al.

v.

LIMELIGHT NETWORKS, INC.

MEMORANDUM AND ORDER

April 24, 2009

ZOBEL, D.J.

Introduction	2
Background and Procedural History	2
Content Delivery Service Providers	2
Operation of the Parties' Content Delivery Services	4
Akamai's '703 Patent	5
The Farber '598 Patent	7
Sandpiper's Footprint System	8
The Digital Island and Speedera Patent Infringement Lawsuits	8
The Instant Lawsuit	10
Discussion	11
Inequitable Conduct During Prosecution of the '703 Patent	12
The Legal Standard for Inequitable Conduct	12
Materiality	13
Intent	13
Evidentiary Rulings	13
DX1209 - Sandpiper Web Site Images	13
PX1004A - Limelight's <u>Markman</u> Brief in the Level 3 Lawsuit ..	17
DX1228 - Leighton's Testimony in the Digital Island Trial	18
The Allegedly Withheld Information	19
Materiality	19
Intent	25
Limelight's Defenses of Laches and Equitable Estoppel	31
Legal Standard	31
Laches	31

Equitable Estoppel	31
Factual Background	32
Discussion	35
Laches	35
Equitable Estoppel	39
Unclean Hands	41
Limelight’s Motion for Reconsideration	42
<u>BMC Resources</u>	43
The Jury Instructions in the Instant Case	45
<u>Muniauction</u>	47
Vicarious Liability	48
The Significance of <u>Muniauction</u>	49
Conclusion	54

I. Introduction

Defendant Limelight Networks, Inc. (“Limelight”) seeks relief from a jury finding of patent infringement and an award of \$45.5 million in damages to plaintiffs Akamai Technologies, Inc., and the Massachusetts Institute of Technology (hereinafter the singular “Akamai”) based on the defenses of inequitable conduct, laches, equitable estoppel and unclean hands. It also moves for reconsideration of my earlier denial of its motion for judgment as a matter of law (“JMOL”). After careful consideration of the evidence presented at the November 2008 bench trial and the arguments in the parties’ papers, I hold that Limelight has failed to prove that Akamai’s conduct was so egregious that its patent rights should not be enforced. However, based on a clarification in the standard for a finding of joint infringement articulated by the Federal Circuit after the jury trial, Limelight is entitled to JMOL on the issue of infringement.

II. Background and Procedural History

A. Content Delivery Service Providers

Both Akamai and Limelight provide Internet content delivery services to their customers, content providers who maintain, inter alia, news and entertainment web sites that supply content to end users' web browsers. Without such a service, a content provider must distribute all its content from its own web servers. This requires the content provider to purchase and maintain servers and telecommunications bandwidth to handle the worst case load, and even then it may be overwhelmed by an unanticipated event, such as a major national disaster, or even a planned event which draws a large number of viewers, such as the Super Bowl. In addition, end users located far from the content provider's servers may experience poor performance due to Internet delays and congestion.

Content service providers Akamai and Limelight both maintain their own content delivery network ("CDN") consisting of hundreds or thousands of servers located in multiple locations across the United States and around the world. Once a content provider has contracted for content delivery services, a portion of its web content, typically large data files such as images, video and/or sound, is supplied by the CDN from a server located close to the end user rather than from the content provider's servers. Because a content delivery service aggregates the data demands of multiple content providers with differing peak usage patterns and serves that content from multiple servers in multiple locations, it is less likely to slow down or fail when an event creates a high demand for particular content. In addition, since content is supplied from a server close to the end user, that content is less likely to be affected by Internet congestion or breakdowns. The result is that the content provider can obtain the

capacity to service its end users under worst case demand conditions without having to pay for capacity that is idle much of the time.

B. Operation of the Parties' Content Delivery Services

A web page typically consists of text interspersed with various types of content such as images, video and sound, which are referred to as page objects. The web page, as well as the page objects, are stored on the content provider's web server. The page objects are normally not included within the web page itself, rather the web page consists of only the text on the page along with links (i.e. an Internet address) pointing to the page objects. Upon receiving a request for a web page, the content provider's web server returns the web page containing these links to the end user's web browser. The end user's web browser then uses these links to request each page object from the content provider's server until all the objects have been retrieved and the web page fully rendered.

To utilize Akamai's or Limelight's content delivery service, the content provider modifies its web pages so that the links, or URLs,¹ to the page objects point to the content service provider's servers, not its own.² The end user's browser fetches the initial web page from the content provider's server and then uses the returned links to request the other objects on the page from the content service provider's servers. The

¹ URL is short for Uniform Resource Locator, the address of a file or resource on the Internet.

² As discussed infra, Limelight's customers may instead modify the way in which the links to page objects are translated into server addresses so that they are fetched from the content service provider's servers rather than the content provider's servers.

content delivery service provider replicates these page objects on some or all of its servers and directs the end user's request for an object to an appropriate server. Thus, only the initial page is supplied by the content provider; the remaining page objects are served by the content delivery service provider's web servers. Because the initial page supplied by the content provider is relatively small compared to the size of the page objects, the majority of the information on the page is served by the content delivery service provider.

C. Akamai's '703 Patent³

On July 14, 1998, Akamai filed a provisional application for what would become the '703 patent. The disclosures in this patent form the basis for Akamai's FreeFlow content delivery service. The utility application was filed on May 19, 1999,⁴ and Akamai filed a petition to make special ("PTMS") with the Patent and Trademark Office ("PTO") in September 1999 to expedite the examination of its application. See 37 C.F.R. § 1.102 (1999). The PTO allowed the PTMS the following month, then, after a single office action in February 2000, allowed the application in April 2000. The patent issued on August 22, 2000.

The '703 patent claims systems and methods for replicating page objects among a distributed set of content delivery service provider servers and redirecting end user requests for those objects to a particular content server. The specification describes,

³ United States Patent No. 6,108,703.

⁴ For simplicity, in this opinion I refer to the application that resulted in the '703 patent as the '703 application. The actual number of the application was 09/314,863.

inter alia, a modification to the Internet's address lookup system, the Domain Name System ("DNS"), to accomplish this redirection. The DNS consists of a set of computers known as domain name servers which translate ("resolve") textual names of computers on the Internet ("hostnames") into numerical Internet addresses ("IP addresses"). The '703 patent describes the use of virtual hostnames which do not resolve to the IP address of a particular server; rather, in conjunction with the modified DNS described in the patent, the virtual hostname resolves to the IP address of a server containing the page object which is near the end user, has the page object and is not overloaded.⁵ Thus, a given virtual hostname could resolve to the IP address of servers at opposite ends of the country or around the world depending on the location of the end user, the location of the content and the delays in routing to the available servers.

The two independent claims asserted in the instant case, however, do not require either the use of virtual hostnames or a modified DNS. They do require modifying the links to at least some of the page objects, termed "tagging" in the patent, so that requests for those objects resolve to a content server in a domain other than the content provider's domain. In this context, a domain is an organization's unique name on the Internet, for example "akamai.com" or "limelightnetworks.com."⁶ In addition, one

⁵ This is a simplified explanation of the methods described in the patent, which actually involve multiple levels of domain name servers and sequential DNS requests to obtain the IP address of the page object server.

⁶ A domain name is unique in the sense that only one entity can own a particular domain name. See Virtual Works, Inc. v. Volkswagen of Am., Inc., 238 F.3d 264, 266 (4th Cir. 2001). However, a single entity can register and respond to multiple domain

of the asserted claims, claim 19, also requires the initial page to be served from the content provider's domain.

The tagging described in the '703 patent adds or "prepends" the hostname of the content service provider's server to the original URL used to retrieve the page object from the content provider's server. Including the original URL in the tagged link provides the content service provider's server with the information necessary to retrieve the object from the content provider's server if it is not already stored ("cached") locally. Once the content service provider's server fetches an object from the content provider's server, it caches it locally so it is available for subsequent requests for that object.

D. The Farber '598 Patent

On February 10, 1998, Sandpiper Networks, Inc. ("Sandpiper") filed a United States patent application for a content delivery system. Just under a year later, Sandpiper filed an international patent application with the World Intellectual Property Organization under the Patent Cooperation Treaty ("PCT Application") containing the same specification as its earlier United States application. The PCT Application was published on August 12, 1999. Although Sandpiper's United States application was filed before Akamai's provisional application, Sandpiper's patent, United States Patent No. 6,185,598 (the '598 patent"), did not issue until February 6, 2001, after the '703 patent issued.⁷

names.

⁷ The '598 patent is alternately known as the "Farber patent" after one of its inventors, David J. Farber ("Farber").

The '598 patent claims systems and methods for a content delivery system similar to those claimed by Akamai. However, rather than use DNS to redirect an end user to an appropriate content server (termed a "repeater" in the '598 patent), the patent adds a component called a reflector to the content provider's server to redirect requests for page objects (termed "resources") to be served from a repeater. In one embodiment, the reflector redirects a resource by rewriting the link to the resource to create a new link which designates the repeater instead of the content provider's server. The rewritten link consists of an identifier for the repeater followed by the original URL to the resource. This identifier can be the hostname of the repeater or the actual IP address of the repeater. The end user's browser then uses the rewritten link to request the resource from the repeater. If the resource is available locally on the repeater, it is returned to the end user. If the resource is not already on the repeater, the repeater uses the original URL portion of the rewritten link to fetch the resource from the content provider's server, returns it to the end user and caches it locally for future requests.

E. Sandpiper's Footprint System

Sandpiper introduced its content delivery service, named Footprint, in September 1998. This initial system (hereinafter "Footprint 1.0") required reflector software running on the content provider's server to intercept end user requests for content and redirect them to Sandpiper's content servers. In spring of 1999, Sandpiper modified its system to use DNS, rather than the redirector software, to redirect end user requests for content. Akamai believed this new system ("Footprint 2.0") was a direct

copy of its then newly deployed content delivery system based on the '703 patent.

F. The Digital Island and Speedera Patent Infringement Lawsuits

In September 2000, Akamai filed suit against Digital Island, Inc. ("Digital Island") alleging, inter alia, infringement of the '703 patent by Digital Island's (nee Sandpiper's)⁸ Footprint 2.0 system. On December 21, 2001, a jury found that the Footprint 2.0 system infringed claims 1, 3, 5, 9, 17, 18 and 22 of the '703 patent, but that claims 17, 18 and 22 were invalid as anticipated by Digital Island's '598 patent or obvious in view of the '598 patent and a product by Cisco Systems, Inc., Distributed Director, disclosed in United States Patent No. 6,178,160. (See Civ. A. No. 00-11851-RWZ (D. Mass.), Docket # 232.) The district court granted a permanent injunction based on the jury verdict.

Digital Island appealed to the Court of Appeals for the Federal Circuit, arguing that claim 9 was not infringed and that claims 1, 3, 5, and 9 were invalid for anticipation and/or obviousness. In its opinion,⁹ the court described the history of Digital Island's

⁸ Digital Island acquired the Footprint system when it merged with Sandpiper in 1999. Digital Island was acquired by Cable and Wireless Plc in 2001, which, in turn, was bought by Savvis Communications. Level 3 Communications purchased Savvis' CDN business in January 2007 and acquired rights to intellectual property developed by Sandpiper. See Level 3 Commc'ns, LLC v. Limelight Networks, Inc., Civ. A. No. 2:07cv589, 2008 WL 5188143, at *1 (E.D. Va. Dec. 10, 2008). This complicated genealogy is useful in understanding the evidentiary dispute concerning the admission of images from Sandpiper's 1999 web site, discussed infra Part III.A.2.a.

⁹ By the time of the appeal, Digital Island had been acquired by Cable & Wireless Plc. Therefore, the Federal Circuit decision refers to Digital Island as "C & W."

'598 patent and its Footprint systems:¹⁰

The '598 patent is directed to similar systems and methods [as the '703 patent] for increasing the accessibility of web pages on the Internet. The '598 patent was filed on February 10, 1998, and issued on February 6, 2001. Thus the '598 patent is prior art to the '703 patent pursuant to 35 U.S.C. § 102(e). C & W marketed and sold products embodying the '598 patent under the name "Footprint." The relevant difference between the disclosure of the '598 patent and Akamai's preferred embodiment disclosed in the '703 patent is the location of the load balancing software. Akamai's preferred embodiment has the load balancing software installed at the DNS servers, while the '598 patent discloses installation of the load balancing software at the content provider, or origin, servers. The '598 patent does not disclose or fairly suggest that the load balancing software can be placed at the DNS servers. It is now understood that placement of the software at the DNS servers allows for load balancing during the resolving process, resulting in a more efficient system for accessing the proper information from the two server networks. Indeed, C & W later created a new product, "Footprint 2.0," the systems subject to the permanent injunction, in which the load balancing software was installed at the DNS servers as opposed to the content provider servers. Footprint 2.0 replaced C & W's Footprint product.

Akamai Techs. Inc. v. Cable & Wireless Internet Servs., Inc., 344 F.3d 1186, 1190-91

(Fed. Cir. 2003) (footnote omitted). The Federal Circuit held that claims 1 and 3 of the '703 patent were invalid because they did not require that the load balancing software be placed at the DNS servers and thus were anticipated by the '598 patent. Id. at 1194-95.

In February 2002, Akamai sued another competitor, Speedera Networks, Inc. ("Speedera"), alleging that Speedera's Universal Delivery Network infringed several Akamai patents, including the '703 patent. (See Civ. A. No. 02-10188-RWZ (D. Mass), Docket # 1.) The case settled in mid-2005 after the district court's order on claim

¹⁰ The Federal Circuit decision uses the term "load balancing" for the process of redirecting a request for a page object to an optimal server.

construction but before any other decision on the merits.

G. The Instant Lawsuit

Limelight began offering its content delivery services in late 2001. In the spring of 2004, Akamai and Limelight began a series of discussions concerning the possible acquisition of Limelight by Akamai. Ultimately, Akamai choose not to proceed with the proposed deal, and the discussions terminated in the fall of 2004.

In early 2006, Akamai and Limelight again resumed acquisition discussions. However, on June 22, 2006, Limelight informed Akamai that it had found alternative funding and was no longer interested in being acquired. On the following day, Akamai filed the instant lawsuit against Limelight alleging patent infringement.

On February 28, 2008, a jury found that Limelight infringed claims 19-21 and claim 34 of the '703 Patent and that none of the infringed claims were invalid due to anticipation, obviousness, indefiniteness, lack of enablement or written description. The jury awarded Akamai damages of \$41.5M based on lost profits and reasonable royalty from April 2005 through December 31, 2007, plus prejudgment interest, along with price erosion damages in the amount of \$4M. (See Jury Verdict (Docket # 287).)

The parties filed a flurry of post-trial motions. Akamai renewed its motion for partial summary judgment on Limelight's equitable defenses of laches, equitable estoppel and unclean hands,¹¹ moved for summary judgment on Limelight's defense of inequitable conduct and sought a permanent injunction. Limelight moved for a new trial

¹¹ Limelight had also asserted a defense of patent misuse, which was resolved by the allowance of Akamai's motion for partial summary judgment on this defense. (See Hr'g Tr. 5, Feb. 5, 2008 (Docket # 240).)

and for a judgment of obviousness and renewed its motion for JMOL on grounds of noninfringement and invalidity. After briefing and a hearing, I denied all six motions, including Akamai's motion for a permanent injunction, which was premature in light of the scheduled bench trial. Shortly thereafter, Limelight moved for reconsideration on the court's denial of its motion for JMOL, citing a newly announced Federal Circuit decision. (See Docket # 377.)

After a delay requested by the parties, the court held a three-day bench trial in November 2008 on Limelight's remaining equitable defenses.

III. Discussion

A. Inequitable Conduct During Prosecution of the '703 Patent

Limelight asserts that the applicants for the '703 patent and/or their attorney: (1) had a duty to, but intentionally failed to, investigate the prior art of their key competitor Sandpiper; and (2) intentionally withheld material from the PTO concerning Sandpiper's Footprint 1.0 CDN. It claims that these actions breached the duty of candor and good faith owed to the PTO and thereby constitute inequitable conduct rendering the patent unenforceable. Although I find that the information that was not disclosed was material, Limelight has failed to prove the requisite intent necessary to find inequitable conduct.

1. The Legal Standard for Inequitable Conduct

Applicants have a duty to prosecute patent applications with candor, good faith, and honesty. Duro-Last, Inc. v. Custom Seal, Inc., 321 F.3d 1098, 1099 (Fed. Cir. 2003); see also 37 C.F.R. § 1.56. A breach of this duty constitutes inequitable conduct and renders a patent unenforceable. See Impax Labs., Inc. v. Aventis Pharms., Inc.,

468 F.3d 1366, 1374 (Fed. Cir. 2006). To prove inequitable conduct, “the alleged infringer must provide clear and convincing evidence of (1) affirmative misrepresentations of a material fact, failure to disclose material information, or submission of false material information and (2) an intent to deceive.” Id.

The analysis of inequitable conduct is a two-step process. First, the court must determine whether both the materiality of the information and the intent to deceive have been established. Bristol-Myers Squibb Co. v. Rhone-Poulenc Rorer, Inc., 326 F.3d 1226, 1234 (Fed. Cir. 2003). If so, it must then “weigh them to determine whether the equities warrant a conclusion that inequitable conduct occurred.” Id. (quoting Molins PLC v. Textron, Inc., 48 F.3d 1172, 1178 (Fed. Cir. 1995)). In balancing, a greater showing of one factor can compensate for a lesser showing of the other. Id.

a. Materiality

Information is material if a reasonable examiner would consider it important in deciding whether to allow the patent application. Digital Control, Inc. v. Charles Mach. Works, 437 F.3d 1309, 1314 (Fed. Cir. 2006) (citations omitted). Information can be material even though disclosure of it would not render the invention unpatentable. Id. at 1318. However, prior art that is merely cumulative or less pertinent than information considered by the examiner is not considered material in analyzing a claim of inequitable conduct. Id. at 1319.

b. Intent

The evidence must also support a finding that the material information was withheld with an intent to deceive or mislead the PTO. Abbott Labs. v. Sandoz, Inc.,

544 F.3d 1341, 1356 (Fed. Cir. 2008). “[T]he involved conduct, viewed in light of all the evidence, including evidence of good faith, must indicate sufficient culpability to require a finding of intent to deceive.” Impax Labs., 468 F.3d at 1375 (citation omitted). Where the alleged conduct is the nondisclosure of information, there must be clear and convincing evidence that the applicant made a deliberate decision to withhold the information from the PTO. Molins, 48 F.3d at 1181.

2. Evidentiary Rulings

a. DX1209 - Sandpiper Web Site Images

As discussed infra, one of the factual issues relevant to the allegations of inequitable conduct is what Akamai knew about Sandpiper’s Footprint system and when it knew it. Daniel Lewin (“Lewin”), one of the inventors of the ’703 patent, admitted studying Sandpiper’s web site frequently during the pendency of the ’703 application. At the bench trial, Limelight moved to admit printouts of Sandpiper’s web site from January and May 1999 (DX1209),¹² which included several press releases describing the Footprint system, to show the extent of Akamai’s knowledge at the time. I admitted the exhibit de bene, and the parties have submitted briefs on its admissibility.

In two separate written lists of objections to Limelight’s exhibit list, provided to Limelight before the November bench trial, Akamai objected to DX1209 on grounds of

¹² “DX_____” refers to Limelight’s Trial Exhibit DX_____ offered at the November 2008 bench trial. Similarly, “PX_____” refers to Akamai’s Trial Exhibit PX_____ from the bench trial.

relevance and hearsay.¹³ (See Docket # 395, Exs. 1, 2.) It argues that the press releases are not relevant because there is no evidence that the inventors of the '703 patent ever saw them. However, Lewin testified that in August or September of 1998 a friend sent him the URL for Sandpiper's web site and suggested that he take a look at what it was doing. He further testified that he and others at Akamai

read the Web site in great detail, which had an overview of what they were doing, what the business was, who the people were that were involved in the company, overall descriptions of what the benefits to a customer were. Their network infrastructure was on the Web site.

(Lewin Dep. 80:5-11, Jan. 18, 2001.) At that time, Lewin discussed Sandpiper as a potential competitor with Tom Leighton ("Leighton"), the other inventor of the '703 patent (collectively with Lewin, the "Inventors"), as well as with other Akamai personnel. He continued to monitor Sandpiper's web site, agreeing that he "looked at the Sandpiper Web page almost daily." (*Id.* at 94:5-7.) Leighton testified that he first learned about Sandpiper in the fall of 1998 "as a consequence of the September 28th press release." (Trial Tr. Day 2, 109:18-22, Nov. 13, 2008.) He also explained that Akamai deduced the operation of Sandpiper's system by looking at how it delivered Sandpiper's own web site.

Several of the press releases in the proffered exhibit include a description of Sandpiper as "a provider of adaptive content distribution services" and describe Sandpiper's technology as "optimiz[ing] network performance by migrating content based on network conditions, content popularity and traffic loads." (*e.g.*, DX1209 at L3-

¹³ Akamai also generally reserved its right to object to any exhibit for lack of foundation.

AKLL-000007.) In addition, a September 28, 1998, press release describes the Footprint “distribution network” as “patent-pending.” (Id. at L3-AKLL-000010.) The evidence proffered supports the conclusion that both press releases were made available on the web site shortly after release. (Compare id. at L3-AKLL-000004 (list of Sandpiper press releases on web site as of Jan. 13, 1999), with id. at L3-AKLL-000025 (list of Sandpiper press releases on web site as of May 2, 1999).)

Lewin’s and Leighton’s testimony, that they paid close attention to Sandpiper’s web site beginning in fall of 1998, permits the inference, which I draw, that one or both of them read the press releases contained therein. Therefore, exhibit DX1209 is relevant to the issue of what the Inventors knew about Sandpiper and its Footprint system during the prosecution of the ’703 patent. See Fed. R. Evid 401.

Akamai’s second objection, that the Sandpiper web site printouts are hearsay, fails because they are not offered for the truth of the information contained within. Rather, Limelight seeks to show the Inventors’ knowledge that in late 1998 and early 1999 Sandpiper claimed to have a content delivery system that took content popularity, network loads and congestion into account in delivering content, and that it had submitted a patent application for some part of that system. Whether these claims were actually true is immaterial to the issue of inequitable conduct by Akamai. Therefore, both of Akamai’s pretrial objections to the exhibits are overruled.

On the first day of the bench trial, however, Akamai raised yet another objection, namely, that “the testimony that authenticates these press releases is itself hearsay.” (Trial. Tr. Day 1, 52, Nov. 12, 2008.) Before the trial, in September 2008, Limelight, to

authenticate DX1209, had provided excerpts from a Rule 30(b)(6) deposition of Andrew Swart ("Swart"), a former Sandpiper employee, taken in a lawsuit between Limelight and Level 3 Communications, LLC ("Level 3"). Akamai objected because it was not a party to that lawsuit and therefore did not have an opportunity to cross-examine Swart. In response, Limelight attempted to depose Swart again, this time with Akamai present. Level 3, however, provided not Swart, but Joseph Lawrence ("Lawrence"), a Level 3 employee, as its 30(b)(6) witness. (See Docket # 395, Exs. 6, 7.) Lawrence authenticated the web site printouts and the press releases based on discussions with Swart and another former Sandpiper employee, Will Crowder. Akamai now objects to Lawrence's authentication because it is based on hearsay, not personal knowledge, and Level 3 is not a party-opponent.

While Akamai did object to Lawrence's deposition testimony as hearsay two weeks before the trial, it did not put Limelight on notice that it objected to the authenticity of DX1209. It therefore waived that objection. See Fed. R. Civ. P. 26(a)(3)(B) (waiving any objection to exhibits disclosed prior to trial other than relevance, prejudice or confusion). DX1209, therefore, is admitted for the purpose of showing what information was on the Sandpiper web site in late 1998 and early 1999 when Lewin testified he was viewing the site.¹⁴

¹⁴ When asked directly by the court whether it was denying that the printouts were what was on the web site, Akamai's counsel demurred, admitting that the Sandpiper web site existed, and that Lewin viewed the web site and learned of at least one press release, but insisting several times that there was "no admissible evidence" that DX1209 was what was on the web site. (Trial Tr. Day 1, 52:25, Nov. 12, 2008.) Because Akamai was present to cross-examine Lawrence, whose testimony is not in conflict with that given by Swart earlier, I deem it to have sufficient circumstantial

b. PX1004A - Limelight's Markman Brief in the Level 3 Lawsuit

Akamai seeks to introduce a Markman brief that Limelight filed in an unrelated case as an admission by Limelight that the specification of the Farber '598 patent describes a system significantly different from that described by the '703 patent. In particular, it argues that Limelight's adoption of portions of an email memorandum Farber sent during the pendency of the '703 application establishes the authenticity and admissibility of that memorandum. Limelight objected to the admission of both the brief and the memorandum, and I invited Akamai to provide additional briefing on their admissibility post-trial. (See Trial Tr. Day 2, 53:23-25, Nov. 13, 2008.) Akamai has provided a paragraph of explanation why it believes these documents are admissible, citing only to Fed. R. Evid. 801(d)(2) (admissions by a party opponent are not hearsay). (See PX1004A, flysheet.)

The objection to this exhibit is sustained both because the briefing is inadequate to resolve the legal issue and because this evidence is, in any event, either not relevant to the issue of inequitable conduct or cumulative of other testimony.

c. DX1228 - Leighton's Testimony in the Digital Island Trial

At the November 2008 bench trial, Leighton testified that Akamai's investigation into the operation of the Footprint 1.0 system showed that it used IP addresses, not hostnames, to redirect requests for page objects from the content provider's server to the content service provider's servers. He denied both knowing that the Footprint 1.0

guarantees of trustworthiness to authenticate DX1209 under Fed. R. Evid. 807 as well.

system could alternately use hostnames to redirect object requests or having seen the Sandpiper PCT application, which described the use of hostnames as well as IP addresses, prior to the issuance of the '703 patent. On cross-examination, Limelight attempted to impeach Leighton's direct testimony using Farber's deposition and testimony from the 2001 Digital Island trial describing the Footprint 1.0 system. I allowed the questioning de bene over Akamai's objection. I now overrule that objection.¹⁵

Limelight post trial seeks to introduce a portion of Leighton's testimony from the Digital Island lawsuit to attempt to show that he learned additional information about Footprint 1.0 during the pendency of the '703 application. Akamai objects because by not identifying and offering this testimony at the bench trial, Limelight denied it the opportunity to counter-designate relevant testimony and denied Leighton the opportunity to explain his prior testimony. For reasons similar to those discussed supra Part III.A.2.a, overruling Akamai's objection to the Sandpiper web site printouts, its objection to this evidence is sustained.

3. The Allegedly Withheld Information

Limelight alleges that, during the pendency of the '703 application, Akamai was aware of Sandpiper and its Footprint 1.0 system, which, Limelight argues, is an embodiment of the '598 Patent, but failed to disclose this information to the PTO.

¹⁵ However, while this cross-examination did establish that Leighton learned that Footprint 1.0 could use hostnames from Farber's testimony at the Digital Island trial in 2000, this fact is not in conflict with his direct testimony that the Footprint 1.0 system, as Akamai observed it in 1998 and 1999, did not use hostnames.

Limelight further asserts that Akamai was aware that Sandpiper had filed a patent covering the Footprint 1.0 technology and, therefore, had a duty report this information to the PTO or at least conduct an investigation into Sandpiper as part of its PTMS.

a. Materiality

Limelight argues that the '598 patent, and thus the Footprint 1.0 system, was material because the Federal Circuit later held that claims 1 and 3 of the '703 patent were invalid as anticipated by the '598 patent. Akamai responds that the disclosures in the '598 patent were not material because they were cumulative of other information Akamai provided to the PTO. In addition, to the extent the Footprint 1.0 system was an embodiment of that patent, Akamai argues that it did not include the features that the Federal Circuit relied on to find anticipation. In particular, it asserts that the Footprint 1.0 system prepended IP addresses, not hostnames, and therefore did not require the use of DNS, both limitations of the claims ruled invalid as anticipated. See Akamai Techs., 344 F.3d at 1192 (a prior art reference is anticipating if it “discloses each and every limitation of the claim expressly or inherently”).

According to the testimony and evidence, the Inventors understood that Sandpiper's Footprint 1.0 system was a framework for delivering web content. As such it attempted to solve the same general problem Akamai was addressing, namely, reducing the bottleneck created by serving all content from a single server while avoiding the limitations of then existing mirroring content delivery systems. Mirroring systems, which duplicate content on multiple servers, do not scale well because it becomes increasingly difficult to keep the content of all the systems synchronized as

the number of servers increases. Footprint 1.0, like the system described in Akamai's '703 patent, implemented a content delivery system that split delivery of the contents of a web page by serving the initial page from the content provider's server, but some or all of the page objects from a server closer to the end user. Page splitting addresses the synchronization problem because the original web page need be kept up-to-date only on a single server, that of the content provider. By always fetching the initial page from the content provider's server, a page-splitting system guarantees that the user receives the current content. In addition, both the Footprint 1.0 system and the disclosure in the '703 application replaced the URL to each page object with a rewritten URL that consisted of an identifier for an optimal content server followed by the original URL for the object. This preserves the original Internet address of the object in the new URL, enabling the content server to fetch the object from the content provider's server if the object was not already cached locally. Therefore, unlike mirroring systems, these systems did not require that all content be copied to the content servers prior to the first end user request. This design ensures that any page objects updated by the content provider are migrated to the content service provider's servers when requested by end users.

The primary differences between the Footprint 1.0 system and that described in the '703 patent is that Footprint 1.0 prepended an IP address not a hostname,¹⁶ and the

¹⁶ Limelight disputes Akamai's contention that the Footprint 1.0 system only prepended IP addresses, not hostnames. As discussed infra Part III.A.3.b., I find credible the testimony and evidence Akamai offered that the Inventors only observed IP addresses prepended in their investigation into the operation of the Footprint 1.0 system.

determination as to which content server to prepend was made at the content provider's server by the reflector, not by the DNS system described in Akamai's application.

Nevertheless, Footprint 1.0 attempted to solve the same synchronization problem that the '703 sought to solve in much the same way. While much of the '703 patent focuses on the use of a virtual hostname and DNS to select an optimal content server, several of the claims of Akamai's patent differ from the Footprint 1.0 system only in the use of a normal hostname rather than an IP address prepended to the original URL or by requiring the resolution of a hostname. (See, e.g., '703 patent claim 14; id. claim 23.) In addition, the Inventors viewed the Footprint 1.0 system as direct competition to their content delivery system, which was based on the invention claimed in the '703 patent, and they expended significant effort to investigate and reverse engineer the operation of this competing system. Given the similarity in the problem each system attempted to solve and in the operation of the respective systems, "a reasonable examiner would consider [Footprint 1.0] important in deciding whether to allow [Akamai's] application to issue as a patent." Digital Control, 437 F.3d at 1315. Thus, Sandpiper's Footprint 1.0 system was material information that should have been disclosed to the PTO during the pendency of Akamai's application, unless it was cumulative of other information provided.

At the bench trial, Akamai's expert, Kevin Jeffay ("Jeffay"), opined that the '598 patent, and thus its embodiment in the Footprint 1.0 system, was merely cumulative of other information Akamai included in its submissions to the PTO. In particular, he asserted that the Kriegsman and Kenner patents, which Akamai did provide to the PTO,

disclosed all of the relevant features of the '598 patent. The Kriegsman patent, United States Patent No. 5,991,809 (the "'809 patent") (PX1016), discloses a system consisting of a primary server and one or more secondary servers which mirror static files copied from the primary server.¹⁷ When the primary server receives a request for data, it returns all the dynamic data files requested, then determines an optimum secondary server to serve each static data file. If a requested dynamic data file contains links to static data, software on the primary server replaces those links with new links pointing to the static data on the optimum secondary server. (See '809 patent, col.9 ll.61-65.) However, unlike the disclosure in the '598 patent, the Kriegsman patent does not include the link to the original object in the rewritten link, and therefore the secondary server cannot use the URL of the requested object to request the object from the primary server if it is not stored locally. Rather, the patent describes a mirrored system in which the primary server copies static data to the secondary servers prior to redirecting users' requests for static content to a secondary server. (See id. col.6 l.67-col.7 l.9.)

The Kenner patent, United States Patent No. 6,003,030 (the "'030 patent") (PX1015), discloses a content delivery system primarily designed to serve video objects and which utilizes software installed on the end user's computer ("Client Software") to route requests for content to an optimal content server. Content to be

¹⁷ The Kriegsman patent defines "static files" as those which are repeatedly transferred without changes, while files which may be modified for each transfer are termed "dynamic files." (See '809 patent, col.1 ll.8-12.) "Static data" includes images, video, computer programs and other non-text data. (Id. col.2 ll.9-11.)

served is copied from the original web site to the content servers prior to end user requests for content. When a user makes a request for content which is marked as also available on the content servers, the Client Software attempts to download the content from the optimal server by constructing a URL pointing to that content on the content servers. (See '030 patent, col.15 ll.4-10.) If the content is not available, the Client Software tries the next available content server. If the content is not found on any of the content servers, the Client Software retrieves it from the original content provider's site. (See id. col.15 ll.18-24.)

According to Jeffay, Kriegsmann discloses all aspects of the '598 patent except for prepending the URL pointing to the original object on the content provider's server to the rewritten URL and retrieving the object from the content provider's server when it is not found on the content server. However, he asserts that these features are disclosed in the Kenner patent. Therefore, the disclosures of the '598 patent are, in his opinion, cumulative of those obtained by combining Kriegsmann and Kenner. But see Molins, 48 F.3d at 1180 (noting that a withheld reference can be material when no single piece of cited prior art taught the combination present in the reference).

Jeffay's conclusion assumes that the URL created by the '030 patent's Client Software to retrieve the object from a content server includes the original URL pointing to the object on the content provider's server. The patent describes passing information to the end user, and thus to the Client Software, about whether content is available on a content server within a text construct known as an "EMBED" statement. The patent's specification describes how it creates the URL to retrieve the object from a

content server, termed a "Smart Mirror:"

If the [page object] does not exist on the local computer, the player creates a new URL (step 78) in the following form: "http://", plus the IP address of the selected Smart Mirror site stored in the configuration file, plus the path name to mirror files (e.g. "/pub/mirror/"), plus the name of the content provider taken from the "SM" parameter in the EMBED statement, plus the [page object] filename taken from the EMBED statement.

('030 patent col.15 ll.4-10 (emphasis added).) If the content is not available on a content server, the Client Software retrieves it "from the original content provider's site as specified directly by the EMBED statement." (Id. col.15 ll.21-24.) Jeffay opines that "the name of the content provider" refers to the hostname of the original server. However, this is not explicitly stated in the '030 patent and hostnames are not necessarily identical to the name of the content provider, for example, content provider Volkswagen uses the hostname www.vw.com. In addition, both the '598 patent and the '703 patent explicitly describe prepending the hostname of the content service provider's server to the origin server and path to the original object and the Footprint 1.0 system prepended the IP address of a content server to the origin server and path to the original object. (See '598 patent col.8 ll.26-35; '703 patent col.8 ll.4-12; PX1009, 9-10.) The '030 patent, however, only describes appending the name of the content provider and the filename of the object, not the path to the original object. In addition, because the '030 patent does not require that a content server be able to retrieve objects which it does not already have mirrored, there is no reason to include the URL to the original object in the constructed URL. Therefore, the evidence is insufficient to conclude that the Kenner '030 patent discloses prepending the identifier of a content server to the URL of the original object as is disclosed in the '598 patent and as was

observed by the Inventors in examining the operation of Footprint 1.0. Thus, the '598 patent and the Footprint 1.0 system are not cumulative of the combination of the Kriegsman and Kenner patents.

In addition, neither Kenner nor Kriegsman disclose the technique described in the '598 patent in which content servers use the URL of the original object to retrieve the object if it is not already cached on the content server. In Kenner, it is the Client Software running on the end user's computer that fetches the object if it is not found on any content server. (See '030 patent col.6 ll.36-37, col.15 ll.21-24.) Kriegsman is a mirrored system with data copied to the secondary (content) servers by the primary server before any requests for content are made; therefore, the patent does not contemplate a need to fetch an object in response to a user request to a secondary server. (See '809 patent col.3 ll.45-48, col.6 l.67-col.7 l.5.)

For all of these reasons, I find that the '598 patent was material because it contained prior art which was later found to have anticipated several claims of the '703 patent and was not cumulative of other information provided to the PTO during the prosecution of the '703 patent. See Fox Indus., Inc. v. Structural Pres. Sys., Inc., 922 F.2d 801, 804 (Fed. Cir. 1990). I also find that the Footprint 1.0 system was not cumulative of other references provided to the PTO and thus was material for the reasons described supra.

b. Intent

Limelight argues that intent by the Inventors to withhold information from the PTO can be inferred from their careful examination of the Sandpiper web site, their

investigation into the operation of the Footprint 1.0 system, the temporal proximity of Akamai's PTMS following the publication of Sandpiper's PCT application and the limited search Akamai conducted as part of that PTMS.

Leighton testified that because the Footprint 1.0 system used IP addresses, not hostnames, it did not occur to him that it was relevant to Akamai's efforts to patent its DNS-based system. He stated that he viewed the Sandpiper system as having many of the same deficiencies as the mirroring systems that Akamai was trying to correct, in particular, the bottleneck of having all redirection determinations made at the content provider's server. Footprint 1.0 prepended IP addresses, so there was no way for Sandpiper to use DNS to distribute requests for page objects to other servers. Because he saw the use of DNS as fundamental to the Akamai invention, he did not consider disclosing Footprint 1.0 to the PTO. In addition, he testified that in late 1998 and early 1999 he was preoccupied with efforts to launch Akamai's own content delivery service.

When Sandpiper released Footprint 2.0 after Akamai launched its service, Leighton believed Sandpiper had copied the Akamai system in response to the problems created by the centralization inherent in the architecture of the Footprint 1.0 system. Therefore, he did not consider Footprint 2.0 relevant to the disclosures in the '703 patent application; rather, he asked his patent counsel to investigate the possibility of an infringement claim against Sandpiper when Akamai's patent finally issued. Finally, Leighton stated that he was unaware of Sandpiper's PCT application or its United States patent application until after the '703 patent issued.

Limelight disputes Leighton's credibility. It points out that Sandpiper's web site, which Akamai employees studied in detail, described Sandpiper's technology as "patent-pending" and cites an Akamai competitive overview (PX1009) which, it asserts, shows that Akamai was aware that Footprint 1.0 used hostnames as well as IP addresses.

While I find that Akamai employees, including Lewin, carefully examined Sandpiper's web site and considered Sandpiper a serious competitor in the content distribution arena, there is no evidence that anyone who saw the Sandpiper press releases recognized the "patent-pending" statement as relevant to Akamai's own patent application. As to Akamai's competitive overview, dated September 30, 1999; it only describes Footprint 1.0 as prepending IP addresses. (See PX1009, 9 ("[Sandpiper 1.0] redirect[s] inbound user requests . . . using proprietary routing tables that provide IP addresses"); id. at 10 ("URLs that used [Footprint 1.0] for content delivery typically appear as an IP address followed by a standard URL.")) The document also notes that "nearly all customers use Footprint 2.0," the copy of Akamai's system. (Id. at 9.) In addition, Lewin testified in an earlier deposition that Footprint 1.0 only prepended IP addresses. There is no evidence that the Inventors were aware that Sandpiper described prepending hostnames as a separate embodiment of its invention until they read it in the specification for the PCT application or the '598 patent. I find credible Leighton's testimony that the Inventors were not aware of Sandpiper's PCT application until after the '703 patent issued. The coincidence of Akamai's decision to file its PTMS six weeks after the PCT application was published is not enough to persuade me

otherwise.

Given the central importance of DNS to the disclosures of the '703 patent and the operation of Footprint 1.0 which only prepended IP addresses, together with the pressures of creating a startup company, I am persuaded that Leighton did not deliberately fail to disclose Footprint 1.0 to the PTO. There is also no evidence that Lewin intentionally failed to disclose Footprint 1.0, only that he studied Sandpiper's web site and therefore may have read Sandpiper's claim that its technology was patent-pending.

Akamai's patent attorney, David Judson ("Judson"), testified that he first became aware of Sandpiper in summer of 1999 because Akamai believed that Footprint 2.0 was a copy of its system. As a result, his investigations into Sandpiper looked to a possible infringement lawsuit, not possible prior art. Although he became aware of the Footprint 1.0 system as a result of his investigation, I credit his testimony that his focus was on infringement by the Footprint 2.0 system.

With respect to the prosecution of the '703 patent, Judson testified that he made two searches in preparation for the PTMS.¹⁸ First, he conducted a keyword search on a patent database, possibly the IBM patent database; however, he was not sure at trial. He is unable to locate any notes or records of the search. This search did not return

¹⁸ While a patent applicant normally has no duty to conduct a search for prior art, an applicant that seeks to have his or her application made special assumes, inter alia, an obligation to conduct a "careful and thorough search of the prior art." Manual of Patent Examining Procedure § 708.02(I)(D) (7th ed. 1998).

Sandpiper because at the time the company had no United States patents.¹⁹ In addition, he searched a patent assignment database for five companies identified by Leighton in the provisional application as involved in the problem of content delivery. Because the provisional application had been filed before Akamai became aware of Sandpiper, it was not one of the five companies listed and therefore was not among the companies searched. Judson testified that his practice was not to search a database of foreign patent assignments, which conceivably might have found Sandpiper's PCT application. Finally, he conducted a manual search through other patents he had in his office that he thought might be relevant.

Akamai did disclose approximately 50 references during the prosecution of the '703 case, including the Kriegsman and Kenner patents as well as a paper by Amir, et al. The Amir paper appeared on its face to have been published after the '703 application priority date, but Akamai disclosed to the PTO that it may have been publicly available before that date. At the jury trial, Limelight relied on the Kenner patent and a presentation by Peterson, one of the authors of the Amir paper, to argue that the '703 patent was obvious. Judson testified that he was not aware of the Sandpiper PCT application or the '598 patent until after the '703 patent issued.

In hindsight, Judson probably should have disclosed the existence of Sandpiper's Footprint system to the PTO, or at least included Sandpiper in his searches, and the lack of records concerning his PTMS search investigation is

¹⁹ It is unclear whether this search would have returned Sandpiper even if the database had included the '598 patent, since the '598 specification had its own lexicon for terms such as "reflector" and "repeater."

troubling. Nonetheless, I credit his testimony that by the time he became aware of Sandpiper he viewed it as a copier not an innovator. See Molins, 48 F.3d at 1181 (expressing concern about “the ease with which a relatively routine act of patent prosecution can be portrayed as intended to mislead or deceive”).

Limelight argues that intent can be inferred from Akamai’s failure to disclose the Footprint 1.0 system to the PTO, its decision to file a PTMS shortly after the Sandpiper PCT application was published²⁰ and the failure by Judson to search for Sandpiper as an assignee or to search foreign databases. But Akamai did disclose references which Limelight later argued to the jury either anticipated claims in the ’703 patent or, when combined, rendered those claims obvious. In addition, it disclosed the Amir paper and explained that the paper could be prior art notwithstanding its apparent date of publication. The Amir paper described the system on which the Peterson presentation was based, a reference heavily relied on by Limelight to invalidate the ’703 patent.

²⁰ Judson would not describe the reasons Akamai choose to file a PTMS or explain the timing of that application, citing client privilege. Limelight asserts that the court may draw a negative inference from this invocation of attorney-client privilege. This is incorrect. While the invocation of privilege denies Akamai the benefit of any benign explanation for the coincidence of its PTMS following so closely the publication of Sandpiper’s PCT application, it cannot contribute a negative inference. Compare Brasseler, U.S.A. I., L.P. v. Stryker Sales Corp., 267 F.3d 1370, 1384 (Fed. Cir. 2001) (describing the invocation of privilege as one reason it was “not surprising that the district court was unable to point to any credible evidence upon which it could reasonably infer that [the patent attorneys] acted in good faith”), with Datapoint Corp. v. Picturitel Corp., 215 F.3d 1344, 1999 WL 507141, at *4 (Fed. Cir. July 15, 1999) (unpublished) (“The district court [] did not err in excluding the evidence that Datapoint’s counsel had invoked the attorney-client privilege and in declining to instruct the jury that it could infer wrongful intent on Datapoint’s part from counsel’s invocation of the privilege.”). See also Knorr-Bremse Systeme Fuer Nutzfahrzeuge GmbH v. Dana Corp., 383 F.3d 1337, 1345 (Fed. Cir. 2004).

Thus, I conclude that Limelight has failed to prove by clear and convincing evidence that the Inventors intended to deceive the PTO or that Judson was deliberately indifferent to notice of material information during the pendency of the '703 application. Cf. Brasseler, U.S.A. I., L.P. v. Stryker Sales Corp., 267 F.3d 1370, 1383 (Fed. Cir. 2001) (finding attorneys willfully ignored notice of a potentially invalidating event in an conscious effort to avoid complying with their duty to disclose). Limelight has not sustained its burden of proof on its charge of inequitable conduct.

B. Limelight's Defenses of Laches and Equitable Estoppel

1. Legal Standard

a. Laches

To prevail on its laches defense, Limelight must prove by a preponderance of the evidence that: (1) "the plaintiff delayed filing suit for an unreasonable and inexcusable length of time from the time he knew or reasonably should have known of his claim against the defendant;" and (2) "the delay operated to the prejudice or injury of the defendant." A.C. Aukerman Co. v. R.L. Chaides Constr. Co., 960 F.2d 1020, 1032 (Fed. Cir. 1992) (en banc). The laches period begins to run when the patentee has actual or constructive knowledge of the infringer's activity. Id. A successful laches defense "bars relief on a patentee's claim only with respect to damages accrued prior to suit." Id. at 1041.

A presumption of laches exists where a patent holder delays more than six years before filing suit. Id. at 1035. Here, Limelight alleges only a five-year delay between Akamai learning of Limelight's CDN in 2001 and its filing of this lawsuit. Therefore,

there is no presumption of laches, and “unreasonable delay and prejudice [] must be proved and judged on the totality of the evidence presented.” Id. at 1038.

b. Equitable Estoppel

To show equitable estoppel, Limelight must prove: (1) that Akamai communicated by statements or conduct that it did not intend to press an infringement claim against Limelight; (2) that Limelight substantially relied on this misleading conduct, i.e. it was lulled into a false sense of security in its actions going forward; and (3) that it would be materially prejudiced if Akamai is allowed to proceed. A.C. Aukerman Co., 960 F.2d at 1041. The inferences Limelight derived from Akamai’s conduct must be reasonable. Id. at 1043. “[S]ilence alone will not create an estoppel unless there was a clear duty to speak” or the silence reinforces a plaintiff’s acquiescence to a defendant’s conduct. Id. Unlike laches, equitable estoppel does not require an unreasonable passage of time prior to filing suit. Id. at 1041. “Where equitable estoppel is established, all relief on a claim may be barred.” Id. The Federal Circuit has “adopted the preponderance of evidence standard in connection with the proof of equitable estoppel factors, absent special circumstances, such as fraud or intentional misconduct.” Id. at 1046.

2. Factual Background

Michael Gordon (“Gordon”), one of the founders of Limelight, testified that Limelight was founded in June 2001 and began marketing its content delivery service in November of that year. Limelight grew and added customers and equipment over the next several years. Late in the first quarter of 2004, officials from Akamai and Limelight

met to discuss the possible acquisition of Limelight by Akamai. Over the course of approximately the next six months, Gordon provided Akamai with information on Limelight's financial results, technology and content delivery architecture. At a meeting in June 2004, Gordon, along with Limelight's CEO and its Chief Technical Officer, met with a group of Akamai employees, including several technical people. At that meeting, Limelight explained that its network did not choose a best server, rather it used Internet routing techniques to route a user's request to a particular group of servers and assigned a particular server to fill the request on a round-robin basis. Gordon understood that Akamai viewed the infrastructure and architectures of Limelight's network as very different from its own. This understanding is confirmed by a summary Akamai prepared during the 2004 negotiations, which described Limelight as "tak[ing] a very different approach to its network architecture than Akamai." (DX1146, AKL064050.) Indeed, according to Gordon, Akamai reached the conclusion that, after an acquisition, the approach that "Akamai would pursue would be to migrate Limelight's customers off of Limelight's architecture and onto Akamai's and then decommission Limelight's network." (Trial Tr. Day 1, 124:11-15, Nov. 12, 2008.)

Ultimately, Akamai decided not to make an offer for Limelight. In a late October 2004 email to another of the founders of Limelight, Robert Wood ("Wood"), the Vice President of Corporate Development for Akamai, concluded the discussions with:

I . . . want to congratulate you on the great business that you and your team have built. I really enjoyed getting to know you and Mike [Gordon] and hope our paths cross again in the future. My understanding from the last conversation with Broadview is that you guys are going to receive tremendous value for the business and I am glad to hear it.

(DX1072.) Gordon testified that, although he had general discussions with Wood about the Speedera lawsuit, no one at Akamai ever indicated that Limelight's CDN infringed any Akamai patents.

During and after the 2004 negotiations, Limelight continued to add customers, purchase equipment and add capacity to its network. Gordon testified that Limelight invested about \$10 million in capital equipment in 2005 alone. It also added three new United States locations for its servers and began building points of presence in several European cities. Limelight purchased equipment at an increasing rate in 2006, at least doubling its purchases in the six months after the lawsuit was filed compared to the prior six months.

In April 2005, Limelight began implementation of a new architecture for its CDN which did not allocate servers on a round-robin basis; rather, it used software that maintained a list of information concerning the "health" of the available servers to select an appropriate server.

In early 2006, the parties renewed acquisition discussions, and this time they progressed to an offer by Akamai to acquire Limelight. However, Limelight decided to stay independent and find alternate funding. On June 22, 2006, it rejected Akamai's final offer, and the parties wished each other well. The following day, Akamai filed this lawsuit charging Limelight with infringing its patents. Gordon testified without contradiction that, prior to notifying him of the lawsuit, Akamai never suggested that Limelight infringed any of its patents or that it had considered suing Limelight.

Leighton testified that Akamai began to investigate the possibility that Limelight

was infringing its patents in mid-2005 because Limelight was starting to be a serious competitive threat and because the Speedera litigation had been resolved. The investigation concluded about the time the parties renewed their acquisition discussions. While the negotiations were ongoing, Akamai refrained from instituting any legal actions, but Limelight's disinclination to be bought removed such restraints.

On cross-examination, however, Leighton admitted that Akamai viewed Limelight as one of its competitors much earlier than 2005, a view confirmed by an April 2002 Akamai confidential document titled "How To Beat LimeLightNetworks." (DX1221.)²¹ In that report, Akamai recommended that its sales force "[p]osition LimeLight as a start up – just like the other 25 or so that are out there." (*Id.* at AKL124384.) In addition, an October 2003 internal Akamai email describes Limelight as "a key competitor in several accounts." (DX1212.)

Akamai clearly recognized Limelight as a potentially serious competitor as early as 2002 and, in its memo, considered how to beat Limelight, not the "other 25 or so" content delivery companies. (DX1221.) In addition, by April 2002 Akamai was aware that Limelight had "snagged one of its notable clients, MusicMatch.com" and was pricing its services aggressively against Akamai's FreeFlow service. (*Id.*) However, I find no evidence that Akamai investigated Limelight's technology in the context of patent infringement prior to mid-2005. Wood participated in both the 2004 and the

²¹ Although Akamai did not object to DX1221 during Leighton's cross-examination, it does so now on grounds of relevance. (See DX1221 flysheet.) The objection is overruled. Not only is it belated, but the document is relevant to the issue of how Akamai viewed Limelight at the time and what information it possessed concerning its competitor.

2004 acquisition discussion, and I credit his testimony that he was unaware of any investigation by Akamai into the question of patent infringement during the earlier negotiations. In addition, there is no evidence that Limelight's round-robin CDN implementation infringed any Akamai patent. While Akamai did include this implementation in its initial infringement contentions, it ultimately dropped it from the lawsuit, and the jury awarded damages only for Limelight's use of the architecture introduced in April 2005.

3. Discussion

a. Laches

Limelight argues that Akamai's delay of five years in filing this lawsuit prejudiced it because, had it known that Akamai believed Limelight's technology infringed its patents, Limelight "would likely not have focused solely on expanding its CDN business but would have also developed and expanded other types of business such as transit and hosting businesses." (Docket # 353, 7.) Limelight also complains about evidentiary disadvantages, including witnesses' loss of memory visited upon it as a result of the delay. According to Limelight, since Akamai viewed it as a competitor in the content delivery field, Akamai had a duty to investigate whether Limelight infringed any of its patents and cannot avoid a finding of laches by willful ignorance. Therefore, Limelight argues, Akamai's infringement damages should be limited to damages accruing after it filed suit in 2006.

Akamai responds that it did not learn that Limelight might be infringing its patents until 2005 and that, because it dropped any infringement claims based on

Limelight's pre-2005 technology, it sued within 14 months of when the infringing conduct began, and thus there can be no inexcusable delay. Akamai also disputes Limelight's claim that it was prejudiced by any delay in the filing of the lawsuit. Limelight's argument that it would have "likely" done something different is insufficient to show economic harm and the claimed lack of memory is rebutted by Limelight's factual allegations in its inequitable conduct claim. Moreover, Akamai states that the information Limelight seeks is available in the 2001 transcripts of the Digital Island case, which were taken much closer in time to the events at issue; thus, it suffered no evidentiary prejudice.

First, whether plaintiff unreasonably delayed filing suit depends on the date the clock began to run which, in turn, depends on the date on which plaintiff knew or should have known of defendant's infringement of the '703 patent. If Akamai first became aware of Limelight's use of the infringing technology in 2005, the 14-month delay in filing suit is too short to sustain a finding of laches. This is particularly true here as the parties were engaged in acquisition discussions for more than a third of that time. Limelight does not explicitly dispute this conclusion; rather, it argues that the unreasonable delay began in October 2001, when Akamai first learned of Limelight's competing service. (See Limelight's Proposed Findings of Facts and Conclusions of Law (Docket # 407, Ex. A), FF123, FF125, CL83.)

As to knowledge, none of the pre-2005 documents on which Limelight relies suggest that Akamai believed Limelight infringed Akamai's patents, only that Limelight posed a competitive threat. (See, e.g., DX1216, DX1225.) Indeed, that is what it told

Gordon, who testified that he was led to believe that Akamai viewed his company's technology as significantly different from its own, and Akamai's internal evaluation of Limelight in June 2004 made precisely that point. (See DX1146, AKL0964050.) Akamai's plan at the time was to purchase Limelight and then shut down Limelight's content delivery system and not even try to integrate it with Akamai's. Thus, there is no evidence that Akamai believed that Limelight was infringing its patents prior to 2005, but that Akamai failed to bring suit.

Nor did Akamai have a duty to investigate Limelight's CDN in 2001. While it "could not simply ignore any and all evidence of potentially infringing activity," it did not have a duty to investigate Limelight merely because it announced a competing service. Wanlass v. Fedders Corp., 145 F.3d 1461, 1466 (Fed. Cir.1998). A duty to investigate a particular product arises "when publicly available information about it should have led [the patent holder] to suspect that product of infringing." Id. Akamai does not claim to have invented the concept of content delivery services, nor does it claim that every content delivery service infringes its patent. Rather, as Leighton explained, the key ideas of the Akamai system are tagging page objects with a virtual hostname and adding intelligence to the DNS system to resolve that virtual hostname into an optimal server. During the 2004 discussions, Limelight described its content delivery service as allocating servers on a round-robin basis, i.e., sequentially rather than using some intelligent selection. This information did not give rise to a duty to investigate Limelight's CDN for infringement at that time.

There is also no evidence that Akamai learned of Limelight's April 2005

deployment of the infringing system or that it should have known of this deployment.²²

There is evidence only that, Akamai, concerned over Limelight's success in the marketplace, went looking for a way to blunt that competition in mid-2005 and discovered the alleged infringement as a result of its investigation. Prior thereto, it had no notice of any possible infringement and, therefore, no legal duty to investigate.

Finally, even if Akamai's two-year delay in bringing suit was unreasonable, Limelight has not shown that it was unduly prejudiced by that delay. While Gordon testified that Limelight looked at opportunities other than content delivery, and invested in some of them, it invested heavily in content delivery because it believed it would grow as broadband access and media content flourished. He suggested that, had Akamai indicated there might be an infringement issue, Limelight might not have made the investment it did, but did not explain how its decisions would have changed. Indeed, Limelight continued to invest even greater amounts in its content delivery business after the lawsuit was filed. In addition, Limelight did not then, and indeed still does not, believe its system meets, inter alia, the "tagging" limitation of the '703 patent or chooses an optimal server as required by the patent. (See, e.g., Docket # 310.) This evidence contradicts Limelight's assertion that it might have changed course had it

²² Nor did Limelight present evidence to suggest that detecting infringement of the '703 patent was "easily testable" such that Akamai had a duty to conduct an investigation of any content service provider competitor who conceivably could be using the patented technology, even without some additional knowledge that it used DNS to assign an optimal server. Wanlass v. GE, 148 F.3d 1334, 1339-40 (Fed. Cir. 1998) ("The frequency with which these types of investigations [into competitor's products] should have occurred is a function of their cost and difficulty.").

been warned of Akamai's claims of infringement. Limelight has failed to meet its burden to show material economic prejudice due to any delay in Akamai's filing suit.

Similarly, Limelight has failed to show that the evidentiary prejudice it alleges would not have occurred if discovery had taken place two years earlier. Lewin died in 2001, Limelight had access to the testimony of the Inventors and Akamai's employees from the two earlier lawsuits and there is no evidence that Judson's notes were lost after the suit was filed.

The defense of laches fails.

b. Equitable Estoppel

The factual question decisive of Limelight's defense of equitable estoppel is whether Akamai intentionally misled Limelight before initiation of this suit as to its position on infringement by Limelight and its plan to enforce its rights under the patent. Given Akamai's aggressive stance against infringers, but its silence as to Limelight's technology, Limelight argues it was reasonable for it to infer that Akamai did not think it infringed.

Akamai states that it had no duty to threaten or warn Limelight prior to suit that it was infringing and therefore there can be no estoppel. Furthermore, even if it had a duty to speak, Limelight can show no prejudice for the same reasons none exists in its laches defense.

Although Limelight asserts that it was lulled into a false sense of security by Akamai's silence during the 2004 negotiations, at that time it used only the round-robin method of selecting a server. At the trial Akamai did not assert that this method

violated the '703 patent, and the jury assessed damages only on Limelight's new system first implemented in the spring of 2005. Thus, Akamai's silence on patent issues, its statements that Limelight's network was very different and Wood's well wishes during the 2004 negotiations cannot be considered conduct intended to mislead Limelight.

By the time of the 2006 negotiations, Akamai clearly had concluded that Limelight's new system infringed its patent, however, it still had no obligation to warn Limelight prior to filing suit. See Aukerman, 960 F.2d at 1042 ("Silence alone will not create an estoppel unless there was a clear duty to speak . . ."). In order to foster reliance, Akamai's silence must "reenforce[] the defendant's inference from the plaintiff's known acquiescence that the defendant will be unmolested." Id. Here, negotiations began shortly after Akamai's investigation concluded that Limelight's system was infringing, and Akamai reasonably did not want to destroy the possibility of an acquisition by filing suit. It did, however, hint that Limelight could face litigation if the negotiations failed. During the 2006 negotiations, Wood responded to attempts by Limelight for a higher valuation by pointing out that Akamai had "a very hard patent portfolio, and that anyone operating a Content Delivery Network the way that Limelight was operating its Content Delivery Network certainly came through that patent portfolio." (Trial Tr. Day 3 Sess. 2, 77:5-8, Nov. 14, 2008.) The evidence is clear, and I find that Akamai did not deliberately mislead Limelight into believing it would not be sued for infringement.

At the same time, the evidence does not support Limelight's claim of reliance on

Akamai's silence in making business decisions. By 2005, Limelight already had contractual obligations to provide delivery services, and it believed and continues to believe that it does not perform several of the limitations of the asserted claims. Therefore, Limelight's equitable estoppel defense fails both for lack of misleading conduct by Akamai and lack of detrimental reliance by Limelight.

C. Unclean Hands

Unclean hands is a catch-all equitable remedy that allows a court to refuse to aid "one tainted with inequity or bad faith relative to the matter in which he seeks relief, however improper may have been the behavior of the defendant." Precision Instrument Mfg. Co. v. Auto. Maint. Mach. Co., 324 U.S. 806, 814 (1945). The misconduct alleged need not be punishable as a crime, but must "rightfully [] be said to transgress equitable standards of conduct." Id. at 815. A finding of unclean hands allows a court to deny relief to the culpable party.

The Supreme Court has found unclean hands and refused to enforce patent rights where a plaintiff settled an interference with a rival and ultimately obtained rights to the rival's invention even though it knew that the original applicant had committed perjury in his application to the PTO (Precision, 324 U.S. at 818-19) and where a patent holder used an earlier successful infringement suit as the basis for an injunction in a new suit, even though it had previously concealed evidence of prior use of the invention. Keystone Driller Co. v. Gen. Excavator Co., 290 U.S. 240, 247 (1933). In addition, the Federal Circuit has refused to enforce several other of a plaintiff's patents under the doctrine where it found inequitable conduct in the prosecution a related

patent. See Consol. Aluminum Corp. v. Foseco Int'l, Ltd., 910 F.2d 804, 812 (Fed. Cir. 1990).

Limelight bases its unclean hands defense on the same conduct that underlies its inequitable conduct and equitable estoppel defenses, discussed above. It argues that “these are the very types of wrongful actions that bar relief under the doctrine of unclean hands.” (Docket # 407, 38.) Even if the evidence supported Limelight’s claim, unclean hands is an exceptional remedy reserved only for truly egregious conduct. Here, the evidence does not support the claim of bad faith and unfair conduct.

D. Limelight’s Motion for Reconsideration

Limelight moves for reconsideration of the decision denying its motion for JMOL, which focused on the issue of whether it directs or controls every step of the asserted claims of the '703 patent. In particular, Limelight points to the Federal Circuit’s opinion in Muniauction, Inc. v. Thomson Corp., released two weeks after I denied its motion for JMOL, to argue that Akamai’s proof was insufficient for the jury to find it infringed the patent. 532 F.3d 1318 (Fed. Cir 2008), cert. denied, — S.Ct. —, 2009 WL 578715 (U.S. Mar. 9, 2009) (No. 08-847).

It is undisputed that Limelight does not itself perform every step of the claims found infringed. While it provides its customers with the information necessary for them to modify their web pages or Internet address routing information to utilize its service,²³ the actual modifications are performed by the customer, not Limelight. These

²³ In the first method, the customer changes the hostname address of one or more page objects in the initial web page to point to Limelight’s servers (the “prepend method”). In the second method, the customer adds or changes alias information in its

modifications are required steps in both of the independent claims found infringed by the jury.²⁴ In addition, the content provider is responsible for providing the initial web page to the end user, therefore, Limelight does not perform the first step of claim 19 as well.²⁵

1. BMC Resources

At trial, both parties relied on the Federal Circuit's opinion in BMC Resources, Inc. v. Paymentech, L.P., 498 F.3d 1373 (Fed. Cir. 2007), for their proposed jury instructions addressing the issue of direct infringement by multiple parties performing different steps of a single patented process. (See Docket # 264, 20; Docket # 267, 11.) In BMC Resources, the court reaffirmed the requirement that "[i]nfringement requires . . . a showing that a defendant has practiced each and every element of the claimed invention." 498 F.3d at 1380. However, a party cannot avoid infringement by "contracting out steps of a patented process to another entity" if it directs or controls the actions of that entity. Id. at 1381. The court acknowledged that "requiring control or

DNS record so that the hostname addresses of the page objects resolve to Limelight's servers without requiring any change to the customer's initial web page (the "CNAME method").

²⁴ (See '703 patent, claim 19 (. . . for a given page normally served from the content provider domain, tagging the embedded objects of the page so that requests for the page objects resolve to the [content delivery service] domain instead of the content provider domain; . . .); id., claim 34 (. . . for a given page normally served from the content provider domain, tagging at least some of the embedded objects of the page so that requests for the objects resolve to the [content delivery service] domain instead of the content provider domain; . . .).)

²⁵ (See '703 patent, claim 19 (" . . . responsive to a request for the given page received at the content provider domain, serving the given page from the content provider domain; . . .").)

direction for a finding of joint infringement may in some circumstances allow parties to enter into arms-length agreements to avoid infringement.” Id.

The accused infringer in BMC Resources received payment information for a customer transaction from a merchant, forwarded it to a participating debit network who then forwarded it to an affiliated financial institution. The patented method included steps performed by all four parties, the merchant, the defendant payment services provider, the debit network and the financial institution. In addressing the question whether the defendant directed or controlled the steps it did not itself perform, the Federal Circuit concluded that the mere provision by the defendant of data describing the financial transaction to a debit network, “absent any evidence that [defendant] also provides instructions or directions regarding the use of those data,” was inadequate to establish that it controlled or directed the activity. Id. It also noted the tenuous relationship between the defendant and the financial institutions as weighing against a finding of direction or control, citing the lack of evidence “even of a contractual relationship” between them. Id. at 1382.

Thus, BMC Resources left open the possibility that direction or control adequate for a finding of direct infringement might exist where an accused infringer provided data to another entity along with instructions or directions regarding the use of those data. Similarly, BMC Resources suggested that the existence of a contractual relationship between the accused infringer and the entity performing other steps of the accused method was a significant consideration.

2. The Jury Instructions in the Instant Case

At the conclusion of the evidence in the instant case, I instructed the jurors that if they found that use of Limelight's service infringed the asserted claims then:

[t]he second issue is that I think it is not disputed that Limelight does not make [the prepend method] substitution. The substitution is made by the content provider. That raises the question whether the content provider, when carrying out this step, acts under the direction and control²⁶ of Limelight such that Limelight can properly be deemed to be the one to do it. And you heard argument about that just before we broke.

If Limelight did not direct and control this action, then this substitution cannot be attributed to Limelight. And Limelight cannot, therefore, infringe. And in the CNAME method, it's similar, but rather than modifying the object on the page served, the content provider modifies its DNS system so that the object is retrieved from Limelight's content delivery network by means of an alias of the hostname.

Again, the first [question] is whether this method of getting to the Defendant content delivery network infringes any claim, and the second question, again, is whether the content provider acted under the direction and control of Limelight. And again, if Limelight directed and controlled this action, it was effectively the action of Limelight, and then it may be found responsible. But if Limelight did not direct and control, both are necessary, the modification at the content provider, then it cannot be deemed to infringe.

Do you understand what I'm trying to tell you? It's like one acting for somebody else and controlling that person, then you can be responsible. But if the other person, if you do not control what the other person does, then you are not responsible for what the other person does, not liable for what the other person does.

So, you should review the evidence, decide how the Limelight systems work, how does the interaction with the content provider work, and, specifically, does Limelight direct and control the modifications or does the content provider carry out these tasks entirely independently. Then compare each of the mechanisms with what is claimed in the certain claims and,

²⁶ I initially instructed the jury that Limelight must both direct and control the actions of the content provider. Upon conferring with the counsel after this charge, I issued a correcting instruction as discussed infra.

specifically, does either of the Defendant's content delivery methods practice each element of whichever claim you are considering.

(Trial Tr. Day 13 Sess. 2, 20-22, Feb. 28, 2008.)

Upon conferring with the parties after this instruction, I issued the following correction:

One of those [things I didn't get right] is that I had told you with respect to activities that Limelight says were really done by somebody else, namely the content provider, that such activities are chargeable to Limelight if Limelight directed and controlled this content provider in the example we're using. I was wrong on that. It is either direct or control, control or direct; it doesn't have to be both.

(Id. at 52-53.)

Although the jury returned a finding of infringement on these instructions, Limelight moved for JMOL arguing that there was "no substantial evidence" that it "directs or controls another party to perform" several steps of the asserted claims. (Docket # 310, 3-4.) I denied the motion because, unlike in BMC Resources, here there was evidence that not only was there a contractual relationship between Limelight and its customers, but that it provided those customers with instructions explaining how to utilize its content delivery service. Therefore, I found that there was sufficient evidence for the jury to have found direction or control. Limelight now moves for reconsideration of that decision, arguing that under the Federal Circuit's subsequent decision in Muniauction "an accused infringer's control over access to an Internet-based system, coupled with instructions to customers on how to use that system, is insufficient to establish direct infringement." (Docket # 377, 2.) It asserts that Muniauction requires a showing that it is vicariously liable for the act committed by

others in order for a jury to find it directly infringed Akamai's patented method.

3. Muniauction

In Muniauction, decided July 14, 2008, MuniAuction, Inc. ("MuniAuction"), the patent owner, brought an action against defendant Thompson for infringement of a patent claiming a method of auctioning municipal bonds using a web browser. The preamble of claim 1 describes an auction system consisting of a "bidder's computer [] located remotely from [the bond] issuer's computer," with the two computers connected by an electronic network such as the Internet. Muniauction, 532 F.3d at 1322 (quoting United States Patent No. 6,161,099). The first step of the method requires "inputting data associated with at least one bid . . . into said bidder's computer . . ." Id. The court noted that this step is completed by the bidder, while "at least a majority of the remaining steps are performed by [defendant's] system." Id. at 1328-29. Therefore, the issue was whether the actions of the bidder could be combined with those of the defendant to give rise to a finding of direct infringement by the latter. Referencing its earlier decision in BMC Resources, the court explained

where the actions of multiple parties combine to perform every step of a claimed method, the claim is directly infringed only if one party exercises "control or direction" over the entire process such that every step is attributable to the controlling party, i.e., the "mastermind." At the other end of this multi-party spectrum, mere "arms-length cooperation" will not give rise to direct infringement by any party.

Id. at 1329 (internal citations omitted). In applying this standard to the facts of the case, the Federal Circuit concluded that

[u]nder BMC Resources, the control or direction standard is satisfied in situations where the law would traditionally hold the accused direct infringer vicariously liable for the acts committed by another party that are required

to complete performance of a claimed method. In this case, Thomson neither performed every step of the claimed methods nor had another party perform steps on its behalf, and Muniauction has identified no legal theory under which Thomson might be vicariously liable for the actions of the bidders. Therefore, Thomson does not infringe the asserted claims as a matter of law.

Id. at 1330 (internal citation omitted).

a. Vicarious Liability

Thus, the first question posed by the Federal Circuit's decision in Muniauction is the following: is vicarious liability a necessary condition to satisfy BMC Resources' control or direction standard, as Limelight asserts, or is it merely a condition sufficient to find infringement within the spectrum of possible interactions ranging from an arms-length agreement to "contracting out steps of a patented process to another entity?" BMC Resources, 498 F.3d at 1381. I conclude that the court intended the latter. First, in Muniauction, the court refers to vicarious liability as an alternative theory of liability to having "another party perform steps on its behalf." 532 F.3d at 1330. Second, in BMC Resources, the court cautioned that "a defendant cannot [] avoid liability for direct infringement by having someone else carry out one or more of the claimed steps on its behalf." 498 F.3d at 1379. Were vicarious liability a requirement for a finding of joint infringement, then an entity could use a patented method with impunity by hiring an independent contractor to perform one or more steps of that method. See, e.g., Graham v. Malone Freight Lines, Inc., 314 F.3d 7, 15 (1st Cir 1999) ("Employers are generally not liable for the negligent acts of the independent contractors they hire.") Yet in BMC Resources, the court explicitly stated that one cannot avoid infringement by "contracting out steps of a patented process to another entity." 498 F.3d at 1381.

There is no indication that the court intended to make a major change to this jurisprudence in its Muniauction decision.²⁷ While a showing of vicarious liability is sufficient to find direction or control, it is not a necessary requirement. Therefore, the fact that Limelight is not vicariously liable for the actions of its customers does not end the inquiry.

b. The Significance of Muniauction

However, Muniauction did establish a new data point on the continuum between an arms-length relationship and vicarious liability for determining direction or control. In BMC Resources, as discussed supra, the court's reasoning left open the possibility that evidence of direction or control might be found from the provision of "instructions or directions" regarding the use of the data the defendant provided to the debit networks. 498 F.3d at 1381. It similarly suggested that the lack of a contractual relationship between the defendant and the financial institutions was relevant in finding a lack of direction or control of the latter's actions by the former. Indeed, in opposing Limelight's original motion for JMOL, that is exactly how Akamai summarized the law on joint infringement:

The BMC decision and subsequent decisions of several District Courts identify factors that can support a finding of direction and [sic] control: (1) a contractual relationship between the accused party and the third party performing one or more steps of the claim; and (2) the accused infringer's provision of instructions, or directions to the third party that direct the third party to perform one or more steps of the claim.

²⁷ Nor was Muniauction decided en banc, a requirement to overruling a prior decision. See El-Shifa Pharm. Indus. Co. v. United States, 378 F.3d 1346, 1352 (Fed. Cir. 2004).

(Docket # 343, 31 (citations omitted).)

In Muniauction, however, the court explicitly held “[t]hat Thomson controls access to its system and instructs bidders on its use is not sufficient to incur liability for direct infringement.” 532 F.3d at 1330. It also described as not “relevant to whether Thomson satisfies the ‘control or direction’ standard of BMC Resources,” a jury instruction which directed the jurors to ask whether “there one party teaching, instructing, or facilitating the other party’s participation in the electronic auction process.” Id. at 1331. While the Muniauction decision makes no explicit mention of a contract between Thompson and its customers, it quotes the lower court’s finding that the defendant charges the bidders a fee for its services. See id. at 1329 (quoting Muniauction, Inc. v. Thomson Corp., 502 F. Supp.2d 477, 492 (W.D. Pa. 2007)). In addition, MuniAuction’s brief to the Federal Circuit highlighted the contractual relationship between the parties as a factor distinguishing Thompson’s actions from those of the defendant in BMC Resources. (See Docket # 378, Ex. D, 26; see also id. at 13 (describing Thompson’s contracts with bidders and issuers).) Thus, Muniauction establishes that direction or control requires something more than merely a contractual agreement to pay for a defendant’s services and instructions or directions on how to utilize those services.

Limelight therefore argues that, under Muniauction, its “control over access to an Internet-based system, coupled with instructions to customers on how to use that system, is insufficient to establish infringement.” (Docket # 378, 2 (emphasis in original).) It asserts that its content delivery services are “remarkably similar” to the

auction bidding system deemed not infringed by Thompson in Muniauction such that that case's holding should be binding on the facts in this case. (Id. at 4-5.)

Akamai attempts to distinguish Muniauction, arguing that “[t]he facts of this case are grossly different” because once Limelight’s customers “sign Limelight’s contract, [they] are under a contractual obligation to – implement Limelight’s content delivery method by serving pages that include a hostname supplied by Limelight to the customer.” (Docket # 379, 3-4.) However, this formulation misstates the basis of the contract between Limelight and its customers. Limelight does not “contract[] out steps of a patented process to another entity.” BMC Resources, 498 F.3d at 1381. Rather, like Thompson, the fundamental agreement between Limelight and its customers is the provision of a service in exchange for payment. Thompson provides its customers with the ability to bid on municipal bonds, while Limelight promises to serve its customers’ page objects from its network. In each case, the customer must perform a step of the patented method in order to obtain the offered service, but the consideration offered by the customer is in the form of financial remuneration. Akamai’s argument -- that direction or control is established by the existence of a contractual relationship in which one entity incidentally has to perform a step of a patented process to receive the benefits of the contract – runs counter to the Federal Circuit’s holding that “mere ‘arms-length cooperation’ will not give rise to direct infringement by any party.” Muniauction, 532 F.3d at 1329; see also BMC Resources, 498 F.3d at 1381 (“[T]he standard requiring control or direction for a finding of joint infringement may in some circumstances allow parties to enter into arms-length agreements to avoid

infringement.”). Therefore, while a contract to perform steps of a patented process on the mastermind’s behalf cannot shield the mastermind from liability for direct infringement, the existence in the instant case of a contract for services does not give rise to direction or control, even if the customer must perform one or more steps of the patented process in order to receive the benefits of those services.

Akamai further argues that because “[t]he customer must use the Limelight-supplied hostname” as instructed by Limelight in order to obtain its content delivery services, and because “[t]here is no other purpose for providing a hostname to the customer,” there was sufficient evidence for the jury to find direction or control of the tagging step. (Docket # 379, 3-4 (emphasis in original).) However, MuniAuction unsuccessfully made a similar argument to the Federal Circuit. It argued that because Thompson supplies a bidder ID and password to its customers and instructs its customers as to their use, and because that ID and password are necessary in order to access its auction system, “Thompson controls the [claimed] inputting step.” (Docket # 378, Ex. D, 11-12.) Nevertheless, the court held that controlling access to its system and instructing its customers on its use was not enough to evince direction or control.²⁸

²⁸ This result is likely to make unenforceable a number of patents claiming similar methods in which a remote user contacts a central server. The Federal Circuit implicitly acknowledged as much in BMC Resources by noting:

The concerns over a party avoiding infringement by arms-length cooperation can usually be offset by proper claim drafting. A patentee can usually structure a claim to capture infringement by a single party. See Mark A. Lemley et al., Divided Infringement Claims, 33 AIPLA Q.J. 255, 272-75 (2005). In this case, for example, BMC could have drafted its claims to focus on one entity. The steps of the claim might have featured references to a single party’s supplying or receiving each element of the claimed process.

I find no material difference between Limelight's interaction with its customers and that of Thompson in Muniauction. There is no suggestion that the agreements between Limelight and its customers for content delivery services were other than the result of an arms-length contract negotiation. Akamai has identified no legal theory under which Limelight might be vicariously liable for the actions of the content providers. The first step of claim 19 of the '703 patent, serving the initial web page from the content provider's domain, is performed by the content provider whether it subscribes to Limelight's services or not. Limelight's customers, following Limelight's instructions, do modify the embedded objects of their web pages or alter their DNS records so that requests for the objects resolve to the content delivery service domain,

However, BMC chose instead to have four different parties perform different acts within one claim. BMC correctly notes the difficulty of proving infringement of this claim format. Nonetheless, this court will not unilaterally restructure the claim or the standards for joint infringement to remedy these ill-conceived claims.

498 F.3d at 1381.

In its petition for a writ of certiorari, MuniAuction raised a similar concern. It argued that the Federal Circuit's holding "threatens drastically to change the meanings of the claims of hundreds or thousands of patents" and warned that "any claim that requires as one method step that a computer or cell phone must be activated by a user, even though that step is controlled by a web site operator or a carrier, can no longer be enforced against infringers." On Petition For A Writ Of Certiorari To The United States Court Of Appeals For The Federal Circuit at 21, 25, Muniauction, Inc. v. Thomson Corp., No. 08-847 (U.S. Dec. 22, 2008). See also id. at 35 (disputing the Federal Circuit's suggestion that the problem can be avoided by proper claim drafting and noting that in any event, it "generally cannot be implemented on issued patents already in litigation"). Nevertheless, the Supreme Court denied certiorari. See Muniauction, Inc. v. Thomson Corp., — S. Ct. —, 2009 WL 578715 (U.S. Mar. 9, 2009) (No. 08-847).

rather than the content provider domain, in order to take advantage of Limelight's service. However, this step is performed by Limelight's customers not because they are contractually obligated to do so; rather, they do so because they wish to avail themselves of Limelight's service. Under Muniauction, this is insufficient to establish the requisite direction or control by Limelight of its customers necessary to find it liable for direct infringement.

Therefore, Limelight's motion for reconsideration is allowed and, based on the standard set forth in Muniauction, Limelight's motion for JMOL of noninfringement is allowed.

IV. Conclusion

For the reasons set forth, Limelight's Motion to Admit Trial Exhibit 1228 (Docket # 399) is DENIED. Limelight's Motion for Leave to File a Reply (Docket # 413) is ALLOWED. Limelight's Motion for Reconsideration of the Court's Denial of its Motion for JMOL Based on New Case Law (Docket # 377) is ALLOWED.

The parties shall submit a proposed form of judgment within 20 days.

April 24, 2009

DATE

/s/Rya W. Zobel

RYA W. ZOBEL

UNITED STATES DISTRICT JUDGE

UNITED STATES DISTRICT COURT
DISTRICT OF MASSACHUSETTS

CIVIL ACTION NO. 06-11109-RWZ

AKAMAI TECHNOLOGIES, INC., et al.

v.

LIMELIGHT NETWORKS, INC.

ORDER REGARDING CLAIM CONSTRUCTION

June 29, 2007

ZOBEL, D.J.

I. Introduction

Plaintiffs Akamai Technologies, Inc. and the Massachusetts Institute of Technology (collectively "Akamai") allege that defendant Limelight Networks, Inc. ("Limelight") has infringed (1) United States Patent No. 6,108,703 ("the '703 Patent"), a "Global Hosting System;" (2) United States Patent No. 6,553,413 ("the '413 Patent"), a "Content Delivery Network Using Edge-of-Network Servers for Providing Content Delivery to a Set of Participating Content Providers;" and (3) United States Patent No. 7,103,645 ("the '645 Patent"), a "Method and System for Providing Content Delivery to a Set of Participating Content Providers" (collectively, the "Akamai Patents"). The three patents share a common specification; only the claims differ between them. The common Abstract describes the invention as an "inventive framework" that "allows a Content Provider to replicate and serve its most popular content at an unlimited number of points throughout the world." (Akamai Patents, Abstract.) The parties dispute the

construction of a total of seventeen claim terms from the three patents.

II. Legal Standard

The construction of patent claims is a matter of law for this court to decide.

Markman v. Westview Instruments, Inc., 517 U.S. 370, 372 (1996). “[T]he words of a claim are generally given their ordinary and customary meaning,” in other words, “the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention, *i.e.*, as of the effective filing date of the patent application.”

Phillips v. AWH Corp., 415 F.3d 1303, 1312-13 (Fed. Cir. 2005) (internal citations omitted); see also Markman v. Westview Instruments, Inc., 52 F.3d 967, 985 (Fed. Cir. 1995) (en banc), *aff'd*, 517 U.S. 370 (1996) (describing the focus in construing disputed terms as applying an objective test of the term’s meaning to “one of ordinary skill in the art at the time” and not a consideration of the subjective intent of the parties creating the patent contract). However, the presumption that words are given their ordinary meaning may be overcome if the patent specification or prosecution history “clearly and deliberately set[s] forth” a different meaning. K-2 Corp. v. Salomon S.A., 191 F.3d 1356, 1363 (Fed. Cir. 1999).

The scope and meaning of a patent’s claims must be ascertained in the context of the specification and the prosecution history. Phillips, 415 F.3d at 1315. While limitations from the specification should not be read into the claims, a patent is a “fully integrated written document” and the “claims must be read in view of the specification, of which they are a part.” Markman, 52 F.3d at 978-79; see also Gentry Gallery, Inc. v. Berkline Corp., 134 F.3d 1473, 1480 (Fed. Cir. 1998) (“claims may be no broader than

the supporting disclosure, and therefore [] a narrow disclosure will limit claim breadth”).

The Federal Circuit has refused to endorse a bright-line rule limiting the scope of the claims to the embodiment disclosed when only a single embodiment is described in the specification. See Teleflex, Inc. v. Ficosa N. Am. Corp., 299 F.3d 1313, 1326 (Fed. Cir. 2002). However, “when the preferred embodiment is described in the specification as the invention itself, the claims are not necessarily entitled to a scope broader than that embodiment.” Modine Mfg. Co. v. United States Int’l Trade Comm’n, 75 F.3d 1545, 1551 (Fed. Cir. 1996), abrogated on other grounds, Fast Corp. v. Checkouts Kinzoku Kogyo Kabushiki Co., Ltd., 234 F.3d 558 (Fed. Cir. 2000) (en banc); see also Microsoft Corp. v. Multi-Tech Systems, Inc., 357 F.3d 1340, 1348 (Fed. Cir. 2004) (“In light of those clear statements in the specification that the invention (‘the present system’) is directed to communications ‘over a standard telephone line,’ we cannot read the claims of [the patents at issue] to encompass data transmission over a packet-switched network”). In addition, statements in the “Summary of the Invention” portion of the specification “are not limited to describing a preferred embodiment, but more broadly describe the overall inventions of [the] patents.” Multi-Tech Systems, 357 F.3d at 1348.

III. Claim Construction and Discussion

Having considered in light of the applicable legal standard the parties’ written submissions as well as the argument of counsel at a hearing held on May 17, 2007, the court construes the disputed claim terms as follows:

A. '645 Patent Terms in Dispute

1. Term 1 ('645 Patent, Claim 1)

Term	Court's Construction
... a given object of a participating content provider is associated with an alphanumeric string a particular object of a participating content provider is associated with an alphanumeric string that includes the URL used to identify the object in the absence of a content delivery network ...

Akamai's suggestion that the term "associated" be given its dictionary meaning¹ ignores the Federal Circuit's warning in Phillips that "[t]he risk of systematic overbreadth is greatly reduced if the court [] focuses at the outset on how the patentee used the claim term in the claims, specification, and prosecution history, rather than starting with a broad definition and whittling it down." Phillips, 415 F.3d at 1321. The '645 Patent specification describes as the present invention a single embodiment in which the Uniform Resource Locator ("URL") used to retrieve an embedded object from the content provider's server(s) in the absence of a content delivery network is modified by prepending it with a virtual server hostname:

According to the present invention, a given Web page (comprising a base HTML document and a set of embedded objects) is served in a distributed manner. . . . To serve the page contents in this manner, the URL associated with an embedded object is modified. As is well known, each embedded object that may be served in a page has its own URL. . . . According to the invention, the embedded object URL is first modified, preferably in an off-line process, to condition the URL to be served by the

¹ (See Akamai's Claim Construction Mem. (Docket # 67) at 9 (citing the definition of "associated" from Merriam-Webster's Collegiate Dictionary.)

global hosting servers² Thus, according to the present invention, a virtual server hostname is prepended into the URL for a given embedded object

(‘645 Patent, col.6 l.35 - col.7 l.40 (emphasis added).)

The specification then continues on to describe “the inventive global framework” in the context of a specific example. (Id. col.7 ll.50-53 (emphasis added).) At step 5 of the example, a copy of the object is retrieved from a content delivery provider (“ghost”) server. The specification goes on to explain:

Step 6: If, however, no copy of the data on the ghost exists, a copy is retrieved from the original server or another ghost server. Note that the ghost knows who the original server was because the name was encoded into the URL that was passed to the ghost from the browser.

(Id. col.12 ll.54-58 (emphasis added).)

Here, the specification describes the invention as associating a particular object of a content provider with an alphanumeric string consisting of a virtual server hostname prepended onto the URL for the object. The URL of the object is necessary to the inventive global framework in order to retrieve the object from the content provider’s server if no copy exists on a ghost server. The specification discloses no

² The ‘645 Patent specification uses the terms “ghost,” “ghost server” and “hosting server” interchangeably to describe the service provider’s servers which replicate and deliver a participating content provider’s content to the user. (See ‘645 Patent, col.5 ll.65-67.) The claims, however, use the term “content server(s),” a term which does not appear in the specification, to describe these servers. (See, e.g., ‘645 Patent, Claim 1 (claiming “a method of content delivery wherein participating content providers identify content to be delivered by a service provider from a set of content servers that are distinct from the participating content provider sites and associated with the service provider”).) The court construes “ghost(s),” “ghost server(s),” and “hosting server(s)” to refer to “content server(s) distinct from the content provider server(s).”

other way that an object is associated with an alphanumeric string, nor is there any suggestion or teaching that an association which did not include the URL for the embedded object could be used in an embodiment of the invention. Therefore, Akamai's proposed construction is overly broad and the court declines to adopt it. Rather, the court adopts a construction that incorporates the association described in the specification as "the . . . invention." (Id. col.7 ll.36-40.)³

2. Term 2 ('645 Patent, Claim 1)

Term	Court's Construction
... an alternative domain name system (DNS), distinct from the Internet domain name system and any client local name server a domain name system, separate from the Internet DNS and the client's name server, that is controlled by a content delivery network service provider and includes control routines that are different from regular name servers ...

Both parties' proposed claim construction for Term 2 are deficient. Akamai's proposal essentially reads out the limitations in the claim requiring the DNS established by the service provider to be "alternative" and "distinct," while Limelight's proposed construction does not read on the preferred embodiment.

The Brief Summary of the Invention describes the "object of the present invention" as "provid[ing] a network architecture that moves content closer to the user." ('645 Patent, col.2 ll.49-50.) This is accomplished by "replicating content over a large

³ Limelight's proposed construction requiring the alphanumeric string to include "the domain name conventionally used . . . to identify the object," is excessively limiting. (See Docket # 90, 1.) Neither the specification nor the claim requires the domain name to be a part of the object URL. (See '645 Patent, col.6 ll.51-54 ("Typically, the URL has a hostname identifying the Content Provider's site from where the object is conventionally served . . .") (emphasis added).)

network of distributed servers.” (Id. col.2 ll.46-47.) The task of determining which of this multiplicity of servers should supply content to a particular user’s request is handled by the alternate and distinct DNS system:

The determination of which hosting server to use to serve a given embedded object is effected by other resources in the hosting framework. In particular, the framework includes a second set of servers (or server resources) that are configured to provide top level Domain Name Service (DNS). In addition, the framework also includes a third set of servers (or server resources) that are configured to provide low level DNS functionality.⁴

To locate the appropriate hosting servers to use, the top-level DNS server determines the user’s location in the network to identify a given low-level DNS server to respond to the request for the embedded object. The top-level DNS server then redirects the request to the identified low-level DNS server that, in turn, resolves the request into an IP address for the given hosting server that serves the object back to the client.

(Id. col.3 ll.29-36, 42-49 (Summary of the Invention) (emphasis added).) The specification emphasizes the difference between the invention and the operation of “regular DNS servers” which return the Internet Protocol (“IP”) addresses of one or more DNS or content servers without any consideration of where the user or the server is located:

[T]he global hosting architecture of the present invention manipulates the DNS system so that the name is resolved to one of the ghosts that is near the client and is likely to have the page already. . . . The top level DNS servers [for the inventive global hosting framework] have a special function that is different from regular DNS servers like those of the .com domain. The top level DNS servers include appropriate control routines that are used to determine where in the network a user is located, and then to direct the user to . . . a low level DNS[] server that is close-by.

⁴ The first set of servers are the framework’s hosting servers (also referred to as ghost servers or ghosts). (See ‘645 Patent, col.5 ll.65-67.)

(Id. col.9 ll.40-44, 49-55 (emphasis added).) Akamai confirmed the character of this alternative, distinct DNS at the Markman hearing:

... the Domain Name System described in the patent is, one, something that is established by, set up by, run by the content delivery network. And, second, it is different in that it has intelligence that will direct – that will inform the translation of character strings into IP addresses.⁵

(Hr’g Tr., 59:10-15, May 17, 2007 (emphasis added).) Thus, read in the context of the specification, the meaning of “distinct” and “alternative” describes a domain name system established and controlled by a content delivery network service provider, which includes control routines “different from regular DNS servers like those of the .com domain.” (‘645 Patent, col.9 ll.50-51.)

3. Term 4 (‘645 Patent, Claim 1)⁶

Term	Court’s Construction
... the given name server that receives the DNS query being close to the client local name server as determined by given location information the particular name server that receives the DNS query is selected by the alternative domain name system and is close in Internet terms to the client local name server ...

The parties’ first disagreement concerns whether “close” refers to geographic

⁵ In addition to the intelligence in the top-level DNS servers that determines the location of the user and chooses a particular close-by low-level DNS server to service that user, the specification also describes intelligence in the low-level DNS servers, not present in regular DNS servers, which provides other claimed functionality. (See ‘645 Patent, col.11 ll.53-55 (“The low-level DNS servers monitor the various ghost servers to take into account their loads while translating virtual ghost names into real addresses.”) (emphasis added).)

⁶ The parties reached an agreement as to the construction of several contested terms prior to the Markman hearing and agreed that one other did not require immediate construction. Therefore, Terms 3, 8 and 20 are not addressed by the court here.

distance or Internet distance.⁷ While Limelight argues that the specification only intends to specify Internet distance where explicitly stated, the language of the specification supports the concept of Internet distance generally where it refers to distance.⁸ For example, the Brief Summary of the Invention describes an object of the invention as “to serve Web content efficiently, effectively, and reliably to end users” by “provid[ing] a network architecture that moves content close to the user” to avoid having to “build[] a massive infrastructure to handle the associated traffic.” (Id. col.2 ll.41-42, 49-53 (emphasis added).) “Close,” read in this context, refers to Internet distance. In addition, the description of the preferred embodiment generally references the network or network delays in its measure of distance:

“Several factors may determine where the hosting servers are placed in the network. . . . By studying [network] traffic patterns, the ISP may optimize the server locations for the given traffic profiles.” (Id. col.6 ll.28-34 (emphasis added).)

“[A]ppropriate control routines [] are used to determine where in the network the user is located, and then to direct the user to a . . . server that is close-by.” (Id. col.9 ll.51-54 (emphasis added).)

⁷ A particular DNS or content server may be closer to a user in geographic terms, but delays in routing or high-traffic conditions along a particular route may mean that information can be obtained in less time from a server that is physically located farther away.

⁸ The exception to the general reference to Internet distance in the specification occurs in the Brief Summary of the Invention which asserts “[a]n additional feature [of the inventive framework], the actual content that is replicated at any one geographic location is specifically tailored to viewers in that location.” (‘645 Patent, col.3 ll.7-9.) However, it does not appear that this particular benefit of the invention is claimed in the ‘645 Patent. (See also id. col.10 ll.11-16 (“Alternatively, the user’s location or IP address could be directly encoded into the request sent to the top level DNS.”).)

“Thus, a given top-level DNS server directs the user to a region of the Internet” (Id. col.9 ll.57-59 (emphasis added).)

After determining where in the network the request originated, the top level DNS server redirects the DNS request to a low level DNS server close to the user in the network. (Id. col.10 ll.28-30 (emphasis added).)

In general, the clients are directed to regions in a way that minimizes the overall latency experienced by clients subject to the constraint that no region becomes overloaded.⁹ (Id. col.10 ll.64-67 (emphasis added).)

While Limelight is correct that none of these descriptions explicitly states that closeness refers to Internet distance, it is implied by the emphasized text. (Cf. id. col.10 ll.9-11 (“[T]he routines make the assumption that the user is located near (in the Internet sense) this server.”).)

The second issue concerns whether there is any limitation on how this closeness is determined. Akamai argues that the words of the claim do not limit the determination of closeness, while Limelight asserts that it must be determined by the alternative domain name system. For the reason discussed supra, Term 1, I believe Limelight has the better argument. The specification describes “the present invention” as “manipulat[ing] the DNS system so the name is resolved to one of the ghosts that is near the client.” (Id. col.9 ll.42-44 (emphasis added); see also id. col.3 ll.42-44 (Brief Summary of the Invention) (“[T]he top-level DNS server determines the user’s location in the network”).) As discussed supra, Term 2, the purpose of establishing “an alternative domain name system (DNS), distinct from the Internet domain name system” is to run “appropriate control routines” to “determine where in the network a user is

⁹ Latency refers to the time delay between making a request for Internet content and receiving the requested content.

located.” (Id. Claim 1; id. col.9 ll.52-53.) Read in light of the specification, the invention claims an alternate DNS system that selects a DNS server in response to a user request based on the location of the user.

4. Term 5 ('645 Patent, Claim 1)

Term	Court's Construction
... the alphanumeric string is resolved without reference to a filename for the given object the alphanumeric string is translated into an IP address without reference to the name of the object ...

The parties do not appear to disagree on the explicit limitation in this term requiring resolution of the string without regard to the object name. Limelight, however, argues that the term should be further limited to require the name resolution to resolve to the “Internet Protocol address of the optimal content server.” (See Docket # 71, 26.) Reading the contested term in light of the rest of the claim demonstrates that the claim itself provides explicit limitations on string resolution. Claim 1 requires that the IP address returned by the contested term be associated with one of the content servers associated with the “close” DNS server selected in the previous step, and also that the content server be selected according to a load-sharing algorithm. ('645 Patent, Claim 1.) The word “optimal” does not appear in the specification (or the claims) of the '645 Patent, and adding it as a limitation, where the claim step itself limits the result of the string resolution, would muddy, rather than clarify, an understanding of the claim step.

5. Term 6 ('645 Patent, Claim 1)

Term	Court's Construction
-------------	-----------------------------

... being selected according to a load sharing algorithm enforced across the subset of the set of content servers associated with the given name server being selected by a procedure that distributes requests for objects among a group of content servers in a content delivery network associated with the particular name server to avoid overloading any single content server ...
---	--

The conflict between the parties arises over the meaning of the words “load sharing.” Limelight seeks to limit this term to an active, real-time algorithm which allocates server requests in order to “even out transient load peaks.” (See Docket # 71, 29.) Akamai attempts to distinguish between load sharing and load balancing, arguing the former is broader and encompasses both the latter, which they argue is active, as well as “other load distribution methods.” (Docket # 68, 24.) The specification does not explicitly define (or use) the term “load sharing,” and, as Akamai points out, prior art references “do[] not establish the meaning of these terms to a person of ordinary skill in the art” because they “contain directly conflicting descriptions.” (Docket # 81 Ex. A, 18.)

The specification uses the term “load balancing” to describe a two-step process to allocate requests for objects to various content servers in order to distribute the load; a pre-processing step that allocates objects randomly across potentially available servers, followed by the active instrumentation and adjustment of actual server loads:

According to the present invention, load balancing across the set of hosting servers is achieved in part through a novel technique for distributing the embedded object requests. In particular, each embedded object URL is preferably modified by prepending a virtual server hostname into the URL. . . . This function serves to randomly distribute the embedded objects over a given set of virtual server hostnames.

....

According to the invention, the virtual ghost names may be hashed into real ghost addresses using a table lookup, where the table is continually updated based on network conditions and traffic in such a way to insure load balancing and fault tolerance. . . . The low-level DNS servers monitor the various ghost servers to take into account their loads while translating virtual ghost names into real addresses. This is handled by a software routine that runs on the ghosts and on the low level DNS servers.

(‘645 Patent, col.4 ll.13-24, col.11 ll.23-27, 53-57 (emphasis added) (describing the preprocessing of embedded object alphanumeric strings to randomly distribute object requests across a set of virtual servers, followed by an active step which translates virtual server hostnames into real server IP addresses to avoid overloading any single server).) The specification describes the goal of load sharing not as evening out transient load peaks, but as distributing object requests among a set of servers so that “no server becomes overloaded.” (id. col.11 l.67; accord id. col.11 ll.7-10 (“The local DNS server is responsible for returning the IP address of one of the ghost servers on the network that is close to the user, not overloaded, and most likely to already have the required data.”).) Given the lack of clarity in the prior art and the lack of an explicit definition of load sharing in the specification, a limitation requiring that the load sharing algorithm avoid overloading the content servers comports with the understanding of a person of ordinary skill in the art’s reading of the patent. Phillips v. AWH Corp., 415 F.3d 1303, 1313 (Fed. Cir. 2005) (“[T]he person of ordinary skill in the art is deemed to read the claim term not only in the context of the particular claim in which the disputed term appears, but in the context of the entire patent, including the specification.”).

B. ‘413 Patent Terms in Dispute

1. Term 7 (‘413 Patent, Claims 8, 18, 20)

Term	Court's Construction
... responsive to a DNS query, selecting a given one of the name servers in the content delivery network ... [claims 8, 18]	... in response to a DNS query, the content delivery network's domain name system selects a particular name server ...
... responsive to a DNS query received from a client local name server, selecting a given one of the name servers in the content delivery network ... [claim 20]	... in response to a DNS query received from a client local name server, the content delivery network's domain name system selects a particular name server ...

The crux of the disagreement between the parties here is whether the selection of the particular name server is a result of a DNS lookup, or whether the claim covers any method which results in the selection of a particular name server in response to a DNS query. Limelight argues that there is no support in the specification for any method of choosing a particular name server other than by a DNS lookup. The Brief Summary of the Invention describes the “hosting framework” as including two sets of DNS servers. (‘413 Patent, col. 3 ll.27-32.) “To locate the appropriate hosting servers to use, the top-level DNS server determines the user's location in the network to identify a given low-level DNS server to respond to the request for the embedded object.” (*Id.* col.3 ll.37-41.) In Limelight’s view, this requires the invention’s domain name system to choose the second-level name server.

Akamai counters that, under Limelight’s proposed construction, if the particular name server is selected by a first DNS lookup, the particular name server chosen must therefore be a member of a second DNS level. This, they assert, is in conflict with language earlier in each of the three claims describing the content provider’s domain name system as “having one or more DNS levels.” (‘413 Patent, Claim 1 (emphasis

added).) In addition, they point to other claims that explicitly require a two-level DNS system as evidence that the patent differentiates between claims that require a two-level DNS implementation and claims that do not. Therefore, Akamai argues for a broad, dictionary definition of “selecting” as “making a choice.” (Docket # 68, 28.)

The error in Akamai’s argument is that limiting the selection of a particular name server to the DNS lookup does not necessitate two DNS levels from the client’s perspective. While the embodiment describes a global hosting system containing a two-level DNS system, it notes that “the functionality of the top and low-level servers” may be combined in “a single DNS level.” (’413 Patent, col.5 ll.59-65 (emphasis added).) Thus, the specification supports language claiming a single-level DNS system, but with the requirement that it accomplish the same steps as the described embodiment. For example, in the described embodiment, the top-level DNS server selects a low-level name server and “[re]direct[s] the user to a . . . low-level DNS[] server that is close-by [the user].” (Id. col.9 ll.49-50.) The user’s local name server then makes a second DNS request to this second level of DNS to obtain the object’s IP address.¹⁰

In a single-level DNS embodiment, as suggested by the specification, the user’s local name server would still contact a content delivery provider’s top-level name server to resolve the IP address of a server to serve an object. This name server, however,

¹⁰ (See ’413 Patent, col.10 ll.26-27 (“[T]he ability to redirect [DNS] requests is a standard feature in the DNS system.”); see also Akamai Technologies, Inc. Tutorial – Corrected Version (Docket # 84) Figure 7 (showing the end user’s local domain name server making two separate DNS lookups, steps 2 and 3, to determine the IP address of the Akamai content server).)

would then directly communicate with a particular local name server, based on the user's location, to resolve the server's IP address and return it to the user, rather than require the user to conduct a second lookup. Thus, the user would obtain the IP address of the appropriate ghost server with only a single DNS request, however the selection of a particular name server would still be the result of a DNS lookup by the service provider's DNS system. Such an embodiment would satisfy the claimed "one" level of DNS, yet not be in conflict with Limelight's proposed claim construction.

However, Limelight's proposed requirement that "the content delivery network service provider conducts a DNS lookup to select a single specific name server" is excessively limiting. (Docket # 71, 23.) Each of the three claims containing the term to be construed requires a "content delivery network service provider" to establish a "domain name system (DNS) having authority to resolve the alphanumeric strings in the URLs of the objects identified by the participating content providers" having "one or more DNS levels." ('413 Patent, Claims 8, 18, 20.) As discussed supra, Term 2, the service provider's domain name system includes "special function[s] that [are] different from regular DNS servers like those of the .com domain" and "appropriate control routines" to select a particular name server. (Id. col.9 ll.44-50.) Read in this light, the selection of a particular name server is made by these special functions in the DNS system in response to a DNS query, but not necessarily by a DNS lookup.¹¹ The construction adopted by the court reflects this broader understanding of the term in

¹¹ While a subtle distinction, the court interprets a "DNS lookup" to be the name resolution function provided by normal DNS servers like those of the .com domain.

dispute.

2. Term 9 ('413 Patent, Claims 8, 18, 20)

Term	Court's Construction
... the alphanumeric string is resolved without reference to a filename for the given object the alphanumeric string is translated into an IP address without reference to the name of the object ...

This term to be construed in the '413 Patent is identical to Term 5, supra, in the '645 Patent. Limelight argues that a limitation of an "optimal content server" should be read into these claims. (See Docket # 71, 26.) Again, the remainder of the claim or subsequent dependent claims provide explicit limitations on the string resolution, so the addition of the undefined term, "optimal," is unnecessary. See supra, Term 5.

C. '703 Patent Terms in Dispute

1. Term 10 ('703 Patent, Claim 5)¹²

Term	Court's Construction
... a routine for modifying at least one embedded object URL of a web page a procedure for modifying a Uniform Resource Locator for an object embedded in a page that is accessible in or through the World Wide Web ...

Akamai objects to Limelight's construction that limits "a routine" to a computer program or other automated process. Rather, Akamai argues for a definition of "a routine" as "a procedure" or a "series of steps" that could be performed by either humans or machines. (Docket # 81 Ex. A, 23.)

¹² The term to be construed appears in claim 1 of the '703 Patent. Akamai is asserting that claim 5, which is dependent on claim 1, is infringed by Limelight.

The specification uses the terms “routine,” “process” and “method” to describe the modification of object URLs. In the preferred embodiment, the embedded object URL is “first modified, preferably in an off-line process, to condition the URL to be served by the global hosting servers.” (’703 Patent, col.6 ll.40-44 (emphasis added).) The flowchart “illustrating the preferred method for modifying the object URL” is described and referred to as “[t]he routine.” (Id. col.6 ll.44-47 (emphasis added).)¹³

Limelight asserts that in an earlier case claiming infringement of the same patent, Akamai limited the same term to a computer program. (Docket # 71, 38 (citing Akamai Techs., Inc. v. Digital Island, Inc. (No. 00-cv-11851-RWZ), Trial Exhibit 96).) In that case, Akamai argued “[a] ‘routine’ has its ordinary meaning of a set of instructions, e.g. a computer program or process.” (Id.) “E.g.,” however, denotes an example and is therefore not limiting.¹⁴

While the hash value calculations described in the preferred embodiment would be difficult to calculate manually, simpler embodiments might be more amenable to hand calculation. For example, the specification describes an alternate method for modifying the object URL with a serial number which contains information about the

¹³ Limelight asserts that a flowchart is a “typical representation[] of a computer program.” (Docket # 71, 37.) While flowcharts are often used to represent computational steps, this is not determinative. Flowcharts are used to document human processes as well; the Internal Revenue Service uses flowcharts to illustrate the procedure necessary for a human to figure out various complicated tax scenarios. See, e.g., Chart 1 - Eligibility for the Self-Correction Program (“SCP”) Qualified Plans and 403(b) Plans, http://www.irs.gov/pub/irs-tege/scp_qual_403b_flowchart.pdf (last visited June 14, 2007).

¹⁴ See Black’s Law Dictionary 555 (Bryan A. Garner ed., 8th ed. 2004) (contrasting “e.g.” with “i.e.”).

object. ('703 Patent, col.6 l.65 - col.7 l.29.) Therefore, there is insufficient support in the specification to limit the “routine” which modifies object URLs solely to automated or computer programs.

2. Term 11 ('703 Patent, Claim 15)

Term	Court's Construction
... modifying a URL for the page object modifying a Uniform Resource Locator for an object embedded in a page ...

Limelight similarly seeks to limit this term to “a computer program that modifies an existing Uniform Resource Locator of an embedded object of a web page.” (Docket # 71, 36.) This construction fails not only for the reasons discussed supra, Term 10, but also because claim 15 is a method claim and Limelight’s proposed construction describes a product claim limitation.

3. Term 12 ('703 Patent, Claims 5, 15)¹⁵

Term	Court's Construction
... hostname a name or other identifier that can be translated by a client machine’s local DNS server into the IP address of a content server ...

These two patent claims describe prepending a hostname to the URL for an embedded web page object. ('703 Patent, Claims 5, 15.) As described in the remaining steps of claim 15, in response “to a browser query to resolve the hostname,” the IP address of a particular one of the “set of content servers distinct from the content

¹⁵ The term to be construed appears in claim 15 and claim 1, upon which claim 5 depends.

provider server" is returned to the client's browser. (Id. Claim 15; accord id. Claim 5.)

Thus, the hostname identifies a server on the Internet which the client's DNS server can resolve into an IP address so his or her browser can fetch the object. (See id. col.9 ll.20-28.) The court's construction strikes a balance between Akamai's broad proposed construction and Limelight's narrow one, opting instead to include the functional limitations described in the claims rather than one specific to the current Internet name resolution system.

4. Term 13 ('703 Patent, Claims 5, 15)¹⁶

Term	Court's Construction
... domain.name and path a name or other identifier that defines a network connection to the embedded object at a content provider server and is used to retrieve the object in the absence of a content delivery network ...

Akamai describes the domain name portion of a URL as “an expression that identifies the content provider’s servers.” (Docket # 81 Ex. A, 26.) This is broader than the normal use of the term to describe the address of a particular host or server that can be resolved to establish a network connection with that server.¹⁷ (Cf. Akamai Technologies, Inc. Tutorial – Corrected Version (Docket # 84), 20 (“The first portion of a URL relates to the computer or a group of computers on which a resource may be located. This is the portion of the URL that is resolved by a domain name service to an IP address.”). There is nothing in the specification or claims to indicate that the term is being redefined by the patent to be broader than its normal usage. At the Markman hearing, Akamai conceded that the domain name “has to be something recognized by the Domain Name System” (Hr’g Tr., 187:25 - 188:3, May 17, 2007.)

¹⁶ The term to be construed appears in claim 15 and claim 1, upon which claim 5 depends.

¹⁷ To support its construction of the term “domain name,” Akamai cites to the glossary definition of the term “domain” alone at an obscure ISP site. (See Docket # 68, 37 n.13). Not only is “domain” a different term than “domain name,” but the popular dictionary definition of “domain name” eschewed by Akamai for its proposed construction is very close to Limelight’s proposed construction. See Webster’s II New College Dictionary 337 (2001) (defining “domain name” as “[a] series of alphanumeric strings separated by periods, such as www.hmco.com, that is the address of a computer network connection and that identifies the owner of the address.”).

The path contains the name and location of the file, e.g., the object, on the server. (Docket # 81 Ex. A, 26.) Together, the domain name and path form the URL which provides a client computer with the information necessary to identify the server and retrieve the object from that server on the network. Both claims describe “modifying” the URL of an embedded object and claim 15 describes the domain name and path as “content provider-supplied.” (‘703 Patent, Claims 5, 15 (emphasis added).) Read in light of the specification explaining that “according to the present invention, a virtual server hostname is prepended into the URL for a given embedded object,” the terms domain name and path in claims 5 and 15 would be understood by one skilled in the art to refer to the URL used to retrieve the embedded object in the absence of a content delivery network. (Id. col.7 ll.24-28 (emphasis added).)

5. Terms 14 and 15 (‘703 Patent, Claim 5)¹⁸

Term	Court’s Construction
... at least one first level name server that provides a first level domain name service (DNS) resolution ... [Term 14]	... a computer or program running on a computer in the hosting framework that receives a request to resolve a name or other identifier into an IP address, and returns the IP address of a name server or servers ...
... and at least one second level name server that provides a second level domain name service (DNS) resolution ... [Term 15]	... a computer or program running on a computer in the hosting framework that receives a request to resolve a name or other identifier into an IP address, and returns the IP address of a content server or servers ...

Akamai argues that a proper construction of these terms should include the

¹⁸ The terms to be construed appear in claim 1, upon which claim 5 depends.

regular Internet DNS root¹⁹ and top-level (e.g. “.com”) name servers, while Limelight insists that the patent limits the invention to name servers within the “distributed hosting framework” cited in the claim preamble. (’703 Patent, Claim 1.) Akamai bases its argument on an appellate decision in a prior case, Akamai Techs. v. Cable & Wireless Internet Servs., 344 F.3d 1186 (Fed. Cir. 2003) (“C&W”). Akamai, however, misreads that decision. The issue addressed by the Federal Circuit in C&W was “a single point of contention--the placement of the load balancing software at either the DNS servers or the origin server.” Id. at 1193. The court found that claim 1 of the ’703 Patent did not require the load balancing software to be located at the DNS server, and therefore was anticipated by an earlier C&W patent. Id. at 1194.

In attempting to save claim 3, which added only a redundant second-level name server to the limitations of claim 1, Akamai argued that the earlier patent did not disclose a hierarchical DNS. Id. The Federal Circuit considered this argument but concluded:

This additional argument, however, fails to address C&W’s contention that hierarchical DNS is inherent in any Internet system. Indeed, C&W proffered documentary evidence and testimony at trial that redundant domain name servers are inherent in any Internet-based application. See Dayco, 329 F.3d at 1369.

Id. at 1195 (emphasis added). Contrary to Akamai’s contention, this language does not construe the terms of claim 1 concerning limitations on the location of the claimed

¹⁹ The root servers are a worldwide set of domain name servers at fixed IP addresses which can be queried to provide the location of the next level of lower-level DNS servers supporting the top-level domains such as .com, .org, etc. They provide a known starting place at the top of the domain name hierarchy for a client’s local name server to begin in order to resolve a hostname. (See Docket # 84, 14.)

name servers. Indeed, the Federal Circuit had no reason to examine claim 1 further, having already determined it invalid. Rather, it holds that, given an invalid claim 1 anticipated by an earlier patent, the additional limitation of a redundant name server in claim 3 does not sufficiently limit claim 1 to create a valid claim.

A plain reading of claim 1 in the '703 Patent describes the two-level DNS service as part of "[a] distributed hosting framework operative in a computer network . . . the framework comprising: . . . at least one first level name server . . . and at least one second level name server" ('703 Patent, Claim 1.) The specification supports a construction of this language that limits the location of the name servers to the claimed invention's hosting framework:

The hosting framework of the present invention comprises a set of servers operating in a distributed manner. . . . In particular, the framework includes a second set of servers (or server resources) that are configured to provide top level Domain Name Service (DNS). In addition, the framework also includes a third set of servers (or server resources) that are configured to provide low level DNS functionality. (Id. col.3 ll.4-5, 19-24 (Summary of the Invention) (emphasis added).)

[T]he global hosting system 35 comprises three (3) basic types of servers (or server resources): hosting servers (sometimes called ghosts) 36, top-level DNS servers 38, and low-level DNS servers 40. (Id. col.3 ll.51-54 (Summary of the Invention) (emphasis added).)

Step 3: As previously described, preferably there are two types of DNS servers in the inventive system: top-level and low-level. The top level DNS servers 38 for ghosting.com have a special function that is different from regular DNS servers like those of the .com domain. (Id. col.9 ll.31-35 (emphasis added) (distinguishing the DNS servers in the inventive framework from those of the Internet root and top-level servers).)

Thus, both the claim and the specification limit the invention to a global framework containing two levels of DNS servers different from the DNS servers

providing root and top-level name resolution.

6. Terms 16 and 17 ('703 Patent, Claims 5, 15, 19, 34)²⁰

Term	Court's Construction
... given one of the content servers ... [Claims 5, 34]	... a particular one of the content servers distinct from the content provider server described previously in the claim ...
... given one of the set of content servers ... [Claim 15]	
... given content server ... [Claim 19]	

The parties both agree that a single construction is sufficient for all three claims terms.²¹ Limelight seeks to add the limitation that the given content server be “a single, optimal content server.” (Docket # 71, 50.) In claim 5, the contested term appears in a limitation of claim 1 requiring “the embedded object [to be] served from a given one of the content servers as identified by the first level and second level name servers.” ('703 Patent, Claim 1; accord id. Claim 15.) In C&W, the Federal Circuit refused to read any requirement for a load balancing mechanism into the word “identified” in this claim. C&W, 344 F.3d at 1193-94 (“[The plain meaning of the claim language] simply requires the embedded object to be served from ‘the content servers as identified by

²⁰ The term to be construed in claim 5 appears in claim 1, upon which claim 5 depends.

²¹ The four claims use three similar terms with slightly different wording. The parties have consolidated these into two separate contested terms in their briefs; however, they each argue for the same (but different between the parties) construction for all three.

the first level and second level name servers.”). Similarly, it would be inappropriate to read a requirement for an “optimal” server into it as well.

Additional language in claim 34 limits the “given one of the content servers” to a server which is “within the given region that is likely to host the embedded object and that is not overloaded.” (’703 Patent, Claim 34.) As discussed supra, Term 5, adding a requirement that the server be “optimal” is unnecessary and confusing, as the claim already provides a specific limitation, consistent with the specification, as to which content server’s IP address should be returned to the client. Similarly, while independent claim 19 provides no particular limitations on which given content server serves the embedded object, subsequent dependent claims 20 through 22 limit the server selected in ways that would be redundant with Limelight’s construction.

Akamai’s proposed construction, however, reads out the word “given” from the claims. It is appropriate to construe the term as a referring to a “particular” server, to make clear that a selection is made.

7. Term 18 (’703 Patent, Claim 5)

Term	Court’s Construction
... the second level name server includes a load balancing mechanism that balances loads across a subset of the set of servers.	... a mechanism in the second level name server monitors the loads on a group of content servers in a content delivery network and distributes requests for objects among them to avoid overloading any single content server.

As discussed supra, Term 6, the specification uses the term “load balancing” to describe a two-step process to allocate requests to various servers in order to

distribute the load, a preprocessing step on embedded object alphanumeric strings to randomly distribute object requests across a set of virtual servers, followed by an active step which translates virtual server hostnames into real server IP addresses to avoid overloading any single server. (See id. col.3 l.66 - col.4 l.4, col.11 ll.6-10, 35-39.) This second step includes active instrumentation and adjustment of server loads. (Id.) Limelight seeks to have this limitation included in the claim construction. (See Docket # 71, 29.) Akamai concedes that load balancing “requires something more than load sharing.” (Docket # 81 Ex. A, 31.) Given the construction of load sharing adopted by the court in Term 6, the court construes the “something more” in load balancing to be the additional limitation of actively monitoring the loading on the content servers.

8. Term 19 ('703 Patent, Claim 17)

Term	Court's Construction
... prepending given data to a content provider-supplied URL to generate an alternate resource locator (ARL) generating an alternative resource locator (ARL) by adding a name or other identifier that can be translated by a domain name system into the IP address of a content server to the beginning of the URL of an embedded object supplied by a content provider ...

Claim 17 describes several steps comprising a “content delivery method” which, inter alia, requires modifying the URL of an embedded object to generate an alternative resource locator (“ARL”). ('703 Patent, Claim 17.) This ARL is then “resolv[ed] to identify a content server in [a domain other than a content provider’s] domain.” (Id.) Thus, the language of the remaining steps in the claim limits “given data” to a string

resolvable to the address of a content server in the content delivery system.

June 29, 2007

DATE

/s/Rya W. Zobel

RYA W. ZOBEL

UNITED STATES DISTRICT JUDGE



US007103645B2

(12) **United States Patent**
Leighton et al.

(10) **Patent No.:** US 7,103,645 B2
(45) **Date of Patent:** Sep. 5, 2006

(54) **METHOD AND SYSTEM FOR PROVIDING CONTENT DELIVERY TO A SET OF PARTICIPATING CONTENT PROVIDERS**

(75) **Inventors:** F. Thomson Leighton, Newtonville, MA (US); Daniel M. Lewin, Cambridge, MA (US)

(73) **Assignee:** Massachusetts Institute of Technology, Cambridge, MA (US)

(* **Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 73 days.

(21) **Appl. No.:** 10/417,607

(22) **Filed:** Apr. 17, 2003

(65) **Prior Publication Data**
US 2003/0191822 A1 Oct. 9, 2003

Related U.S. Application Data

(63) Continuation of application No. 09/604,878, filed on Jun. 28, 2000, now Pat. No. 6,553,413, which is a continuation of application No. 09/314,863, filed on May 19, 1999, now Pat. No. 6,108,703.

(60) Provisional application No. 60/092,710, filed on Jul. 14, 1998.

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** 709/219; 709/229; 709/217

(58) **Field of Classification Search** 709/201, 709/217, 219, 229

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

- 4,922,417 A 5/1990 Churm et al.
- 5,136,716 A * 8/1992 Harvey et al. 709/228
- 5,287,499 A 2/1994 Nemes
- 5,341,477 A 8/1994 Pitkin et al.

- 5,542,087 A 7/1996 Neimat et al.
- 5,638,443 A 6/1997 Stefik et al.
- 5,646,676 A 7/1997 Dewkett et al.
- 5,715,453 A 2/1998 Stewart
- 5,740,423 A 4/1998 Logan et al.
- 5,751,961 A 5/1998 Smyk
- 5,761,507 A 6/1998 Govett

(Continued)

FOREIGN PATENT DOCUMENTS

CA 2202572 10/1998

(Continued)

OTHER PUBLICATIONS

Stanford-Clark, "Atlanta Olympics WOMplex," AIXpert Magazine, Mar. 1997.

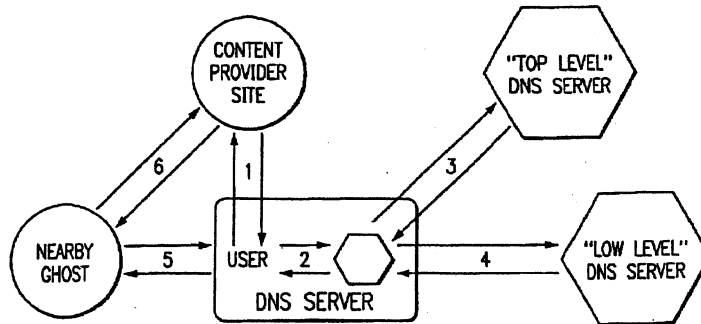
(Continued)

Primary Examiner—Rupal Dharja
Assistant Examiner—Kristie D. Shingles
(74) *Attorney, Agent, or Firm*—David H. Judson

(57) **ABSTRACT**

The present invention is a network architecture or framework that supports hosting and content distribution on a truly global scale. The inventive framework allows a Content Provider to replicate and serve its most popular content at an unlimited number of points throughout the world. The inventive framework comprises a set of servers operating in a distributed manner. The actual content to be served is preferably supported on a set of hosting servers (sometimes referred to as ghost servers). This content comprises HTML page objects that, conventionally, are served from a Content Provider site. In accordance with the invention, however, a base HTML document portion of a Web page is served from the Content Provider's site while one or more embedded objects for the page are served from the hosting servers, preferably, those hosting servers near the client machine. By serving the base HTML document from the Content Provider's site, the Content Provider maintains control over the content.

18 Claims, 2 Drawing Sheets



U.S. PATENT DOCUMENTS

5,774,660	A	6/1998	Brendel et al.	
5,774,668	A *	6/1998	Choquier et al.	709/223
5,777,989	A *	7/1998	McGarvey	370/254
5,802,291	A	9/1998	Balick et al.	
5,815,664	A *	9/1998	Asano	709/227
5,832,506	A	11/1998	Kuzma	
5,856,974	A	1/1999	Gervais et al.	
5,870,559	A	2/1999	Leshem et al.	
5,878,212	A	3/1999	Civanlar et al.	
5,884,038	A	3/1999	Kapoor	
5,894,554	A	4/1999	Lowery et al.	
5,903,723	A	5/1999	Beck et al.	
5,919,247	A	7/1999	Van Hoff et al.	
5,920,701	A *	7/1999	Miller et al.	709/228
5,933,832	A	8/1999	Suzuoka et al.	
5,945,989	A	8/1999	Freishtat et al.	
5,956,716	A *	9/1999	Kenner et al.	707/10
5,961,596	A	10/1999	Takubo et al.	
5,991,809	A	11/1999	Kriegsman	
6,003,030	A	12/1999	Kenner et al.	
6,006,264	A *	12/1999	Colby et al.	709/226
6,052,718	A *	4/2000	Gifford	709/219
6,112,239	A *	8/2000	Kenner et al.	709/224
6,115,752	A *	9/2000	Chauhan	709/241
6,119,143	A	9/2000	Dias et al.	
6,134,583	A *	10/2000	Herriot	709/217
6,144,996	A *	11/2000	Starnes et al.	709/217
6,151,624	A *	11/2000	Teare et al.	709/217
6,154,738	A *	11/2000	Call	707/4
6,154,744	A *	11/2000	Kenner et al.	707/10
6,167,427	A *	12/2000	Rabinovich et al.	709/201
6,178,160	B1	1/2001	Bolton et al.	
6,181,867	B1	1/2001	Kenner et al.	
6,185,598	B1 *	2/2001	Farber et al.	709/200
6,185,619	B1	2/2001	Joffe et al.	
6,230,196	B1	5/2001	Guenther et al.	
6,243,760	B1 *	6/2001	Armbruster et al.	709/243
6,256,675	B1	7/2001	Rabinovich	
6,266,699	B1 *	7/2001	Sevcik	709/229
6,269,394	B1	7/2001	Kenner et al.	
6,282,569	B1 *	8/2001	Wallis et al.	709/224
6,286,045	B1 *	9/2001	Griffiths et al.	709/224
6,311,214	B1 *	10/2001	Rhoads	709/217
6,314,565	B1	11/2001	Kenner et al.	
6,332,195	B1 *	12/2001	Green et al.	713/201
6,347,085	B1 *	2/2002	Kelly	370/352
6,360,256	B1 *	3/2002	Lim	709/223
6,370,571	B1	4/2002	Medin, Jr.	
6,442,549	B1 *	8/2002	Schneider	707/10
6,502,125	B1	12/2002	Kenner et al.	
6,665,706	B1	12/2003	Kenner et al.	
6,973,485	B1 *	12/2005	Ebata et al.	709/219
7,047,300	B1 *	5/2006	Oehrke et al.	709/226

FOREIGN PATENT DOCUMENTS

EP	0817 444	A2	1/1998
EP	0817444	A2 *	7/1998
EP	0865 180	A2	9/1998
WO	WO9804985		2/1998

OTHER PUBLICATIONS

Goldszmidt, et al., "Load Distribution for Scalable Web Servers: Summer Olympics 1996—A Case Study," published in the following journal: Proceedings of the 8th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, Sydney, Australia, Oct. 1997.

Berners-Lee et al., RFC 1738—Uniform Resource Locators, Dec. 1994.

Postel, RFC 1591—Domain Name System Structure and Delegation, Mar. 1994.

Mockapetris et al.—Development of the Domain Name System, Proceedings of SIGCOMM '88 Computer Communications Review, vol. 18, No. 4, Aug. 1988.

Wessels, Intelligent Caching for World-Wide Web Objects, Masters Thesis, University of Colorado, 1995.

Smith, What can Archives offer the World Wide Web? Mar. 22, 1994.

Albitz et al., How Does DNS Work? Chapter 2, DNS and BIND, O'Reilly & Associates, Inc. 1992, pp. 13-38.

Overview of the Cisco Distributed Director 2500 Series, actual publication date unknown, but believed to be in 1997.

Crovella et al.—Dynamic server selection in the Internet, 3rd IEEE Workshop on the Arch. and Implementation of High Performance Computer Sys. '95, pp. 158-162, Aug. 1995.

Bhattacharjee et al.—Application-layer anycasting, Proceedings of the IEEE INFOCOM '97, 1997.

Fei et al., A novel server selection technique for improving the response time of a replicated service, Proceedings of the IEEE INFOCOM '98, Mar. 1998.

Carter et al., Server selection using dynamic path characterization in Wide-Area Networks, IEEE INFOCOM '97, 1997.

Guyton et al., Locating nearby copies of replicated Internet servers, Proceedings of ACM SIG/COMM '95, pp. 288-298, 1995.

Seltzer et al., The Case for Geographical Push Caching, Proceedings of the 1995 Workshop on Hot Operating Systems, 1995.

Bestavros, et al., Server-Initiated Document Dissemination for the WWW, IEEE Data Engineering Bulletin 19, pp. 3-11, 1996.

Carter et al., Dynamic server selection using bandwidth probing in wide-area networks, Technical Report BU-CS-96-007, Boston University, Mar. 1996.

Braun et al., Web traffic characterization: an assessment of the impact of caching documents from NCSA's web server, Proceedings of the 2d Int'l WWW Conference, Sep. 1994.

Baker et al., Distributed Cooperative Web servers, Computer Networks, Elsevier Science Publishers, BV, vol. 31, No. 11-16, pp. 1215-1229, May 17, 1999.

Karger et al., Web caching with consistent hashing, Computer Networks, Elsevier Science Publishers, BV, vol. 31, No. 11-16, pp. 1203-1213, May 17, 1999.

"Cisco Distributed Director," http://www.cisco.com/warp/public/751/distdir/dd_wp.htm, posted Feb. 21, 1997.

"How to Cost-Effectively Scale Web Servers," <http://www.cisco.com/warp/public/784/5.html>, posted Nov. 12, 1996.

"Ibnamed, a load balancing name server written in Perl," <http://www.stanford.edu/~schemers/doc/ibnamed/ibnamed.html>, Jan. 19, 1995.

"Exporting Web Server Final Report," http://www.cs.technion.ac.il/Labs/Loon/projects/spring97/project4/final_report.html, Spring 1997.

Jeffrey et al., Proxy-Sharing Proxy Servers, IEEE, pp. 116-119, 1996.

Excerpts from www.sandpiper.com Web site, Nov. 27, 1999 (14 pages).

Luotonen et al., World-Wide Web Proxies, CERN, Apr. 1994.

Oguchi et al., A Study of Caching Proxy Mechanisms Realized on Wide Area Distributed Networks, High Performance Distributed Computing, 5th Int'l Symposium, pp. 443-449, 1996.

Kwan et al., NCSA's World Wide Web Server: Design and Performance, IEEE, pp. 68-74, Nov. 1995.

Malpani et al., Making World Wide Web Caching Servers Cooperate, <http://bmerc.berkeley.edu/papers/1995/138/paper-59.html>, 1995.

Ross, Hash Routing for Collection of Shared Web Caches, IEEE Network, pp. 37-44, Nov./Dec. 1997.

"Reverse Proxy Content Re-mapper for Netscape Proxy 2.52 & 2.53," <http://help.netscape.com/products/server/proxy/documentation/rpMapper.html>, Nov. 1997.

"Super Proxy Script—How to make distributed proxy servers by URL hashing," Sharp, <http://naragw.sharp.co.jp/sps/> 1996-2000.

Shaw, A Low Latency, High Throughput Web Service Using Internet-Wide Replication, Department of Computer Science, Johns Hopkins University, Aug. 1998.

- Amir et al., Seamlessly Selecting the Best Copy From Internet-Wide Replicated Web Servers, Department of Computer Science, Johns Hopkins University, Jun. 1998.
- Beavan, Web Life They're Watching You, Esquire, pp. 104-105, Aug. 1997.
- Bestavros, Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time In Distributed Info. Sys., Proceedings of the ICDE '96, Mar. 1996.
- Carter et al., Universal Classes of Hash Functions, Journal of Computer and Systems Sciences, vol. 18, No. 2, pp. 143-154, Apr. 1979.
- Chankhunthod, et al., A Hierarchical Internet Object Cache, USENIX Proceedings, pp. 153-163, Jan. 1996.
- Cormen et al., Introduction to Algorithms, The MIT Press, pp. 219-243, 991-993, 1994.
- Deering et al., Multicast Routing in Datagram Internetworks and Extended LANs, ACM Transactions on Computer Systems, vol. 8, No. 2, pp. 85-110, May 1990.
- Devine, Design and Implementation of DDH: A Dist. Dynamic Hashing Algorithm, Proc. of 4th Int'l Conf. on Foundations of Data Orgs and Algorithms, pp. 101-114, 1993.
- Grigni et al., Tight Bounds on Minimum Broadcasts Networks, SIAM Journal of Discrete Mathematics, vol. 4, No. 2, pp. 207-222, May 1991.
- Gwertzman et al., World-Wide Web Cache Consistency, Proceedings of the 1996 USENIX Technical Conference, Jan. 1996.
- Feeley et al., Implementing Global Memory Management in a Workstation Cluster, Proceedings of the 15th ACM Symposium on Operating Systems Principles, pp. 201-212, 1995.
- Floyd et al., A Reliable Multicast Framework for Lightweight Sessions and Application Level Framing, Proceedings of ACM SIGCOMM'95, pp. 342-356, 1995.
- Fredman et al., Storing a Sparse Table with $O(1)$ Worst Case Access Time, Journal of the Ass'n. for Computer Machinery, vol. 31, No. 3, pp. 538-544, Jul. 1984.
- Karger et al., Consistent Hashing and Random Trees, Proceedings of the 29th ACM Symposium on Theory of Computing, pp. 654-663, May 1997.
- Litwin et al., LH-A Scaleable, Distributed Data Structure, ACM Transactions on Database Systems, vol. 21, No. 4, pp. 480-525, Dec. 1996.
- Naor et al., The Load, Capacity and Availability of Quorum Systems, Proceedings of the 35th IEEE Symposium on Foundations of Computer Science, pp. 214-225, Nov. 1994.
- Nisam, Pseudorandom Generators for Space-Bounded Computation, Proceedings of the 22nd ACM Symposium on Theory of Computing, pp. 204-212, May 1990.
- Palmer et al., Fido, A Cache That Learns To Fetch, Proceedings of the 17th Int'l Conf. on Very Large Data Bases, pp. 255-264, Sep. 1991.
- Panigraphy, Relieving Hot Spots on the World Wide Web, MIT, pp. 1-66, Jun. 1997.
- Peleg et al., The Availability of Quorum Systems, Information and Computation 123, pp. 210-223, 1995.
- Plaxton et al., Fast Fault-Tolerant Concurrent Access to Shared Objects, Proceedings of 37th IEEE Symposium on Foundations of Computer Science, pp. 570-579, 1996.
- Rabin, Efficient Dispersal of Information for Security, Load Balancing and Fault Tolerance, Journal of the ACM, vol. 36, No. 2, pp. 335-348, Apr. 1989.
- Ravi, Rapid Rumor Ramification: Approximating the Minimum Broadcast Time, Proceedings of the 35th IEEE Symposium on Foundations of Computer Science, pp. 202-213, Nov. 1994.
- Schmidt, Chernoff-Hoeffding Bounds for Applications with Limited Independence, Proceedings of the 4th ACS-SIAM Symposium on Discrete Algorithms, pp. 331-340, 1993.
- Tarjan et al., Storing a Sparse Table, Communications of the ACM, vol. 22, No. 11, pp. 606-611, Nov. 1979.
- Vitter et al., Optimal Prefetching via Data Compression, Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science, pp. 121-130, Nov. 1991.
- Wegman et al., New Hash Functions and Their Use In Authentication and Set Equality, Journal of Computer and System Sciences, vol. 22, pp. 265-279, Jun. 1981.
- Yao, Should Tables be Stored, Journal of the ACM, vol. 28, No. 3, pp. 615-628, Jul. 1981.

* cited by examiner

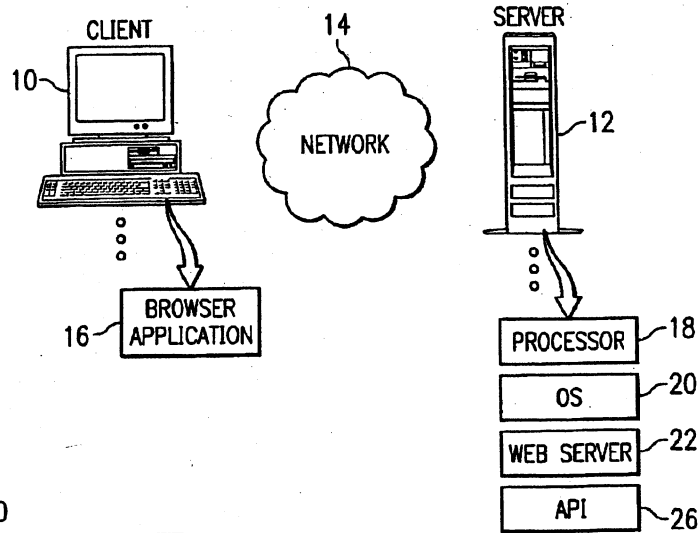


FIG. 1

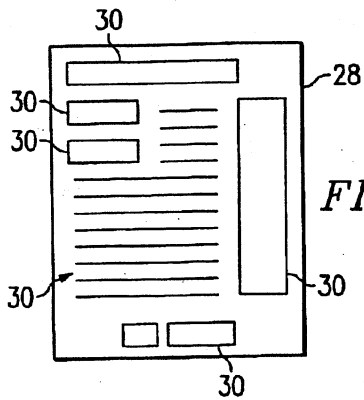


FIG. 2

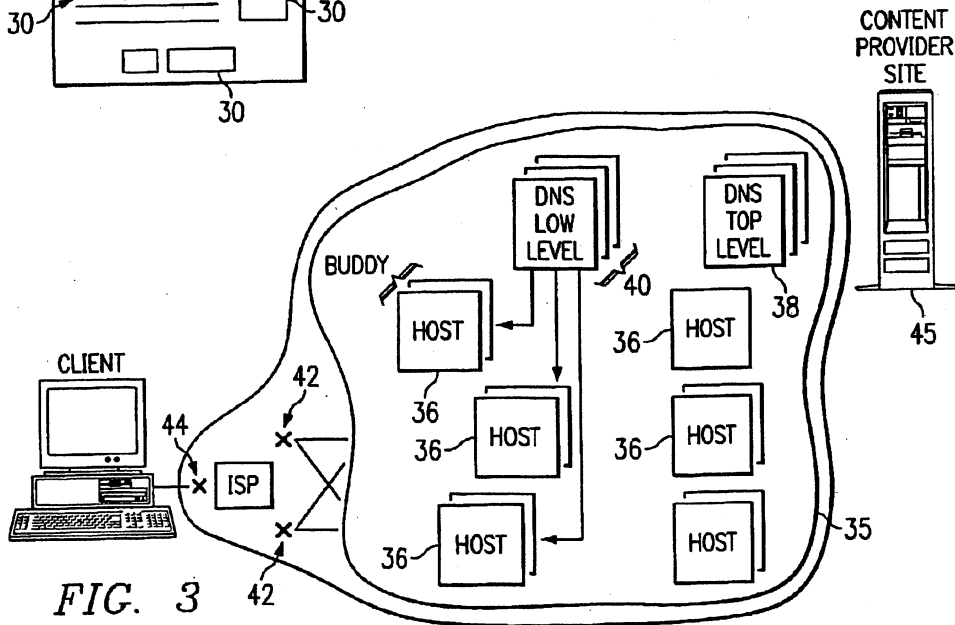


FIG. 3

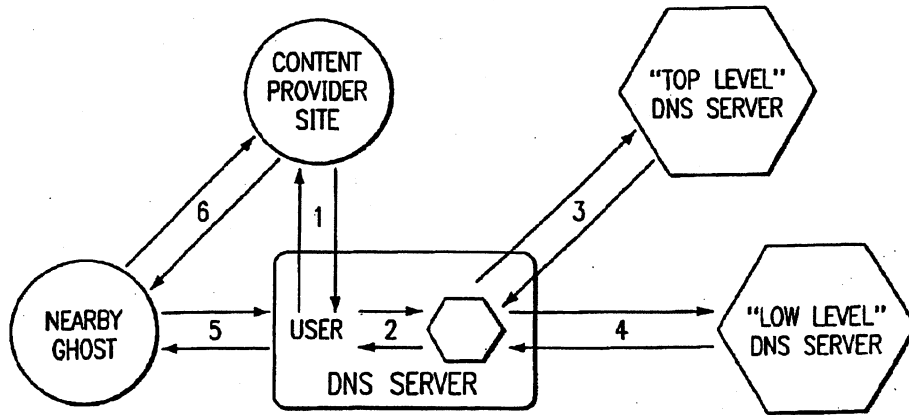
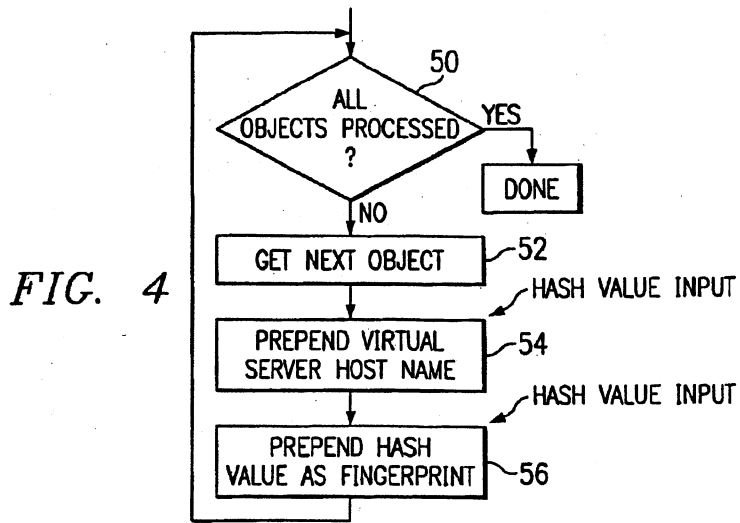


FIG. 5

1

METHOD AND SYSTEM FOR PROVIDING CONTENT DELIVERY TO A SET OF PARTICIPATING CONTENT PROVIDERS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation of prior application Ser. No. 09/604,878, filed Jun. 28, 2000, now U.S. Pat. No. 6,553,413, which application was a continuation of prior application Ser. No. 09/314,863, filed May 19, 1999, now U.S. Pat. No. 6,108,703, which application was based on and claimed priority from Provisional Application No. 60/092,710, filed Jul. 14, 1998.

BACKGROUND OF THE INVENTION

1. Technical Field

This invention relates generally to information retrieval in a computer network. More particularly, the invention relates to a novel method of hosting and distributing content on the Internet that addresses the problems of Internet Service Providers (ISPs) and Internet Content Providers.

2. Description of the Related Art

The World Wide Web is the Internet's multimedia information retrieval system. In the Web environment, client machines effect transactions to Web servers using the Hypertext Transfer Protocol (HTTP), which is a known application protocol providing users access to files (e.g., text, graphics, images, sound, video, etc.) using a standard page description language known as Hypertext Markup Language (HTML). HTML provides basic document formatting and allows the developer to specify "links" to other servers and files. In the Internet paradigm, a network path to a server is identified by a so-called Uniform Resource Locator (URL) having a special syntax for defining a network connection. Use of an HTML-compatible browser (e.g., Netscape Navigator or Microsoft Internet Explorer) at a client machine involves specification of a link via the URL. In response, the client makes a request to the server identified in the link and, in return, receives a document or other object formatted according to HTML. A collection of documents supported on a Web server is sometimes referred to as a Web site.

It is well known in the prior art for a Web site to mirror its content at another server. Indeed, at present, the only method for a Content Provider to place its content closer to its readers is to build copies of its Web site on machines that are located at Web hosting farms in different locations domestically and internationally. These copies of Web sites are known as mirror sites. Unfortunately, mirror sites place unnecessary economic and operational burdens on Content Providers, and they do not offer economies of scale. Economically, the overall cost to a Content Provider with one primary site and one mirror site is more than twice the cost of a single primary site. This additional cost is the result of two factors: (1) the Content Provider must contract with a separate hosting facility for each mirror site, and (2) the Content Provider must incur additional overhead expenses associated with keeping the mirror sites synchronized.

In an effort to address problems associated with mirroring, companies such as Cisco, Resonate, Bright Tiger, F5 Labs and Alteon, are developing software and hardware that will help keep mirror sites synchronized and load balanced. Although these mechanisms are helpful to the Content Provider, they fail to address the underlying problem of scalability. Even if a Content Provider is willing to incur the

2

costs associated with mirroring, the technology itself will not scale beyond a few (i.e., less than 10) Web sites.

In addition to these economic and scalability issues, mirroring also entails operational difficulties. A Content Provider that uses a mirror site must not only lease and manage physical space in distant locations, but it must also buy and maintain the software or hardware that synchronizes and load balances the sites. Current solutions require Content Providers to supply personnel, technology and other items necessary to maintain multiple Web sites. In summary, mirroring requires Content Providers to waste economic and other resources on functions that are not relevant to their core business of creating content.

Moreover, Content Providers also desire to retain control of their content. Today, some ISPs are installing caching hardware that interrupts the link between the Content Provider and the end-user. The effect of such caching can produce devastating results to the Content Provider, including (1) preventing the Content Provider from obtaining accurate hit counts on its Web pages (thereby decreasing revenue from advertisers), (2) preventing the Content Provider from tailoring content and advertising to specific audiences (which severely limits the effectiveness of the Content Provider's Web page), and (3) providing outdated information to its customers (which can lead to a frustrated and angry end user).

There remains a significant need in the art to provide a decentralized hosting solution that enables users to obtain Internet content on a more efficient basis (i.e., without burdening network resources unnecessarily) and that likewise enables the Content Provider to maintain control over its content.

The present invention solves these and other problems associated with the prior art.

BRIEF SUMMARY OF THE INVENTION

It is a general object of the present invention to provide a computer network comprising a large number of widely deployed Internet servers that form an organic, massively fault-tolerant infrastructure designed to serve Web content efficiently, effectively, and reliably to end users.

Another more general object of the present invention is to provide a fundamentally new and better method to distribute Web-based content. The inventive architecture provides a method for intelligently routing and replicating content over a large network of distributed servers, preferably with no centralized control.

Another object of the present invention is to provide a network architecture that moves content close to the user. The inventive architecture allows Web sites to develop large audiences without worrying about building a massive infrastructure to handle the associated traffic.

Still another object of the present invention is to provide a fault-tolerant network for distributing Web content. The network architecture is used to speed-up the delivery of richer Web pages, and it allows Content Providers with large audiences to serve them reliably and economically, preferably from servers located close to end users.

A further feature of the present invention is the ability to distribute and manage content over a large network without disrupting the Content Provider's direct relationship with the end user.

Yet another feature of the present invention is to provide a distributed scalable infrastructure for the Internet that shifts the burden of Web content distribution from the

Content Provider to a network of preferably hundreds of hosting servers deployed, for example, on a global basis.

In general, the present invention is a network architecture that supports hosting on a truly global scale. The inventive framework allows a Content Provider to replicate its most popular content at an unlimited number of points throughout the world. As an additional feature, the actual content that is replicated at any one geographic location is specifically tailored to viewers in that location. Moreover, content is automatically sent to the location where it is requested, without any effort or overhead on the part of a Content Provider.

It is thus a more general object of this invention to provide a global hosting framework to enable Content Providers to retain control of their content. The hosting framework of the present invention comprises a set of servers operating in a distributed manner. The actual content to be served is preferably supported on a set of hosting servers (sometimes referred to as ghost servers). This content comprises HTML page objects that, conventionally, are served from a Content Provider site. In accordance with the invention, however, a base HTML document portion of a Web page is served from the Content Provider's site while one or more embedded objects for the page are served from the hosting servers, preferably, those hosting servers nearest the client machine. By serving the base HTML document from the Content Provider's site, the Content Provider maintains control over the content.

The determination of which hosting server to use to serve a given embedded object is effected by other resources in the hosting framework. In particular, the framework includes a second set of servers (or server resources) that are configured to provide top level Domain Name Service (DNS). In addition, the framework also includes a third set of servers (or server resources) that are configured to provide low level DNS functionality.

When a client machine issues an HTTP request to the Web site for a given Web page, the base HTML document is served from the Web site as previously noted. Embedded objects for the page preferably are served from particular hosting servers identified by the top- and low-level DNS servers. To locate the appropriate hosting servers to use, the top-level DNS server determines the user's location in the network to identify a given low-level DNS server to respond to the request for the embedded object. The top-level DNS server then redirects the request to the identified low-level DNS server that, in turn, resolves the request into an IP address for the given hosting server that serves the object back to the client.

More generally, it is possible (and, in some cases, desirable) to have a hierarchy of DNS servers that consisting of several levels. The lower one moves in the hierarchy, the closer one gets to the best region.

A further aspect of the invention is a means by which content can be distributed and replicated through a collection of servers so that the use of memory is optimized subject to the constraints that there are a sufficient number of copies of any object to satisfy the demand, the copies of objects are spread so that no server becomes overloaded, copies tend to be located on the same servers as time moves forward, and copies are located in regions close to the clients that are requesting them. Thus, servers operating within the framework do not keep copies of all of the content database. Rather, given servers keep copies of a minimal amount of data so that the entire system provides the required level of service.

This aspect of the invention allows the hosting scheme to be far more efficient than schemes that cache everything everywhere, or that cache objects only in pre-specified locations.

The global hosting framework is fault tolerant at each level of operation. In particular, the top level DNS server returns a list of low-level DNS servers that may be used by the client to service the request for the embedded object. Likewise, each hosting server preferably includes a buddy server that is used to assume the hosting responsibilities of its associated hosting server in the event of a failure condition.

According to the present invention, load balancing across the set of hosting servers is achieved in part through a novel technique for distributing the embedded object requests. In particular, each embedded object URL is preferably modified by prepending a virtual server hostname into the URL. More generally, the virtual server hostname is inserted into the URL. Preferably, the virtual server hostname includes a value (sometimes referred to as a serial number) generated by applying a given hash function to the URL or by encoding given information about the object into the value. This function serves to randomly distribute the embedded objects over a given set of virtual server hostnames. In addition, a given fingerprint value for the embedded object is generated by applying a given hash function to the embedded object itself. This given value serves as a fingerprint that identifies whether the embedded object has been modified. Preferably, the functions used to generate the values (i.e., for the virtual server hostname and the fingerprint) are applied to a given Web page in an off-line process. Thus, when an HTTP request for the page is received, the base HTML document is served by the Web site and some portion of the page's embedded objects are served from the hosting servers near (although not necessarily the closest) to the client machine that initiated the request.

The foregoing has outlined some of the more pertinent objects and features of the present invention. These objects should be construed to be merely illustrative of some of the more prominent features and applications of the invention. Many other beneficial results can be attained by applying the disclosed invention in a different manner or modifying the invention as will be described. Accordingly, other objects and a fuller understanding of the invention may be had by referring to the following Detailed Description of the Preferred Embodiment.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference should be made to the following Detailed Description taken in connection with the accompanying drawings in which:

FIG. 1 is a representative system in which the present invention is implemented;

FIG. 2 is a simplified representation of a markup language document illustrating the base document and a set of embedded objects;

FIG. 3 is a high level diagram of a global hosting system according to the present invention;

FIG. 4 is a simplified flowchart illustrating a method of processing a Web page to modified embedded object URLs that is used in the present invention;

FIG. 5 is a simplified state diagram illustrating how the present invention responds to a HTTP request for a Web page.

DETAILED DESCRIPTION OF THE
PREFERRED EMBODIMENT

A known Internet client-server system is implemented as illustrated in FIG. 1. A client machine 10 is connected to a Web server 12 via a network 14. For illustrative purposes, network 14 is the Internet, an intranet, an extranet or any other known network. Web server 12 is one of a plurality of servers which are accessible by clients, one of which is illustrated by machine 10. A representative client machine includes a browser 16, which is a known software tool used to access the servers of the network. The Web server supports files (collectively referred to as a "Web" site) in the form of hypertext documents and objects. In the Internet paradigm, a network path to a server is identified by a so-called Uniform Resource Locator (URL).

A representative Web server 12 is a computer comprising a processor 18, an operating system 20, and a Web server program 22, such as Netscape Enterprise Server. The server 12 also includes a display supporting a graphical user interface (GUI) for management and administration, and an Application Programming Interface (API) 26 that provides extensions to enable application developers to extend and/or customize the core functionality thereof through software programs including Common Gateway Interface (CGI) programs, plug-ins, servlets, active server pages, server side include (SSI) functions or the like.

A representative Web client is a personal computer that is x86-, PowerPC®- or RISC-based, that includes an operating system such as IBM® OS/2® or Microsoft Windows 95, and that includes a Web browser, such as Netscape Navigator 4.0 (or higher), having a Java Virtual Machine (JVM) and support for application plug-ins or helper applications. A client may also be a notebook computer, a handheld computing device (e.g., a PDA), an Internet appliance, or any other such device connectable to the computer network.

As seen in FIG. 2, a typical Web page comprises a markup language (e.g. HTML) master or base document 28, and many embedded objects (e.g., images, audio, video, or the like) 30. Thus, in a typical page, twenty or more embedded images or objects are quite common. Each of these images is an independent object in the Web, retrieved (or validated for change) separately. The common behavior of a Web client, therefore, is to fetch the base HTML document, and then immediately fetch the embedded objects, which are typically (but not always) located on the same server. According to the present invention, preferably the markup language base document 28 is served from the Web server (i.e., the Content Provider site) whereas a given number (or perhaps all) of the embedded objects are served from other servers. As will be seen, preferably a given embedded object is served from a server (other than the Web server itself) that is close to the client machine, that is not overloaded, and that is most likely to already have a current version of the required file.

Referring now to FIG. 3, this operation is achieved by the hosting system of the present invention. As will be seen, the hosting system 35 comprises a set of widely-deployed servers (or server resources) that form a large, fault-tolerant infrastructure designed to serve Web content efficiently, effectively, and reliably to end users. The servers may be deployed globally, or across any desired geographic regions. As will be seen, the hosting system provides a distributed architecture for intelligently routing and replicating such content. To this end, the global hosting system 35 comprises three (3) basic types of servers (or server resources): hosting servers (sometimes called ghosts) 36, top-level DNS servers

38, and low-level DNS servers 40. Although not illustrated, there may be additional levels in the DNS hierarchy. Alternatively, there may be a single DNS level that combines the functionality of the top level and low-level servers. In this illustrative embodiment, the inventive framework 35 is deployed by an Internet Service Provider (ISP), although this is not a limitation of the present invention. The ISP or ISPs that deploy the inventive global hosting framework 35 preferably have a large number of machines that run both the ghost server component 36 and the low-level DNS component 40 on their networks. These machines are distributed throughout the network; preferably, they are concentrated around network exchange points 42 and network access points 44, although this is not a requirement. In addition, the ISP preferably has a small number of machines running the top-level DNS 38 that may also be distributed throughout the network.

Although not meant to be limiting, preferably a given server used in the framework 35 includes a processor, an operating system (e.g., Linux, UNIX, Windows NT, or the like), a Web server application, and a set of application routines used by the invention. These routines are conveniently implemented in software as a set of instructions executed by the processor to perform various process or method steps as will be described in more detail below. The servers are preferably located at the edges of the network (e.g., in points of presence, or POPs).

Several factors may determine where the hosting servers are placed in the network. Thus, for example, the server locations are preferably determined by a demand driven network map that allows the provider (e.g., the ISP) to monitor traffic requests. By studying traffic patterns, the ISP may optimize the server locations for the given traffic profiles.

According to the present invention, a given Web page (comprising a base HTML document and a set of embedded objects) is served in a distributed manner. Thus, preferably, the base HTML document is served from the Content Provider Site 45 that normally hosts the page. The embedded objects, or some subset thereof, are preferentially served from the hosting servers 36 and, specifically, given hosting servers 36 that are near the client machine that in the first instance initiated the request for the Web page. In addition, preferably loads across the hosting servers are balanced to ensure that a given embedded object may be efficiently served from a given hosting server near the client when such client requires that object to complete the page.

To serve the page contents in this manner, the URL associated with an embedded object is modified. As is well-known, each embedded object that may be served in a page has its own URL. Typically, the URL has a hostname identifying the Content Provider's site from where the object is conventionally served, i.e., without reference to the present invention. According to the invention, the embedded object URL is first modified, preferably in an off-line process, to condition the URL to be served by the global hosting servers. A flowchart illustrating the preferred method for modifying the object URL is illustrated in FIG. 4.

The routine begins at step 50 by determining whether all of the embedded objects in a given page have been processed. If so, the routine ends. If not, however, the routine gets the next embedded object at step 52. At step 54, a virtual server hostname is prepended into the URL for the given embedded object. The virtual server hostname includes a value (e.g., a number) that is generated, for example, by applying a given hash function to the URL. As is well-known, a hash function takes arbitrary length bit strings as

inputs and produces fixed length bit strings (hash values) as outputs. Such functions satisfy two conditions: (1) it is infeasible to find two different inputs that produce the same hash value, and (2) given an input and its hash value, it is infeasible to find a different input with the same hash value. In step 54, the URL for the embedded object is hashed into a value xx,xxx that is then included in the virtual server hostname. This step randomly distributes the object to a given virtual server hostname.

The present invention is not limited to generating the virtual server hostname by applying a hash function as described above. As an alternative and preferred embodiment, a virtual server hostname is generated as follows. Consider the representative hostname a1234.g.akamai.net. The 1234 value, sometimes referred to as a serial number, preferably includes information about the object such as its size (big or small), its anticipated popularity, the date on which the object was created, the identity of the Web site, the type of object (e.g., movie or static picture), and perhaps some random bits generated by a given random function. Of course, it is not required that any given serial number encode all of such information or even a significant number of such components. Indeed, in the simplest case, the serial number may be a simple integer. In any event, the information is encoded into a serial number in any convenient manner. Thus, for example, a first bit is used to denote size, a second bit is used to denote popularity, a set of additional bits is used to denote the date, and so forth. As noted above in the hashing example, the serial number is also used for load balancing and for directing certain types of traffic to certain types of servers. Typically, most URLs on the same page have the same serial number to minimize the number of distinguished name (DN) accesses needed per page. This requirement is less important for larger objects.

Thus, according to the present invention, a virtual server hostname is prepended into the URL for a given embedded object, and this hostname includes a value (or serial number) that is generated by applying a given function to the URL or object. That function may be a hash function, an encoding function, or the like.

Turning now back to the flowchart, the routine then continues at step 56 to include a given value in the object's URL. Preferably, the given value is generated by applying a given hash function to the embedded object. This step creates a unique fingerprint of the object that is useful for determining whether the object has been modified. Thereafter, the routine returns to step 50 and cycles.

With the above as background, the inventive global hosting framework is now described in the context of a specific example. In particular, it is assumed that a user of a client machine in Boston requests a Content Provider Web page normally hosted in Atlanta. For illustrative purposes, it is assumed that the Content Provider is using the global hosting architecture within a network, which may be global, international, national, regional, local or private. FIG. 5 shows the various components of the system and how the request from the client is processed. This operation is not to be taken by way of limitation, as will be explained.

Step 1: The browser sends a request to the Provider's Web site (Item 1). The Content Provider site in Atlanta receives the request in the same way that it does as if the global hosting framework were not being implemented. The difference is in what is returned by the Provider site. Instead of returning the usual page, according to the invention, the Web site returns a page with embedded object URLs that are

modified according to the method illustrated in the flowchart of FIG. 4. As previously described, the URLs preferably are changed as follows:

Assume that there are 100,000 virtual ghost servers, even though there may only be a relatively small number (e.g., 100) physically present on the network. These virtual ghost servers or virtual ghosts are identified by the hostname: ghostxxxx.ghosting.com, where xxxxx is replaced by a number between 0 and 99,999. After the Content Provider Web site is updated with new information, a script executing on the Content Provider site is run that rewrites the embedded URLs. Preferably, the embedded URLs names are hashed into numbers between 0 and 99,999, although this range is not a limitation of the present invention. An embedded URL is then switched to reference the virtual ghost with that number. For example, the following is an embedded URL from the Provider's site:

```
<IMG SRC=http://www.provider.com/TECH/images/
space.story.gif>
```

If the serial number for the object referred to by this URL is the number 1467, then preferably the URL is rewritten to read:

```
<IMG SRC=http://ghost1467.ghosting.akamai.com/www-
provider.com/TECH/images/space.story.gif>
```

The use of serial numbers in this manner distributes the embedded URLs roughly evenly over the 100,000 virtual ghost server names. Note that the Provider site can still personalize the page by rearranging the various objects on the screen according to individual preferences. Moreover, the Provider can also insert advertisements dynamically and count how many people view each ad.

According to the preferred embodiment, an additional modification to the embedded URLs is made to ensure that the global hosting system does not serve stale information. As previously described, preferably a hash of the data contained in the embedded URL is also inserted into the embedded URL itself. That is, each embedded URL may contain a fingerprint of the data to which it points. When the underlying information changes, so does the fingerprint, and this prevents users from referencing old data.

The second hash takes as input a stream of bits and outputs what is sometimes referred to as a fingerprint of the stream. The important property of the fingerprint is that two different streams almost surely produce two different fingerprints. Examples of such hashes are the MD2 and MD5 hash functions, however, other more transparent methods such as a simple checksum may be used. For concreteness, assume that the output of the hash is a 128 bit signature. This signature can be interpreted as a number and then inserted into the embedded URL. For example, if the hash of the data in the picture space.story.gif from the Provider web site is the number 28765, then the modified embedded URL would actually look as follows:

```
<IMG SRC=http://ghost1467.ghosting.akamai.com/28765/
www.provider.com/TECH/images/space.story.gif">
```

Whenever a page is changed, preferably the hash for each embedded URL is recomputed and the URL is rewritten if necessary. If any of the URL's data changes, for example, a new and different picture is inserted with the name space.s-tory.gif, then the hash of the data is different and therefore the URL itself will be different. This scheme prevents the system from serving data that is stale as a result of updates to the original page.

For example, assume that the picture space.story.gif is replaced with a more up-to-date version on the Content

Provider server. Because the data of the pictures changes, the hash of the URL changes as well. Thus, the new embedded URL looks the same except that a new number is inserted for the fingerprint. Any user that requests the page after the update receives a page that points to the new picture. The old picture is never referenced and cannot be mistakenly returned in place of the more up-to-date information.

In summary, preferably there are two hashing operations that are done to modify the pages of the Content Provider. First, hashing can be a component of the process by which a serial number is selected to transform the domain name into a virtual ghost name. As will be seen, this first transformation serves to redirect clients to the global hosting system to retrieve the embedded URLs. Next, a hash of the data pointed to by the embedded URLs is computed and inserted into the URL. This second transformation serves to protect against serving stale and out-of-date content from the ghost servers. Preferably, these two transformations are performed off-line and therefore do not pose potential performance bottlenecks.

Generalizing, the preferred URL schema is as follows. The illustrative domain `www.domainname.com/frontpage.jpg` is transformed into:

```
xxxx.yy.zzzz.net/aaaa/www.domainname.com/frontpage-  
-jpg.
```

where:

```
xxxx=serial number field  
yy=lower level DNS field  
zzzz=top level DNS field  
aaaa=other information (e.g., fingerprint) field.
```

If additional levels of the DNS hierarchy are used, then there may be additional lower level DNS fields, e.g., `xxxx.y1.y2.zzz.net/aaaa/...`

Step 2: After receiving the initial page from the Content Provider site, the browser needs to load the embedded URLs to display the page. The first step in doing this is to contact the DNS server on the user's machine (or at the user's ISP) to resolve the altered hostname, in this case: `ghost1467.ghosting.akamai.com`. As will be seen, the global hosting architecture of the present invention manipulates the DNS system so that the name is resolved to one of the ghosts that is near the client and is likely to have the page already. To appreciate how this is done, the following describes the progress of the DNS query that was initiated by the client.

Step 3: As previously described, preferably there are two types of DNS servers in the inventive system: top-level and low-level. The top level DNS servers 38 for `ghosting.com` have a special function that is different from regular DNS servers like those of the `.com` domain. The top level DNS servers 38 include appropriate control routines that are used to determine where in the network a user is located, and then to direct the user to a `akamai.com` (i.e., a low level DNS) server 40 that is close-by. Like the `.com` domain, `akamai.com` preferably has a number of top-level DNS servers 38 spread throughout the network for fault tolerance. Thus, a given top level DNS server 38 directs the user to a region in the Internet (having a collection of hosting servers 36 that may be used to satisfy the request for a given embedded object) whereas the low level DNS server 40 (within the identified region) identifies a particular hosting server within that collection from which the object is actually served.

More generally, as noted above, the DNS process can contain several levels of processing, each of which serves to better direct the client to a ghost server. The ghost server name can also have more fields. For example,

"`a123.g.g.akamaitech.net`" may be used instead of "`a123.ghost.akamai.com`." If only one DNS level is used, a representative URL could be "`a123.akamai.com`."

Although other techniques may be used, the user's location in the network preferably is deduced by looking at the IP address of the client machine making the request. In the present example, the DNS server is running on the machine of the user, although this is not a requirement. If the user is using an ISP DNS server, for example, the routines make the assumption that the user is located near (in the Internet sense) this server. Alternatively, the user's location or IP address could be directly encoded into the request sent to the top level DNS. To determine the physical location of an IP address in the network, preferably, the top level DNS server builds a network map that is then used to identify the relevant location.

Thus, for example, when a request comes in to a top level DNS for a resolution for `a1234.g.akamaitech.net`, the top level DNS looks at the return address of the requester and then formulates the response based on that address according to a network map. In this example, the `a1234` is a serial number, the `g` is a field that refers to the lower level DNS, and `akamaitech` refers to the top level DNS. The network map preferably contains a list of all Internet Protocol (IP) blocks and, for each IP block, the map determines where to direct the request. The map preferably is updated continually based on network conditions and traffic.

After determining where in the network the request originated, the top level DNS server redirects the DNS request to a low level DNS server close to the user in the network.

The ability to redirect requests is a standard feature in the DNS system. In addition, this redirection can be done in such a way that if the local low level DNS server is down, there is a backup server that is contacted.

Preferably, the TTL (time to live) stamp on these top level DNS redirections for the `ghosting.com` domain is set to be long. This allows DNS caching at the user's DNS servers and/or the ISP's DNS servers to prevent the top level DNS servers from being overloaded. If the TTL for `ghosting.akamai.com` in the DNS server at the user's machine or ISP has expired, then a top level server is contacted, and a new redirection to a local low level `ghosting.akamai.com` DNS server is returned with a new TTL stamp. It should be noted the system does not cause a substantially larger number of top level DNS lookups than what is done in the current centralized hosting solutions. This is because the TTL of the top level redirections are set to be high and, thus, the vast majority of users are directed by their local DNS straight to a nearby low level `ghosting.akamai.com` DNS server.

Moreover, fault tolerance for the top level DNS servers is provided automatically by DNS similarly to what is done for the popular `.com` domain. Fault tolerance for the low level DNS servers preferably is provided by returning a list of possible low level DNS servers instead of just a single server. If one of the low level DNS servers is down, the user will still be able to contact one on the list that is up and running.

Fault tolerance can also be handled via an "overflow control" mechanism wherein the client is redirected to a low-level DNS in a region that is known to have sufficient capacity to serve the object. This alternate approach is very useful in scenarios where there is a large amount of demand from a specific region or when there is reduced capacity in a region. In general, the clients are directed to regions in a way that minimizes the overall latency experienced by clients subject to the constraint that no region becomes overloaded. Minimizing overall latency subject to the

regional capacity constraints preferably is achieved using a min-cost multicommodity flow algorithm.

Step 4: At this point, the user has the address of a close-by ghosting.com DNS server 38. The user's local DNS server contacts the close-by low level DNS server 40 and requests a translation for the name ghost1467.ghosting.akamai.com. The local DNS server is responsible for returning the IP address of one of the ghost servers 36 on the network that is close to the user, not overloaded, and most likely to already have the required data.

The basic mechanism for mapping the virtual ghost names to real ghosts is hashing.

One preferred technique is so-called consistent hashing, as described in U.S. Pat. No. 6,430,618, and in U.S. Pat. No. 6,553,420 each titled Method And Apparatus For Distributing Requests Among A Plurality Of Resources, and owned by the Massachusetts Institute of Technology, which are incorporated herein by reference. Consistent hash functions make the system robust under machine failures and crashes. It also allows the system to grow gracefully, without changing where most items are located and without perfect information about the system.

According to the invention, the virtual ghost names may be hashed into real ghost addresses using a table lookup, where the table is continually updated based on network conditions and traffic in such a way to insure load balancing and fault tolerance.

Preferably, a table of resolutions is created for each serial number. For example, serial number 1 resolves to ghost 2 and 5, serial number 2 resolves to ghost 3, serial number 3 resolves to ghosts 2,3,4, and so forth. The goal is to define the resolutions so that no ghost exceeds its capacity and that the total number of all ghosts in all resolutions is minimized.

This is done to assure that the system can take maximal advantage of the available memory at each region. This is a major advantage over existing load balancing schemes that tend to cache everything everywhere or that only cache certain objects in certain locations no matter what the loads are. In general, it is desirable to make assignments so that resolutions tend to stay consistent over time provided that the loads do not change too much in a short period of time. This mechanism preferably also takes into account how close the ghost is to the user, and how heavily loaded the ghost is at the moment.

Note that the same virtual ghost preferably is translated to different real ghost addresses according to where the user is located in the network. For example, assume that ghost server 18.98.0.17 is located in the United States and that ghost server 132.68.1.28 is located in Israel. A DNS request for ghost1487.ghosting.akamai.com originating in Boston will resolve to 18.98.0.17, while a request originating in Tel-Aviv will resolve to 132.68.1.28.

The low-level DNS servers monitor the various ghost servers to take into account their loads while translating virtual ghost names into real addresses. This is handled by a software routine that runs on the ghosts and on the low level DNS servers. In one embodiment, the load information is circulated among the servers in a region so that they can compute resolutions for each serial number. One algorithm for computing resolutions works as follows. The server first computes the projected load (based on number of user requests) for each serial number. The serial numbers are then processed in increasing order of load. For each serial number, a random priority list of desired servers is assigned using a consistent hashing method. Each serial number is then resolved to the smallest initial segment of servers from the priority list so that no server becomes overloaded. For

example, if the priority list for a serial number is 2,5,3,1,6, then an attempt is made first to try to map the load for the serial number to ghost 2. If this overloads ghost 2, then the load is assigned to both ghosts 2 and 5. If this produced too much load on either of those servers, then the load is assigned to ghosts 2,3, and 5, and so forth. The projected load on a server can be computed by looking at all resolutions that contain that server and by adding the amount of load that is likely to be sent to that server from that serial number. This method of producing resolutions is most effective when used in an iterative fashion, wherein the assignments starts in a default state, where every serial number is mapped to every ghost. By refining the resolution table according to the previous procedure, the load is balanced using the minimum amount of replication (thereby maximally conserving the available memory in a region).

The TTL for these low level DNS translations is set to be short to allow a quick response when heavy load is detected on one of the ghosts. The TTL is a parameter that can be manipulated by the system to insure a balance between timely response to high load on ghosts and the load induced on the low level DNS servers. Note, however, that even if the TTL for the low level DNS translation is set to 1-2 minutes, only a few of the users actually have to do a low level DNS lookup. Most users will see a DNS translation that is cached on their machine or at their ISP. Thus, most users go directly from their local DNS server to the close-by ghost that has the data they want. Those users that actually do a low level DNS lookup have a very small added latency, however this latency is small compared to the advantage of retrieving most of the data from close by.

As noted above, fault tolerance for the low level DNS servers is provided by having the top level DNS return a list of possible low level DNS servers instead of a single server address. The user's DNS system caches this list (part of the standard DNS system), and contacts one of the other servers on the list if the first one is down for some reason. The low level DNS servers make use of a standard feature of DNS to provide an extra level of fault tolerance for the ghost servers. When a name is translated, instead of returning a single name, a list of names is returned. If for some reason the primary fault tolerance method for the ghosts (known as the Buddy system, which is described below) fails, the client browser will contact one of the other ghosts on the list.

Step 5: The browser then makes a request for an object named a123.ghosting.akamai.com/.../www.provider.com/TECH/images/space.story.gif from the close-by ghost. Note that the name of the original server (www.provider.com) preferably is included as part of the URL. The software running on the ghost parses the page name into the original host name and the real page name. If a copy of the file is already stored on the ghost, then the data is returned immediately.

Step 6: If, however, no copy of the data on the ghost exists, a copy is retrieved from the original server or another ghost server. Note that the ghost knows who the original server was because the name was encoded into the URL that was passed to the ghost from the browser. Once a copy has been retrieved it is returned to the user, and preferably it is also stored on the ghost for answering future requests.

As an additional safeguard, it may be preferable to check that the user is indeed close to the server. This can be done by examining the IP address of the client before responding to the request for the file. This is useful in the rare case when the client's DNS server is far away from the client. In such a case, the ghost server can redirect the user to a closer server (or to another virtual address that is likely to be

resolved to a server that is closer to the client). If the redirect is to a virtual server, then it must be tagged to prevent further redirections from taking place. In the preferred embodiment, redirection would only be done for large objects; thus, a check may be made before applying a redirection to be sure that the object being requested exceeds a certain overall size.

Performance for long downloads can also be improved by dynamically changing the server to which a client is connected based on changing network conditions. This is especially helpful for audio and video downloads (where the connections can be long and where quality is especially important). In such cases, the user can be directed to an alternate server in mid-stream. The control structure for redirecting the client can be similar to that described above, but it can also include software that is placed in the client's browser or media player. The software monitors the performance of the client's connection and perhaps the status of the network as well. If it is deemed that the client's connection can be improved by changing the server, then the system directs the client to a new server for the rest of the connection.

Fault tolerance for the ghosts is provided by a buddy system, where each ghost has a designated buddy ghost. If a ghost goes down, its buddy takes over its work (and IP address) so that service is not interrupted. Another feature of the system is that the buddy ghost does not have to sit idle waiting for a failure. Instead, all of the machines are always active, and when a failure happens, the load is taken over by the buddy and then balanced by the low level DNS system to the other active ghosts. An additional feature of the buddy system is that fault tolerance is provided without having to wait for long timeout periods.

As yet another safety feature of the global hosting system, a gating mechanism can be used to keep the overall traffic for certain objects within specified limits. One embodiment of the gating mechanism works as follows. When the number of requests for an object exceeds a certain specified threshold, then the server can elect to not serve the object. This can be very useful if the object is very large. Instead, the client can be served a much smaller object that asks the client to return later. Or, the client can be redirected. Another method of implementing a gate is to provide the client with a "ticket" that allows the client to receive the object at a pre-specified future time. In this method, the ghost server needs to check the time on the ticket before serving the object. The inventive global hosting scheme is a way for global ISPs or conglomerates of regional ISPs to leverage their network infrastructure to generate hosting revenue, and to save on network bandwidth. An ISP offering the inventive global hosting scheme can give content providers the ability to distribute content to their users from the closest point on the ISP's network, thus ensuring fast and reliable access. Guaranteed web site performance is critical for any web-based business, and global hosting allows for the creation of a service that satisfies this need.

Global hosting according to the present invention also allows an ISP to control how and where content traverses its network. Global hosting servers can be set up at the edges of the ISP's network (at the many network exchange and access points, for example). This enables the ISP to serve content for sites that it hosts directly into the network exchange points and access points. Expensive backbone links no longer have to carry redundant traffic from the content provider's site to the network exchange and access points. Instead, the content is served directly out of the ISP's network, freeing valuable network resources for other traffic.

Although global hosting reduces network traffic, it is also a method by which global ISPs may capture a piece of the rapidly expanding hosting market, which is currently estimated at over a billion dollars a year.

The global hosting solution also provides numerous advantages to Content Providers, and, in particular, an efficient and cost-effective solution to improve the performance of their Web sites both domestically and internationally. The inventive hosting software ensures Content Providers with fast and reliable Internet access by providing a means to distribute content to their subscribers from the closest point on an ISP's network. In addition to other benefits described in more detail below, the global hosting solution also provides the important benefit of reducing network traffic.

Once inexpensive global hosting servers are installed at the periphery of an ISP's network (i.e., at the many network exchange and access points), content is served directly into network exchange and access points. As a result of this efficient distribution of content directly from an ISP's network, the present invention substantially improves Web site performance. In contrast to current content distribution systems, the inventive global hosting solution does not require expensive backbone links to carry redundant traffic from the Content Provider's Web site to the network exchange and access points.

A summary of the specific advantages afforded by the inventive global hosting scheme are set forth below:

1. Decreased Operational Expenses for Content Providers:

Most competing solutions require Content Providers to purchase servers at each Web site that hosts their content. As a result, Content Providers often must negotiate separate contracts with different ISPs around the world. In addition, Content Providers are generally responsible for replicating the content and maintaining servers in these remote locations.

With the present invention, ISPs are primarily responsible for the majority of the aspects of the global hosting. Content Providers preferably maintain only their single source server. Content on this server is automatically replicated by software to the locations where it is being accessed. No intervention or planning is needed by the Provider (or, for that matter, the ISP). Content Providers are offered instant access to all of the servers on the global network; there is no need to choose where content should be replicated or to purchase additional servers in remote locations.

2. Intelligent and Efficient Data Replication:

Most competing solutions require Content Providers to replicate their content on servers at a commercial hosting site or to mirror their content on geographically distant servers. Neither approach is particularly efficient. In the former situation, content is still located at a single location on the Internet (and thus it is far away from most users). In the latter case, the entire content of a Web site is copied to remote servers, even though only a small portion of the content may actually need to be located remotely. Even with inexpensive memory, the excessive cost associated with such mirroring makes it uneconomical to mirror to more than a few sites, which means that most users will still be far away from a mirror site. Mirroring also has the added disadvantage that Content Providers must insure that all sites remain consistent and current, which is a nontrivial task for even a few sites.

With the present invention, content is automatically replicated to the global server network in an intelligent and efficient fashion. Content is replicated in only those loca-

tions where it is needed. Moreover, when the content changes, new copies preferably are replicated automatically throughout the network.

3. Automatic Content Management:

Many existing solutions require active management of content distribution, content replication and load balancing between different servers. In particular, decisions about where content will be hosted must be made manually, and the process of replicating data is handled in a centralized push fashion. On the contrary, the invention features passive management. Replication is done in a demand-based pull fashion so that content preferably is only sent to where it is truly needed. Moreover, the process preferably is fully automated; the ISP does not have to worry about how and where content is replicated and/or the content provider.

4. Unlimited, Cost Effective Scalability:

Competing solutions are not scalable to more than a small number of sites. For example, solutions based on mirroring are typically used in connection with at most three or four sites. The barriers to scaling include the expense of replicating the entire site, the cost of replicating computing resources at all nodes, and the complexity of supporting the widely varying software packages that Content Providers use on their servers.

The unique system architecture of the present invention is scaleable to hundreds, thousands or even millions of nodes. Servers in the hosting network can malfunction or crash and the system's overall function is not affected. The global hosting framework makes efficient use of resources; servers and client software do not need to be replicated at every node because only the hosting server runs at each node. In addition, the global hosting server is designed to run on standard simple hardware that is not required to be highly fault tolerant.

5. Protection against Flash Crowds:

Competing solutions do not provide the Content Provider with protection from unexpected flash crowds. Although mirroring and related load-balancing solutions do allow a Content Provider to distribute load across a collection of servers, the aggregate capacity of the servers must be sufficient to handle peak demands. This means that the Provider must purchase and maintain a level of resources commensurate with the anticipated peak load instead of the true average load. Given the highly variable and unpredictable nature of the Internet, such solutions are expensive and highly wasteful of resources.

The inventive hosting architecture allows ISPs to utilize a single network of hosting servers to offer Content Providers flash crowd insurance. That is, insurance that the network will automatically adapt to and support unexpected higher load on the Provider's site. Because the ISP is aggregating many Providers together on the same global network, resources are more efficiently used.

6. Substantial Bandwidth Savings:

Competing solutions do not afford substantial bandwidth savings to ISPs or Content Providers. Through the use of mirroring, it is possible to save bandwidth over certain links (i.e., between New York and Los Angeles). Without global hosting, however, most requests for content will still need to transit the Internet, thus incurring bandwidth costs. The inventive hosting framework saves substantial backbone bandwidth for ISPs that have their own backbones. Because content is distributed throughout the network and can be placed next to network exchange points, both ISPs and Content Providers experience substantial savings because backbone charges are not incurred for most content requests.

7. Instant Access to the Global Network:

Competing solutions require the Content Provider to choose manually a small collection of sites at which content will be hosted and/or replicated. Even if the ISP has numerous hosting sites in widely varied locations, only those sites specifically chosen (and paid for) will be used to host content for that Content Provider.

On the contrary, the global hosting solution of the present invention allows ISPs to offer their clients instant access to the global network of servers. To provide instant access to the global network, content is preferably constantly and dynamically moved around the network. For example, if a Content Provider adds content that will be of interest to customers located in Asia, the Content Provider will be assured that its content will be automatically moved to servers that are also located in Asia. In addition, the global hosting framework allows the content to be moved very close to end users (even as close as the user's building in the case of the Enterprise market).

8. Designed for Global ISPs and Conglomerates:

Most competing solutions are designed to be purchased and managed by Content Providers, many of whom are already consistently challenged and consumed by the administrative and operational tasks of managing a single server. The inventive hosting scheme may be deployed by a global ISP, and it provides a new service that can be offered to Content Providers. A feature of the service is that it minimizes the operational and managerial requirements of a Content Provider, thus allowing the Content Provider to focus on its core business of creating unique content.

9. Effective Control of Proprietary Databases and Confidential Information:

Many competing solutions require Content Providers to replicate their proprietary databases to multiple geographically distant sites. As a result, the Content Provider effectively loses control over its proprietary and usually confidential databases. To remedy these problems, the global hosting solution of the present invention ensures that Content Providers retain complete control over their databases. As described above, initial requests for content are directed to the Content Provider's central Web site, which then implements effective and controlled database access. Preferably, high-bandwidth, static parts for page requests are retrieved from the global hosting network.

10. Compatibility with Content Provider Software:

Many competing solutions require Content Providers to utilize a specific set of servers and databases. These particular, non-uniform requirements constrain the Content Provider's ability to most effectively use new technologies, and may require expensive changes to a Content Provider's existing infrastructure. By eliminating these problems, the inventive global hosting architecture effectively interfaces between the Content Provider and the ISP, and it does not make any assumptions about the systems or servers used by the Content Provider. Furthermore, the Content Provider's systems can be upgraded, changed or completely replaced without modifying or interrupting the inventive architecture.

11. No Interference with Dynamic Content, Personalized Advertising or E-Commerce, and No stale content:

Many competing solutions (such as naive caching of all content) can interfere with dynamic content, personalized advertising and E-commerce and can serve the user with stale content. While other software companies have attempted to partially eliminate these issues (such as keeping counts on hits for all cached copies), each of these solutions causes a partial or complete loss of functionality (such as the ability to personalize advertising). On the contrary, the global hosting solution does not interfere with generation of

dynamic content, personalized advertising or E-commerce, because each of these tasks preferably is handled by the central server of the Content Provider.

12. Designed for the Global Network:

The global hosting architecture is highly scaleable and thus may be deployed on a world-wide network basis.

The above-described functionality of each of the components of the global hosting architecture preferably is implemented in software executable in a processor, namely, as a set of instructions or program code in a code module resident in the random access memory of the computer. Until required by the computer, the set of instructions may be stored in another computer memory, for example, in a hard disk drive, or in a removable memory such as an optical disk (for eventual use in a CD ROM) or floppy disk (for eventual use in a floppy disk drive), or downloaded via the Internet or other computer network.

In addition, although the various methods described are conveniently implemented in a general purpose computer selectively activated or reconfigured by software, one of ordinary skill in the art would also recognize that such methods may be carried out in hardware, in firmware, or in more specialized apparatus constructed to perform the required method steps.

Further, as used herein, a Web "client" should be broadly construed to mean any computer or component thereof directly or indirectly connected or connectable in any known or later-developed manner to a computer network, such as the Internet. The term Web "server" should also be broadly construed to mean a computer, computer platform, an adjunct to a computer or platform, or any component thereof. Of course, a "client" should be broadly construed to mean one who requests or gets the file, and "server" is the entity which downloads the file.

Having thus described our invention, what we claim as new and desire to secure by Letters Patent is set forth in the following claims.

The invention claimed is:

1. In a wide area network in which an Internet domain name system (DNS) is useable to resolve DNS queries directed to participating content provider content that is available from participating content provider sites, a method of content delivery wherein participating content providers identify content to be delivered by a service provider from a set of content servers that are distinct from the participating content provider sites and associated with the service provider, wherein a given object of a participating content provider is associated with an alphanumeric string, the method comprising:

having the service provider establish an alternative domain name system (DNS), distinct from the Internet domain name system and any client local name server, and having authority to resolve the alphanumeric strings associated with the objects identified by the participating content providers so that the objects identified by the participating content providers are available to be served from the service provider's content servers, the service provider's alternative domain name system having one or more DNS levels, wherein at least one DNS level comprises a set of one or more name servers;

for each of one or more participating content providers, delivering a given object on behalf of the participating content provider, wherein the given object is delivered by the following steps:

responsive to a DNS query to the given object's associated alphanumeric string, the DNS query originat-

ing from a client local name server, receiving the DNS query at a given name server of a lowest level of the one or more DNS levels in the service provider's alternative domain name system, the given name server that receives the DNS query being close to the client local name server as determined by given location information;

having the given name server that receives the DNS query resolve the alphanumeric string into an IP address that the given name server then returns to the client local name server, wherein the alphanumeric string is resolved without reference to a filename for the given object, wherein the IP address returned as a result of the resolution is associated with a content server within a given subset of the set of content servers, the subset of the set of content servers being associated with the given name server, the content server associated with the IP address returned by the given name server being selected according to a load sharing algorithm enforced across the subset of the set of content servers associated with the given name server;

at the content server associated with the IP address, receiving a request for the given object, the request having the filename associated therewith;

if the given object is available for delivery from the content server associated with the IP address, serving the given object from the content server.

2. The method as described in claim 1 wherein each of the set of content servers is adapted to host given objects from a set of participating content providers.

3. The method as described in claim 1 wherein the one or more levels of the alternative domain name subsystem are arranged as a hierarchy of levels, and wherein a name server in a given level selects a name server in a next succeeding level of the hierarchy.

4. The method as described in claim 1 wherein the load sharing algorithm evaluates a load on the content server, wherein the load on the content server is a function of an identifier in the alphanumeric string, the identifier being associated with a virtual content bucket in which the given object is associated.

5. The method as described in claim 1 wherein the given subset of the content servers is located at a given Internet Point of Presence.

6. The method as described in claim 1 wherein the given subset of the content servers is located at a given network access point.

7. The method as described in claim 1 wherein the given object of the participating content provider is an embedded object in a markup language page.

8. The method as described in claim 1 wherein the IP address is identified by the alternative domain name system as a function of where the DNS query originates.

9. The method as described in claim 1 wherein the IP address is identified by the alternative domain name system as a function of where the DNS query originates and Internet traffic conditions.

10. The method as described in claim 1 further including the step of caching the given object at the content server associated with the IP address for a given time to live (TTL) so that the given object is available for delivery in response to a new request received at the content server associated with the IP address.

19

11. The method as described in claim 1 wherein the load sharing algorithm is a hashing algorithm.

12. The method as described in claim 1 wherein the given location information is an IP address of the client local name server.

13. The method as described in claim 1 wherein the given location information is an IP address of the client from which the DNS query originates.

14. The method as described in claim 1 further including the steps of:
caching the given object at the content server associated with the IP address for a given time to live (TTL);
receiving a new request for the given object at the content server; and
in response to the new request, serving the given object if the TTL has not expired.

20

15. The method as described in claim 1 wherein the alphanumeric string includes given text characters identifying or associated with the service provider.

16. The method as described in claim 1 wherein the alphanumeric string includes a number together with given text characters identifying or associated with the service provider.

17. The method as described in claim 1 wherein the set of content servers are organized into a set of regions, with each region in the set of regions having a given name server and an associated subset of content servers.

18. The method as described in claim 17 wherein a given region in the set of regions is located at one of: an Internet Point of Presence, or a given network access point.

* * * * *



US006108703A

United States Patent [19]
Leighton et al.

[11] Patent Number: 6,108,703
[45] Date of Patent: Aug. 22, 2000

[54] GLOBAL HOSTING SYSTEM
[75] Inventors: F. Thomson Leighton, Newtonville;
Daniel M. Lewin, Cambridge, both of
Mass.
[73] Assignee: Massachusetts Institute of
Technology, Cambridge, Mass.

5,933,832	8/1999	Suzuoka et al.	707/101
5,945,989	8/1999	Freishtat et al.	345/329
5,956,716	9/1999	Kenner et al.	707/10
5,961,596	10/1999	Takubo et al.	709/224
5,991,809	11/1999	Kriegsman	709/226
6,003,030	12/1999	Kenner et al.	707/10
6,006,264	12/1999	Colby et al.	709/226

[21] Appl. No.: 09/314,863
[22] Filed: May 19, 1999

FOREIGN PATENT DOCUMENTS

2202572	10/1998	Canada
865180A2	9/1998	European Pat. Off.
9804985	2/1998	WIPO

Related U.S. Application Data
[60] Provisional application No. 60/092,710, Jul. 14, 1998.
[51] Int. Cl.⁷ G06F 13/00
[52] U.S. Cl. 709/226; 709/105; 709/219;
709/223; 709/224; 709/235
[58] Field of Search 707/10, 2, 104,
707/203, 500, 501, 511, 512, 513, 515;
709/200, 201, 203, 218, 219, 230, 235,
238, 245, 246, 226, 224, 105, 220; 711/118,
119, 120, 122, 126, 130, 200, 202, 216

OTHER PUBLICATIONS
Shaw, David M. "A Low Latency, High Throughput Web Service Using Internet-wide Replication." Department of Computer Science, Johns Hopkins University, Aug. 1998, 33 pgs.

(List continued on next page.)

Primary Examiner—Dung C. Dinh
Assistant Examiner—Abdullahi E. Salad
Attorney, Agent, or Firm—David H. Judson

[56] References Cited

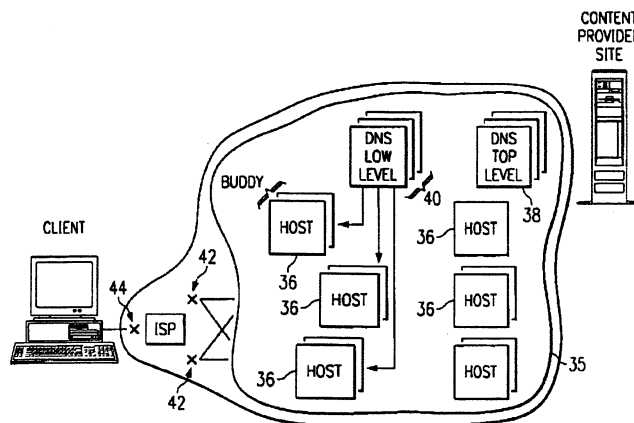
[57] ABSTRACT

U.S. PATENT DOCUMENTS

4,922,417	5/1990	Churm et al.	707/1
5,287,499	2/1994	Nemes	707/2
5,341,477	8/1994	Pitkin et al.	709/226
5,542,087	7/1996	Neimat et al.	707/10
5,638,443	6/1997	Stefik et al.	705/54
5,646,676	7/1997	Dewkett et al.	348/7
5,715,453	2/1998	Stewart	707/104
5,740,423	4/1998	Logan et al.	707/10
5,751,961	5/1998	Smyk	709/217
5,761,507	12/1999	Govett	395/684
5,774,660	6/1998	Brendel et al.	709/201
5,777,989	7/1998	McGarvey	370/254
5,802,291	9/1998	Balick et al.	709/202
5,832,506	11/1998	Kuzma	707/200
5,856,974	1/1999	Gervais et al.	370/392
5,870,559	2/1999	Leshem et al.	709/224
5,878,212	3/1999	Civanlar et al.	709/203
5,884,038	3/1999	Kapoor	709/226
5,903,723	5/1999	Beck et al.	709/200
5,919,247	12/1999	Van Hoff et al.	709/217

The present invention is a network architecture or framework that supports hosting and content distribution on a truly global scale. The inventive framework allows a Content Provider to replicate and serve its most popular content at an unlimited number of points throughout the world. The inventive framework comprises a set of servers operating in a distributed manner. The actual content to be served is preferably supported on a set of hosting servers (sometimes referred to as ghost servers). This content comprises HTML page objects that, conventionally, are served from a Content Provider site. In accordance with the invention, however, a base HTML document portion of a Web page is served from the Content Provider's site while one or more embedded objects for the page are served from the hosting servers, preferably, those hosting servers near the client machine. By serving the base HTML document from the Content Provider's site, the Content Provider maintains control over the content.

34 Claims, 2 Drawing Sheets



OTHER PUBLICATIONS

- Amir, Yair, et al. "Seamlessly Selecting the Best Copy from Internet-Wide Replicated Web Servers." Department of Computer Science, Johns Hopkins University, Jun. 1998, 14 pgs.
- Bestavros, Azer. "Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time in Distributed Information Systems." In *Proceedings of ICDE '96: The 1996 International Conference on Data Engineering*, Mar. 1996, 4 pgs.
- Carter, J. Lawrence, et al. "Universal Classes of Hash Function." *Journal of Computer and System Sciences*, vol. 18, No. 2, Apr. 1979, pp. 143-154.
- Chankhunthod, Anawat, et al. "A Hierarchical Internet Object Cache." In *Usenix Proceedings*, Jan. 1996, pgs. 153-163.
- Cormen, Thomas H., et al. *Introduction to Algorithms*, The MIT Press, Cambridge, Massachusetts, 1994, pgs. 219-243, 991-993.
- Deering, Stephen, et al. "Multicast Routing in Datagram Internetworks and Extended LANs." *ACM Transactions on Computer Systems*, vol. 8, No. 2, May 1990, pgs. 85-110.
- Devine, Robert. "Design and Implementation of DDH: A Distributed Dynamic Hashing Algorithm." In *Proceedings of 4th International Conference on Foundations of Data Organizations and Algorithms*, 1993, pgs. 101-114.
- Grigni, Michelangelo, et al. "Tight Bounds on Minimum Broadcasts Networks." *SIAM Journal of Discrete Mathematics*, vol. 4, No. 2, May 1991, pgs. 207-222.
- Gwertzman, James, et al. "The Case for Geographical Push-Caching." *Technical Report HU TR 34-94*(excerpt), Harvard University, DAS, Cambridge, MA 02138, 1994, 2 pgs.
- Gwertzman, James, et al. "World-Wide Web Cache Consistency." In *Proceedings of the 1996 USENIX Technical Conference*, Jan. 1996, 8 pgs.
- Feeley, Michael, et al. "Implementing Global Memory Management in a Workstation Cluster." In *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, 1995, pgs. 201-212.
- Floyd, Sally, et al. "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing." In *Proceeding of ACM SIGCOMM'95*, pgs. 342-356.
- Fredman, Michael, et al. "Storing a Sparse Table with $O(1)$ Worst Case Access Time." *Journal of the Association for Computing Machinery*, vol. 31., No. 3, Jul. 1984, pgs. 538-544.
- Karger, David, et al. "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web." In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, May 1997, pgs. 654-663.
- Litwin, Withold, et al. "LH—A Scaleable, Distributed Data Structure." *ACM Transactions on Database Systems*, vol. 21, No. 4, Dec. 1996, pgs. 480-525.
- Malpani, Radhika, et al. "Making World Wide Web Caching Servers Cooperate." In *Proceedings of World Wide Web Conference*, 1996, 6 pgs.
- Naor, Moni, et al. "The Load, Capacity and Availability of Quorum Systems." In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, Nov. 1994, pgs. 214-225.
- Nisan, Noam. "Pseudorandom Generators for Space-Bounded Computation." In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, May 1990, pgs. 204-212.
- Palmer, Mark, et al. "Fido: A Cache that Learns to Fetch." In *Proceedings of the 17th International Conference on Very Large Data Bases*, Sep. 1991, pgs. 255-264.
- Panigrahy, Rina. *Relieving Hot Spots on the World Wide Web*. Massachusetts Institute of Technology, Jun. 1997, pgs. 1-66.
- Peleg, David, et al. "The Availability of Quorum Systems." *Information and Computation* 123, 1995, 210-223.
- Plaxton, C. Greg, et al. "Fast Fault-Tolerant Concurrent Access to Shared Objects." In *Proceedings of 37th IEEE Symposium on Foundations of Computer Science*, 1996, pgs. 570-579.
- Rabin, Michael. "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance." *Journal of the ACM*, vol. 36, No. 2, Apr. 1989, pgs. 335-348.
- Ravi, R., "Rapid Rumor Ramification: Approximating the Minimum Broadcast Time." In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, Nov. 1994, pgs. 202-213.
- Schmidt, Jeanette, et al. "Chernoff-Hoeffding Bounds for Applications with Limited Independence." In *Proceedings of the 4th ACS-SIAM Symposium on Discrete Algorithms*, 1993, pgs. 331-340.
- Tarjan, Robert Endre, et al. "Storing a Sparse Table." *Communications of the ACM*, vol. 22, No. 11, Nov. 1979, pgs. 606-611.
- Vitter, Jeffrey Scott, et al. "Optimal Prefetching via Data Compression." In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, Nov. 1991, pgs. 121-130.
- Wegman, Mark, et al. "New Hash Functions and Their Use in Authentication and Set Equality." *Journal of Computer and System Sciences* vol. 22, Jun. 1981, pgs. 265-279.
- Yao, Andrew Chi-Chih. "Should Tables be Sorted?" *Journal of the Association for Computing Machinery*, vol. 28, No. 3, Jul. 1981, pgs. 615-628.
- Beavan, Colin "Web Life They're Watching You." *Esquire*, Aug. 1997, pgs. 104-105.
- Beavan, Colin "Web Life They're Watching You." *Esquire*, Aug. 1997, pp. 104-105.

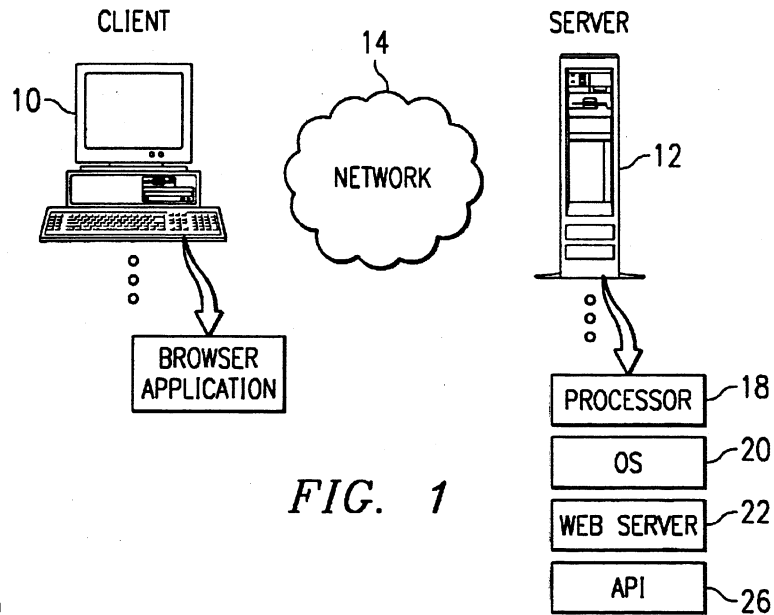


FIG. 1

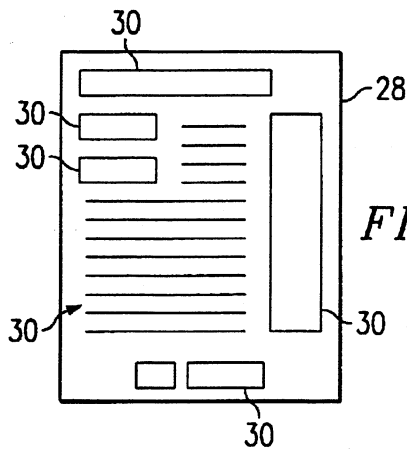


FIG. 2

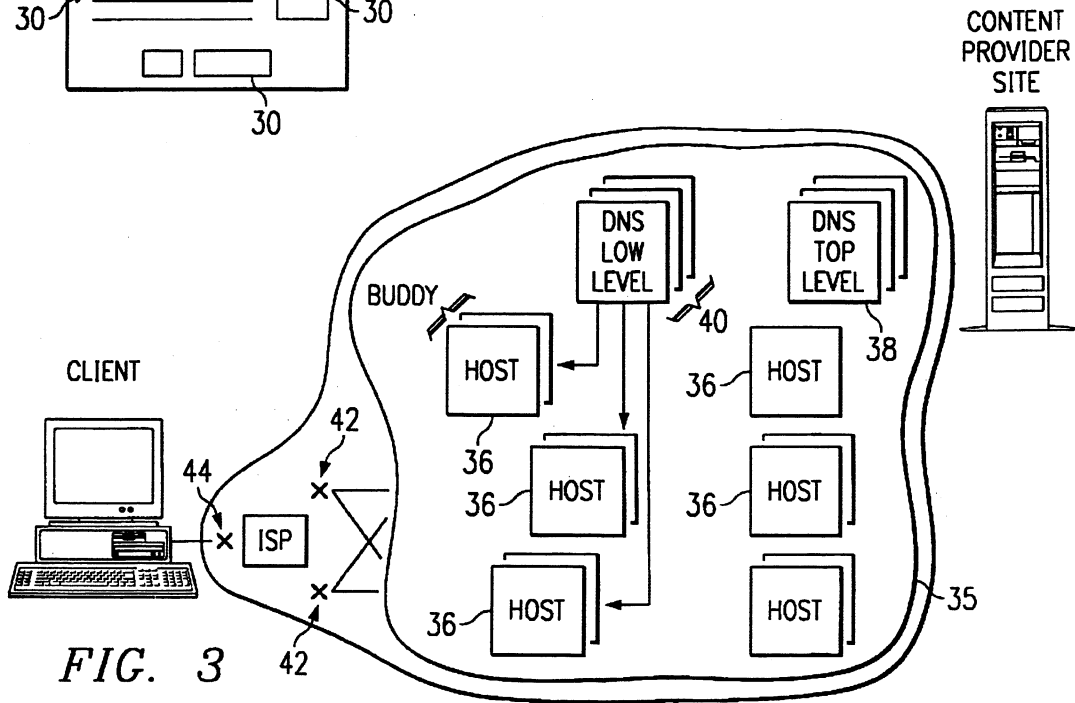


FIG. 3

FIG. 4

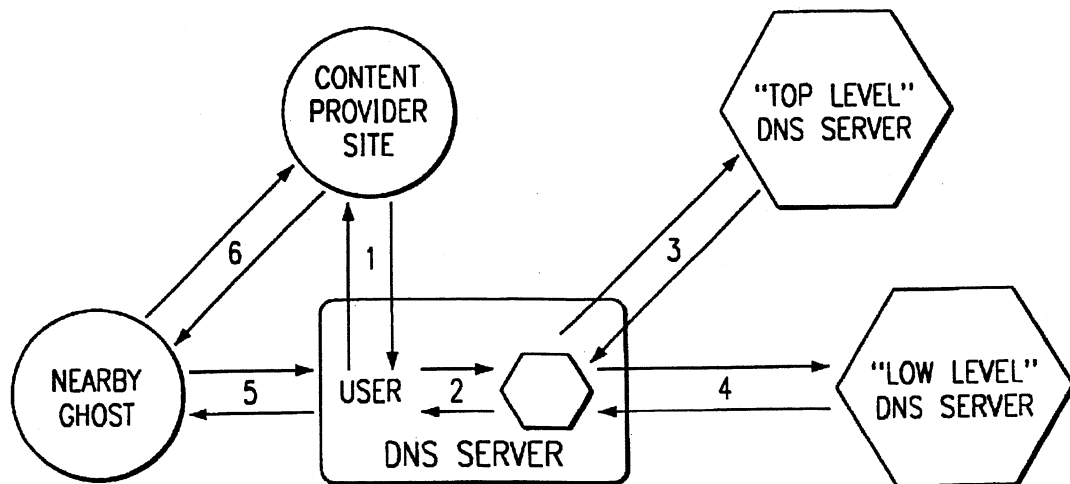
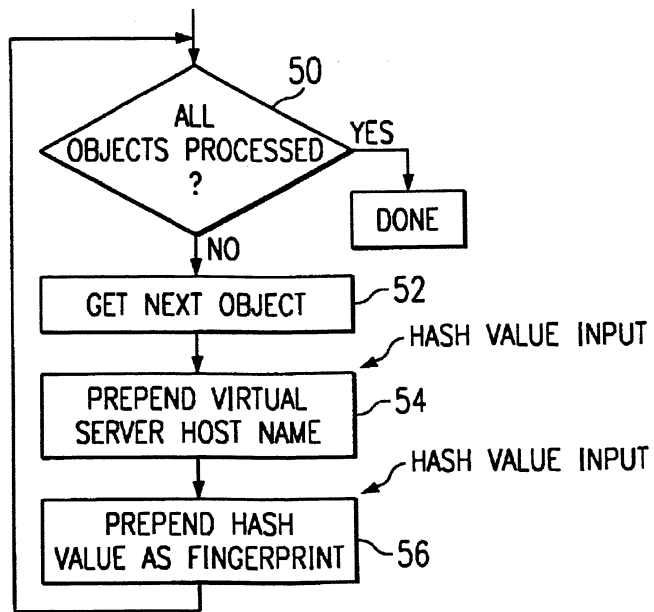


FIG. 5

GLOBAL HOSTING SYSTEM

This application is based on Provisional Application No. 60/092,710, filed Jul. 14, 1998. This application includes subject matter protected by copyright.

BACKGROUND OF THE INVENTION

1. Technical Field

This invention relates generally to information retrieval in a computer network. More particularly, the invention relates to a novel method of hosting and distributing content on the Internet that addresses the problems of Internet Service Providers (ISPs) and Internet Content Providers.

2. Description of the Related Art

The World Wide Web is the Internet's multimedia information retrieval system. In the Web environment, client machines effect transactions to Web servers using the Hypertext Transfer Protocol (HTTP), which is a known application protocol providing users access to files (e.g., text, graphics, images, sound, video, etc.) using a standard page description language known as Hypertext Markup Language (HTML). HTML provides basic document formatting and allows the developer to specify "links" to other servers and files. In the Internet paradigm, a network path to a server is identified by a so-called Uniform Resource Locator (URL) having a special syntax for defining a network connection. Use of an HTML-compatible browser (e.g., Netscape Navigator or Microsoft Internet Explorer) at a client machine involves specification of a link via the URL. In response, the client makes a request to the server identified in the link and, in return, receives a document or other object formatted according to HTML. A collection of documents supported on a Web server is sometimes referred to as a Web site.

It is well known in the prior art for a Web site to mirror its content at another server. Indeed, at present, the only method for a Content Provider to place its content closer to its readers is to build copies of its Web site on machines that are located at Web hosting farms in different locations domestically and internationally. These copies of Web sites are known as mirror sites. Unfortunately, mirror sites place unnecessary economic and operational burdens on Content Providers, and they do not offer economies of scale. Economically, the overall cost to a Content Provider with one primary site and one mirror site is more than twice the cost of a single primary site. This additional cost is the result of two factors: (1) the Content Provider must contract with a separate hosting facility for each mirror site, and (2) the Content Provider must incur additional overhead expenses associated with keeping the mirror sites synchronized.

In an effort to address problems associated with mirroring, companies such as Cisco, Resonate, Bright Tiger, F5 Labs and Alteon, are developing software and hardware that will help keep mirror sites synchronized and load balanced. Although these mechanisms are helpful to the Content Provider, they fail to address the underlying problem of scalability. Even if a Content Provider is willing to incur the costs associated with mirroring, the technology itself will not scale beyond a few (i.e., less than 10) Web sites.

In addition to these economic and scalability issues, mirroring also entails operational difficulties. A Content Provider that uses a mirror site must not only lease and manage physical space in distant locations, but it must also buy and maintain the software or hardware that synchronizes and load balances the sites. Current solutions require Content Providers to supply personnel, technology and other items necessary to maintain multiple Web sites. In summary,

mirroring requires Content Providers to waste economic and other resources on functions that are not relevant to their core business of creating content.

Moreover, Content Providers also desire to retain control of their content. Today, some ISPs are installing caching hardware that interrupts the link between the Content Provider and the end-user. The effect of such caching can produce devastating results to the Content Provider, including (1) preventing the Content Provider from obtaining accurate hit counts on its Web pages (thereby decreasing revenue from advertisers), (2) preventing the Content Provider from tailoring content and advertising to specific audiences (which severely limits the effectiveness of the Content Provider's Web page), and (3) providing outdated information to its customers (which can lead to a frustrated and angry end user).

There remains a significant need in the art to provide a decentralized hosting solution that enables users to obtain Internet content on a more efficient basis (i.e., without burdening network resources unnecessarily) and that likewise enables the Content Provider to maintain control over its content.

The present invention solves these and other problems associated with the prior art.

BRIEF SUMMARY OF THE INVENTION

It is a general object of the present invention to provide a computer network comprising a large number of widely deployed Internet servers that form an organic, massively fault-tolerant infrastructure designed to serve Web content efficiently, effectively, and reliably to end users.

Another more general object of the present invention is to provide a fundamentally new and better method to distribute Web-based content. The inventive architecture provides a method for intelligently routing and replicating content over a large network of distributed servers, preferably with no centralized control.

Another object of the present invention is to provide a network architecture that moves content close to the user. The inventive architecture allows Web sites to develop large audiences without worrying about building a massive infrastructure to handle the associated traffic.

Still another object of the present invention is to provide a fault-tolerant network for distributing Web content. The network architecture is used to speed-up the delivery of richer Web pages, and it allows Content Providers with large audiences to serve them reliably and economically, preferably from servers located close to end users.

A further feature of the present invention is the ability to distribute and manage content over a large network without disrupting the Content Provider's direct relationship with the end user.

Yet another feature of the present invention is to provide a distributed scalable infrastructure for the Internet that shifts the burden of Web content distribution from the Content Provider to a network of preferably hundreds of hosting servers deployed, for example, on a global basis.

In general, the present invention is a network architecture that supports hosting on a truly global scale. The inventive framework allows a Content Provider to replicate its most popular content at an unlimited number of points throughout the world. As an additional feature, the actual content that is replicated at any one geographic location is specifically tailored to viewers in that location. Moreover, content is automatically sent to the location where it is requested, without any effort or overhead on the part of a Content Provider.

3

It is thus a more general object of this invention to provide a global hosting framework to enable Content Providers to retain control of their content.

The hosting framework of the present invention comprises a set of servers operating in a distributed manner. The actual content to be served is preferably supported on a set of hosting servers (sometimes referred to as ghost servers). This content comprises HTML page objects that, conventionally, are served from a Content Provider site. In accordance with the invention, however, a base HTML document portion of a Web page is served from the Content Provider's site while one or more embedded objects for the page are served from the hosting servers, preferably, those hosting servers nearest the client machine. By serving the base HTML document from the Content Provider's site, the Content Provider maintains control over the content.

The determination of which hosting server to use to serve a given embedded object is effected by other resources in the hosting framework. In particular, the framework includes a second set of servers (or server resources) that are configured to provide top level Domain Name Service (DNS). In addition, the framework also includes a third set of servers (or server resources) that are configured to provide low level DNS functionality. When a client machine issues an HTTP request to the Web site for a given Web page, the base HTML document is served from the Web site as previously noted. Embedded objects for the page preferably are served from particular hosting servers identified by the top- and low-level DNS servers. To locate the appropriate hosting servers to use, the top-level DNS server determines the user's location in the network to identify a given low-level DNS server to respond to the request for the embedded object. The top-level DNS server then redirects the request to the identified low-level DNS server that, in turn, resolves the request into an IP address for the given hosting server that serves the object back to the client.

More generally, it is possible (and, in some cases, desirable) to have a hierarchy of DNS servers that consisting of several levels. The lower one moves in the hierarchy, the closer one gets to the best region.

A further aspect of the invention is a means by which content can be distributed and replicated through a collection of servers so that the use of memory is optimized subject to the constraints that there are a sufficient number of copies of any object to satisfy the demand, the copies of objects are spread so that no server becomes overloaded, copies tend to be located on the same servers as time moves forward, and copies are located in regions close to the clients that are requesting them. Thus, servers operating within the framework do not keep copies of all of the content database. Rather, given servers keep copies of a minimal amount of data so that the entire system provides the required level of service. This aspect of the invention allows the hosting scheme to be far more efficient than schemes that cache everything everywhere, or that cache objects only in pre-specified locations.

The global hosting framework is fault tolerant at each level of operation. In particular, the top level DNS server returns a list of low-level DNS servers that may be used by the client to service the request for the embedded object. Likewise, each hosting server preferably includes a buddy server that is used to assume the hosting responsibilities of its associated hosting server in the event of a failure condition.

According to the present invention, load balancing across the set of hosting servers is achieved in part through a novel

4

technique for distributing the embedded object requests. In particular, each embedded object URL is preferably modified by prepending a virtual server hostname into the URL. More generally, the virtual server hostname is inserted into the URL. Preferably, the virtual server hostname includes a value (sometimes referred to as a serial number) generated by applying a given hash function to the URL or by encoding given information about the object into the value. This function serves to randomly distribute the embedded objects over a given set of virtual server hostnames. In addition, a given fingerprint value for the embedded object is generated by applying a given hash function to the embedded object itself. This given value serves as a fingerprint that identifies whether the embedded object has been modified. Preferably, the functions used to generate the values (i.e., for the virtual server hostname and the fingerprint) are applied to a given Web page in an off-line process. Thus, when an HTTP request for the page is received, the base HTML document is served by the Web site and some portion of the page's embedded objects are served from the hosting servers near (although not necessarily the closest) to the client machine that initiated the request.

The foregoing has outlined some of the more pertinent objects and features of the present invention. These objects should be construed to be merely illustrative of some of the more prominent features and applications of the invention. Many other beneficial results can be attained by applying the disclosed invention in a different manner or modifying the invention as will be described. Accordingly, other objects and a fuller understanding of the invention may be had by referring to the following Detailed Description of the Preferred Embodiment.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference should be made to the following Detailed Description taken in connection with the accompanying drawings in which:

FIG. 1 is a representative system in which the present invention is implemented;

FIG. 2 is a simplified representation of a markup language document illustrating the base document and a set of embedded objects;

FIG. 3 is a high level diagram of a global hosting system according to the present invention;

FIG. 4 is a simplified flowchart illustrating a method of processing a Web page to modified embedded object URLs that is used in the present invention;

FIG. 5 is a simplified state diagram illustrating how the present invention responds to a HTTP request for a Web page.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A known Internet client-server system is implemented as illustrated in FIG. 1. A client machine 10 is connected to a Web server 12 via a network 14. For illustrative purposes, network 14 is the Internet, an intranet, an extranet or any other known network. Web server 12 is one of a plurality of servers which are accessible by clients, one of which is illustrated by machine 10. A representative client machine includes a browser 16, which is a known software tool used to access the servers of the network. The Web server supports files (collectively referred to as a "Web" site) in the form of hypertext documents and objects. In the Internet

5

paradigm, a network path to a server is identified by a so-called Uniform Resource Locator (URL).

A representative Web server 12 is a computer comprising a processor 18, an operating system 20, and a Web server program 22, such as Netscape Enterprise Server. The server 12 also includes a display supporting a graphical user interface (GUI) for management and administration, and an Application Programming Interface (API) that provides extensions to enable application developers to extend and/or customize the core functionality thereof through software programs including Common Gateway Interface (CGI) programs, plug-ins, servlets, active server pages, server side include (SSI) functions or the like.

A representative Web client is a personal computer that is x86-, PowerPC®- or RISC-based, that includes an operating system such as IBM® OS/2® or Microsoft Windows '95, and that includes a Web browser, such as Netscape Navigator 4.0 (or higher), having a Java Virtual Machine (JVM) and support for application plug-ins or helper applications. A client may also be a notebook computer, a handheld computing device (e.g., a PDA), an Internet appliance, or any other such device connectable to the computer network.

As seen in FIG. 2, a typical Web page comprises a markup language (e.g. HTML) master or base document 28, and many embedded objects (e.g., images, audio, video, or the like) 30. Thus, in a typical page, twenty or more embedded images or objects are quite common. Each of these images is an independent object in the Web, retrieved (or validated for change) separately. The common behavior of a Web client, therefore, is to fetch the base HTML document, and then immediately fetch the embedded objects, which are typically (but not always) located on the same server. According to the present invention, preferably the markup language base document 28 is served from the Web server (i.e., the Content Provider site) whereas a given number (or perhaps all) of the embedded objects are served from other servers. As will be seen, preferably a given embedded object is served from a server (other than the Web server itself) that is close to the client machine, that is not overloaded, and that is most likely to already have a current version of the required file.

Referring now to FIG. 3, this operation is achieved by the hosting system of the present invention. As will be seen, the hosting system 35 comprises a set of widely-deployed servers (or server resources) that form a large, fault-tolerant infrastructure designed to serve Web content efficiently, effectively, and reliably to end users. The servers may be deployed globally, or across any desired geographic regions. As will be seen, the hosting system provides a distributed architecture for intelligently routing and replicating such content. To this end, the global hosting system 35 comprises three (3) basic types of servers (or server resources): hosting servers (sometimes called ghosts) 36, top-level DNS servers 38, and low-level DNS servers 40. Although not illustrated, there may be additional levels in the DNS hierarchy. Alternatively, there may be a single DNS level that combines the functionality of the top level and low-level servers. In this illustrative embodiment, the inventive framework 35 is deployed by an Internet Service Provider (ISP), although this is not a limitation of the present invention. The ISP or ISPs that deploy the inventive global hosting framework 35 preferably have a large number of machines that run both the ghost server component 36 and the low-level DNS component 40 on their networks. These machines are distributed throughout the network; preferably, they are concentrated around network exchange points 42 and network access points 44, although this is not a requirement. In addition, the

6

ISP preferably has a small number of machines running the top-level DNS 38 that may also be distributed throughout the network.

Although not meant to be limiting, preferably a given server used in the framework 35 includes a processor, an operating system (e.g., Linux, UNIX, Windows NT, or the like), a Web server application, and a set of application routines used by the invention. These routines are conveniently implemented in software as a set of instructions executed by the processor to perform various process or method steps as will be described in more detail below. The servers are preferably located at the edges of the network (e.g., in points of presence, or POPs).

Several factors may determine where the hosting servers are placed in the network. Thus, for example, the server locations are preferably determined by a demand driven network map that allows the provider (e.g., the ISP) to monitor traffic requests. By studying traffic patterns, the ISP may optimize the server locations for the given traffic profiles.

According to the present invention, a given Web page (comprising a base HTML document and a set of embedded objects) is served in a distributed manner. Thus, preferably, the base HTML document is served from the Content Provider that normally hosts the page. The embedded objects, or some subset thereof, are preferentially served from the hosting servers 36 and, specifically, given hosting servers 36 that are near the client machine that in the first instance initiated the request for the Web page. In addition, preferably loads across the hosting servers are balanced to ensure that a given embedded object may be efficiently served from a given hosting server near the client when such client requires that object to complete the page.

To serve the page contents in this manner, the URL associated with an embedded object is modified. As is well-known, each embedded object that may be served in a page has its own URL. Typically, the URL has a hostname identifying the Content Provider's site from where the object is conventionally served, i.e., without reference to the present invention. According to the invention, the embedded object URL is first modified, preferably in an off-line process, to condition the URL to be served by the global hosting servers. A flowchart illustrating the preferred method for modifying the object URL is illustrated in FIG. 4.

The routine begins at step 50 by determining whether all of the embedded objects in a given page have been processed. If so, the routine ends. If not, however, the routine gets the next embedded object at step 52. At step 54, a virtual server hostname is prepended into the URL for the given embedded object. The virtual server hostname includes a value (e.g., a number) that is generated, for example, by applying a given hash function to the URL. As is well-known, a hash function takes arbitrary length bit strings as inputs and produces fixed length bit strings (hash values) as outputs. Such functions satisfy two conditions: (1) it is infeasible to find two different inputs that produce the same hash value, and (2) given an input and its hash value, it is infeasible to find a different input with the same hash value. In step 54, the URL for the embedded object is hashed into a value xx,xxx that is then included in the virtual server hostname. This step randomly distributes the object to a given virtual server hostname.

The present invention is not limited to generating the virtual server hostname by applying a hash function as described above. As an alternative and preferred

embodiment, a virtual server hostname is generated as follows. Consider the representative hostname a1234.g.akamaitech.net. The 1234 value, sometimes referred to as a serial number, preferably includes information about the object such as its size (big or small), its anticipated popularity, the date on which the object was created, the identity of the Web site, the type of object (e.g., movie or static picture), and perhaps some random bits generated by a given random function. Of course, it is not required that any given serial number encode all of such information or even a significant number of such components. Indeed, in the simplest case, the serial number may be a simple integer. In any event, the information is encoded into a serial number in any convenient manner. Thus, for example, a first bit is used to denote size, a second bit is used to denote popularity, a set of additional bits is used to denote the date, and so forth. As noted above in the hashing example, the serial number is also used for load balancing and for directing certain types of traffic to certain types of servers. Typically, most URLs on the same page have the same serial number to minimize the number of distinguished name (DN) accesses needed per page. This requirement is less important for larger objects.

Thus, according to the present invention, a virtual server hostname is prepended into the URL for a given embedded object, and this hostname includes a value (or serial number) that is generated by applying a given function to the URL or object. That function may be a hash function, an encoding function, or the like.

Turning now back to the flowchart, the routine then continues at step 56 to include a given value in the object's URL. Preferably, the given value is generated by applying a given hash function to the embedded object. This step creates a unique fingerprint of the object that is useful for determining whether the object has been modified. Thereafter, the routine returns to step 50 and cycles.

With the above as background, the inventive global hosting framework is now described in the context of a specific example. In particular, it is assumed that a user of a client machine in Boston requests a Content Provider Web page normally hosted in Atlanta. For illustrative purposes, it is assumed that the Content Provider is using the global hosting architecture within a network, which may be global, international, national, regional, local or private. FIG. 5 shows the various components of the system and how the request from the client is processed. This operation is not to be taken by way of limitation, as will be explained.

Step 1: The browser sends a request to the Provider's Web site (Item 1). The Content Provider site in Atlanta receives the request in the same way that it does as if the global hosting framework were not being implemented. The difference is in what is returned by the Provider site. Instead of returning the usual page, according to the invention, the Web site returns a page with embedded object URLs that are modified according to the method illustrated in the flowchart of FIG. 4. As previously described, the URLs preferably are changed as follows:

Assume that there are 100,000 virtual ghost servers, even though there may only be a relatively small number (e.g., 100) physically present on the network. These virtual ghost servers or virtual ghosts are identified by the hostname: ghostxxxx.ghosting.com, where xxxxx is replaced by a number between 0 and 99,999. After the Content Provider Web site is updated with new information, a script executing on the Content Provider site is run that rewrites the embedded URLs. Preferably, the embedded URLs names are

hashed into numbers between 0 and 99,999, although this range is not a limitation of the present invention. An embedded URL is then switched to reference the virtual ghost with that number. For example, the following is an embedded URL from the Provider's site:

```
<IMG SRC=http://www.provider.com/TECH/images/
space.story.gif>
```

If the serial number for the object referred to by this URL is the number 1467, then preferably the URL is rewritten to read:

```
<IMG SRC=http://ghost467.ghosting.akamai.com/
www.provider.com/TECH/images/sp ace.story.gif>
```

The use of serial numbers in this manner distributes the embedded URLs roughly evenly over the 100,000 virtual ghost server names. Note that the Provider site can still personalize the page by rearranging the various objects on the screen according to individual preferences. Moreover, the Provider can also insert advertisements dynamically and count how many people view each ad.

According to the preferred embodiment, an additional modification to the embedded URLs is made to ensure that the global hosting system does not serve stale information. As previously described, preferably a hash of the data contained in the embedded URL is also inserted into the embedded URL itself. That is, each embedded URL may contain a fingerprint of the data to which it points. When the underlying information changes, so does the fingerprint, and this prevents users from referencing old data.

The second hash takes as input a stream of bits and outputs what is sometimes referred to as a fingerprint of the stream. The important property of the fingerprint is that two different streams almost surely produce two different fingerprints. Examples of such hashes are the MD2 and MD5 hash functions, however, other more transparent methods such as a simple checksum may be used. For concreteness, assume that the output of the hash is a 128 bit signature. This signature can be interpreted as a number and then inserted into the embedded URL. For example, if the hash of the data in the picture space.story.gif from the Provider web site is the number 28765, then the modified embedded URL would actually look as follows:

```
<IMG SRC=http://ghost1467.ghosting.akamai.com/28765/
www.provider.com /TECH/images/space.story.gif">
```

Whenever a page is changed, preferably the hash for each embedded URL is recomputed and the URL is rewritten if necessary. If any of the URL's data changes, for example, a new and different picture is inserted with the name space.story.gif, then the hash of the data is different and therefore the URL itself will be different. This scheme prevents the system from serving data that is stale as a result of updates to the original page.

For example, assume that the picture space.story.gif is replaced with a more up-to-date version on the Content Provider server. Because the data of the pictures changes, the hash of the URL changes as well. Thus, the new embedded URL looks the same except that a new number is inserted for the fingerprint. Any user that requests the page after the update receives a page that points to the new picture. The old picture is never referenced and cannot be mistakenly returned in place of the more up-to-date information.

In summary, preferably there are two hashing operations that are done to modify the pages of the Content Provider. First, hashing can be a component of the process by which a serial number is selected to transform the domain name into a virtual ghost name. As will be seen, this first transformation serves to redirect clients to the global hosting

system to retrieve the embedded URLs. Next, a hash of the data pointed to by the embedded URLs is computed and inserted into the URL. This second transformation serves to protect against serving stale and out-of-date content from the ghost servers. Preferably, these two transformations are performed off-line and therefore do not pose potential performance bottlenecks.

Generalizing, the preferred URL schema is as follows. The illustrative domain `www.domainname.com/frontpage.jpg` is transformed into:

```
xxxx.yy.zzzz.net/aaaa/www.domainname.com/
frontpage.jpg,
```

where:

xxxx=serial number field

yy=lower level DNS field

zzzz=top level DNS field

aaaa=other information (e.g., fingerprint) field.

If additional levels of the DNS hierarchy are used, then there may be additional lower level DNS fields, e.g., `xxxx.y1.y2.zzz.net/aaaa/...`

Step 2: After receiving the initial page from the Content Provider site, the browser needs to load the embedded URLs to display the page. The first step in doing this is to contact the DNS server on the user's machine (or at the user's ISP) to resolve the altered hostname, in this case: `ghost1467.ghosting.akamai.com`. As will be seen, the global hosting architecture of the present invention manipulates the DNS system so that the name is resolved to one of the ghosts that is near the client and is likely to have the page already. To appreciate how this is done, the following describes the progress of the DNS query that was initiated by the client.

Step 3: As previously described, preferably there are two types of DNS servers in the inventive system: top-level and low-level. The top level DNS servers 38 for `ghosting.com` have a special function that is different from regular DNS servers like those of the `.com` domain. The top level DNS servers 38 include appropriate control routines that are used to determine where in the network a user is located, and then to direct the user to a `akamai.com` (i.e., a low level DNS) server 40 that is close-by. Like the `.com` domain, `akamai.com` preferably has a number of top-level DNS servers 38 spread throughout the network for fault tolerance. Thus, a given top level DNS server 38 directs the user to a region in the Internet (having a collection of hosting servers 36 that may be used to satisfy the request for a given embedded object) whereas the low level DNS server 40 (within the identified region) identifies a particular hosting server within that collection from which the object is actually served.

More generally, as noted above, the DNS process can contain several levels of processing, each of which serves to better direct the client to a ghost server. The ghost server name can also have more fields. For example, "`a123.g.g.akamaitech.net`" may be used instead of "`a123.ghost.akamai.com`." If only one DNS level is used, a representative URL could be "`a123.akamai.com`."

Although other techniques may be used, the user's location in the network preferably is deduced by looking at the IP address of the client machine making the request. In the present example, the DNS server is running on the machine of the user, although this is not a requirement. If the user is using an ISP DNS server, for example, the routines make the assumption that the user is located near (in the Internet sense) this server. Alternatively, the user's location or IP address could be directly encoded into the request sent to the top level DNS. To determine the physical location of an IP address in the network, preferably, the top level DNS server builds a network map that is then used to identify the relevant location.

Thus, for example, when a request comes in to a top level DNS for a resolution for `a1234.g.akamaitech.net`, the top level DNS looks at the return address of the requester and then formulates the response based on that address according to a network map. In this example, the `a1234` is a serial number, the `g` is a field that refers to the lower level DNS, and `akamaitech` refers to the top level DNS. The network map preferably contains a list of all Internet Protocol (IP) blocks and, for each IP block, the map determines where to direct the request. The map preferably is updated continually based on network conditions and traffic.

After determining where in the network the request originated, the top level DNS server redirects the DNS request to a low level DNS server close to the user in the network. The ability to redirect requests is a standard feature in the DNS system. In addition, this redirection can be done in such a way that if the local low level DNS server is down, there is a backup server that is contacted.

Preferably, the TTL (time to live) stamp on these top level DNS redirections for the `ghosting.com` domain is set to be long. This allows DNS caching at the user's DNS servers and/or the ISP's DNS servers to prevent the top level DNS servers from being overloaded. If the TTL for `ghosting.akamai.com` in the DNS server at the user's machine or ISP has expired, then a top level server is contacted, and a new redirection to a local low level `ghosting.akamai.com` DNS server is returned with a new TTL stamp. It should be noted the system does not cause a substantially larger number of top level DNS lookups than what is done in the current centralized hosting solutions. This is because the TTL of the top level redirections are set to be high and, thus, the vast majority of users are directed by their local DNS straight to a nearby low level `ghosting.akamai.com` DNS server.

Moreover, fault tolerance for the top level DNS servers is provided automatically by DNS similarly to what is done for the popular `.com` domain. Fault tolerance for the low level DNS servers preferably is provided by returning a list of possible low level DNS servers instead of just a single server. If one of the low level DNS servers is down, the user will still be able to contact one on the list that is up and running.

Fault tolerance can also be handled via an "overflow control" mechanism wherein the client is redirected to a low-level DNS in a region that is known to have sufficient capacity to serve the object. This alternate approach is very useful in scenarios where there is a large amount of demand from a specific region or when there is reduced capacity in a region. In general, the clients are directed to regions in a way that minimizes the overall latency experienced by clients subject to the constraint that no region becomes overloaded. Minimizing overall latency subject to the regional capacity constraints preferably is achieved using a min-cost multicommodity flow algorithm.

Step 4: At this point, the user has the address of a close-by `ghosting.com` DNS server 38. The user's local DNS server contacts the close-by low level DNS server 40 and requests a translation for the name `ghost1467.ghosting.akamai.com`. The local DNS server is responsible for returning the IP address of one of the ghost servers 36 on the network that is close to the user, not overloaded, and most likely to already have the required data.

The basic mechanism for mapping the virtual ghost names to real ghosts is hashing. One preferred technique is so-called consistent hashing, as described in U.S. Ser. No. 09/042,228, filed Mar. 13, 1998, and in U.S. Ser. No. 09/088,825, filed Jun. 2, 1998, each titled Method And Apparatus For Distributing Requests Among A Plurality Of

Resources, and owned by the Massachusetts Institute of Technology, which applications are incorporated herein by reference. Consistent hash functions make the system robust under machine failures and crashes. It also allows the system to grow gracefully, without changing where most items are located and without perfect information about the system.

According to the invention, the virtual ghost names may be hashed into real ghost addresses using a table lookup, where the table is continually updated based on network conditions and traffic in such a way to insure load balancing and fault tolerance. Preferably, a table of resolutions is created for each serial number. For example, serial number 1 resolves to ghost 2 and 5, serial number 2 resolves to ghost 3, serial number 3 resolves to ghosts 2,3,4, and so forth. The goal is to define the resolutions so that no ghost exceeds its capacity and that the total number of all ghosts in all resolutions is minimized. This is done to assure that the system can take maximal advantage of the available memory at each region. This is a major advantage over existing load balancing schemes that tend to cache everything everywhere or that only cache certain objects in certain locations no matter what the loads are. In general, it is desirable to make assignments so that resolutions tend to stay consistent over time provided that the loads do not change too much in a short period of time. This mechanism preferably also takes into account how close the ghost is to the user, and how heavily loaded the ghost is at the moment.

Note that the same virtual ghost preferably is translated to different real ghost addresses according to where the user is located in the network. For example, assume that ghost server 18.98.0.17 is located in the United States and that ghost server 132.68.1.28 is located in Israel. A DNS request for ghost1487.ghosting.akamai.com originating in Boston will resolve to 18.98.0.17, while a request originating in Tel-Aviv will resolve to 132.68.1.28.

The low-level DNS servers monitor the various ghost servers to take into account their loads while translating virtual ghost names into real addresses. This is handled by a software routine that runs on the ghosts and on the low level DNS servers. In one embodiment, the load information is circulated among the servers in a region so that they can compute resolutions for each serial number. One algorithm for computing resolutions works as follows. The server first computes the projected load (based on number of user requests) for each serial number. The serial numbers are then processed in increasing order of load. For each serial number, a random priority list of desired servers is assigned using a consistent hashing method. Each serial number is then resolved to the smallest initial segment of servers from the priority list so that no server becomes overloaded. For example, if the priority list for a serial number is 2,5,3,1,6, then an attempt is made first to try to map the load for the serial number to ghost 2. If this overloads ghost 2, then the load is assigned to both ghosts 2 and 5. If this produced too much load on either of those servers, then the load is assigned to ghosts 2,3, and 5, and so forth. The projected load on a server can be computed by looking at all resolutions that contain that server and by adding the amount of load that is likely to be sent to that server from that serial number. This method of producing resolutions is most effective when used in an iterative fashion, wherein the assignments starts in a default state, where every serial number is mapped to every ghost. By refining the resolution table according to the previous procedure, the load is balanced using the minimum amount of replication (thereby maximally conserving the available memory in a region).

The TTL for these low level DNS translations is set to be short to allow a quick response when heavy load is detected

on one of the ghosts. The TTL is a parameter that can be manipulated by the system to insure a balance between timely response to high load on ghosts and the load induced on the low level DNS servers. Note, however, that even if the TTL for the low level DNS translation is set to 1-2 minutes, only a few of the users actually have to do a low level DNS lookup. Most users will see a DNS translation that is cached on their machine or at their ISP. Thus, most users go directly from their local DNS server to the close-by ghost that has the data they want. Those users that actually do a low level DNS lookup have a very small added latency, however this latency is small compared to the advantage of retrieving most of the data from close by.

As noted above, fault tolerance for the low level DNS servers is provided by having the top level DNS return a list of possible low level DNS servers instead of a single server address. The user's DNS system caches this list (part of the standard DNS system), and contacts one of the other servers on the list if the first one is down for some reason. The low level DNS servers make use of a standard feature of DNS to provide an extra level of fault tolerance for the ghost servers. When a name is translated, instead of returning a single name, a list of names is returned. If for some reason the primary fault tolerance method for the ghosts (known as the Buddy system, which is described below) fails, the client browser will contact one of the other ghosts on the list.

Step 5: The browser then makes a request for an object named a123.ghosting.akamai.com/.../www.provider.com/TECH/images/space.story.gif from the close-by ghost. Note that the name of the original server (www.provider.com) preferably is included as part of the URL. The software running on the ghost parses the page name into the original host name and the real page name. If a copy of the file is already stored on the ghost, then the data is returned immediately. If, however, no copy of the data on the ghost exists, a copy is retrieved from the original server or another ghost server. Note that the ghost knows who the original server was because the name was encoded into the URL that was passed to the ghost from the browser. Once a copy has been retrieved it is returned to the user, and preferably it is also stored on the ghost for answering future requests.

As an additional safeguard, it may be preferable to check that the user is indeed close to the server. This can be done by examining the IP address of the client before responding to the request for the file. This is useful in the rare case when the client's DNS server is far away from the client. In such a case, the ghost server can redirect the user to a closer server (or to another virtual address that is likely to be resolved to a server that is closer to the client). If the redirect is to a virtual server, then it must be tagged to prevent further redirections from taking place. In the preferred embodiment, redirection would only be done for large objects; thus, a check may be made before applying a redirection to be sure that the object being requested exceeds a certain overall size.

Performance for long downloads can also be improved by dynamically changing the server to which a client is connected based on changing network conditions. This is especially helpful for audio and video downloads (where the connections can be long and where quality is especially important). In such cases, the user can be directed to an alternate server in mid-stream. The control structure for redirecting the client can be similar to that described above, but it can also include software that is placed in the client's browser or media player. The software monitors the performance of the client's connection and perhaps the status of the network as well. If it is deemed that the client's connection can be improved by changing the server, then the system directs the client to a new server for the rest of the connection.

Fault tolerance for the ghosts is provided by a buddy system, where each ghost has a designated buddy ghost. If a ghost goes down, its buddy takes over its work (and IP address) so that service is not interrupted. Another feature of the system is that the buddy ghost does not have to sit idle waiting for a failure. Instead, all of the machines are always active, and when a failure happens, the load is taken over by the buddy and then balanced by the low level DNS system to the other active ghosts. An additional feature of the buddy system is that fault tolerance is provided without having to wait for long timeout periods.

As yet another safety feature of the global hosting system, a gating mechanism can be used to keep the overall traffic for certain objects within specified limits. One embodiment of the gating mechanism works as follows. When the number of requests for an object exceeds a certain specified threshold, then the server can elect to not serve the object. This can be very useful if the object is very large. Instead, the client can be served a much smaller object that asks the client to return later. Or, the client can be redirected. Another method of implementing a gate is to provide the client with a "ticket" that allows the client to receive the object at a prespecified future time. In this method, the ghost server needs to check the time on the ticket before serving the object.

The inventive global hosting scheme is a way for global ISPs or conglomerates of regional ISPs to leverage their network infrastructure to generate hosting revenue, and to save on network bandwidth. An ISP offering the inventive global hosting scheme can give content providers the ability to distribute content to their users from the closest point on the ISP's network, thus ensuring fast and reliable access. Guaranteed web site performance is critical for any web-based business, and global hosting allows for the creation of a service that satisfies this need.

Global hosting according to the present invention also allows an ISP to control how and where content traverses its network. Global hosting servers can be set up at the edges of the ISP's network (at the many network exchange and access points, for example). This enables the ISP to serve content for sites that it hosts directly into the network exchange points and access points. Expensive backbone links no longer have to carry redundant traffic from the content provider's site to the network exchange and access points. Instead, the content is served directly out of the ISP's network, freeing valuable network resources for other traffic.

Although global hosting reduces network traffic, it is also a method by which global ISPs may capture a piece of the rapidly expanding hosting market, which is currently estimated at over a billion dollars a year.

The global hosting solution also provides numerous advantages to Content Providers, and, in particular, an efficient and cost-effective solution to improve the performance of their Web sites both domestically and internationally. The inventive hosting software ensures Content Providers with fast and reliable Internet access by providing a means to distribute content to their subscribers from the closest point on an ISP's network. In addition to other benefits described in more detail below, the global hosting solution also provides the important benefit of reducing network traffic.

Once inexpensive global hosting servers are installed at the periphery of an ISP's network (i.e., at the many network exchange and access points), content is served directly into network exchange and access points. As a result of this efficient distribution of content directly from an ISP's network, the present invention substantially improves Web

site performance. In contrast to current content distribution systems, the inventive global hosting solution does not require expensive backbone links to carry redundant traffic from the Content Provider's Web site to the network exchange and access points.

A summary of the specific advantages afforded by the inventive global hosting scheme are set forth below:

1. Decreased Operational Expenses for Content Providers:

Most competing solutions require Content Providers to purchase servers at each Web site that hosts their content. As a result, Content Providers often must negotiate separate contracts with different ISPs around the world. In addition, Content Providers are generally responsible for replicating the content and maintaining servers in these remote locations.

With the present invention, ISPs are primarily responsible for the majority of the aspects of the global hosting. Content Providers preferably maintain only their single source server. Content on this server is automatically replicated by software to the locations where it is being accessed. No intervention or planning is needed by the Provider (or, for that matter, the ISP). Content Providers are offered instant access to all of the servers on the global network; there is no need to choose where content should be replicated or to purchase additional servers in remote locations.

2. Intelligent and Efficient Data Replication:

Most competing solutions require Content Providers to replicate their content on servers at a commercial hosting site or to mirror their content on geographically distant servers. Neither approach is particularly efficient. In the former situation, content is still located at a single location on the Internet (and thus it is far away from most users). In the latter case, the entire content of a Web site is copied to remote servers, even though only a small portion of the content may actually need to be located remotely. Even with inexpensive memory, the excessive cost associated with such mirroring makes it uneconomical to mirror to more than a few sites, which means that most users will still be far away from a mirror site. Mirroring also has the added disadvantage that Content Providers must insure that all sites remain consistent and current, which is a nontrivial task for even a few sites.

With the present invention, content is automatically replicated to the global server network in an intelligent and efficient fashion. Content is replicated in only those locations where it is needed. Moreover, when the content changes, new copies preferably are replicated automatically throughout the network.

3. Automatic Content Management:

Many existing solutions require active management of content distribution, content replication and load balancing between different servers. In particular, decisions about where content will be hosted must be made manually, and the process of replicating data is handled in a centralized push fashion. On the contrary, the invention features passive management. Replication is done in a demand-based pull fashion so that content preferably is only sent to where it is truly needed. Moreover, the process preferably is fully automated; the ISP does not have to worry about how and where content is replicated and/or the content provider.

4. Unlimited, Cost Effective Scalability:

Competing solutions are not scalable to more than a small number of sites. For example, solutions based on mirroring are typically used in connection with at most three or four sites. The barriers to scaling include the expense of replicating the entire site, the cost of replicating computing

resources at all nodes, and the complexity of supporting the widely varying software packages that Content Providers use on their servers.

The unique system architecture of the present invention is scalable to hundreds, thousands or even millions of nodes. Servers in the hosting network can malfunction or crash and the system's overall function is not affected. The global hosting framework makes efficient use of resources; servers and client software do not need to be replicated at every node because only the hosting server runs at each node. In addition, the global hosting server is designed to run on standard simple hardware that is not required to be highly fault tolerant.

5. Protection against Flash Crowds:

Competing solutions do not provide the Content Provider with protection from unexpected flash crowds. Although mirroring and related load-balancing solutions do allow a Content Provider to distribute load across a collection of servers, the aggregate capacity of the servers must be sufficient to handle peak demands. This means that the Provider must purchase and maintain a level of resources commensurate with the anticipated peak load instead of the true average load. Given the highly variable and unpredictable nature of the Internet, such solutions are expensive and highly wasteful of resources.

The inventive hosting architecture allows ISPs to utilize a single network of hosting servers to offer Content Providers flash crowd insurance. That is, insurance that the network will automatically adapt to and support unexpected higher load on the Provider's site. Because the ISP is aggregating many Providers together on the same global network, resources are more efficiently used.

6. Substantial Bandwidth Savings:

Competing solutions do not afford substantial bandwidth savings to ISPs or Content Providers. Through the use of mirroring, it is possible to save bandwidth over certain links (i.e., between New York and Los Angeles). Without global hosting, however, most requests for content will still need to transit the Internet, thus incurring bandwidth costs. The inventive hosting framework saves substantial backbone bandwidth for ISPs that have their own backbones. Because content is distributed throughout the network and can be placed next to network exchange points, both ISPs and Content Providers experience substantial savings because backbone charges are not incurred for most content requests.

7. Instant Access to the Global Network:

Competing solutions require the Content Provider to choose manually a small collection of sites at which content will be hosted and/or replicated. Even if the ISP has numerous hosting sites in widely varied locations, only those sites specifically chosen (and paid for) will be used to host content for that Content Provider.

On the contrary, the global hosting solution of the present invention allows ISPs to offer their clients instant access to the global network of servers. To provide instant access to the global network, content is preferably constantly and dynamically moved around the network. For example, if a Content Provider adds content that will be of interest to customers located in Asia, the Content Provider will be assured that its content will be automatically moved to servers that are also located in Asia. In addition, the global hosting framework allows the content to be moved very close to end users (even as close as the user's building in the case of the Enterprise market).

8. Designed for Global ISPs and Conglomerates:

Most competing solutions are designed to be purchased and managed by Content Providers, many of whom are

already consistently challenged and consumed by the administrative and operational tasks of managing a single server. The inventive hosting scheme may be deployed by a global ISP, and it provides a new service that can be offered to Content Providers. A feature of the service is that it minimizes the operational and managerial requirements of a Content Provider, thus allowing the Content Provider to focus on its core business of creating unique content.

9. Effective Control of Proprietary Database s and Confidential Information:

Many competing solutions require Content Providers to replicate their proprietary databases to multiple geographically distant sites. As a result, the Content Provider effectively loses control over its proprietary and usually confidential databases. To remedy these problems, the global hosting solution of the present invention ensures that Content Providers retain complete control over their databases. As described above, initial requests for content are directed to the Content Provider's central Web site, which then implements effective and controlled database access. Preferably, high-bandwidth, static parts for page requests are retrieved from the global hosting network.

10. Compatibility with Content Provider Software:

Many competing solutions require Content Provider s to utilize a specific set of servers and databases. These particular, non-uniform requirements constrain the Content Provider's ability to most effectively use new technologies, and may require expensive changes to a Content Provider's existing infrastructure. By eliminating these problems, the inventive global hosting architecture effectively interfaces between the Content Provider and the ISP, and it does not make any assumptions about the systems or servers used by the Content Provider. Furthermore, the Content Provider's systems can be upgraded, changed or completely replaced without modifying or interrupting the inventive architecture.

11. No Interference with Dynamic Content, Personalized Advertising or E-Commerce, and No Stale content:

Many competing solutions (such as naive caching of all content) can interfere with dynamic content, personalized advertising and E-commerce and can serve the user with stale content. While other software companies have attempted to partially eliminate these issues (such as keeping counts on hits for all cached copies), each of these solutions causes a partial or complete loss of functionality (such as the ability to personalize advertising). On the contrary, the global hosting solution does not interfere with generation of dynamic content, personalized advertising or E-commerce, because each of these tasks preferably is handled by the central server of the Content Provider.

12. Designed for the Global Network:

The global hosting architecture is highly scaleable and thus may be deployed on a world-wide network basis.

The above-described functionality of each of the components of the global hosting architecture preferably is implemented in software executable in a processor, namely, as a set of instructions or program code in a code module resident in the random access memory of the computer. Until required by the computer, the set of instructions may be stored in another computer memory, for example, in a hard disk drive, or in a removable memory such as an optical disk (for eventual use in a CD ROM) or floppy disk (for eventual use in a floppy disk drive), or downloaded via the Internet or other computer network.

In addition, although the various methods described are conveniently implemented in a general purpose computer selectively activated or reconfigured by software, one of ordinary skill in the art would also recognize that such

methods may be carried out in hardware, in firmware, or in more specialized apparatus constructed to perform the required method steps.

Further, as used herein, a Web "client" should be broadly construed to mean any computer or component thereof directly or indirectly connected or connectable in any known or later-developed manner to a computer network, such as the Internet. The term Web "server" should also be broadly construed to mean a computer, computer platform, an adjunct to a computer or platform, or any component thereof. Of course, a "client" should be broadly construed to mean one who requests or gets the file, and "server" is the entity which downloads the file.

Having thus described our invention, what we claim as new and desire to secure by Letters Patent is set forth in the following claims:

1. A distributed hosting framework operative in a computer network in which users of client machines connect to a content provider server, the framework comprising:

a routine for modifying at least one embedded object URL of a web page to include a hostname pretended to a domain name and path;

a set of content servers, distinct from the content provider server, for hosting at least some of the embedded objects of web pages that are normally hosted by the content provider server;

at least one first level name server that provides a first level domain name service (DNS) resolution; and

at least one second level name server that provides a second level domain name service (DNS) resolution;

wherein in response to requests for the web page, generated by the client machines the web page including the modified embedded object URL is served from the content provider server and the embedded object identified by the modified embedded object URL is served from a given one of the content servers as identified by the first level and second level name servers.

2. The hosting framework as described in claim 1 further including a redundant first level name server.

3. The hosting framework as described in claim 1 further including a redundant second level name server.

4. The hosting framework as described in claim 1 wherein a given one of the set of servers includes a buddy server for assuming the hosting responsibilities of the given one of the set of servers upon a given failure condition.

5. The hosting framework as described in claim 1 wherein the second level name server includes a load balancing mechanism that balances loads across a subset of the set of servers.

6. The hosting framework as described in claim 5 wherein the load balancing mechanism minimizes the amount of replication required for the embedded objects while not exceeding a capacity of any of the set of servers.

7. The hosting framework as described in claim 1 further including an overflow control mechanism for minimizing an overall amount of latency experienced by client machines while not exceeding the capacity of any given subset of the set of servers.

8. The hosting framework as described in claim 7 wherein the overflow control mechanism includes a min-cost multi-commodity flow algorithm.

9. The hosting framework as described in claim 1 wherein the first level name server includes a network map for use in directing a request for the embedded object generated by a client.

10. The hosting framework as described in claim 1 wherein a server in the set of servers includes a gating

mechanism for maintaining overall traffic for a given embedded object within specified limits.

11. The hosting framework as described in claim 10 wherein the gating mechanism comprises:

means for determining whether a number of requests for the given embedded object exceeds a given threshold; and

means responsive to the determining means for restricting service of the given embedded object.

12. The hosting framework as described in claim 11 wherein the restricting means comprises means for serving an object that is smaller than the given embedded object.

13. The hosting framework as described in claim 11 wherein the object is a ticket that allows a client to receive the given embedded object at a later time.

14. A method of serving a page supported at a content provider server, the page comprising a markup language base document having associated therewith a set of embedded objects, each embedded object identified by a URL, comprising the steps of:

rewriting the URL of an embedded object to generate a modified URL, the modified URL including a new hostname prepended to an original hostname, wherein the original hostname is maintained as part of the modified URL for use in retrieving the embedded object whenever a cached copy of the embedded object is not available;

in response to a request to serve the page received at the content provider site, serving the page with the modified URL;

attempting to serve the embedded object from a content server other than the content provider server as identified by the new hostname; and

if the cached copy of the embedded object is not available from the content server, serving the embedded object from the content provider server.

15. A method of serving a page and an associated page object, wherein the page is stored on a content provider server and copies of the page object are stored on a set of content servers distinct from the content provider server, comprising the steps of:

(a) modifying a URI for the page object to include a hostname prepended to a content provider-supplied domain name and path;

(b) serving the page from the content provider server with the modified URL;

(c) responsive to a browser query to resolve the hostname, identifying a given one of the set of content servers from which the object may be retrieved; and

(d) returning to the browser an IP address of the identified content server to enable the browser to attempt to retrieve the object from that content server.

16. The method as described in claim 15 wherein the copies of the page object are stored on a subset of the set of content servers.

17. A content delivery method, comprising:

tagging an embedded object in a page to resolve to a domain other than a content provider domain by prepending given data to a content provider-supplied URL to generate an alternate resource locator (ARL); serving the page from a content provider server with the ARL; and

resolving the ARL to identify a content server in the domain; and

serving the embedded object from the identified content server.

19

18. The method as described in claim 17 wherein the step of resolving the ARL comprises:
- utilizing a requesting user's location and data identifying then-current Internet traffic conditions to identify the content server.
19. A content delivery service, comprising:
- replicating a set of page objects across a wide area network of content servers managed by a domain other than a content provider domain;
 - for a given page normally served from the content provider domain, tagging the embedded objects of the page so that requests for the page objects resolve to the domain instead of the content provider domain;
 - responsive to a request for the given page received at the content provider domain, serving the given page from the content provider domain; and
 - serving at least one embedded object of the given page from a given content server in the domain instead of from the content provider domain.
20. The content delivery method as described in claim 19 wherein the serving step comprises:
- for each embedded object, identifying one or more content servers from which the embedded object may be retrieved.
21. The method as described in claim 20 wherein the identifying step comprises:
- resolving a request to the domain as a function of a requesting user's location.
22. The method as described in claim 21 wherein the identifying step comprises:
- resolving a request to the domain as a function of a requesting user's location and then-current Internet traffic conditions.
23. A method for Internet content delivery, comprising:
- at the content provider server, modifying at least one embedded object URL of a page to include a hostname prepended to a domain name and a path normally used to retrieve the embedded object;
 - responsive to a request for the page issued from a client machine, serving the page with the modified embedded object URL to the client machine from the content provider server;
 - responsive to a request for the embedded object, resolving the hostname to an IP address of a content server, other than the content provider server, that is likely to host the embedded object; and
 - attempting to serve the embedded object to the client from the content server.
24. The method as described in claim 23 wherein the hostname includes a value generated by applying a given function to the embedded object.

20

25. The method as described in claim 24 wherein the value is generated by encoding given information, the given information selected from a group of information consisting essentially of: size data, popularity data, creation data and object type data.
26. The method as described in claim 4 wherein the given function randomly associates the embedded object with a virtual content bucket.
27. The method as described in claim 26 wherein the given function is an encoding function.
28. The method as described in claim 26 wherein the given function is a hash function.
29. The method as described in claim 23 wherein the modified URL also includes a fingerprint value generated by applying a given function to the embedded object.
30. The method as described in claim 29 wherein the value is a number generated by hashing the embedded object.
31. The method as described in claim 23 wherein the page is formatted according to a markup language.
32. The method as described in claim 23 further including the step of rewriting the embedded object URL as the content provider modifies the page.
33. The method as described in claim 23 wherein the step of resolving the hostname includes:
- identifying a subset of content servers that may be available to serve the embedded object based on a location of the client machine and current Internet traffic conditions; and
 - identifying the content server from the subset of content servers.
34. A content delivery method, comprising:
- distributing a set of page objects across a network of content servers managed by a domain other than a content provider domain, wherein the network of content servers are organized into a set of regions;
 - for a given page normally served from the content provider domain, tagging at least some of the embedded objects of the page so that requests for the objects resolve to the domain instead of the content provider domain;
 - in response to a client request for an embedded object of the page:
 - resolving the client request as a function of a location of the client machine making the request and current Internet traffic conditions to identify a given region; and
 - returning to the client an IP address of a given one of the content servers within the given region that is likely to host the embedded object and that is not overloaded.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

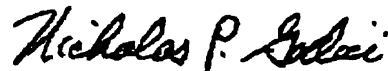
PATENT NO. : 6,108,703
DATED : Aug. 22, 2000
INVENTOR(S) : F. Thomson Leighton, Daniel M. Lewin

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 11, line 1, delete "assachusetts" and substitute -- Massachusetts --.
In Column 11, line 2, delete "ncorporated" and substitute -- incorporated --.
In Column 11, line 3, delete "ake" and substitute -- make --.
In Column 16, line 9, delete "Database s a nd" and substitute -- Databases and --.
In Column 16, line 20, delete "a nd" and substitute -- and --.
In Column 16, line 24, delete "Provider s" and substitute -- Providers --.
In Claim 1, Column 17, line 21, delete "pretended" and substitute -- prepedended --.
In Claim 1, Column 17, line 32, after "machines" insert -- , --.
In Claim 17, Column 18, line 64, delete "ARt" and substitute -- ARL --.

Signed and Sealed this
Fifteenth Day of May, 2001

Attest:



NICHOLAS P. GODICI

Attesting Officer

Acting Director of the United States Patent and Trademark Office



US006553413B1

(12) **United States Patent**
Leighton et al.

(10) Patent No.: **US 6,553,413 B1**
(45) Date of Patent: **Apr. 22, 2003**

(54) **CONTENT DELIVERY NETWORK USING EDGE-OF-NETWORK SERVERS FOR PROVIDING CONTENT DELIVERY TO A SET OF PARTICIPATING CONTENT PROVIDERS**

EP 0817444 A2 * 7/1998 H04L/29/06
EP 865180 A2 9/1998
WO 9804985 2/1998

OTHER PUBLICATIONS

(75) Inventors: **F. Thomson Leighton**, Newtonville, MA (US); **Daniel M. Lewin**, Charlestown, MA (US)

Beavan, Colin "Web Life They're Watching You." *Esquire*, Aug. 1997, pp. 104-105.

(73) Assignee: **Massachusetts Institute of Technology**, Cambridge, MA (US)

Bestavros, Azer. "Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time in Distributed Information Systems." In *Proceedings of ICDE '96: The 1996 International Conference on Data Engineering*, Mar. 1996, 4 pgs.

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 329 days.

Carter, J. Lawrence, et al. "Universal Classes of Hash Functions." *Journal of Computer and System Sciences*, vol. 18, No. 2, Apr. 1979, pp. 143-154.

(21) Appl. No.: **09/604,878**

(List continued on next page.)

(22) Filed: **Jun. 28, 2000**

Related U.S. Application Data

Primary Examiner—David Wiley
Assistant Examiner—April L Baugh
(74) *Attorney, Agent, or Firm*—David H. Judson

(63) Continuation of application No. 09/314,863, filed on May 19, 1999, now Pat. No. 6,108,703.

(60) Provisional application No. 60/092,710, filed on Jul. 14, 1998.

(51) **Int. Cl.⁷ G06F 13/00**

(52) **U.S. Cl. 709/219; 709/200; 709/217; 709/218; 395/800**

(58) **Field of Search 395/800; 709/200, 709/217, 218, 219**

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,922,417 A	5/1990	Churm et al.	707/1
5,136,716 A *	8/1992	Harvey et al.	709/203
5,287,499 A	2/1994	Nemes	707/2
5,341,477 A	8/1994	Pitkin et al.	709/226
5,542,087 A	7/1996	Neimat et al.	707/10

(List continued on next page.)

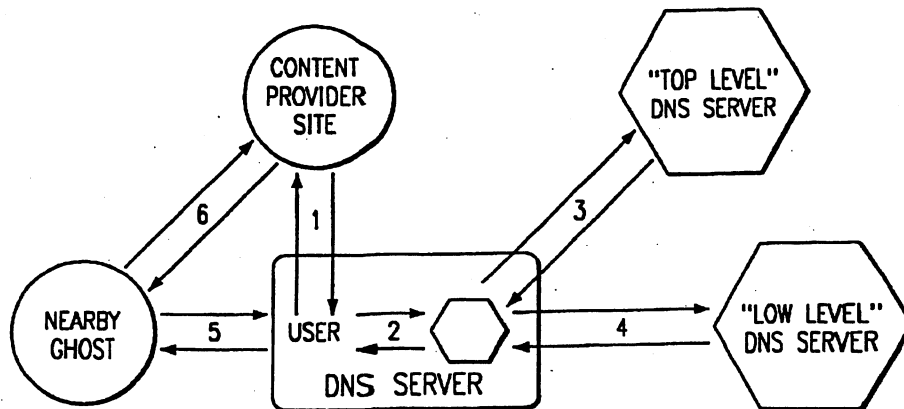
FOREIGN PATENT DOCUMENTS

CA	2202572	10/1998
EP	0 817 444 A	1/1998

(57) **ABSTRACT**

The present invention is a network architecture or framework that supports hosting and content distribution on a truly global scale. The inventive framework allows a Content Provider to replicate and serve its most popular content at an unlimited number of points throughout the world. The inventive framework comprises a set of servers operating in a distributed manner. The actual content to be served is preferably supported on a set of hosting servers (sometimes referred to as ghost servers). This content comprises HTML page objects that, conventionally, are served from a Content Provider site. In accordance with the invention, however, a base HTML document portion of a Web page is served from the Content Provider's site while one or more embedded objects for the page are served from the hosting servers, preferably, those hosting servers near the client machine. By serving the base HTML document from the Content Provider's site, the Content Provider maintains control over the content.

22 Claims, 2 Drawing Sheets



U.S. PATENT DOCUMENTS

5,638,443 A	6/1997	Stefik et al.	705/54
5,646,676 A	7/1997	Dewkett et al.	348/7
5,715,453 A	2/1998	Stewart	707/104
5,740,423 A	4/1998	Logan et al.	707/10
5,751,961 A	5/1998	Smyk	709/217
5,761,507 A *	6/1998	Govett	395/684
5,774,660 A *	6/1998	Brendel et al.	709/201
5,777,989 A *	7/1998	McGarvey	370/254
5,802,291 A	9/1998	Balick et al.	709/202
5,832,506 A	11/1998	Kuzma	707/200
5,856,974 A	1/1999	Gervais et al.	370/392
5,870,559 A	2/1999	Leshem et al.	709/224
5,878,212 A	3/1999	Civanlar et al.	709/203
5,884,038 A *	3/1999	Kapoor	709/226
5,894,554 A	4/1999	Lowery et al.	
5,903,723 A	5/1999	Beck et al.	709/200
5,919,247 A *	7/1999	Van Hoff et al.	709/217
5,920,701 A *	7/1999	Miller et al.	709/228
5,933,832 A	8/1999	Suzuoka et al.	707/101
5,945,989 A	8/1999	Freishtat et al.	345/329
5,956,716 A *	9/1999	Kenner et al.	707/10
5,961,596 A *	10/1999	Takubo et al.	709/224
5,991,809 A	11/1999	Kriegsman	709/226
6,003,030 A	12/1999	Kenner et al.	707/10
6,006,264 A *	12/1999	Colby et al.	709/226
6,052,718 A *	4/2000	Gifford	709/219
6,112,239 A *	8/2000	Kenner et al.	709/105
6,115,752 A *	9/2000	Chauhan	709/238
6,119,143 A	9/2000	Dias et al.	
6,134,583 A *	10/2000	Herriot	707/501.1
6,144,996 A *	11/2000	Starnes et al.	709/217
6,154,744 A	11/2000	Kenner et al.	
6,178,160 B1	1/2001	Bolton et al.	
6,181,867 B1	1/2001	Kenner et al.	
6,185,598 B1	2/2001	Farber et al.	
6,185,619 B1	2/2001	Joffe et al.	
6,230,196 B1	5/2001	Guenthner et al.	
6,256,675 B1	7/2001	Rabinovich	
6,269,394 B1	7/2001	Kenner et al.	
6,314,565 B1	11/2001	Kenner et al.	
6,370,571 B1	4/2002	Medin, Jr.	

OTHER PUBLICATIONS

- Chankhunthod, Anawat, et al. "A Hierarchical Internet Object Cache." In *USENIX Proceedings*, Jan. 1996, pp. 153-163.
- Cormen, Thomas H., et al. *Introduction to Algorithms*, The MIT Press, Cambridge, Massachusetts, 1994, pp. 219-243, 991-993.
- Deering, Stephen, et al. "Multicast Routing in Datagram Internetworks and Extended LANs." *ACM Transactions on Computer Systems*, vol. 8, No. 2, May 1990, pp. 85-110.
- Devine, Robert. "Design and Implementation of DDH: A Distributed Dynamic Hashing Algorithm." In *Proceedings of 4th International Conference on Foundations of Data Organizations and Algorithms*, 1993, pp. 101-114.
- Grigni, Michelangelo, et al. "Tight Bounds on Minimum Broadcasts Networks." *SIAM Journal of Discrete Mathematics*, vol. 4, No. 2, May 1991, pp. 207-222.
- Gwartzman, James, et al. "The Case for Geographical Push-Caching." *Technical Report HU TR 34-94 (excerpt)*, Harvard University, DAS, Cambridge, MA 02138, 1994, 2 pgs.
- Gwartzman, James, et al. "World-Wide Web Cache Consistency." In *Proceedings of the 1996 USENIX Technical Conference*, Jan. 1996, 8 pgs.

- Feeley, Michael, et al. "Implementing Global Memory Management in a Workstation Cluster." In *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, 1995, pp. 201-212.
- Floyd, Sally, et al. "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing." In *Proceeding of ACM SIGCOMM'95*, pp. 342-356.
- Fredman, Michael, et al. "Storing a Sparse Table with $O(1)$ Worst Case Access Time." *Journal of the Association for Computing Machinery*, vol. 31., No. 3, Jul. 1984, pp. 538-544.
- Karger, David, et al. "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web." In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, May 1997, pp. 654-663.
- Litwin, Withold, et al. "LH—A Scaleable, Distributed Data Structure." *ACM Transactions on Database Systems*, vol. 21, No. 4, Dec. 1996, pp. 480-525.
- Malpani, Radhika, et al. "Making World Wide Web Caching Servers Cooperate." In *Proceedings of World Wide Web Conference*, 1996, 6 pgs.
- Naor, Moni, et al. "The Load, Capacity and Availability of Quorum Systems." In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, Nov. 1994, pp. 214-225.
- Nisan, Noam. "Pseudorandom Generators for Space-Bounded Computation." In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, May 1990, pp. 204-212.
- Palmer, Mark, et al. "Fido: A Cache that Learns to Fetch." In *Proceedings of the 17th International Conference on Very Large Data Bases*, Sep. 1991, pp. 255-264.
- Panigrahy, Rina. *Relieving Hot Spots on the World Wide Web*. Massachusetts Institute of Technology, Jun. 1997, pp. 1-66.
- Peleg, David, et al. "The Availability of Quorum Systems." *Information and Computation* 123, 1995, 210-223.
- Plaxton, C. Greg, et al. "Fast Fault-Tolerant Concurrent Access to Shared Objects." In *Proceedings of 37th IEEE Symposium on Foundations of Computer Science*, 1996, pp. 570-579.
- Rabin, Michael. "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance." *Journal of the ACM*, vol. 36, No. 2, Apr. 1989, pp. 335-348.
- Ravi, R., "Rapid Rumor Ramification: Approximating the Minimum Broadcast Time." In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, Nov. 1994, pp. 202-213.
- Schmidt, Jeanette, "Chernoff-Hoeffding Bounds for Applications with Limited Independence." In *Proceedings of the 4th ACS-SIAM Symposium on Discrete Algorithms*, 1993, pp. 331-340.
- Tarjan, Robert Endre, et al. "Storing a Sparse Table." *Communications of the ACM*, vol. 22, No. 11, Nov. 1979, pp. 606-611.
- Vitter, Jeffrey Scott, et al. "Optimal Prefetching via Data Compression." In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, Nov. 1991, pp. 121-130.
- Wegman, Mark, et al. "New Hash Functions and Their Use in Authentication and Set Equality." *Journal of Computer and System Sciences* vol. 22, Jun. 1981, pp. 265-279.

- Yao, Andrew Chi-Chih. "Should Tables be Sorted?" *Journal of the Association for Computing Machinery*, vol. 28, No. 3, Jul. 1981, pp. 615-628.
- Shaw, David M. "A Low Latency, High Throughput Web Service Using Internet-wide Replication." Department of Computer Science, Johns Hopkins University, Aug. 1998, 33 pgs.
- Amir, Yair, et al. "Seamlessly Selecting the Best Copy from Internet-Wide Replicated Web Servers." Department of Computer Science, Johns Hopkins University, Jun. 1998, 14 pgs.
- Braun et al., "Web traffic characterization: an assessment of the impact of caching documents from NCSA's web server, Proceedings of the Second Int'l WWW Conference, Sep. 1994.
- "Cisco Distributed Director", http://www.cisco.com/warp/public/751/distdir/dd_wp.htm posted Feb. 21, 1997.
- "How to Cost-Effectively Scale Web Servers", <http://www.cisco.com/warp/public/784/5.html> posted Nov. 12, 1996.
- "Ibnamed, a load balancing name server written in Perl", <http://www.stanford.edu/~schemers/docs/Ibnamed/Ibnamed.html> Jan. 19, 1995.
- "Exporting Web Server Final Report", http://www.cs.technion.ac.il/Labs/Lccn/projects/spring97/project4/final_report.html, Spring 1997.
- Jeffrey et al., "Proxy-Sharing Proxy Servers", 1996 IEEE, pp. 116-119.
- Excerpts from www.sandpiper.com Web site, Nov. 27, 1999 (14 pages).
- Luotonen et al., "World-Wide Web Proxies" CERN Apr. 1994.
- Oguchi et al., "A Study of Caching Proxy Mechanisms Realized on Wide Area Distributed Networks", High Performance Distributed Computing, 1996 5th Int'l Symposium, pp. 443-449.
- Kwan et al., "NCSA's World Wide Web Server: Design and Performance" IEEE Nov. 1995, pp. 68-74.
- Malpani et al., "Making World Wide Web Caching Servers Cooperate", <http://bmc.berkeley.edu/papers/1995/138/paper-59.html>, 1995.
- Ross, "Hash Routing for Collections of Shared Web Caches", IEEE Network Nov./Dec. 1997 pp. 37-44.
- Reverse Proxy Content Re-Mapper for Netscape Proxy 2.52 & 2.53 <http://help.netscape.com/products/server/proxy/documentation/rpMapper.html>, Nov. 1997.
- "Super Proxy Script—How to make distributed proxy servers by URL hashing" Sharp, <http://naragw.sharp.co.jp/sps/> Copyright 1996-2000.
- Mark E Crovella and Robert L. Carter. Dynamic server selection in the internet. In Third IEEE Workshop on the Architecture and Implementation of High Performance Computer Systems'95, pp. 158-162, Mystic, Connecticut, Aug. 1995.
- S. Bhattacharjee, M. H. Ammar, E. W. Zegura, V. Shah, and Z. Fei. Application-layer anycasting. In Proceedings of the IEEE INFOCOM '97, 1997.
- Z. Fei, S. Bhattacharjee, E. W. Zegura, and M. H. Ammar, "A novel server selection technique for improving the response time of a replicated service," in Proceedings of INFOCOM '98, Mar. 1998.
- R. Carter and M. Crovella. Server selection using dynamic path characterization in Wide-Area Networks. in IEEE Infocom'97, 1997.
- J. Guyton and M. Schwartz. Locating nearby copies of replicated internet servers. In Proceedings of ACM SIGCOMM'95, pp. 288-298, 1995.
- M. Seltzer and J. Gwertzman. The Case for Geographical Pushcaching. In Proceedings of the 1995 Workshop on Hot Operating Systems, 1995.
- Bestavros, A. and C. Cunha (1996), "Server-initiated Document Dissemination for the WWW," IEEE Data Engineering Bulletin 19, 3-11.
- R. L. Carter and M. E. Crovella. Dynamic server selection using bandwidth probing in wide-area networks. Technical Report BU-CS-96-007, Computer Science Department, Boston University, Mar. 1996.
- Baker S. M. et al. "Distributed cooperative Web servers" Computer Networks, Elsevier Science Publishers, B.V., vol. 31, No. 11-16, May 17, 1999, pp. 1215-1229.
- Karger D. et al. "Web caching with consistent hashing" Computer Networks, Elsevier Science Publishers, B.V., vol. 31, No. 11-16, May 17, 1999, pp. 1203-1213.
- Berners-Lee et al., RFC 1738—Uniform Resource Locators, Dec., 1994.
- Postel, RFC 1591—Domain Name System Structure and Delegation, Mar., 1994.
- Mockapetris et al., Development of the Domain Name System, Proceedings of SIGCOMM '88 Computer Communication Review, vol. 18, No. 4, Aug. 1988.
- Wessels, Intelligent Caching For World-Wide Web Objects, Masters Thesis, University of Colorado, 1995.
- Smith, What can Archives offer the World Wide Web? Mar. 22, 1994.
- Albitz et al., "How Does DNS Work?" Chapter 2, DNS and BIND, O'Reilly & Associates, Inc., 1992, pp. 13-38.
- Overview of the Cisco DistributedDirector 2500 Series, actual publication date unknown, but believed to be in 1997.

* cited by examiner

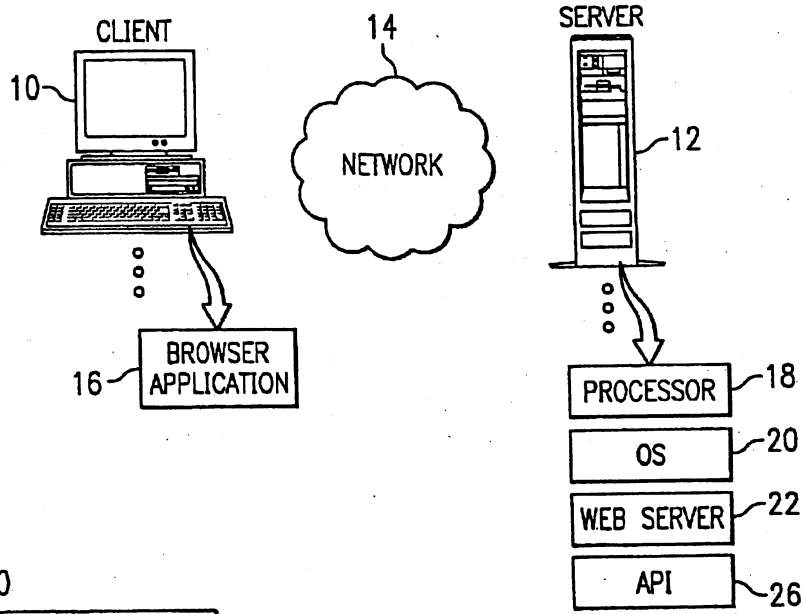


FIG. 1

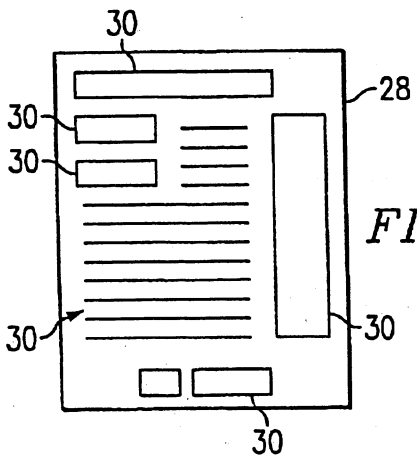


FIG. 2

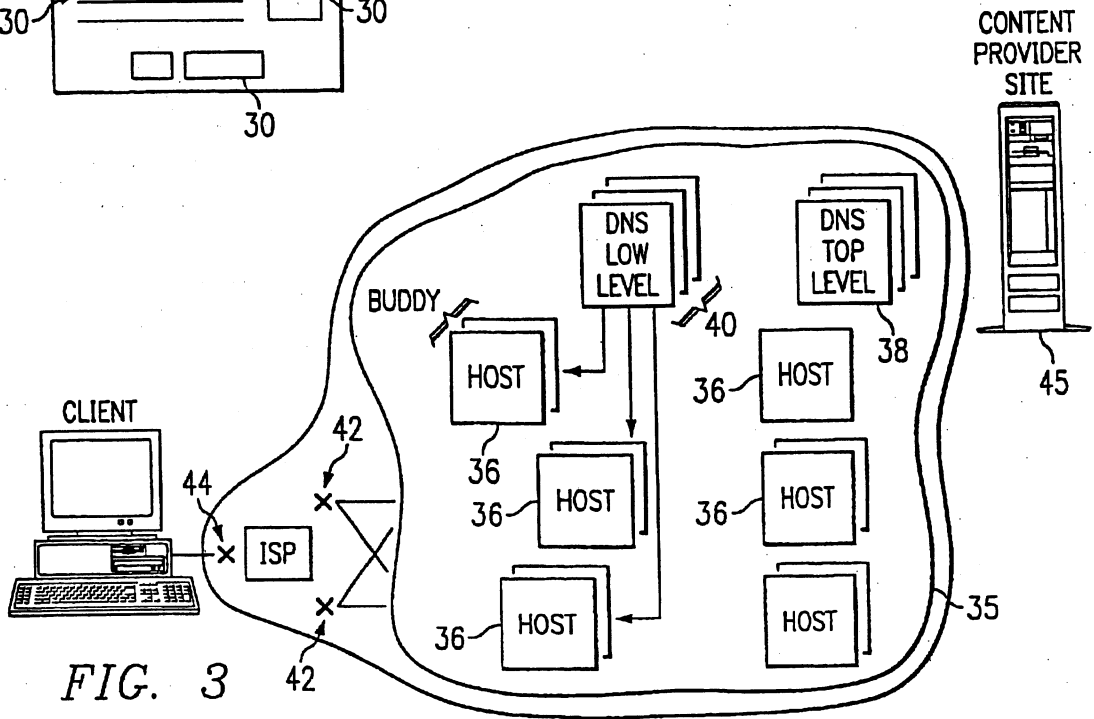


FIG. 3

FIG. 4

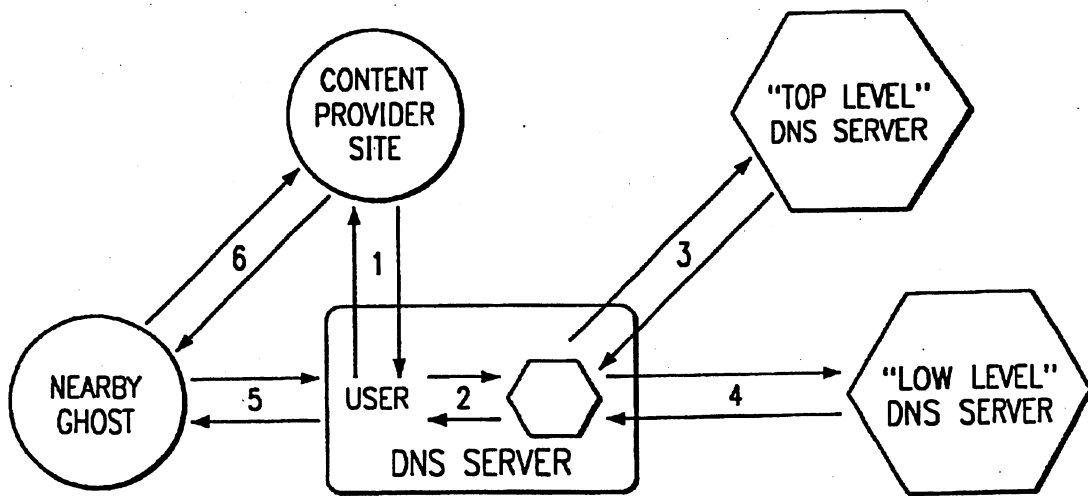
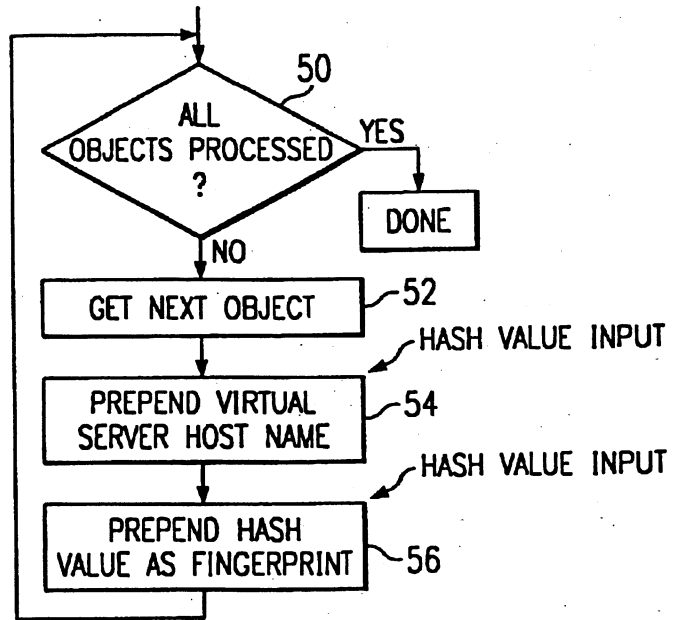


FIG. 5

1

**CONTENT DELIVERY NETWORK USING
EDGE-OF-NETWORK SERVERS FOR
PROVIDING CONTENT DELIVERY TO A
SET OF PARTICIPATING CONTENT
PROVIDERS**

This application is a continuation of prior application Ser. No. 09/314,863, filed May 19, 1999, now U.S. Pat. No. 6,108,703, which application is based on and claims priority from Provisional Application No. 60/092,710, filed Jul. 14, 1998.

BACKGROUND OF THE INVENTION

1. Technical Field

This invention relates generally to information retrieval in a computer network. More particularly, the invention relates to a novel method of hosting and distributing content on the Internet that addresses the problems of Internet Service Providers (ISPs) and Internet Content Providers.

2. Description of the Related Art

The World Wide Web is the Internet's multimedia information retrieval system. In the Web environment, client machines effect transactions to Web servers using the Hypertext Transfer Protocol (HTTP), which is a known application protocol providing users access to files (e.g., text, graphics, images, sound, video, etc.) using a standard page description language known as Hypertext Markup Language (HTML). HTML provides basic document formatting and allows the developer to specify "links" to other servers and files. In the Internet paradigm, a network path to a server is identified by a so-called Uniform Resource Locator (URL) having a special syntax for defining a network connection. Use of an HTML-compatible browser (e.g., Netscape Navigator or Microsoft Internet Explorer) at a client machine involves specification of a link via the URL. In response, the client makes a request to the server identified in the link and, in return, receives a document or other object formatted according to HTML. A collection of documents supported on a Web server is sometimes referred to as a Web site.

It is well known in the prior art for a Web site to mirror its content at another server. Indeed, at present, the only method for a Content Provider to place its content closer to its readers is to build copies of its Web site on machines that are located at Web hosting farms in different locations domestically and internationally. These copies of Web sites are known as mirror sites. Unfortunately, mirror sites place unnecessary economic and operational burdens on Content Providers, and they do not offer economies of scale. Economically, the overall cost to a Content Provider with one primary site and one mirror site is more than twice the cost of a single primary site. This additional cost is the result of two factors: (1) the Content Provider must contract with a separate hosting facility for each mirror site, and (2) the Content Provider must incur additional overhead expenses associated with keeping the mirror sites synchronized.

In an effort to address problems associated with mirroring, companies such as Cisco, Resonate, Bright Tiger, F5 Labs and Alteon, are developing software and hardware that will help keep mirror sites synchronized and load balanced. Although these mechanisms are helpful to the Content Provider, they fail to address the underlying problem of scalability. Even if a Content Provider is willing to incur the costs associated with mirroring, the technology itself will not scale beyond a few (i.e., less than 10) Web sites.

In addition to these economic and scalability issues, mirroring also entails operational difficulties. A Content

2

Provider that uses a mirror site must not only lease and manage physical space in distant locations, but it must also buy and maintain the software or hardware that synchronizes and load balances the sites. Current solutions require Content Providers to supply personnel, technology and other items necessary to maintain multiple Web sites. In summary, mirroring requires Content Providers to waste economic and other resources on functions that are not relevant to their core business of creating content.

Moreover, Content Providers also desire to retain control of their content. Today, some ISPs are installing caching hardware that interrupts the link between the Content Provider and the end-user. The effect of such caching can produce devastating results to the Content Provider, including (1) preventing the Content Provider from obtaining accurate hit counts on its Web pages (thereby decreasing revenue from advertisers), (2) preventing the Content Provider from tailoring content and advertising to specific audiences (which severely limits the effectiveness of the Content Provider's Web page), and (3) providing outdated information to its customers (which can lead to a frustrated and angry end user).

There remains a significant need in the art to provide a decentralized hosting solution that enables users to obtain Internet content on a more efficient basis (i.e., without burdening network resources unnecessarily) and that likewise enables the Content Provider to maintain control over its content.

The present invention solves these and other problems associated with the prior art.

BRIEF SUMMARY OF THE INVENTION

It is a general object of the present invention to provide a computer network comprising a large number of widely deployed Internet servers that form an organic, massively fault-tolerant infrastructure designed to serve Web content efficiently, effectively, and reliably to end users.

Another more general object of the present invention is to provide a fundamentally new and better method to distribute Web-based content. The inventive architecture provides a method for intelligently routing and replicating content over a large network of distributed servers, preferably with no centralized control.

Another object of the present invention is to provide a network architecture that moves content close to the user. The inventive architecture allows Web sites to develop large audiences without worrying about building a massive infrastructure to handle the associated traffic.

Still another object of the present invention is to provide a fault-tolerant network for distributing Web content. The network architecture is used to speed-up the delivery of richer Web pages, and it allows Content Providers with large audiences to serve them reliably and economically, preferably from servers located close to end users.

A further feature of the present invention is the ability to distribute and manage content over a large network without disrupting the Content Provider's direct relationship with the end user.

Yet another feature of the present invention is to provide a distributed scalable infrastructure for the Internet that shifts the burden of Web content distribution from the Content Provider to a network of preferably hundreds of hosting servers deployed, for example, on a global basis.

In general, the present invention is a network architecture that supports hosting on a truly global scale. The inventive

framework allows a Content Provider to replicate its most popular content at an unlimited number of points throughout the world. As an additional feature, the actual content that is replicated at any one geographic location is specifically tailored to viewers in that location. Moreover, content is automatically sent to the location where it is requested, without any effort or overhead on the part of a Content Provider.

It is thus a more general object of this invention to provide a global hosting framework to enable Content Providers to retain control of their content.

The hosting framework of the present invention comprises a set of servers operating in a distributed manner. The actual content to be served is preferably supported on a set of hosting servers (sometimes referred to as ghost servers). This content comprises HTML page objects that, conventionally, are served from a Content Provider site. In accordance with the invention, however, a base HTML document portion of a Web page is served from the Content Provider's site while one or more embedded objects for the page are served from the hosting servers, preferably, those hosting servers nearest the client machine. By serving the base HTML document from the Content Provider's site, the Content Provider maintains control over the content.

The determination of which hosting server to use to serve a given embedded object is effected by other resources in the hosting framework. In particular, the framework includes a second set of servers (or server resources) that are configured to provide top level Domain Name Service (DNS). In addition, the framework also includes a third set of servers (or server resources) that are configured to provide low level DNS functionality. When a client machine issues an HTTP request to the Web site for a given Web page, the base HTML document is served from the Web site as previously noted. Embedded objects for the page preferably are served from particular hosting servers identified by the top- and low-level DNS servers. To locate the appropriate hosting servers to use, the top-level DNS server determines the user's location in the network to identify a given low-level DNS server to respond to the request for the embedded object. The top-level DNS server then redirects the request to the identified low-level DNS server that, in turn, resolves the request into an IP address for the given hosting server that serves the object back to the client.

More generally, it is possible (and, in some cases, desirable) to have a hierarchy of DNS servers that consisting of several levels. The lower one moves in the hierarchy, the closer one gets to the best region.

A further aspect of the invention is a means by which content can be distributed and replicated through a collection of servers so that the use of memory is optimized subject to the constraints that there are a sufficient number of copies of any object to satisfy the demand, the copies of objects are spread so that no server becomes overloaded, copies tend to be located on the same servers as time moves forward, and copies are located in regions close to the clients that are requesting them. Thus, servers operating within the framework do not keep copies of all of the content database. Rather, given servers keep copies of a minimal amount of data so that the entire system provides the required level of service. This aspect of the invention allows the hosting scheme to be far more efficient than schemes that cache everything everywhere, or that cache objects only in pre-specified locations.

The global hosting framework is fault tolerant at each level of operation. In particular, the top level DNS server

returns a list of low-level DNS servers that may be used by the client to service the request for the embedded object. Likewise, each hosting server preferably includes a buddy server that is used to assume the hosting responsibilities of its associated hosting server in the event of a failure condition.

According to the present invention, load balancing across the set of hosting servers is achieved in part through a novel technique for distributing the embedded object requests. In particular, each embedded object URL is preferably modified by prepending a virtual server hostname into the URL. More generally, the virtual server hostname is inserted into the URL. Preferably, the virtual server hostname includes a value (sometimes referred to as a serial number) generated by applying a given hash function to the URL or by encoding given information about the object into the value. This function serves to randomly distribute the embedded objects over a given set of virtual server hostnames. In addition, a given fingerprint value for the embedded object is generated by applying a given hash function to the embedded object itself. This given value serves as a fingerprint that identifies whether the embedded object has been modified. Preferably, the functions used to generate the values (i.e., for the virtual server hostname and the fingerprint) are applied to a given Web page in an off-line process. Thus, when an HTTP request for the page is received, the base HTML document is served by the Web site and some portion of the page's embedded objects are served from the hosting servers near (although not necessarily the closest) to the client machine that initiated the request.

The foregoing has outlined some of the more pertinent objects and features of the present invention. These objects should be construed to be merely illustrative of some of the more prominent features and applications of the invention. Many other beneficial results can be attained by applying the disclosed invention in a different manner or modifying the invention as will be described. Accordingly, other objects and a fuller understanding of the invention may be had by referring to the following Detailed Description of the Preferred Embodiment.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference should be made to the following Detailed Description taken in connection with the accompanying drawings in which:

FIG. 1 is a representative system in which the present invention is implemented;

FIG. 2 is a simplified representation of a markup language document illustrating the base document and a set of embedded objects;

FIG. 3 is a high level diagram of a global hosting system according to the present invention;

FIG. 4 is a simplified flowchart illustrating a method of processing a Web page to modified embedded object URLs that is used in the present invention;

FIG. 5 is a simplified state diagram illustrating how the present invention responds to a HTTP request for a Web page.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A known Internet client-server system is implemented as illustrated in FIG. 1. A client machine 10 is connected to a Web server 12 via a network 14. For illustrative purposes,

5

network 14 is the Internet, an intranet, an extranet or any other known network. Web server 12 is one of a plurality of servers which are accessible by clients, one of which is illustrated by machine 10. A representative client machine includes a browser 16, which is a known software tool used to access the servers of the network. The Web server supports files (collectively referred to as a "Web" site) in the form of hypertext documents and objects. In the Internet paradigm, a network path to a server is identified by a so-called Uniform Resource Locator (URL).

A representative Web server 12 is a computer comprising a processor 18, an operating system 20, and a Web server program 22, such as Netscape Enterprise Server. The server 12 also includes a display supporting a graphical user interface (GUI) for management and administration, and an Application Programming Interface (API) 26 that provides extensions to enable application developers to extend and/or customize the core functionality thereof through software programs including Common Gateway Interface (CGI) programs, plug-ins, servlets, active server pages, server side include (SSI) functions or the like.

A representative Web client is a personal computer that is x86-, PowerPC®- or RISC-based, that includes an operating system such as IBM® OS/2® or Microsoft Windows '95, and that includes a Web browser, such as Netscape Navigator 4.0 (or higher), having a Java Virtual Machine (JVM) and support for application plug-ins or helper applications. A client may also be a notebook computer, a handheld computing device (e.g., a PDA), an Internet appliance, or any other such device connectable to the computer network.

As seen in FIG. 2, a typical Web page comprises a markup language (e.g. HTML) master or base document 28, and many embedded objects (e.g., images, audio, video, or the like) 30. Thus, in a typical page, twenty or more embedded images or objects are quite common. Each of these images is an independent object in the Web, retrieved (or validated for change) separately. The common behavior of a Web client, therefore, is to fetch the base HTML document, and then immediately fetch the embedded objects, which are typically (but not always) located on the same server. According to the present invention, preferably the markup language base document 28 is served from the Web server (i.e., the Content Provider site) whereas a given number (or perhaps all) of the embedded objects are served from other servers. As will be seen, preferably a given embedded object is served from a server (other than the Web server itself) that is close to the client machine, that is not overloaded, and that is most likely to already have a current version of the required file.

Referring now to FIG. 3, this operation is achieved by the hosting system of the present invention. As will be seen, the hosting system 35 comprises a set of widely-deployed servers (or server resources) that form a large, fault-tolerant infrastructure designed to serve Web content efficiently, effectively, and reliably to end users. The servers may be deployed globally, or across any desired geographic regions. As will be seen, the hosting system provides a distributed architecture for intelligently routing and replicating such content. To this end, the global hosting system 35 comprises three (3) basic types of servers (or server resources): hosting servers (sometimes called ghosts) 36, top-level DNS servers 38, and low-level DNS servers 40. Although not illustrated, there may be additional levels in the DNS hierarchy. Alternatively, there may be a single DNS level that combines the functionality of the top level and low-level servers. In this illustrative embodiment, the inventive framework 35 is deployed by an Internet Service Provider (ISP), although

6

this is not a limitation of the present invention. The ISP or ISPs that deploy the inventive global hosting framework 35 preferably have a large number of machines that run both the ghost server component 36 and the low-level DNS component 40 on their networks. These machines are distributed throughout the network; preferably, they are concentrated around network exchange points 42 and network access points 44, although this is not a requirement. In addition, the ISP preferably has a small number of machines running the top-level DNS 38 that may also be distributed throughout the network.

Although not meant to be limiting, preferably a given server used in the framework 35 includes a processor, an operating system (e.g., Linux, UNIX, Windows NT, or the like), a Web server application, and a set of application routines used by the invention. These routines are conveniently implemented in software as a set of instructions used by the processor to perform various process or method steps as will be described in more detail below. The servers are preferably located at the edges of the network (e.g., in points of presence, or POPs).

Several factors may determine where the hosting servers are placed in the network. Thus, for example, the server locations are preferably determined by a demand driven network map that allows the provider (e.g., the ISP) to monitor traffic requests. By studying traffic patterns, the ISP may optimize the server locations for the given traffic profiles.

According to the present invention, a given Web page (comprising a base HTML document and a set of embedded objects) is served in a distributed manner. Thus, preferably, the base HTML document is served from the Content Provider Site 45 that normally hosts the page. The embedded objects, or some subset thereof, are preferentially served from the hosting servers 36 and, specifically, given hosting servers 36 that are near to the client machine that in the first instance initiated the request for the Web page. In addition, preferably loads across the hosting servers are balanced to ensure that a given embedded object may be efficiently served from a given hosting server near the client when such client requires that object to complete the page.

To serve the page contents in this manner, the URL associated with an embedded object is modified. As is well-known, each embedded object that may be served in a page has its own URL. Typically, the URL has a hostname identifying the Content Provider's site from where the object is conventionally served, i.e., without reference to the present invention. According to the invention, the embedded object URL is first modified, preferably in an off-line process, to condition the URL to be served by the global hosting servers. A flowchart illustrating the preferred method for modifying the object URL is illustrated in FIG. 4.

The routine begins at step 50 by determining whether all of the embedded objects in a given page have been processed. If so, the routine ends. If not, however, the routine gets the next embedded object at step 52. At step 54, a virtual server hostname is prepended into the URL for the given embedded object. The virtual server hostname includes a value (e.g., a number) that is generated, for example, by applying a given hash function to the URL. As is well-known, a hash function takes arbitrary length bit strings as inputs and produces fixed length bit strings (hash values) as outputs. Such functions satisfy two conditions: (1) it is infeasible to find two different inputs that produce the same hash value, and (2) given an input and its hash value, it is

infeasible to find a different input with the same hash value. In step 54, the URL for the embedded object is hashed into a value xx,xxx that is then included in the virtual server hostname. This step randomly distributes the object to a given virtual server hostname.

The present invention is not limited to generating the virtual server hostname by applying a hash function as described above. As an alternative and preferred embodiment, a virtual server hostname is generated as follows. Consider the representative hostname a1234.g.akamaitech.net. The 1234 value, sometimes referred to as a serial number, preferably includes information about the object such as its size (big or small), its anticipated popularity, the date on which the object was created, the identity of the Web site, the type of object (e.g., movie or static picture), and perhaps some random bits generated by a given random function. Of course, it is not required that any given serial number encode all of such information or even a significant number of such components. Indeed, in the simplest case, the serial number may be a simple integer. In any event, the information is encoded into a serial number in any convenient manner. Thus, for example, a first bit is used to denote size, a second bit is used to denote popularity, a set of additional bits is used to denote the date, and so forth. As noted above in the hashing example, the serial number is also used for load balancing and for directing certain types of traffic to certain types of servers. Typically, most URLs on the same page have the same serial number to minimize the number of distinguished name (DN) accesses needed per page. This requirement is less important for larger objects.

Thus, according to the present invention, a virtual server hostname is prepended into the URL for a given-embedded object, and this hostname includes a value (or serial number) that is generated by applying a given function to the URL or object. That function may be a hash function, an encoding function, or the like.

Turning now back to the flowchart, the routine then continues at step 56 to include a given value in the object's URL. Preferably, the given value is generated by applying a given hash function to the embedded object. This step creates a unique fingerprint of the object that is useful for determining whether the object has been modified. Thereafter, the routine returns to step 50 and cycles.

With the above as background, the inventive global hosting framework is now described in the context of a specific example. In particular, it is assumed that a user of a client machine in Boston requests a Content Provider Web page normally hosted in Atlanta. For illustrative purposes, it is assumed that the Content Provider is using the global hosting architecture within a network, which may be global, international, national, regional, local or private. FIG. 5 shows the various components of the system and how the request from the client is processed. This operation is not to be taken by way of limitation, as will be explained.

Step 1: The browser sends a request to the Provider's Web site (Item 1). The Content Provider site in Atlanta receives the request in the same way that it does as if the global hosting framework were not being implemented. The difference is in what is returned by the Provider site. Instead of returning the usual page, according to the invention, the Web site returns a page with embedded object URLs that are modified according to the method illustrated in the flowchart of FIG. 4. As previously described, the URLs preferably are changed as follows:

Assume that there are 100,000 virtual ghost servers, even though there may only be a relatively small number (e.g.,

100) physically present on the network. These virtual ghost servers or virtual ghosts are identified by the hostname: ghostxxxx.ghosting.com, where xxxxx is replaced by a number between 0 and 99,999. After the Content Provider Web site is updated with new information, a script executing on the Content Provider site is run that rewrites the embedded URLs. Preferably, the embedded URLs names are hashed into numbers between 0 and 99,999, although this range is not a limitation of the present invention. An embedded URL is then switched to reference the virtual ghost with that number. For example, the following is an embedded URL from the Provider's site:

```
<IMG SRC = http://www.provider.com/TECH/images/
space.story.gif>
```

If the serial number for the object referred to by this URL is the number 1467, then preferably the URL is rewritten to read:

```
<IMG SRC = http:
//ghost1467.ghosting.akamai.com/www.provider.com/
TECH/images/space.story.gif>
```

The use of serial numbers in this manner distributes the embedded URLs roughly evenly over the 100,000 virtual ghost server names. Note that the Provider site can still personalize the page by rearranging the various objects on the screen according to individual preferences. Moreover, the Provider can also insert advertisements dynamically and count how many people view each ad.

According to the preferred embodiment, an additional modification to the embedded URLs is made to ensure that the global hosting system does not serve stale information. As previously described, preferably a hash of the data contained in the embedded URL is also inserted into the embedded URL itself. That is, each embedded URL may contain a fingerprint of the data to which it points. When the underlying information changes, so does the fingerprint, and this prevents users from referencing old data.

The second hash takes as input a stream of bits and outputs what is sometimes referred to as a fingerprint of the stream. The important property of the fingerprint is that two different streams almost surely produce two different fingerprints. Examples of such hashes are the MD2 and MD5 hash functions, however, other more transparent methods such as a simple checksum may be used. For concreteness, assume that the output of the hash is a 128 bit signature. This signature can be interpreted as a number and then inserted into the embedded URL. For example, if the hash of the data in the picture space.story.gif from the Provider web site is the number 28765, then the modified embedded URL would actually look as follows:

```
<IMG
SRC=http://ghost1467.ghosting.akamai.com/28765/
www.provider.com/TECH/images/space.story.gif">
```

Whenever a page is changed, preferably the hash for each embedded URL is recomputed and the URL is rewritten if necessary. If any of the URL's data changes, for example, a new and different picture is inserted with the name space.story.gif, then the hash of the data is different and therefore the URL itself will be different. This scheme prevents the system from serving data that is stale as a result of updates to the original page.

For example, assume that the picture space.story.gif is replaced with a more up-to-date version on the Content Provider server. Because the data of the pictures changes, the hash of the URL changes as well. Thus, the new embedded URL looks the same except that a new number is inserted for the fingerprint. Any user that requests the page after the update receives a page that points to the new

picture. The old picture is never referenced and cannot be mistakenly returned in place of the more up-to-date information.

In summary, preferably there are two hashing operations that are done to modify the pages of the Content Provider. First, hashing can be a component of the process by which a serial number is selected to transform the domain name into a virtual ghost name. As will be seen, this first transformation serves to redirect clients to the global hosting system to retrieve the embedded URLs. Next, a hash of the data pointed to by the embedded URLs is computed and inserted into the URL. This second transformation serves to protect against serving stale and out-of-date content from the ghost servers. Preferably, these two transformations are performed off-line and therefore do not pose potential performance bottlenecks.

Generalizing, the preferred URL schema is as follows. The illustrative domain `www.domainname.com/frontpage.jpg` is transformed into:

```
xxxx.yy.zzzz.net/aaaa/www.domainname.com/frontpage.jpg,
```

where:

```
xxxx=serial number field
yy=lower level DNS field
zzzz=top level DNS field
aaaa=other information (e.g., fingerprint) field.
```

If additional levels of the DNS hierarchy are used, then there may be additional lower level DNS fields, e.g., `xxxx.y1y1.y2y2.zzzz.net/aaaa/...`

Step 2: After receiving the initial page from the Content Provider site, the browser needs to load the embedded URLs to display the page. The first step in doing this is to contact the DNS server on the user's machine (or at the user's ISP) to resolve the altered hostname, in this case:

`ghost1467.ghosting.akamai.com`. As will be seen, the global hosting architecture of the present invention manipulates the DNS system so that the name is resolved to one of the ghosts that is near the client and is likely to have the page already. To appreciate how this is done, the following describes the progress of the DNS query that was initiated by the client.

Step 3: As previously described, preferably there are two types of DNS servers in the inventive system: top-level and low-level. The top level DNS servers 38 for ghosting.com have a special function that is different from regular DNS servers like those of the .com domain. The top level DNS servers 38 include appropriate control routines that are used to determine where in the network a user is located, and then to direct the user to a `akamai.com` (i.e., a low level DNS) server 40 that is close-by. Like the .com domain, `akamai.com` preferably has a number of top-level DNS servers 38 spread throughout the network for fault tolerance. Thus, a given top level DNS server 38 directs the user to a region in the Internet (having a collection of hosting servers 36 that may be used to satisfy the request for a given embedded object) whereas the low level DNS server 40 (within the identified region) identifies a particular hosting server within that collection from which the object is actually served.

More generally, as noted above, the DNS process can contain several levels of processing, each of which serves to better direct the client to a ghost server. The ghost server name can also have more fields. For example, "a123.g.g.akamaitech.net" may be used instead of "a123.ghost.akamai.com." If only one DNS level is used, a representative URL could be "a123.akamai.com."

Although other techniques may be used, the user's location in the network preferably is deduced by looking at the

IP address of the client machine making the request. In the present example, the DNS server is running on the machine of the user, although this is not a requirement. If the user is using an ISP DNS server, for example, the routines make the assumption that the user is located near (in the Internet sense) this server. Alternatively, the user's location or IP address could be directly encoded into the request sent to the top level DNS. To determine the physical location of an IP address in the network, preferably, the top level DNS server builds a network map that is then used to identify the relevant location.

Thus, for example, when a request comes in to a top level DNS for a resolution for `a1234.g.akamaitech.net`, the top level DNS looks at the return address of the requester and then formulates the response based on that address according to a network map. In this example, the `a1234` is a serial number, the `g` is a field that refers to the lower level DNS, and `akamaitech` refers to the top level DNS. The network map preferably contains a list of all Internet Protocol (IP) blocks and, for each IP block, the map determines where to direct the request. The map preferably is updated continually based on network conditions and traffic.

After determining where in the network the request originated, the top level DNS server redirects the DNS request to a low level DNS server close to the user in the network. The ability to redirect requests is a standard feature in the DNS system. In addition, this redirection can be done in such a way that if the local low level DNS server is down, there is a backup server that is contacted.

Preferably, the TTL (time to live) stamp on these top level DNS redirections for the `ghosting.com` domain is set to be long. This allows DNS caching at the user's DNS servers and/or the ISP's DNS servers to prevent the top level DNS servers from being overloaded. If the TTL for `ghosting.akamai.com` in the DNS server at the user's machine or ISP has expired, then a top level server is contacted, and a new redirection to a local low level `ghosting.akamai.com` DNS server is returned with a new TTL stamp. It should be noted the system does not cause a substantially larger number of top level DNS lookups than what is done in the current centralized hosting solutions. This is because the TTL of the top level redirections are set to be high and, thus, the vast majority of users are directed by their local DNS straight to a nearby low level `ghosting.akamai.com` DNS server.

Moreover, fault tolerance for the top level DNS servers is provided automatically by DNS similarly to what is done for the popular .com domain. Fault tolerance for the low level DNS servers preferably is provided by returning a list of possible low level DNS servers instead of just a single server. If one of the low level DNS servers is down, the user will still be able to contact one on the list that is up and running.

Fault tolerance can also be handled via an "overflow control" mechanism wherein the client is redirected to a low-level DNS in a region that is known to have sufficient capacity to serve the object. This alternate approach is very useful in scenarios where there is a large amount of demand from a specific region or when there is reduced capacity in a region. In general, the clients are directed to regions in a way that minimizes the overall latency experienced by clients subject to the constraint that no region becomes overloaded. Minimizing overall latency subject to the regional capacity constraints preferably is achieved using a min-cost multicommodity flow algorithm.

Step 4: At this point, the user has the address of a close-by `ghosting.com` DNS server 38. The user's local DNS server contacts the close-by low level DNS server 40 and requests

a translation for the name ghost1467.ghosting.akamai.com. The local DNS server is responsible for returning the IP address of one of the ghost servers 36 on the network that is close to the user, not overloaded, and most likely to already have the required data.

The basic mechanism for mapping the virtual ghost names to real ghosts is hashing. One preferred technique is so-called consistent hashing, as described in U.S. Ser. No. 09/042,228, filed Mar. 13, 1998, and in U.S. Ser. No. 09/088,825, filed Jun. 2, 1998, each titled Method And Apparatus For Distributing Requests Among A Plurality Of Resources, and owned by the Massachusetts Institute of Technology, which applications are incorporated herein by reference. Consistent hash functions make the system robust under machine failures and crashes. It also allows the system to grow gracefully, without changing where most items are located and without perfect information about the system.

According to the invention, the virtual ghost names may be hashed into real ghost addresses using a table lookup, where the table is continually updated based on network conditions and traffic in such a way to insure load balancing and fault tolerance. Preferably, a table of resolutions is created for each serial number. For example, serial number 1 resolves to ghost 2 and 5, serial number 2 resolves to ghost 3, serial number 3 resolves to ghosts 2,3,4, and so forth. The goal is to define the resolutions so that no ghost exceeds its capacity and that the total number of all ghosts in all resolutions is minimized. This is done to assure that the system can take maximal advantage of the available memory at each region. This is a major advantage over existing load balancing schemes that tend to cache everything everywhere or that only cache certain objects in certain locations no matter what the loads are. In general, it is desirable to make assignments so that resolutions tend to stay consistent over time provided that the loads do not change too much in a short period of time. This mechanism preferably also takes into account how close the ghost is to the user, and how heavily loaded the ghost is at the moment.

Note that the same virtual ghost preferably is translated to different real ghost addresses according to where the user is located in the network. For example, assume that ghost server 18.98.0.17 is located in the United States and that ghost server 132.68.1.28 is located in Israel. A DNS request for ghost1487.ghosting.akamai.com originating in Boston will resolve to 18.98.0.17, while a request originating in Tel-Aviv will resolve to 132.68.1.28.

The low-level DNS servers monitor the various ghost servers to take into account their loads while translating virtual ghost names into real addresses. This is handled by a software routine that runs on the ghosts and on the low level DNS servers. In one embodiment, the load information is circulated among the servers in a region so that they can compute resolutions for each serial number. One algorithm for computing resolutions works as follows. The server first computes the projected load (based on number of user requests) for each serial number. The serial numbers are then processed in increasing order of load. For each serial number, a random priority list of desired servers is assigned using a consistent hashing method. Each serial number is then resolved to the smallest initial segment of servers from the priority list so that no server becomes overloaded. For example, if the priority list for a serial number is 2,5,3,1,6, then an attempt is made first to try to map the load for the serial number to ghost 2. If this overloads ghost 2, then the load is assigned to both ghosts 2 and 5. If this produced too much load on either of those servers, then the load is assigned to ghosts 2,3, and 5, and so forth. The projected

load on a server can be computed by looking at all resolutions that contain that server and by adding the amount of load that is likely to be sent to that server from that serial number. This method of producing resolutions is most effective when used in an iterative fashion, wherein the assignments starts in a default state, where every serial number is mapped to every ghost. By refining the resolution table according to the previous procedure, the load is balanced using the minimum amount of replication (thereby maximally conserving the available memory in a region).

The TTL for these low level DNS translations is set to be short to allow a quick response when heavy load is detected on one of the ghosts. The TTL is a parameter that can be manipulated by the system to insure a balance between timely response to high load on ghosts and the load induced on the low level DNS servers. Note, however, that even if the TTL for the low level DNS translation is set to 1-2 minutes, only a few of the users actually have to do a low level DNS lookup. Most users will see a DNS translation that is cached on their machine or at their ISP. Thus, most users go directly from their local DNS server to the close-by ghost that has the data they want. Those users that actually do a low level DNS lookup have a very small added latency, however this latency is small compared to the advantage of retrieving most of the data from close by.

As noted above, fault tolerance for the low level DNS servers is provided by having the top level DNS return a list of possible low level DNS servers instead of a single server address. The user's DNS system caches this list (part of the standard DNS system), and contacts one of the other servers on the list if the first one is down for some reason. The low level DNS servers make use of a standard feature of DNS to provide an extra level of fault tolerance for the ghost servers. When a name is translated, instead of returning a single name, a list of names is returned. If for some reason the primary fault tolerance method for the ghosts (known as the Buddy system, which is described below) fails, the client browser will contact one of the other ghosts on the list.

Step 5: The browser then makes a request for an object named a123.ghosting.akamai.com/... /www.provider.com/TECH/images/space.story.gif from the close-by ghost. Note that the name of the original server (www.provider.com) preferably is included as part of the URL. The software running on the ghost parses the page name into the original host name and the real page name. If a copy of the file is already stored on the ghost, then the data is returned immediately. If, however, no copy of the data on the ghost exists, a copy is retrieved from the original server or another ghost server. Note that the ghost knows who the original server was because the name was encoded into the URL that was passed to the ghost from the browser. Once a copy has been retrieved it is returned to the user, and preferably it is also stored on the ghost for answering future requests.

As an additional safeguard, it may be preferable to check that the user is indeed close to the server. This can be done by examining the IP address of the client before responding to the request for the file. This is useful in the rare case when the client's DNS server is far away from the client. In such a case, the ghost server can redirect the user to a closer server (or to another virtual address that is likely to be resolved to a server that is closer to the client). If the redirect is to a virtual server, then it must be tagged to prevent further redirections from taking place. In the preferred embodiment, redirection would only be done for large objects; thus, a check may be made before applying a redirection to be sure that the object being requested exceeds a certain overall size.

Performance for long downloads can also be improved by dynamically changing the server to which a client is con-

nected based on changing network conditions. This is especially helpful for audio and video downloads (where the connections can be long and where quality is especially important). In such cases, the user can be directed to an alternate server in mid-stream. The control structure for redirecting the client can be similar to that described above, but it can also include software that is placed in the client's browser or media player. The software monitors the performance of the client's connection and perhaps the status of the network as well. If it is deemed that the client's connection can be improved by changing the server, then the system directs the client to a new server for the rest of the connection.

Fault tolerance for the ghosts is provided by a buddy system, where each ghost has a designated buddy ghost. If a ghost goes down, its buddy takes over its work (and IP address) so that service is not interrupted. Another feature of the system is that the buddy ghost does not have to sit idle waiting for a failure. Instead, all of the machines are always active, and when a failure happens, the load is taken over by the buddy and then balanced by the low level DNS system to the other active ghosts. An additional feature of the buddy system is that fault tolerance is provided without having to wait for long timeout periods.

As yet another safety feature of the global hosting system, a gating mechanism can be used to keep the overall traffic for certain objects within specified limits. One embodiment of the gating mechanism works as follows. When the number of requests for an object exceeds a certain specified threshold, then the server can elect to not serve the object. This can be very useful if the object is very large. Instead, the client can be served a much smaller object that asks the client to return later. Or, the client can be redirected. Another method of implementing a gate is to provide the client with a "ticket" that allows the client to receive the object at a pre-specified future time. In this method, the ghost server needs to check the time on the ticket before serving the object.

The inventive global hosting scheme is a way for global ISPs or conglomerates of regional ISPs to leverage their network infrastructure to generate hosting revenue, and to save on network bandwidth. An ISP offering the inventive global hosting scheme can give content providers the ability to distribute content to their users from the closest point on the ISP's network, thus ensuring fast and reliable access. Guaranteed web site performance is critical for any web-based business, and global hosting allows for the creation of a service that satisfies this need.

Global hosting according to the present invention also allows an ISP to control how and where content traverses its network. Global hosting servers can be set up at the edges of the ISP's network (at the many network exchange and access points, for example). This enables the ISP to serve content for sites that it hosts directly into the network exchange points and access points. Expensive backbone links no longer have to carry redundant traffic from the content provider's site to the network exchange and access points. Instead, the content is served directly out of the ISP's network, freeing valuable network resources for other traffic.

Although global hosting reduces network traffic, it is also a method by which global ISPs may capture a piece of the rapidly expanding hosting market, which is currently estimated at over a billion dollars a year.

The global hosting solution also provides numerous advantages to Content Providers, and, in particular, an efficient and cost-effective solution to improve the performance of their Web sites both domestically and internation-

ally. The inventive hosting software ensures Content Providers with fast and reliable Internet access by providing a means to distribute content to their subscribers from the closest point on an ISP's network. In addition to other benefits described in more detail below, the global hosting solution also provides the important benefit of reducing network traffic.

Once inexpensive global hosting servers are installed at the periphery of an ISP's network (i.e., at the many network exchange and access points), content is served directly into network exchange and access points. As a result of this efficient distribution of content directly from an ISP's network, the present invention substantially improves Web site performance. In contrast to current content distribution systems, the inventive global hosting solution does not require expensive backbone links to carry redundant traffic from the Content Provider's Web site to the network exchange and access points.

A summary of the specific advantages afforded by the inventive global hosting scheme are set forth below:

1. Decreased Operational Expenses for Content Providers:

Most competing solutions require Content Providers to purchase servers at each Web site that hosts their content. As a result, Content Providers often must negotiate separate contracts with different ISPs around the world. In addition, Content Providers are generally responsible for replicating the content and maintaining servers in these remote locations.

With the present invention, ISPs are primarily responsible for the majority of the aspects of the global hosting. Content Providers preferably maintain only their single source server. Content on this server is automatically replicated by software to the locations where it is being accessed. No intervention or planning is needed by the Provider (or, for that matter, the ISP). Content Providers are offered instant access to all of the servers on the global network; there is no need to choose where content should be replicated or to purchase additional servers in remote locations.

2. Intelligent and Efficient Data Replication:

Most competing solutions require Content Providers to replicate their content on servers at a commercial hosting site or to mirror their content on geographically distant servers. Neither approach is particularly efficient. In the former situation, content is still located at a single location on the Internet (and thus it is far away from most users). In the latter case, the entire content of a Web site is copied to remote servers, even though only a small portion of the content may actually need to be located remotely. Even with inexpensive memory, the excessive cost associated with such mirroring makes it uneconomical to mirror to more than a few sites, which means that most users will still be far away from a mirror site. Mirroring also has the added disadvantage that Content Providers must insure that all sites remain consistent and current, which is a nontrivial task for even a few sites.

With the present invention, content is automatically replicated to the global server network in an intelligent and efficient fashion. Content is replicated in only those locations where it is needed. Moreover, when the content changes, new copies preferably are replicated automatically throughout the network.

3. Automatic Content Management:

Many existing solutions require active management of content distribution, content replication and load balancing between different servers. In particular, decisions about where content will be hosted must be made manually, and

the process of replicating data is handled in a centralized push fashion. On the contrary, the invention features passive management. Replication is done in a demand-based pull fashion so that content preferably is only sent to where it is truly needed. Moreover, the process preferably is fully automated; the ISP does not have to worry about how and where content is replicated and/or the content provider.

4. Unlimited, Cost Effective Scalability:

Competing solutions are not scalable to more than a small number of sites. For example, solutions based on mirroring are typically used in connection with at most three or four sites. The barriers to scaling include the expense of replicating the entire site, the cost of replicating computing resources at all nodes, and the complexity of supporting the widely varying software packages that Content Providers use on their servers.

The unique system architecture of the present invention is scalable to hundreds, thousands or even millions of nodes. Servers in the hosting network can malfunction or crash and the system's overall function is not affected. The global hosting framework makes efficient use of resources; servers and client software do not need to be replicated at every node because only the hosting server runs at each node. In addition, the global hosting server is designed to run on standard simple hardware that is not required to be highly fault tolerant.

5. Protection against Flash Crowds:

Competing solutions do not provide the Content Provider with protection from unexpected flash crowds. Although mirroring and related load-balancing solutions do allow a Content Provider to distribute load across a collection of servers, the aggregate capacity of the servers must be sufficient to handle peak demands. This means that the Provider must purchase and maintain a level of resources commensurate with the anticipated peak load instead of the true average load. Given the highly variable and unpredictable nature of the Internet, such solutions are expensive and highly wasteful of resources.

The inventive hosting architecture allows ISPs to utilize a single network of hosting servers to offer Content Providers flash crowd insurance. That is, insurance that the network will automatically adapt to and support unexpected higher load on the Provider's site. Because the ISP is aggregating many Providers together on the same global network, resources are more efficiently used.

6. Substantial Bandwidth Savings:

Competing solutions do not afford substantial bandwidth savings to ISPs or Content Providers. Through the use of mirroring, it is possible to save bandwidth over certain links (i.e., between New York and Los Angeles). Without global hosting, however, most requests for content will still need to transit the Internet, thus incurring bandwidth costs. The inventive hosting framework saves substantial backbone bandwidth for ISPs that have their own backbones. Because content is distributed throughout the network and can be placed next to network exchange points, both ISPs and Content Providers experience substantial savings because backbone charges are not incurred for most content requests.

7. Instant Access to the Global Network:

Competing solutions require the Content Provider to choose manually a small collection of sites at which content will be hosted and/or replicated. Even if the ISP has numerous hosting sites in widely varied locations, only those sites specifically chosen (and paid for) will be used to host content for that Content Provider.

On the contrary, the global hosting solution of the present invention allows ISPs to offer their clients instant access to

the global network of servers. To provide instant access to the global network, content is preferably constantly and dynamically moved around the network. For example, if a Content Provider adds content that will be of interest to customers located in Asia, the Content Provider will be assured that its content will be automatically moved to servers that are also located in Asia. In addition, the global hosting framework allows the content to be moved very close to end users (even as close as the user's building in the case of the Enterprise market).

8. Designed for Global ISPs and Conglomerates:

Most competing solutions are designed to be purchased and managed by Content Providers, many of whom are already consistently challenged and consumed by the administrative and operational tasks of managing a single server. The inventive hosting scheme may be deployed by a global ISP, and it provides a new service that can be offered to Content Providers. A feature of the service is that it minimizes the operational and managerial requirements of a Content Provider, thus allowing the Content Provider to focus on its core business of creating unique content.

9. Effective Control of Proprietary Databases and Confidential Information:

Many competing solutions require Content Providers to replicate their proprietary databases to multiple geographically distant sites. As a result, the Content Provider effectively loses control over its proprietary and usually confidential databases. To remedy these problems, the global hosting solution of the present invention ensures that Content Providers retain complete control over their databases. As described above, initial requests for content are directed to the Content Provider's central Web site, which then implements effective and controlled database access. Preferably, high-bandwidth, static parts for page requests are retrieved from the global hosting network.

10. Compatibility with Content Provider Software:

Many competing solutions require Content Providers to utilize a specific set of servers and databases. These particular, non-uniform requirements constrain the Content Provider's ability to most effectively use new technologies, and may require expensive changes to a Content Provider's existing infrastructure. By eliminating these problems, the inventive global hosting architecture effectively interfaces between the Content Provider and the ISP, and it does not make any assumptions about the systems or servers used by the Content Provider. Furthermore, the Content Provider's systems can be upgraded, changed or completely replaced without modifying or interrupting the inventive architecture.

11. No Interference with Dynamic Content, Personalized Advertising or E-Commerce, and No stale content:

Many competing solutions (such as naive caching of all content) can interfere with dynamic content, personalized advertising and E-commerce and can serve the user with stale content. While other software companies have attempted to partially eliminate these issues (such as keeping counts on hits for all cached copies), each of these solutions causes a partial or complete loss of functionality (such as the ability to personalize advertising). On the contrary, the global hosting solution does not interfere with generation of dynamic content, personalized advertising or E-commerce, because each of these tasks preferably is handled by the central server of the Content Provider.

12. Designed for the Global Network:

The global hosting architecture is highly scalable and thus may be deployed on a world-wide network basis.

The above-described functionality of each of the components of the global hosting architecture preferably is imple-

mented in software executable in a processor, namely, as a set of instructions or program code in a code module resident in the random access memory of the computer. Until required by the computer, the set of instructions may be stored in another computer memory, for example, in a hard disk drive, or in a removable memory such as an optical disk (for eventual use in a CD ROM) or floppy disk (for eventual use in a floppy disk drive), or downloaded via the Internet or other computer network.

In addition, although the various methods described are conveniently implemented in a general purpose computer selectively activated or reconfigured by software, one of ordinary skill in the art would also recognize that such methods may be carried out in hardware, in firmware, or in more specialized apparatus constructed to perform the required method steps.

Further, as used herein, a Web "client" should be broadly construed to mean any computer or component thereof directly or indirectly connected or connectable in any known or later-developed manner to a computer network, such as the Internet. The term Web "server" should also be broadly construed to mean a computer, computer platform, an adjunct to a computer or platform, or any component thereof. Of course, a "client" should be broadly construed to mean one who requests or gets the file, and "server" is the entity which downloads the file.

Having thus described our invention, what we claim as new and desire to secure by letters patent is set forth in the following claims.

What is claimed is:

1. A method of content delivery wherein participating content providers identify content to be delivered by a content delivery network service provider from a set of content servers associated with the content delivery network service provider, wherein a given object of a participating content provider is associated with a Uniform Resource Locator (URL) that includes, in addition to a filename, an alphanumeric string that is resolvable in a domain name system (DNS) associated with the content delivery network service provider, the domain name system including a set of name servers, comprising:

during a first DNS query performed on a first portion of the alphanumeric string, selecting a given one of the name servers in the domain name system associated with the content delivery network service provider;

during a second DNS query performed at the given one of the name servers in the domain name system associated with the content delivery network service provider and on a second portion of the alphanumeric string together with the first portion, identifying a given server from the set of content servers associated with the content delivery network service provider;

at the given server, receiving a request for the given object; and

in response to the request, serving the given object from the given server.

2. The method as described in claim 1 wherein the domain name system associated with the content delivery network service provider selects the given one of the name servers as a function of where the first DNS query originates.

3. The method as described in claim 1 wherein the given server is identified from the set of content servers associated with the content delivery network service provider as a function of where the first DNS query originates and Internet traffic conditions.

4. The method as described in claim 1 wherein the second portion of the alphanumeric string includes a value identi-

fying a virtual content bucket in which the given object is likely to be associated.

5. The method as described in claim 1 wherein the first and second DNS queries are performed without reference to the filename the given object.

6. The method as described in claim 1 further including the step of caching the given object at the server associated with the IP address for a given time to live (TTL) so that the given object is available for delivery in response to a new request received at the server associated with the IP address.

7. The method as described in claim 1 wherein the given object of the participating content provider is an embedded object in a markup language page.

8. A method of content delivery wherein participating content providers identify content to be delivered by a content delivery network service provider from a set of content servers associated with the content delivery network service provider, wherein a given object of a participating content provider is associated with a 12293:10 PATENT Uniform Resource Locator (URL) that includes, in addition to a filename, an alphanumeric string, comprising:

having the content delivery network service provider establish a domain name system (DNS) having authority to resolve the alphanumeric strings in the URLs of the objects identified by the participating content providers, the content delivery network service provider's domain name system having one or more DNS levels, wherein at least one DNS level comprises a set of one or more name servers;

for each of one or more participating content providers, delivering a given object on behalf of the participating content provider, wherein the given object is delivered by the following steps:

responsive to a DNS query, selecting a given one of the name servers in the content delivery network service provider's domain name system;

at the given one of the name servers, resolving the alphanumeric string to an IP address, wherein the alphanumeric string is resolved without reference to the filename for the given object;

at a server associated with the IP address, the server being one of the set of content servers, receiving a request for the given object, the request having the filename associated therewith;

from the server, serving the given object; and

caching the given object at the server so that the given object is available for delivery from the server for a given time period in the event that a new DNS query to resolve the alphanumeric string is received at the domain name system and is resolved to the IP address of the server.

9. The method as described in claim 8 wherein the given one of the name servers is selected as a function of where the DNS query originates.

10. The method as described in claim 8 wherein the server associated with the IP address is identified as a function of where the DNS query originates and Internet traffic conditions.

11. The method as described in claim 8 wherein the given object of the participating content provider is an embedded object in a markup language page.

12. The method as described in claim 8 wherein the request includes a value for use at the server associated with the IP address in determining whether a version of the given object is fresh.

13. The method as described in claim 8 wherein the DNS query is initiated from an end user local name server.

14. Apparatus for use in a content delivery network wherein participating content providers identify content to be delivered by a content delivery network service provider from a set of content servers in the content delivery network, wherein a given object of a participating content provider is associated with a Uniform Resource Locator (URL) that includes, in addition to a filename, an alphanumeric string that is resolvable in a domain name system associated with the content delivery network service provider, the domain name system associated with the content delivery network service provider including a set of name servers, comprising:

domain name service (DNS) code for resolving, to an IP address, a DNS query identifying the alphanumeric string, wherein the IP address is associated with a content server selected from the set of content servers; wherein the domain name service (DNS) code comprises a first set of code executing on a first processor for resolving a first portion of the alphanumeric string to identify a given one of the set of name servers, and a second set of code executing on the given one of the set of name servers for resolving a second portion of the alphanumeric string, together with the first portion of the alphanumeric string, to the IP address.

15. The apparatus as described in claim 14 wherein the DNS code resolves the DNS query as a function of a location of an entity that initiates the DNS query.

16. The apparatus as described in claim 14 wherein the DNS code resolves the DNS query as a function of a location of an entity that initiates the DNS query and given network traffic conditions.

17. The apparatus as described in claim 14 wherein the given object is an embedded object in a markup language page.

18. A method of content delivery wherein participating content providers identify content to be delivered by a content delivery network service provider from a set of content servers associated with the content delivery network service provider, wherein a given object of a participating content provider is associated with a Uniform Resource Locator (URL) that includes, in addition to a filename, an alphanumeric string, comprising:

having the content delivery network service provider establish a domain name system (DNS) having authority to resolve the alphanumeric strings in the URLs of the objects identified by the participating content providers, the content delivery network service provider's domain name system having one or more DNS levels, wherein at least one DNS level comprises a set of one or more name servers;

for each of one or more participating content providers, delivering a given object on behalf of the participating content provider, wherein the given object is delivered by the following steps:

responsive to a DNS query, selecting a given one of the name servers in the content delivery network service provider's domain name system as a function of where the DNS query originates;

at the given one of the name servers, resolving the alphanumeric string to an IP address, wherein the alphanumeric string is resolved without reference to the filename for the given object;

at a server associated with the IP address, the server being one of the set of content servers, receiving a request for the given object, the request having the filename associated therewith; and from the server, serving the given object.

19. The method of content delivery as described in claim 18 wherein the step of delivering a given object on behalf of the participating content provider further includes the step of:

caching the given object at the server so that the given object is available for delivery from the server at a later time in the event that a new DNS query to resolve the alphanumeric string is received at the domain name system and is resolved to the IP address of the server.

20. A method of content delivery wherein participating content providers identify content to be delivered by a content delivery network service provider from a set of content servers associated with the content delivery network service provider, wherein a given object of a participating content provider is associated with an alphanumeric string, comprising:

having the content delivery network service provider establish a domain name system (DNS) having authority to resolve the alphanumeric strings associated with the objects identified by the participating content providers, the content delivery network service provider's domain name system having one or more DNS levels, wherein at least one DNS level comprises a set of one or more name servers;

for each of one or more participating content providers, delivering a given object on behalf of the participating content provider, wherein the given object is delivered by the following steps:

responsive to a DNS query received from a client local name server, selecting a given one of the name servers in the content delivery network service provider's domain name system;

at the given one of the name servers, resolving the alphanumeric string to an IP address, wherein the alphanumeric string is resolved without reference to the filename for the given object;

at a server associated with the IP address, the server being one of the set of content servers, receiving a request for the given object, the request having the filename associated therewith; and

from the server, serving the given object.

21. The method as described in claim 20 wherein the server associated with the IP address is identified as a function of where the DNS query originates.

22. The method as described in claim 20 wherein the server associated with the IP address is identified as a function of where the DNS query originates and Internet traffic conditions.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,553,413 B1
DATED : April 22, 2003
INVENTOR(S) : F. Thomson Leighton and Daniel M. Lewin

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 5,

Line 12, delete "i8" and insert therefor -- 18 --.

Column 20,

Line 19, delete "12293:10 PATENT".

Signed and Sealed this

Twenty-fourth Day of February, 2004



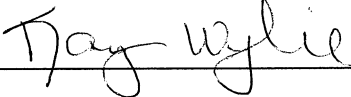
JON W. DUDAS
Acting Director of the United States Patent and Trademark Office

CERTIFICATE OF SERVICE

I hereby certify that copies of the foregoing CORRECTED BRIEF OF PLAINTIFFS-APPELLANTS AKAIMAI TECHNOLOGIES, INC. AND THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY were served by Federal Express on this 8th day of March 2010, on the following counsel of record:

ROBERT G. KRUPKA
KIRKLAND & ELLIS LLP
777 South Figueroa Street, Suite 3700
Los Angeles, CA 90017

ROBERT S. FRANK, JR.
CHOATE, HALL & STEWART LLP
Two International Place
Boston, MA 02110



CERTIFICATE OF COMPLIANCE

I certify that the foregoing CORRECTED BRIEF OF PLAINTIFFS-
APPELLANTS AKAMAI TECHNOLOGIES, INC. AND THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY contains 13,964 words as
measured by the word processing software used to prepare this brief.

Dated: March 8, 2010

Respectfully submitted,

