

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

AKAMAI TECHNOLOGIES, INC.,

Petitioner,

v.

EQUIL IP HOLDINGS LLC,

Patent Owner.

Case IPR2023-00329

U.S. Patent No. 6,792,575

PETITION FOR *INTER PARTES* REVIEW

TABLE OF CONTENTS

TABLE OF ABBREVIATIONS	iii
EXHIBIT LIST	iv
LIST OF CHALLENGED CLAIMS	vii
I. INTRODUCTION	1
II. MANDATORY NOTICES	3
A. Real Parties-in-Interest	3
B. Related Matters	3
C. Counsel and Service Information:	4
III. PAYMENT OF FEES	5
IV. REQUIREMENTS FOR <i>INTER PARTES</i> REVIEW	5
A. Grounds for Standing	5
B. Identification of Challenge	5
1. Specific Art on Which the Challenge Is Based	5
2. Statutory Grounds on Which the Challenge Is Based	6
V. '575 PATENT AND PROSECUTION HISTORY	7
A. '575	7
B. Prosecution History	11
VI. §325(d) AND §314(a) DISCRETION DOES NOT APPLY	12
1. §325(d)	12
2. §314(a)	13
VII. LEVEL OF ORDINARY SKILL	14
VIII. CLAIM CONSTRUCTION	14
A. [1.f]-[1.i]/[9.d]-[9.f.4]	14
B. Preambles	15
IX. GROUNDS OF UNPATENTABILITY	15
A. Ground 1: Lafer in View of Tso (Claim 1)	16
1. Overview of Lafer	16

2.	Overview of Tso and Motivation to Modify Lafer with Tso’s Teachings	20
3.	Claim Chart	25
B.	Ground 2: Lafer in View of Tso and Cao (Claim 9)	62
1.	Overview of Cao and Motivation to Modify Lafer (in view of Tso) with Cao	62
2.	Claim Chart	67
C.	Grounds 3-4: Lafer in view of Tso, [Cao for Claim 9,] and Moore ...	84
X.	SECONDARY CONSIDERATIONS	86
XI.	CONCLUSION.....	86

TABLE OF ABBREVIATIONS

'242	U.S. Patent No. 8,495,242
'575	U.S. Patent No. 6,792,575
'575 FH	File History of U.S. Application No. 09/425,326
'745	U.S. Patent No. 9,158,745
Barbará	U.S. Patent No. 5,581,704
Boucher	U.S. Patent No. 6,745,368
Claims	Claims 1 and 9 of U.S. Patent No. 6,792,575
Delp	U.S. Patent No. 5,420,967
IPR	<i>Inter Partes</i> Review
Lafer	U.S. Patent No. 6,192,382
Moore	U.S. Patent No. 6,310,601
Petitioner	Petitioner Akamai Technologies, Inc
PO	Patent Owner Equil IP Holdings LLC
POSITA	Person of Ordinary Skill in the Art
PTAB	Patent Trial and Appeal Board
Starnes	U.S. Patent No. 6,144,996
Tso	U.S. Patent No. 6,421,733
USPTO	United States Patent and Trademark Office

EXHIBIT LIST

Ex. 1001	U.S. Patent No. 6,792,575 (“’575”)
Ex. 1002	File History of U.S. Application No. 09/425,326 (“’575 FH)
Ex. 1003	Declaration of Vijay K. Madiseti in Support of Petition for <i>Inter Partes</i> Review of U.S. Patent No. 6,792,575
Ex. 1004	U.S. Patent No. 6,192,382 (“Lafer”)
Ex. 1005	U.S. Patent No. 6,421,733 (“Tso”)
Ex. 1006	Pei Cao & Chengjie Liu, <i>Maintaining strong cache consistency in the World Wide Web</i> , 47 IEEE TRANSACTIONS ON COMPUTS. 445 - 457 (1998)
Ex. 1007	U.S. Patent No. 6,310,601 (“Moore”)
Ex. 1008	Declaration of Gordon MacPherson
Ex. 1009	(Reserved)
Ex. 1010	U.S. Patent No. 5,420,967 (“Delp”)
Ex. 1011	U.S. Patent No. 6,144,996 (“Starnes”)
Ex. 1012	Harini Bharadvaj et al., <i>An Active Transcoding Proxy to Support Mobile Web Access</i> , PROCS. 17TH IEEE SYMP. ON RELIABLE DISTRIBUTED SYS. 118 (1998), https://ieeexplore.ieee.org/document/740482
Ex. 1013	U.S. Patent No. 6,745,368 (“Boucher”)
Ex. 1014	Rakesh Mohan et al., <i>Adapting Multimedia Internet Content for Universal Access</i> , 1 IEEE TRANSACTIONS ON MULTIMEDIA 104 (1999), https://ieeexplore.ieee.org/document/748175
Ex. 1015	Armando Fox et al., <i>Adapting to Network and Client Variation Using Infrastructural Proxies: Lessons and Perspectives</i> , 5 IEEE PERS.

	COMMC'NS. 10 (1998), https://ieeexplore.ieee.org/abstract/document/709365
Ex. 1016	U.S. Patent No. 5,581,704 (“Barbará”)
Ex. 1017	Fielding et al., <i>Hypertext Transfer Protocol – HTTP/1.1</i> , RFC 2068, Jan. 1997, https://www.rfc-editor.org/rfc/rfc2068
Ex. 1018	(Reserved)
Ex. 1019	Chaman L. Sabharwal and Sanjiv K. Bhatia, <i>Image databases and near-perfect hash table</i> , 30 PATTERN RECOGNITION 1867 (November 19, 1997), https://www.sciencedirect.com/science/article/pii/S0031320396001872
Ex. 1020	Letter from Tony Corsini of the British Library
Ex. 1021	British Library copy of Pei Cao & Chengjie Liu, <i>Maintaining strong cache consistency in the World Wide Web</i> , 47 IEEE TRANSACTIONS ON COMPUTS. 445 - 457 (1998)
Ex. 1022	IEEE Transactions on Computers About Journal IEEE Xplore (part 1), https://ieeexplore.ieee.org/xpl/aboutJournal.jsp?punumber=12
Ex. 1023	IEEE Transactions on Computers About Journal IEEE Xplore (part 2), https://ieeexplore.ieee.org/xpl/aboutJournal.jsp?punumber=12
Ex. 1024	IEEE Transactions on Computers About Journal IEEE Xplore (part 3), https://ieeexplore.ieee.org/xpl/aboutJournal.jsp?punumber=12
Ex. 1025	IEEE Transactions on Computers All Issues IEEE Xplore, https://ieeexplore.ieee.org/xpl/issues?punumber=12&isnumber=9984045
Ex. 1026	Table of Contents, 47 IEEE TRANSACTIONS ON COMPUTS., https://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=14873&punumber=12

Ex. 1027	Maintaining strong cache consistency in the World Wide Web IEEE Journals & Magazine IEEE Xplore, Citations by Papers, https://ieeexplore.ieee.org/document/675713/citations?tabFilter=papers#citations
Ex. 1028	Maintaining strong cache consistency in the World Wide Web IEEE Journals & Magazine IEEE Xplore, Citations by Patents, https://ieeexplore.ieee.org/document/675713/citations?tabFilter=patents#citations
Ex. 1029	Maarten van Steen et al., <i>A scalable middleware solution for advanced wide-area Web services</i> , 6 DISTRIBUTED SYS. ENG'G 34 (1999), https://iopscience.iop.org/article/10.1088/0967-1846/6/1/304
Ex. 1030	James B. Lim & Ali R. Hurson, <i>Data duplication and consistency in a mobile, multidatabase environment</i> , PROCS. 1998 INT'L CONF. ON PARALLEL AND DISTRIBUTED SYS. 50 (1998) (Cat. No.98TB100250), https://ieeexplore.ieee.org/document/800066
Ex. 1031	Xiangping Chen & Prasant Mohapatra, <i>Lifetime behavior and its impact on Web caching</i> , PROCS. 1999 IEEE WORKSHOP ON INTERNET APPLICATIONS 54 (1999) (Cat. No.PR00197), https://ieeexplore.ieee.org/document/788017
Ex. 1032	K.Y. Wong, K.H. Yeung, <i>Site-based approach in HTTP proxy design</i> , PROCS. 1999 ICPP WORKSHOPS ON COLLABORATION AND MOBILE COMPUTING (1999), https://ieeexplore.ieee.org/document/800066
Ex. 1033	U.S. Patent No. 6,088,721 (“Lin”)
Ex. 1034	PCT International Publication No. WO2000062185 (“Lea”)
Ex. 1035	Declaration of Jonathan Bradford

LIST OF CHALLENGED CLAIMS

[1.pre] A process for delivering an original media, comprising [the steps of]:

[1.a] placing said original media in a network system;

[1.b] creating an HTML document or browser language having proprietary media tags and placing said HTML document or browser language onto a Web server;

[1.c] actuating a user request from a customer browser for a Web page by having said Web server pass said requested Web page to an HTML parser on said network system;

[1.d] parsing said HTML document or browser language on said parser by looking for said proprietary media tags;

[1.e] looking up said proprietary media tags in a media tags database on said network system

[1.f] if at least one media tag of said proprietary media tags is not found, generating a Web media by using said at least one media tag;

[1.g] placing said generated Web media in a media cache on said network system;

[1.h] converting said at least one media tag to at least one standard HTML equivalent tag that refers to said media in said cache; and

[1.i] storing said at least one media tag and said at least one standard HTML equivalent tag in said database;

[1.j] modifying said HTML document or browser language by replacing said at least one media tag by said at least one standard HTML equivalent tag;

[1.k] delivering said modified HTML document or browser language from said network system to said Web server; and

[1.l] delivering said modified HTML document or browser language from said Web server to said customer browser for said customer to view.

[9.pre] A process for creating a media associated with an HTML equivalent tag and associated with a media tag for a system, comprising the steps of:

[9.a] receiving a request from said system for said HTML equivalent tag;

[9.b] combining said media tag with bandwidth information;

[9.c] determining if a media tag entry associated with said media tag exists in a media tag database;

[9.d] if said media tag entry exists in said media database,

determining if at least one original asset used to create said media has changed;

[9.d.1] if said at least one original asset used to create said media has not changed,

retrieving said HTML equivalent tag from said database; and

[9.d.2] return said HTML equivalent tag to said system;

[9.e] if said at least one original asset used to create said media has changed,
removing said media tag entry from said database;

[9.e.1] create said media using said media tag;

[9.e.2] storing said media in a media cache;

[9.e.3] generating an HTML document referring to said generated media;

[9.e.4] placing said media tag and said HTML equivalent tag in said media
database; and

[9.e.5] return said HTML equivalent tag to said system;

[9.f] if said media tag entry does not exist in said media database,
create said media using said media tag;

[9.f.1] storing said media in a media cache;

[9.f.2] generating HTML or browser language referring to said generated
media;

[9.f.3] placing said media tag and said HTML equivalent tag in said media
database; and

[9.f.4] return said HTML equivalent tag to said system.

Pursuant to §§311-319 and §42.1,¹ Akamai Technologies, Inc. (“Petitioner”) petitions for *inter partes* review (“IPR”) of claims 1 and 9 (“Claims”) of U.S. Patent No. 6,792,575 (Ex. 1001, “’575”), assigned to Equil IP Holdings LLC (“PO”). Petitioner requests review of the Claims, and judgment finding them unpatentable under §103.

I. INTRODUCTION

The purported invention of the ’575 is a process for delivering media using proprietary media tags. Specifically, the claims recite a method of “actuating” a user’s request, including the steps of parsing an HTML document for “proprietary media tags,” looking those tags up in a database, converting those proprietary tags to standard HTML equivalent tags, and delivering an HTML document with the standard tags in response to the request. If a proprietary tag is not cached, then the tag from the document is used to generate web media. Ex. 1003 (“Madisetti”), ¶32.

But such use of proprietary media tags was known in the art. For example, **Lafer** (Ex. 1004) discloses a system that responds to a user’s page request by parsing

¹ Section cites reference 35 U.S.C. (pre-AIA) or 37 C.F.R. as context indicates. All emphasis/annotations added unless noted. Figure annotations herein generally quote the Claims for reference. Citations herein are exemplary and not meant to be limiting.

a “tag-extended HTML file” for embedded proprietary tags that are not “part of the standard HTML tag library.” Lafer, 2:65-3:7, 3:52-55, 3:65-67, 4:8-15. These proprietary tags are looked up in a “tag cache,” and replaced in the HTML file with “HTML fragments” (*e.g.*, including fragments that are standard HTML tags) linked to underlying data stored in a “Content DB.” Lafer, 4:32-35, 3:32-43, Fig. 1. If the tag is not already cached, then the system uses the tag from the tag-extended HTML file to create a tag-content pair linking the proprietary tag to an HTML fragment and content stored in the Content DB. Lafer, 4:46-54, 3:22-36; Madisetti, ¶33. **Lafer’s** proprietary tag includes “substitution tokens” identifying “content data” and “user session information” used to “dynamically substitut[e]” the proprietary tag with an HTML fragment. Lafer, 4:24-31, 3:15-21.

Tso (Ex. 1005) further discloses that, when requested content has not been cached, the system retrieves the original content, transcodes it based on “selection criteri[a]” within the request specifying transcoding parameters, then caches and returns the transcoded content to the user. Tso, 4:1-5, 6:28-50, 6:64-8:13. A POSITA would have been motivated to modify **Lafer** with **Tso’s** teachings, yielding a system in which **Lafer’s** substitution tokens include information used to transcode content. Lafer, 4:24-31; Tso, 4:1-5, 6:28-50, 6:64-8:13; Madisetti, ¶34.

Claims 1 and 9 are rendered obvious by the combinations of **Tso, Huang, Cao** (which teaches removing cache entries that are not up to date), **and Moore** (which teaches HTML tag fragments) set forth herein. Madisetti, ¶34. Petitioner requests that the Board institute trial and find the Claims unpatentable.

II. MANDATORY NOTICES

A. Real Parties-in-Interest

Petitioner Akamai Technologies, Inc. is the real party-in-interest. No other party had access to or control over the present Petition, and no other party funded or participated in preparation of the present Petition.

B. Related Matters

Petitioner is concurrently filing IPR petitions challenging claims of related U.S. Patent Nos. 8,495,242 (“’242”) (IPR2023-00330) and 9,158,745 (“’745”) (IPR2023-00332). There are no other IPRs pending for the ’575 or patents related to the ’575 beyond those being concurrently filed.

The ’575 is the subject of the following co-pending civil action:

Equil IP Holdings LLC v. Akamai Technologies, Inc., No. 1-22-cv-00677 (D. Del.).

C. Counsel and Service Information:

James L. Davis, Jr. (Reg. No. 57,325) (Lead)
ROPES & GRAY LLP
1900 University Avenue, 6th Floor
East Palo Alto, CA 94303-2284
Phone: 650-617-4000
Fax: 617-235-9492
james.l.davis@ropesgray.com
akamai-equil-ropes-ipr-service@ropesgray.com

Scott McKeown (Reg. No. 42,866) (Back-Up)
ROPES & GRAY LLP
2099 Pennsylvania Avenue NW
Washington, DC 20006
Phone: 202-508-4740
Fax: 202-383-9335
scott.mckeown@ropesgray.com

Daniel W. Richards (Reg. No. 69,652) (Back-Up)
ROPES & GRAY LLP
1900 University Avenue, 6th Floor
East Palo Alto, CA 94303-2284
Phone: 650-617-4000
Fax: 617-235-9492
daniel.richards@ropesgray.com

Customer No. 28120

Mailing address for all PTAB correspondence:
ROPES & GRAY LLP, IPRM—Floor 43
Prudential Tower, 800 Boylston Street,
Boston, MA 02199-3600

Petitioner consents to electronic service of documents to the email addresses
of the counsel identified above.

III. PAYMENT OF FEES

The USPTO is authorized to charge any fees due during this proceeding to Deposit Account No. 18-1945, under Order No. AKTI-011-651.

IV. REQUIREMENTS FOR *INTER PARTES* REVIEW

A. Grounds for Standing

Pursuant to §42.104(a), Petitioner certifies that the '575 is available for IPR. Petitioner and any real parties-in-interest are not barred or estopped from requesting IPR challenging the claims on the grounds identified herein.

B. Identification of Challenge

Pursuant to §§42.104(b)-(b)(1), Petitioner requests IPR of claims 1 and 9, and that the Board cancel the same as unpatentable.

1. Specific Art on Which the Challenge Is Based

Petitioner relies on the following prior art:

Name	US Patent/ Title	Exhibit	Filed	Published/ Issued	Prior art under at least
Lafer	U.S. Patent 6,192,382	1004	9/24/1997	2/20/2001	§102(e)
Tso	U.S. Patent 6,421,733	1005	9/8/1997	7/16/2002	§102(e)
Cao	Maintaining Strong Cache Consistency	1006		4/1998	§102(b)

Name	US Patent/ Title	Exhibit	Filed	Published/ Issued	Prior art under at least
	in the World Wide Web				
Moore	U.S. Patent 6,310,601	1007	5/12/1998	10/30/2001	§102(e)

Cao was publicly accessible to, and indeed accessed by, POSITAs and interested researchers well before October 1999: it was published in, disseminated by, and publicly available through IEEE Transactions on Computers in April 1998, bears a 1998 copyright date, was available from the British Library by May 1998 and cited by multiple other publications/patents. Ex. 1006; Madisetti, ¶57 (IEEE publications were well-known resources for POSITAs and interested researchers); *Power Integrations, Inc. v. Semiconductor Components Industries, LLC*, IPR2018-00377, Paper 10, *10 (IEEE is “a well-known, reputable compiler and publisher,” taking official notice that relevant persons “rely on ... the copyright line of IEEE publications.”) (quotations and citations omitted); Ex. 1008 (IEEE declaration); Ex. 1020 (British Library letter); Exs. 1022-26 (describing the journal); Exs. 1027-1028 (patents/publications citing Cao); Exs. 1029-1034 (citing Cao).

2. Statutory Grounds on Which the Challenge Is Based

Ground (§103)	Claims	References
1	1	Lafer in view of Tso

Ground (§103)	Claims	References
2	9	Lafer in view of Tso and Cao
3	1	Lafer in view of Tso and Moore
4	9	Lafer in view of Tso, Cao and Moore

V. '575 PATENT AND PROSECUTION HISTORY

A. '575

Challenged '575 Claims 1 and 9 generally recite performing methods in a system in which, in response to a user request, the system reads a proprietary tag in an HTML file, processes it to generate and cache optimized media, and replaces the tag with standard HTML referencing the optimized media. '575, 1:9-11, 5:21-35; Madisetti, ¶35. Fig. 6 shows an embodiment of the system:

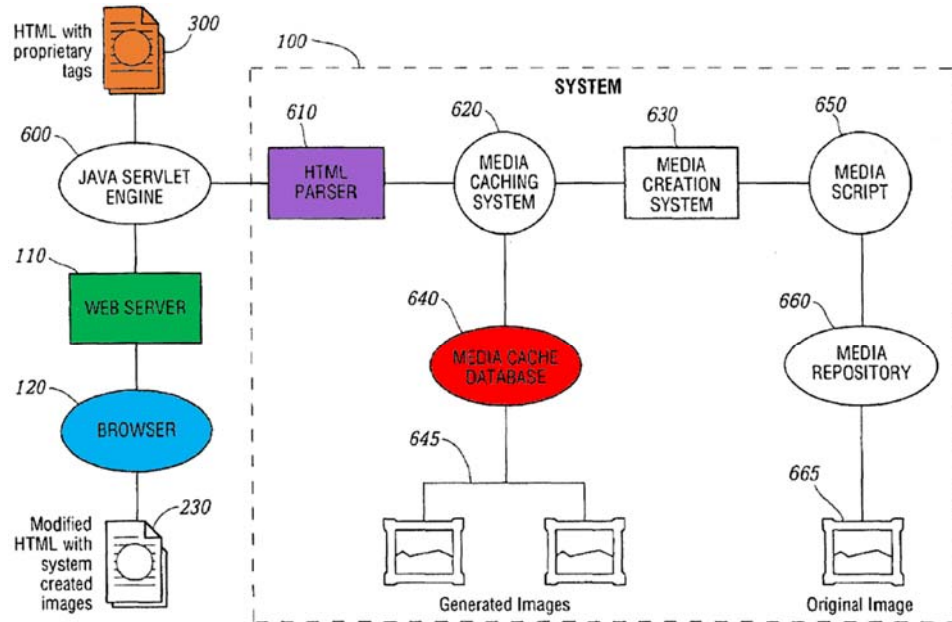


FIG. 6

A **web server** communicates with a user's **browser**. '575, Fig. 6, 7:28-29. The server stores **HTML with proprietary tags** for content requested by a user, and is connected to a system that includes an **HTML parser** and a **media cache database**. '575, Fig. 6, 8:18-20, 7:21-26, 8:21-24. The **database** stores both proprietary tags and generated web media. '575, Fig. 6, 7:47-57; Madisetti, ¶35.

Figure 9 shows the process for handling user requests:

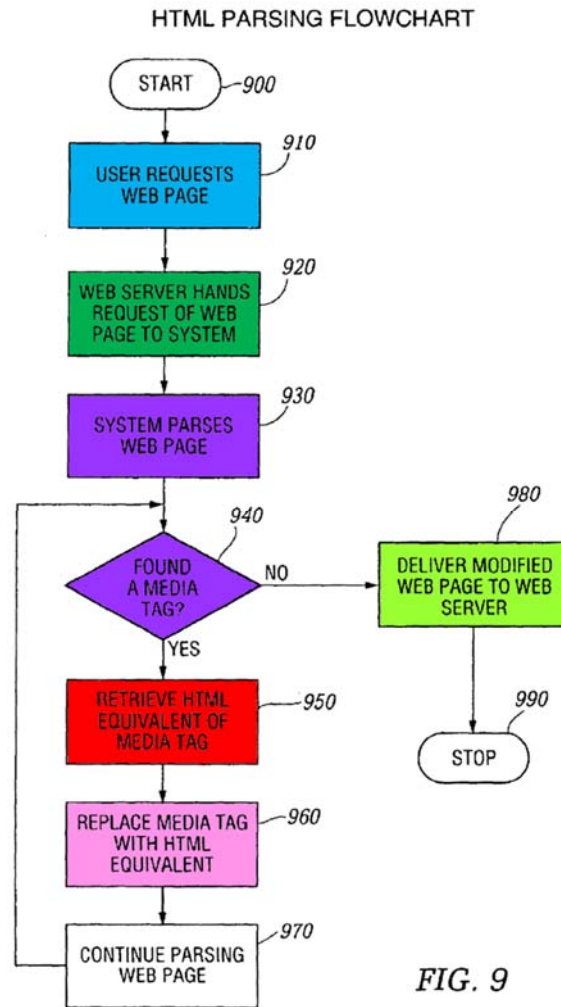
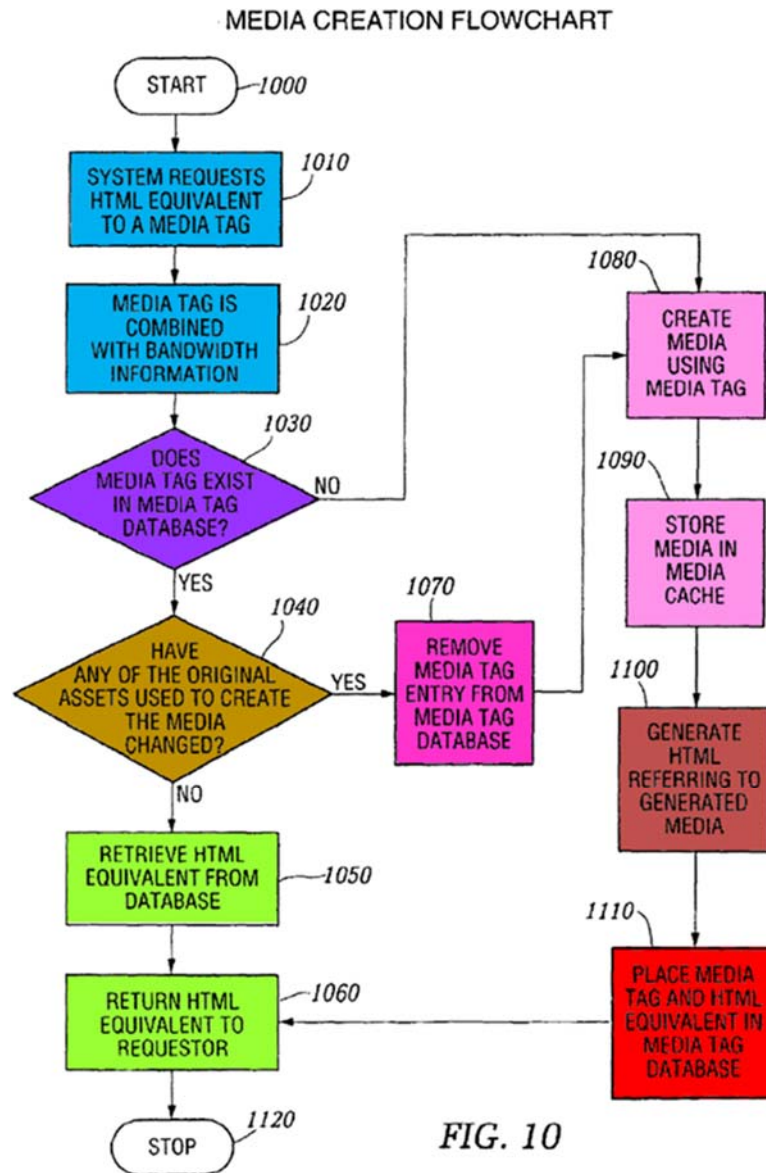


FIG. 9

When a **user requests a web page**, the **server passes the request to the system**. '575, 8:48-49. The system then **parses the requested web page for media tags**. '575, 8:49-50. When a media tag is found in the media cache database, the system **retrieves an HTML equivalent of the tag from the database**, and **replaces the tag in the web page with an HTML equivalent**. '575, 8:49-54. When the system finishes parsing the page, it **delivers the page back to the server**, which then returns it to the user. '575, 8:54-56; Madisetti, ¶36.

Figure 10 shows the process for media creation and retrieval from cache:



First, the system **requests a media tag, combined with bandwidth information, from the media tag database.** '575, 8:57-61. The database is checked to **determine whether the requested media tag exists.** '575, 8:61-63. If so, the system **checks if the original asset is unchanged** and, if so, **retrieves and**

returns the HTML equivalent tag. '575, 8:63-67. If the original asset changed, the system updates the database by **removing the old media tag entry**, using the page's media tag to **create and store a new media in cache, generating HTML referencing the generated media**, and **storing the media tag and HTML equivalent in a media tag database** before **returning the generated HTML back to the system.** See '575, 8:61-9:9, Fig. 10; Madisetti, ¶37.

The system checks whether a media tag exists in a database at step 1030, and if the tag does not exist, the system uses same steps 1080-1110 to create and store the generated **media**, the **media tag, and the HTML equivalent tag.** *Id.*; Madisetti, ¶38.

B. Prosecution History

After the Applicant elected claims to address a restriction requirement, the Examiner allowed draft claim 35 (which issued as claim 1) and indicated that draft claim 45 (which issued as claim 9) would be allowable if amended to address a §112 rejection by adding the word “media” in elements reciting “to create said media [has/has not] changed.” Ex. 1002, 80-83, 86-115. These Claims were allowed without further rejections. Ex. 1002, 139-41; Madisetti, ¶39.

While the Examiner did reject other claims based on prior art, the Applicant canceled those claims. Ex. 1002, 124-38. Those claims did not recite, *e.g.*, “actuating

a user request” or “receiving a request” ([1.c], [9.a]), or “removing said media tag entry from said database” ([9.g]). *See* Ex. 1002, 89-115; Madisetti, ¶40.

VI. §325(d) AND §314(a) DISCRETION DOES NOT APPLY

1. §325(d)

Under *Advanced Bionics v. Med-El Elektromedizinische Gerate*, IPR2019-01469, Pap. 6, *8-9, the Board should not exercise §325(d) discretion.

The grounds raised by this Petition are not the same or substantially the same as the art and arguments raised during ’575’s prosecution. Lafer, Tso, Cao, and Moore were not considered during prosecution. The Claims were not rejected based on any prior art prior to allowance. *See* Ex. 1002, 102-15, 139-41. While other claims were rejected based on prior art, those claims did not recite all of the elements of the challenged Claims. Ex. 1002, 89-115, 102-15, 132-38. The specification’s citations to additional references (’575, 2:38-3:42) were not included in an invention disclosure statement (37 C.F.R. §1.98(a)(1)), and those disclosures identified to the USPTO do not disclose or render obvious the Claims or include the same teachings discussed in the Grounds. As explained below, the Claims are obvious under the identified Grounds. *See* §IX.

Even if the art or arguments were substantially the same, the Examiner erred in a manner material to the patentability of the Claims. The Examiner “did

not expressly consider” **Lafer, Tso, Cao** or **Moore** as prior art, rendering it difficult, if not impossible, to explain “how the Examiner might have considered the arguments presented in the Petition.” *Bowtech, Inc. v. MCP IP, LLC*, IPR2019-00379, Pap. 14, *20 (declining to exercise §325(d) discretion). Even if any of the art considered by the USPTO was substantially similar to these references (none is), the Examiner would have erred in not rejecting the Claims.

2. §314(a)

The co-pending litigation also does not warrant exercising discretion under §314(a) based on the *Fintiv* factors.

1: Following institution, Petitioner intends to seek a stay of the Delaware case pending the outcome of this IPR and concurrently-filed IPR2023-00330 and IPR2023-00332.

2: No trial date is set.

3: The case is in the pleading stage and the court has not ruled on the initial motion to dismiss.

4: After the final written decision, which is expected before trial even without a stay, the same grounds, arguments, and evidence could not be presented in the litigation.

5: The litigation and PTAB parties are the same.

6: The merits of this Petition are compelling as shown herein.

VII. LEVEL OF ORDINARY SKILL

For purposes of this Petition only, Petitioner assumes the Claims are entitled to the '575's filing date, 10/21/1999. Madisetti, ¶41.

A person of ordinary skill in the art ("POSITA") as of the claimed priority date of the '575 would have had a bachelor's degree in computer systems, computer science, or the equivalent thereof, and at least two years of experience with networked media delivery or related technology. Madisetti, ¶¶42-44. More education can supplement practical experience, and vice-versa. *Id.*

VIII. CLAIM CONSTRUCTION

Claim terms subject to IPR are construed using *Phillips v. AWH Corp.*, 415 F.3d 1303, 1313 (Fed. Cir. 2005). §42.100(b). Only terms necessary to resolve the controversy need construction. *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor*, 868 F.3d 1013, 1017 (Fed. Cir. 2017). Because the prior art asserted herein discloses embodiments within the Claims' indisputable scope, the Board need not construe the Claims' outer bounds. All claim terms should be construed according to their plain and ordinary meaning. Madisetti, ¶¶45-46.

A. [1.f]-[1.i]/[9.d]-[9.f.4]

Although the prior art meets all elements of the Claims, [1.f]-[1.i] and [9.d]-[9.f.4] recite conditional limitations that need not be performed. *Microsoft Corp. v.*

Uniloc 2017 LLC, IPR2019-01187, Pap. 20, *12-14 (claim with conditional steps encompasses a claim “where only the non-conditional steps are performed”); *see also* MPEP §608.01(m).

[1.f]-[1.i] recite that “if at least one media tag ... is not found,” then the “placing,” “converting ... and” “storing” steps ([1.g]-[1.i]) are performed. But if the media tag *is* found, those steps need not be performed. Claim 1 therefore is rendered obvious regardless of whether the art meets [1.f]-[1.i]. Madisetti, ¶¶47, 49-51, 53.

Similarly, claim 9 covers methods in which (1) said “media tag entry exists” and the “original asset” “has not changed” ([9.d]-[9.d.2]); (2) the entry exists and the “original asset” “has changed” ([9.d], [9.e]-[9.e.5]); and (3) the entry “does not exist” ([9.f]-[9.f.4]). These are “mutually exclusive conditional steps,” and meeting any one of these sets of steps is sufficient to meet the claim. *Microsoft*, IPR2019-01187, Pap. 20, *14-17. Madisetti, ¶¶48-50, 52-53.

B. Preambles

Regardless of whether the preambles are limiting, the prior art discloses the preambles. *See* §IX; Madisetti ¶54.

IX. GROUNDS OF UNPATENTABILITY

This Petition is supported by the Declaration of Dr. Vijay Madisetti, which describes the prior art’s scope and content at the time of the ’575. Madisetti, ¶¶1-31.

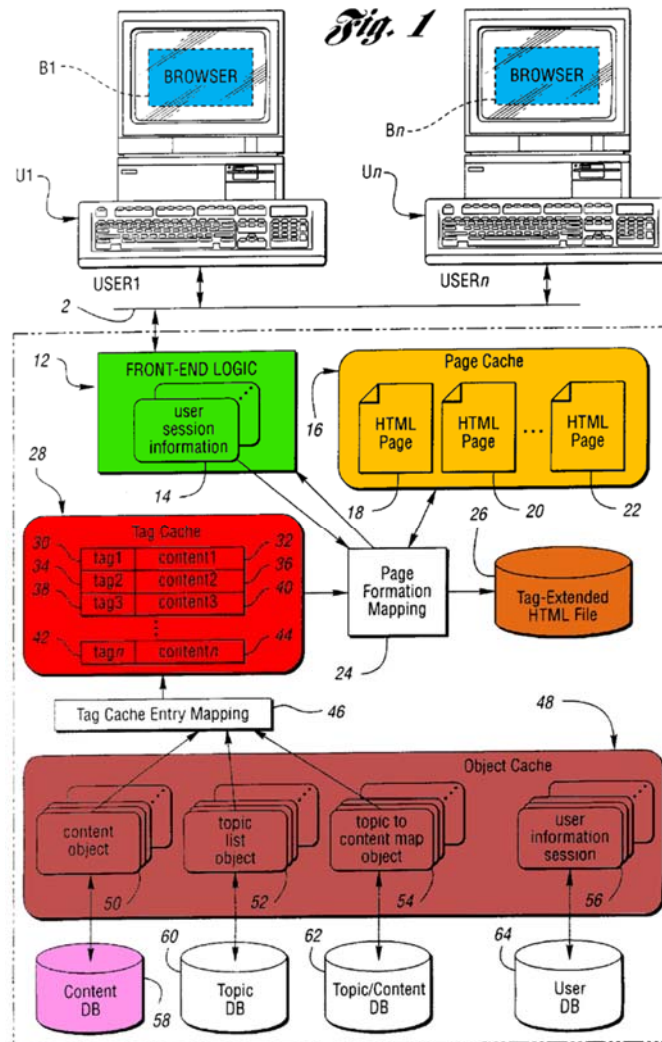
The prior art renders the Claims unpatentable for two independent reasons. Madisetti, ¶¶55-59.

A. Ground 1: Lafer in View of Tso (Claim 1)

Lafer in view of **Tso** renders obvious Claim 1; any purported differences between this art and the claim would have been obvious to a POSITA. Madisetti, ¶60.

1. Overview of Lafer

Lafer teaches creating “personalize[d] individual pages” by using an “HTML page” stored with embedded special “tags,” replacing those tags with cached HTML fragments, and returning the personalized page to the user. Lafer, Abstract, 1:45-48; Madisetti, ¶61. Figure 1 shows Lafer’s system:



Madisetti, ¶61. **Browsers** communicate with the system's **front-end logic** (e.g., implemented as a server). Lafer, 2:65-3:7. “Tag-extended HTML files,” which include “special tags” that are not “part of the standard HTML tag library,” are stored in a **storage medium or device**. Lafer, 3:15-16, 4:10-13, 4:55-60. These special tags have “substitution tokens” specifying attributes including “content data” and “user session information.” Lafer, 4:24-31. The system stores “HTML fragments” as content objects in the **object cache** and associates these fragments with content data

(which a POSITA would have understood includes media content) stored in, *e.g.*, a **content database**. Lafer, 3:37-43; Madisetti, ¶62. As was well-understood in the art, “HTML fragments” are HTML code, such as HTML tags. Madisetti, ¶¶62-63; Ex. 1007, 3:41-53, 4:5-13, 5:29-30, 6:59-7:13. Webpages display images, *e.g.*, using commands implemented with “img” tags. Ex. 1007, 3:41-4:35, 5:29-30. These tags provide “options” for basic image adjustments, including changing its “height” or “width.” *Id.* Lafer thus not only renders obvious, but affirmatively discloses, using an HTML “image” tag as a “fragment” referencing an image to be inserted in a webpage. *Id.* A **“tag cache”** stores tag-content pairs associating the special tags with the HTML fragments (including fragments that are standard HTML tags). Lafer, 3:22-43. A **page cache** stores previously-generated personalized pages for reuse. Lafer, 3:8-14; Madisetti, ¶62.

Lafer further discloses that Figure 1’s logical units “could be combined into larger logical units or further subdivided to conveniently implement the system.” Lafer, 4:65-5:2. For example, a POSITA would have understood that the back-end **object cache** accessed using the **tag cache**, along with the content items stored in, *e.g.*, the **content DB**, could be part of a back-end system, while front-end server components, including the **front-end logic**, the **page cache** of previously-generated

pages, and the **storage for tag-extended HTML files**, could be integrated into a front-end system. *Id.*; Madisetti, ¶64.

In Fig. 2, **Lafer** illustrates a disclosed process performed in the system:

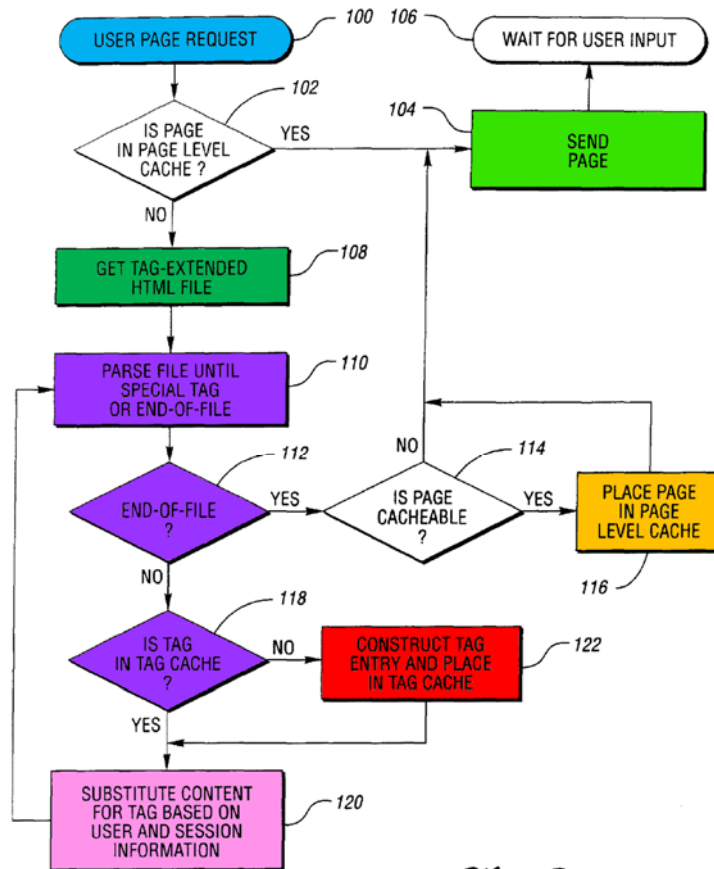


Fig. 2

Lafer, Fig. 2; Madisetti, ¶65. Upon receiving a **user’s page request**, if the requested page is not already cached, the system **retrieves a tag-extended HTML file**. *Id.*; Lafer, 3:52-65. The system **parses this file for special tags** “until either a particular tag or the end-of-file is encountered.” Lafer, 3:8-10, 3:65-67. Each identified “special” tag **is replaced with an HTML fragment**. Lafer, 4:8-10, 3:22-25. If a

“special” tag does not exist in the cache, a **“tag-content pair” is created and saved in cache**. Lafer, 4:46-48, Fig. 2. Once the end of the file is reached, the page is **placed in the page cache** then **sent to the user**. Lafer, 3:67-4:7; Madisetti, ¶66.

2. Overview of Tso and Motivation to Modify Lafer with Tso’s Teachings

In the same field of endeavor, Tso discloses a “system for dynamically transcoding data” to be “transmitted between two computers.” Tso, 1:13-14, 1:24-30, 2:47-49; Madisetti, ¶¶67-69. Requests from a client for “image, video, or HTML” content include “selection criteria” used to transcode the content. Tso, 3:46-54, 4:6-10, 6:64-8:9; Madisetti, ¶67. The “selection criteria” specify user-specific parameters, including “user,” “network,” and “proxy” server parameters such as “bandwidth.” Tso, 7:15-8:9; Madisetti, ¶67.

After receiving a request for content, the server checks whether a transcoded version of the requested content is already cached. Tso, 9:49-56, 10:24-27, 5:62-6:28. If not, the server retrieves an “original” version of content from the network. Tso, 10:27-32, 6:28-30, 5:8-10. The server then uses “selection criteria” to determine how to transcode and then transcodes the content. Tso, 10:32-47, 6:37-44, 6:64-7:14. The transcoded and original versions of the content are both cached, and the transcoded version is returned to the user. Tso, 10:47-49, 4:1-5, 5:8-26, 6:28-31, 6:48-50. Madisetti, ¶68.

Lafer and **Tso** are analogous prior art and in the same field as the '575—networked media caching and delivery—and reasonably pertinent to the alleged problems identified in the '575 of automating media delivery and reducing required overhead and time to generate Web pages. *E.g.*, '575, 4:47-67, 5:33-37, 6:36-40, 13:1-24; Lafer, 1:15-18, 1:24-35, 4:46-61; Tso, 1:10-31, 1:59-2:5, 4:1-5; Madisetti, ¶69.

Lafer leaves it to a POSITA to implement its broad disclosures of caching media objects and using user preferences and session information stored in “substitution tokens” to generate personalized content. *E.g.*, Lafer, 3:2-7, 3:15-43, Fig. 1; Madisetti, ¶70. **Tso** provides detailed teachings about how to perform such caching, and personalization. For example, **Tso** discloses generating and storing original and transcoded versions of media objects in a server-side cache. Tso, 4:1-5; Madisetti, ¶71. **Tso** also discloses “selectively transcod[ing] content based on a predetermined selection criterion” or “combinations thereof” that indicate user and session information, including criteria relating to the “network client,” “network characteristics” including “bandwidth,” and “user preferences.” Tso, 3:48-54, 7:20-8:21; Madisetti, ¶71. A POSITA would have been motivated to modify **Lafer** with these teachings from **Tso** by implementing a system in which (1) original media objects are retrieved and transcoded, and the original and transcoded versions are

stored in the “Content DB”; and (2) “substitution tokens” in Lafer’s special tags carry selection criteria used to transcode content, including bandwidth for at least the following reasons. Madisetti, ¶71.

First, Lafer leaves it to a POSITA to implement its teachings of substituting tags with HTML fragments referencing media content “based on stored user preferences and session information” carried in a special tag’s “substitution tokens.” Lafer, 3:2-35, 4:7-31. **Tso** teaches that the received request includes “selection criteria” and provides detailed teachings of criteria including the “network client,” “network,” and “user preferences,” including “bandwidth,” as “selection criteria” to “selectively transcode content,” and provides a flexible teaching of providing such information to the server in a request. Tso, 3:48-54, 7:20-8:21, 9:26-33; Madisetti, ¶72. A POSITA would have looked to modify **Lafer** with **Tso’s** teachings to implement Lafer’s “substitution tokens” because Tso teaches a known technique for providing session information to the system, and provides a broad disclosure of including such information as selection criteria in a request, including, *e.g.*, describing the use of session information such as “bandwidth” to adapt to the “condition of a communications link” during a session, improving reliability of the communication. Tso, 7:4-41, 16:63-17:9. Modified with **Tso’s** teachings, **Lafer’s** substitution tokens carry user preference and session information, including **Tso’s**

selection criteria such as “bandwidth.” Madisetti, ¶72; *see also* Ex. 1011 (Starnes), 2:24-41; Ex. 1012 (Bharadvaj), 1-4.

Second, both **Lafer** and **Tso** address similar needs to improve the efficiency of a system. Lafer, 1:15-18, 1:27-35, 4:46-61; Tso, 4:45-47; Madisetti, ¶73. **Lafer** teaches that “caching of the information stored in a database accessed by a server” will “improve distribution efficiency.” Lafer, 1:15-18, 1:27-35, 4:45-47. **Tso** similarly teaches that caching original and transcoded versions of content advantageously eliminates “the need to re-retrieve” and “re-transcode” content, further improving efficiency. Tso, 4:1-5. A POSITA further would have recognized that applying **Tso’s** teaching of caching original media content in a server-side cache would further improve the efficiency of **Lafer’s** caching teachings, allowing content to be delivered to users more quickly. A POSITA therefore would have been motivated to modify **Lafer** with **Tso’s** teachings by caching both original and transcoded versions of media in **Lafer’s** Content DB, as taught by **Tso**, to advantageously minimize re-retrieval and re-transcoding of content. Madisetti, ¶73; *see also* Ex. 1013 (Boucher), 1:41-50, 5:1-5; Ex. 1014 (Mohan), 106; Ex. 1015, (Fox), 13.

Third, **Tso** provides detail on how to implement an efficient caching process applicable to **Lafer**. **Lafer** discloses that a “Content DB” stores cached “data” for

HTML fragments, but leaves it to a POSITA to implement populating this cache. Lafer, 3:32-43. **Tso** teaches retrieving and caching both “original” and “transcoded” versions of content when the content is transcoded to eliminate the need for re-transcoding and re-retrieval. Tso, 14:37-46, 3:48-4:5. **Tso** thus teaches how to implement **Lafer’s** “Content DB” by storing both the transcoded version of content data to be accessed by a user as taught in **Lafer**, and the original content data, which allows the cache to efficiently deliver requested content to the user quickly. Madisetti, ¶74; Tso, 4:1-5; *see also* Ex.1010 (Delp), 3:64-4:10, 6:20-32; Boucher, 6:66-7:2, 7:19-25, 9:13-39.

In light of the above teachings, a POSITA would have had a reasonable expectation of success in modifying **Lafer** with **Tso’s** teachings by (1) using **Tso’s** selection criteria in “substitution tokens,” and (2) transcoding and caching content. **Tso**, like **Lafer**, already teaches that its disclosures can be used for “customizing content.” Tso, 7:62-8:4, 9:26-29; Lafer, 3:44-47, Abstract, 1:27-30; Madisetti, ¶75. Indeed, it was well understood for a user request to specify parameters for customizing content, and for a cache to store original and customized versions of content, using even standard HTML. *E.g.*, Ex. 1007 (Moore), 1:30-41, 4:5-60, 5:15-33; Mohan, 104, 106, 108; Delp, 3:64-4:10, 6:20-32, 7:10-18, Fig. 12; Fox, 13; Boucher, 7:14-17. Modifying **Lafer** with **Tso’s** teachings would have been a

straightforward application of those teachings in a manner that advantageously improves the efficiency of a system that endeavors to quickly provide customized content to fulfill user requests. Madisetti, ¶75.

3. Claim Chart

'575 Claim	Lafer in view of Tso
[1.pre] A process for delivering an original media, comprising the steps of:	Lafer discloses a process for delivering an original media (<i>e.g.</i>, “distributing” “content” in an “HTML page[]” in response to “user requests”). <u>E.g., Lafer:</u> Lafer discloses “distributing personalized HTML pages” containing “content” “map[ped]” with underlying “content data”—which a POSITA would have understood includes media content data—that is created by “dynamic[ally] substituti[ng] ... content for tag values.” Madisetti, ¶76-77. • 2:15-33 (“[A] system for constructing and selectively distributing Web pages comprises a server processor for translating user requests into HTML page requests, and a tag cache containing a plurality of predetermined HTML fragments stored therein. A first storage device is provided for storing a plurality of HTML files, one or more of which are selectively augmented with a plurality of particular tags each associated with one or more of the predetermined HTML fragments. ... A mapping logic processor communicates with the first storage device, second storage device and the tag cache so as to construct an HTML file by substituting one or more predetermined HTML fragments for each particular tag embedded in a requested HTML file based on the stored user preference information. The server processor is further operative to distribute the constructed HTML files to the user.”)

'575 Claim	Lafer in view of Tso
	3:23-42 (“A <i>tag cache</i> as shown in reference 28 <i>contains HTML fragments that are substituted for tag values. The tag cache is composed of a plurality of tag-content pairs</i> as shown in blocks 30 and 32, 34 and 36, 38 and 40, and 42 and 44. In the preferred embodiment, <i>the tag cache is implemented using a persistent, e.g., disk-based, hash table.</i> However, as will be appreciated by one of ordinary skill in the art, <i>other software or hardware methods may be used to implement the tag cache.</i> <i>A tag cache entry mapping, as shown in block 46, is used to map HTML fragments into the tag.</i> Depending on <i>the type of tag into which the HTML data will be substituted, the mapping may require content data</i>, topic list data, or topic-to-content mapping data. More specifically, <i>HTML fragments mapped into the tag cache are fetched from an object cache</i> referenced as block 48. <i>The object cache holds content objects 50</i>, topic list objects 52, and topic-to-content map objects 54 as well as user information objects shown as block 56. The <i>data for these objects are maintained in respective databases 58, 60, 62 and 64.</i>”) 4:55-61 (“Thus, the present invention provides a system and <i>method for selectively constructing and distributing personalized HTML pages by storing selected fragments of HTML text in a tag cache</i>, and rapidly rendering the personalized HTML pages via <i>special tags embedded in extended HTML files which allow identification and retrieval of the desired fragments.</i>”) See also Abstract, 1:54-58, 3:2-3, 3:16-20, 4:48-54. Tso further explicitly discloses delivering original media (e.g., “transmission” of “content” with an “original” version). <u>E.g., Tso:</u>

'575 Claim	Lafer in view of Tso
	Lafer already discloses that the “Content DB” caches the underlying “data” for content, which a POSITA would have understood includes original media content, but Tso provides additional explicit disclosure of caching “original” versions of content, including “image” and “video” content for “transmission” to users. Tso, 3:41-54, 4:1-5. As discussed in §IX.A.2, a POSITA would have been motivated to modify Lafer with Tso’s detailed teachings of caching and delivering media content to advantageously and efficiently deliver personalized pages including media content. Madisetti, ¶¶78-80. Figure 3 FIG.3 3:51-54 (“For example, one or more transcode service providers 24 may provide the capability to compress and/or scale <i>different types of data content, such as image, video</i>, or HTML (HyperText Markup Language).”) 4:1-5 (“<i>Server-side cache memory 30 may be used to store both original and transcoded versions of content for later transmission to network client 12 without the need to re-retrieve the content from Internet 18 or to re-transcode the content.</i>”)

'575 Claim	Lafer in view of Tso
	• <i>See also</i> 2:44-53, 3:31-34, 5:8-10, 5:62-64, 6:9-13, 6:19-23, 10:37-44.
[1.a] placing said original media in a network system;	Lafer discloses placing said original media (<i>e.g.</i>, placing “data for” “content”) in a network system (<i>e.g.</i>, in the “Content DB” of a network system that also includes “logical units” for an “Object Cache” and “Tag Cache,” and communicates over the “Internet”). <u>E.g., Lafer:</u> <i>See</i> [1.pre]. Lafer also discloses that the “data” for “content” on webpages is stored in a “Content DB.” Lafer, 3:23-43, Fig. 1. “Content” refers to, <i>e.g.</i>, HTML fragments, associated with underlying “data” and not only renders obvious, but affirmatively discloses, media associated with an HTML tag—for example, an HTML “img” tag fragment is associated with an underlying image media file. <i>See id.</i>; Madisetti, ¶¶81-82, 84; §IX.A.1 (regarding understanding of a POSITA/obviousness). Lafer discloses that the “Content DB” is on a server-side back-end system with logical components including a “Tag Cache” and “Object Cache” that are used to manage tags and content. Lafer, Fig. 1. These components are shown as “logical units” that Lafer discloses combining “into larger logical units ... to conveniently implement the system.” Lafer, 4:65-5:2. Indeed Lafer discloses implementing the “front-end logic” block as a “standard web server”—confirming that it teaches “server 10” broken up into separate server-side devices. Lafer, 2:65-3:3. Madisetti, ¶85. Lafer further discloses using the “Tag Cache,” “Object Cache,” and “Content DB” to substitute “HTML fragments” for “tag values.” Lafer, 3:23-43. The “HTML fragments” are stored in the “Object Cache” and reference “data for the[] objects” stored in, <i>e.g.</i>, the Content DB. <i>Id.</i> Because Lafer discloses that these components work together to replace tags with values, and combining or

'575 Claim	Lafer in view of Tso
	subdividing the “logical units” in Fig. 1 “to conveniently implement the system,” Lafer not only renders obvious, but discloses implementing these components together as a back-end network system—disclosing this element. Madisetti, ¶86. Figure 1 1:59-2:1 (“[A] method for selectively distributing information including personalized Web pages on a network such as the Internet is characterized by the steps of

'575 Claim	Lafer in view of Tso
	storing in a tag cache at least one HTML fragment derived from information stored in a database as determined from a user's input preference information, and <i>associating the at least one HTML fragment</i> stored in the tag cache <i>with a particular tag to be selectively embedded in an HTML document</i> or file stored in a server.”) • 3:15-21 (“<i>A storage medium or device, shown as block 26, is used to hold tag-extended HTML files.</i> In accordance with the present invention, the tags form an extension to the HTML language that allow <i>for dynamic substitution of content for tag values based on stored user preferences and session information. The storage device may be implemented as</i> a directory system or as <i>a database.</i>”) • 3:23-42 (see [1.pre]) • 4:58-5:2 (“[A]ctual implementation of the system architecture is susceptible to modification. For example, <i>the logic blocks as described in conjunction with FIG. 1 can be implemented in hardware, software or a combination thereof, and that the blocks could be combined into larger logical units or further subdivided to conveniently implement the system.</i>”) • See also Abstract, 2:50-57, 4:32-35. Tso further explicitly discloses placing said original media in a network system (e.g., original media (see [1.pre]) is cached in “[s]erver-side cache memory”). <u>E.g., Tso:</u> See [1.pre]. As discussed for [1.pre], Lafer already discloses caching in the “Content DB” the underlying “data” for content, which a POSITA would have understood includes original media, but Tso provides

'575 Claim	Lafer in view of Tso
	additional explicit disclosure of original media—this media is cached in “[s]erver-side cache memory 30” on a server connected to the “Internet.” Tso, 3:41-54, 4:1-5. As discussed in [1.pre] and §IX.A.2, a POSITA would have been motivated to modify Lafer with this teaching to advantageously efficiently deliver personalized pages without unnecessary re-retrieval of media data. Madisetti, ¶¶83, 87.
[1.b] creating an HTML document or browser language having proprietary media tags and placing said HTML document or browser language onto a Web server;	Lafer discloses creating an HTML document or browser language having proprietary media tags (<i>e.g.</i>, using a “page formation mapping process” to create “tag-extended HTML files” having “specialized tags” that are not “part of the standard HTML tag library”) and placing said HTML document or browser language onto a Web server (<i>e.g.</i>, storing “tag-extended HTML files” in the “storage medium” of a server that also comprises “front-end logic” and “Page Cache”). <u>E.g., Lafer:</u> See [1.pre]-[1.a]. As explained for [1.a], Lafer not only renders obvious, but discloses that the “Tag Cache,” “Object Cache,” and “Content DB” together form a network system. See [1.a]. Similarly, Lafer discloses that in part of the “page formation mapping process,” the “front-end logic,” “page cache,” and “storage medium” together serve front-end functions of receiving user requests, returning pre-cached copies of pages, and retrieving tag-extended HTML files to be further processed by the network system (<i>see</i> [1.a]). Lafer, 2:65-3:22. Because these logical units can be “combined” (Lafer, 4:65-5:2), Lafer thus not only renders obvious, but discloses that the claimed Web server is formed by combining the “front-end logic” (which “can be implemented as a standard web server”), “page cache,” and “storage medium.” Lafer, 2:65-3:22, 4:65-5:2; Madisetti, ¶¶88-90. Lafer further discloses using, on this Web server, a “page formation mapping process” to create, and then store in the server’s “storage

'575 Claim	Lafer in view of Tso
	medium,” “tag-extended HTML files.” 2:65-3:22. The HTML files are “selectively augmented with” “specialized tags” that are proprietary at least because they are not “a part of the standard HTML tag library.” 2:20-25, 4:10-22; Madisetti, ¶91. Figure 1 Figure 1 illustrates a system architecture for a web server. At the top, multiple users (U1, Un) are shown, each with a browser (B1, Bn) and a computer. They are connected to a central Web Server (2) via a communication medium. The Web Server is enclosed in a dashed green box and contains several components: Front-End Logic (12): Receives requests from users and manages user session information (14). Tag Cache (14): A table-like structure with columns for tags and content. It lists entries: tag1/content11, tag2/content12, tag3/content13, ..., tagn/contentn. Page Cache (16): Stores HTML Pages (20, 22). Page Formation Mapping (18): A component that maps tags from the Tag Cache to specific HTML pages in the Page Cache. Object Cache (48): Stores various objects: content objects (50), topic list objects (52), topic to content map objects (54), and user information sessions (56). Databases: At the bottom, four databases are shown: Content DB (58), Topic DB (60), Topic/Content DB (62), and User DB (64). Arrows indicate data flow between the Object Cache and these databases. 2:15-33 (see [1.pre]) 2:65-3:22 (“Considering the server 10 in detail, <i>front-end logic is responsible for interfacing with users U1 to Un through the communication medium 2</i>. As represented by block 12, <i>the front-end logic can be implemented as a</i>

'575 Claim	Lafer in view of Tso								
	<i>standard web server</i>. The front-end logic services the user by returning requested information in the form of HTML pages. <i>The front-end logic generally operates to collect session information as shown in block 14. This information can be obtained, for example, through the use of HTML forms as is commonly done with web pages.</i> <i>A page formation mapping process, as shown in reference 24, provides construction of the HTML pages</i> in accordance with the present invention. A page cache, represented by block 16, reduces latency by holding HTML pages that are likely to be used again in the future, as represented by references 18, 20 and 22. Techniques for the caching of pages is well known. <i>A storage medium or device, shown as block 26, is used to hold tag-extended HTML files.</i> In accordance with the present invention, the tags form an extension to the HTML language that allow for dynamic substitution of content for tag values based on stored user preferences and session information.”) • 4:10-22 (“More specifically, <i>a tag in accordance with the present invention does not constitute a part of the standard HTML tag library.</i> The tag syntax chosen for the preferred embodiment uses a special begin symbol and end symbol to mark the particular tag as follows: <table> <tr> <td><! - - CW></td><td>begin symbol</td></tr> <tr> <td>type=typename</td><td>tag type attribute</td></tr> <tr> <td>. . .</td><td>additional tag attributes</td></tr> <tr> <td><CW - - ></td><td>end symbol</td></tr> </table> ”) • 4:58-5:2 (see [1.a]) • See also 1:54-58, 2:50-51, 2:67-3:2, 4:55-61.	<! - - CW>	begin symbol	type=typename	tag type attribute	. . .	additional tag attributes	<CW - - >	end symbol
<! - - CW>	begin symbol								
type=typename	tag type attribute								
. . .	additional tag attributes								
<CW - - >	end symbol								

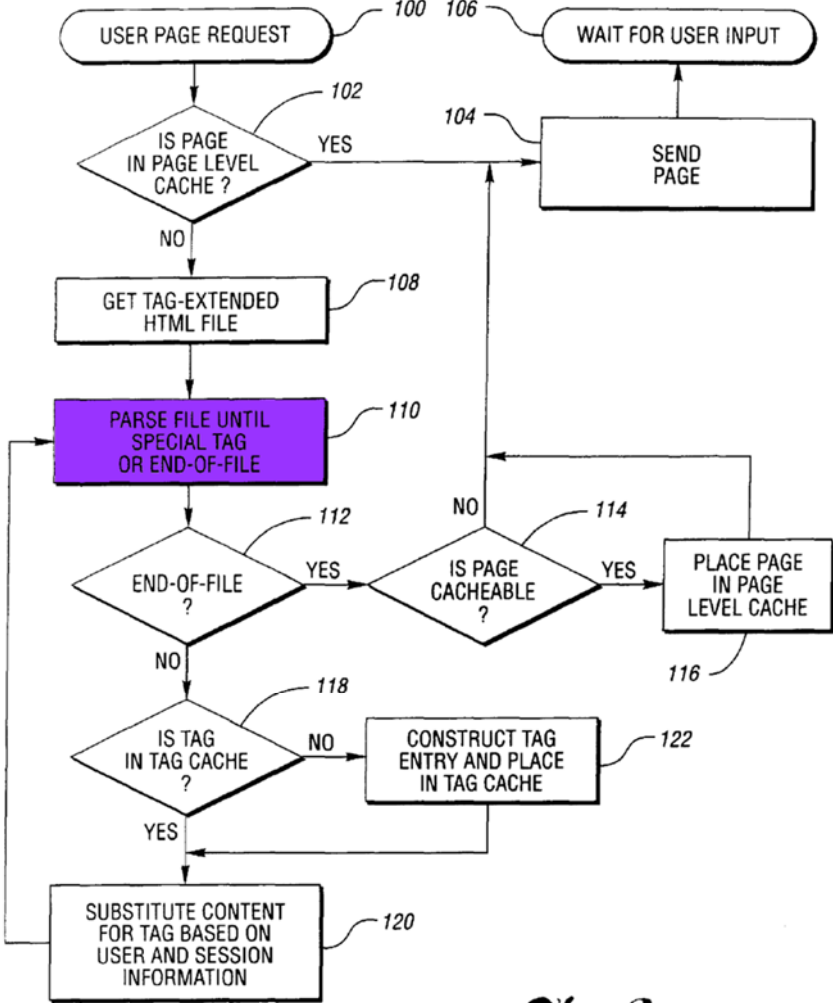
'575 Claim	Lafer in view of Tso
[1.c] actuating a user request from a customer browser for a Web page by having said Web server pass said requested Web page to an HTML parser on said network system;	Lafer discloses actuating a user request from a customer browser for a Web page (<i>e.g.</i>, a “user” views pages on a “browser,” and the system “services the user by returning requested information”) by having said Web server pass said requested Web page to an HTML parser on said network system (<i>e.g.</i>, web server (<i>see</i> [1.b]) “obtain[s]” the “tag-extended HTML file” from cache and provides it to “appropriate software, hardware, or a combination of both” on a network system (<i>see</i> [1.a]) used to “parse[]” the page). <u>E.g., Lafer:</u> <i>See</i> [1.a]-[1.b]. Lafer also discloses that when a user issues a request to the Web server “front-end logic” (<i>see</i> [1.b]), the Web server obtains “requested information” by interfacing with the network system to pass a tag-extended HTML file to the network system (<i>see</i> [1.a]) when additional back-end processing is required to form the page that will be returned to the user. Lafer, 2:65-3:22, 3:62-67, 1:15-18. The network system then implements part of the “page formation mapping” process in which “the tag-extended HTML file” is “parsed until ... a particular tag ... is encountered.” Lafer, 3:8-10, 2:65-3:22, 3:62-67. Components of the network system used to parse this file form an HTML parser. <i>See id.</i>; Madisetti, ¶¶92-94.

'575 Claim	Lafer in view of Tso
	• Figure 1 Figure 1 illustrates a system architecture for a web server and network system. The system includes multiple users (USER1, USERn) interacting with a Web Server (2) and a Network System (4). The Web Server (2) contains: - FRONT-END LOGIC (12) which includes user session information (28). - Page Cache (16) containing HTML Pages (18, 20, 22). The Network System (4) contains: - Tag Cache (30) containing tag-content pairs (32, 34, 36, 38, 40, 42, 44). - Tag-Extended HTML File (26). - Object Cache (50) containing content objects (52, 54, 56). - Tag Cache Entry Mapping (46). - Page Formation Mapping (24). The system also includes several databases: - Content DB (58). - Topic DB (60). - Topic/Content DB (62). - User DB (64). Arrows indicate data flow and connections between these components, showing how user requests are processed and content is retrieved from the databases through the caches and mapping logic.

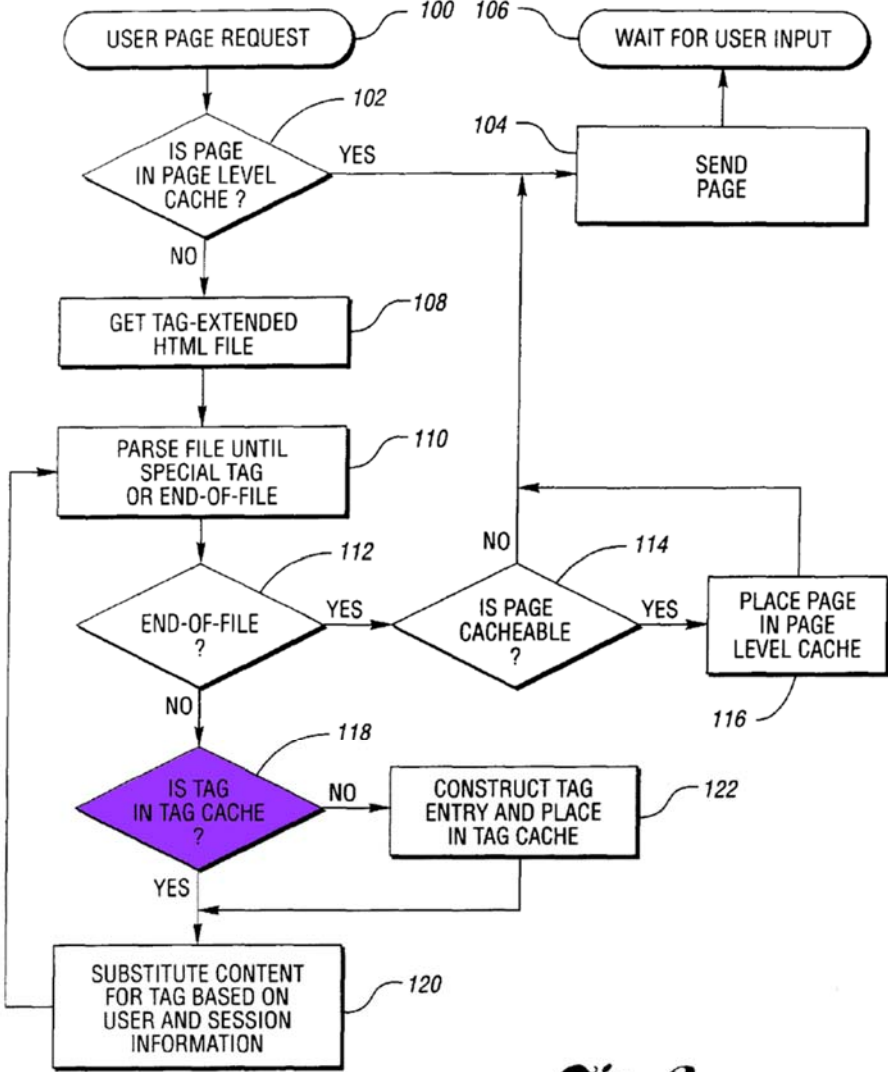
'575 Claim	Lafer in view of Tso
	• Figure 2 <pre> graph TD 100([USER PAGE REQUEST]) --> 102{IS PAGE IN PAGE LEVEL CACHE?} 102 -- YES --> 104[SEND PAGE] 102 -- NO --> 108[GET TAG-EXTENDED HTML FILE] 108 --> 110[PARSE FILE UNTIL SPECIAL TAG OR END-OF-FILE] 110 --> 112{END-OF-FILE?} 112 -- YES --> 114{IS PAGE CACHEABLE?} 112 -- NO --> 118{IS TAG IN TAG CACHE?} 114 -- YES --> 116[PLACE PAGE IN PAGE LEVEL CACHE] 114 -- NO --> 122[CONSTRUCT TAG ENTRY AND PLACE IN TAG CACHE] 116 --> 104 118 -- NO --> 122 118 -- YES --> 120[SUBSTITUTE CONTENT FOR TAG BASED ON USER AND SESSION INFORMATION] 120 --> 110 122 --> 114 104 --> 106([WAIT FOR USER INPUT]) </pre> <i>Fig. 2</i> • 1:15-18 (“[A] key factor in optimizing distribution efficiency is reducing the amount of time required by a server to <i>construct or generate a particular Web page requested by a user.</i>”) • 2:15-33 (see [1.pre])

'575 Claim	Lafer in view of Tso
	• 2:62-64 (“<i>Each user may view the HTML pages using a conventional Internet browser</i> as represented by B1 through Bn.”) • 2:65-3:22 (see [1.b]) • 3:8-10 (“<i>A page formation mapping process, as shown in reference 24, provides construction of the HTML pages in accordance with the present invention.</i>”) • 3:44-55 (“<i>The illustrated steps are not necessarily sequential operations, and may be implemented using appropriate software, hardware, or a combination of both.</i> ... Block 100 of the flow diagram represents <i>a user request for information in the form of an HTML page. As is well known, this can be a Uniform Resource Locator (URL) address for Internet servers.</i>”) • 3:62-4:7 (“<i>If the page is not in the cache, the page must be constructed or generated. The generation process begins by obtaining the tag-extended HTML file as shown in block 108. Block 110 shows that the tag-extended file is then parsed until either a particular tag or the end-of-file is encountered.</i> If the end-of-file is encountered, as is shown in block 112, a check is made to determine if the page should be cached as in block 114. If so, <i>block 116 indicates that the page is placed in the page cache.</i> This process can be implemented using any known algorithm or technique available for managing a page cache. Flow then passes to block 104 where <i>the page is sent to the user</i> as previously described.”) • 4:58-5:2 (see [1.a]).
[1.d] parsing	Lafer discloses parsing said HTML document or browser language on said parser by looking for said proprietary media

'575 Claim	Lafer in view of Tso
said HTML document or browser language on said parser by looking for said proprietary media tags;	tags (<i>e.g.</i>, “the tag extended file is ... parsed” by the parser (<i>see</i> [1.c]) to look for a “specialized tag[]” before the “end-of-file is encountered” (<i>see</i> [1.b])). <u>E.g., Lafer:</u> <i>See</i> [1.b]-[1.c]. Lafer also discloses that “the tag-extended [HTML] file is ... parsed until either a particular tag [such as a “specialized tag” that is not “part of the standard HTML tag library”] or the end-of-file is encountered.” Lafer, 1:54-58, 3:62-67, 4:11-13; Madisetti, ¶¶95-97.

'575 Claim	Lafer in view of Tso
	• Figure 2  <i>Fig. 2</i> • 1:54-58 (“[T]he present invention provides a method and system for <i>specifying fragments of HTML text using specialized tags</i>, holding these fragments in a tag cache, and rapidly rendering personalized HTML pages using the cached fragments.”) • 3:8-10, 3:62-4:5 (see [1.c]) • 4:10-22 (see [1.b])

'575 Claim	Lafer in view of Tso
	• <i>See also</i> 4:8-22.
[1.e] looking up said proprietary media tags in a media tags database on said network system	Lafer discloses looking up said proprietary media tags in a media tags database on said network system (<i>e.g.</i>, “check[]” for the “specialized tag[]” (<i>see</i> [1.b]) in a “tag cache” on the network system (<i>see</i> [1.a])). <u>E.g., Lafer:</u> <i>See</i> [1.a]-[1.b], [1.d]. As discussed for [1.a], Lafer discloses that the network system includes a “tag cache.” <i>See</i> [1.a]. During parsing, if a “specialized tag” is found before the “end-of-file, the tag cache is checked” for a corresponding tag entry. Lafer, 4:32-35, 1:54-58. Each entry maps a tag value to a corresponding “HTML fragment.” Lafer, 3:23-42. The “tag cache” is implemented using a “hash table” or other “software or hardware methods”—not only rendering obvious, but affirmatively disclosing implementing the “tag cache” as a database. <i>Id.</i>; Madisetti, ¶¶98-100.

'575 Claim	Lafer in view of Tso
	• Figure 2  <i>Fig. 2</i> • 1:54-58 (see [1.d]) • 3:23-42 (see [1.pre]) • 4:32-35 (“If a particular tag is encountered prior to the end-of-file, the tag cache is checked as is shown in block 118. If the tag is already associated with contents in the tag

'575 Claim	Lafer in view of Tso
	<i>cache</i>, the contents are substituted for the tag as in block 120.”) • <i>See also</i> 3:37-43, 4:55-61.
[1.f] if at least one media tag of said proprietary media tags is not found, generating a Web media by using said at least one media tag;	Lafer discloses if at least one media tag of said proprietary media tags is not found (<i>e.g.</i>, “[i]f the tag does not exist in the tag cache”), using said at least one media tag (<i>e.g.</i>, “a tag-content pair is constructed from information in the object cache or databases” based on “tag attributes” from the proprietary media tag (<i>see</i> [1.b]) in the tag-extended HTML file indicating “content data from a database” and “user session information”). <u>E.g., Lafer:</u> <i>See</i> [1.b], [1.d]-[1.e]. As discussed above (§VIII.A), [1.f] is a conditional limitation and [1.f]-[1.i] do not need to be met to render obvious claim 1. Madisetti, ¶101. Regardless, Lafer also discloses that when a special tag in the tag-extended HTML file “does not exist in the tag cache,” the tag is used to “construct[]” a “tag-content pair” linking a “tag” to an “HTML fragment.” Lafer, 3:32-39, 4:13-31, 4:46-54. As discussed in §IX.A.1, it was known by a POSITA that an “HTML fragment” is a piece of standard HTML, such as an image tag when an image is being referenced; this fragment references a piece of “content data” stored in the “Content DB.” Lafer, 3:22-30, Fig. 1; Madisetti, ¶102. Each specialized tag contains “tag attributes” including “substitution tokens” for “user session information” and “content data from a database.” Lafer, 4:24-31. This not only renders obvious, but affirmatively discloses that, <i>e.g.</i>, as also discussed in §IX.A.1, a “fragment” for a special tag associated with image content uses a standard “img” HTML tag to link to an image media file. Madisetti, ¶¶102, 104.

'575 Claim	Lafer in view of Tso
	• Figure 2 <pre> graph TD 100([USER PAGE REQUEST]) --> 102{IS PAGE IN PAGE LEVEL CACHE?} 102 -- YES --> 104[SEND PAGE] 104 --> 106([WAIT FOR USER INPUT]) 102 -- NO --> 108[GET TAG-EXTENDED HTML FILE] 108 --> 110[PARSE FILE UNTIL SPECIAL TAG OR END-OF-FILE] 110 --> 112{END-OF-FILE?} 112 -- YES --> 114{IS PAGE CACHEABLE?} 114 -- YES --> 116[PLACE PAGE IN PAGE LEVEL CACHE] 116 --> 104 112 -- NO --> 118{IS TAG IN TAG CACHE?} 118 -- YES --> 120[SUBSTITUTE CONTENT FOR TAG BASED ON USER AND SESSION INFORMATION] 120 --> 104 118 -- NO --> 122[CONSTRUCT TAG ENTRY AND PLACE IN TAG CACHE] 122 --> 120 </pre> <i>Fig. 2</i> • 3:23-42 (see [1.pre]) • 4:24-31 (“<i>The content of a particular tag includes a set of tag attributes. One tag attribute determines the tag type. Other tag attributes have text values that can contain substitution tokens. These tokens are replaced by values depending on the tag type. Possible token replacements include but are not limited to content data from a database,</i>”)”)

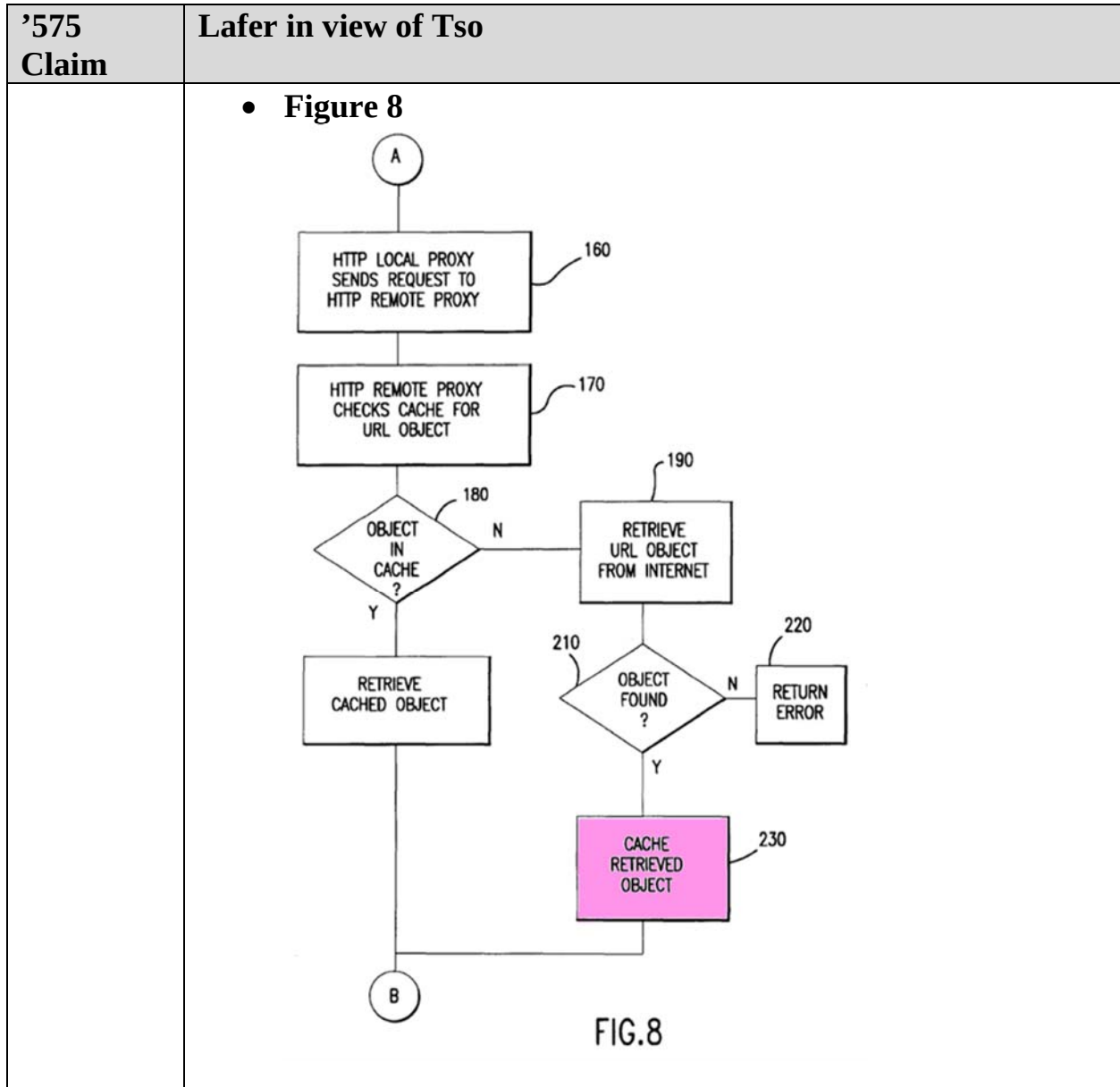
'575 Claim	Lafer in view of Tso
	URLs for predefined forms, <i>user session information</i>, and URL attributes and form fields.”) • 4:46-54 (“<i>If the tag does not exist in the tag cache, a tag-content pair is constructed from information in the object cache or databases as shown in block 122. The use of the object cache shortens database access latency.</i>”) • See also 4:13-31, 5:36-39. Tso discloses if said media is not found (<i>e.g.</i>, if “no entry exists in server-side cache memory 30 for the requested hypertext object” media), generating a Web media (<i>e.g.</i>, “retriev[ing]” the media and “transcod[ing] the data stream appropriately”) using said at least one media tag (<i>e.g.</i>, using reference to “hypertext object[]” embedded in “web page” and “selection criteria” relating to “client,” “network,” and “proxy” characteristics such that the system can “transmit[] the transcoded data stream to network client 12” appropriately). <u>E.g., Tso:</u> Lafer leaves it to a POSITA to select the “content data from a database” and “user session information” included in a tag. Madisetti, ¶105. Tso further discloses that a user issues a request for a webpage with embedded hypertext objects. Tso, 9:60-65, 14:1-17. A POSITA would have understood that hypertext objects are embedded using HTML tags—for example, an “img” tag is used to embed an image, along with certain parameters used to specify how the image is presented in the page. Madisetti, ¶¶103-05; Moore, 3:41-53, 4:5-13, 5:29-30, 6:59-7:13. The tags within the page cause the processing browser to issue separate requests for hypertext objects. Tso, 9:60-65; Madisetti, ¶105. Tso teaches that these requests are processed, along with selection criteria, to identify and transcode content responsive to the request. Tso, 10:24-49, 6:64-7:3, 8:10-13. The selection criteria specify

'575 Claim	Lafer in view of Tso
	“network characteristics,” including “bandwidth,” as well as information about the “network client,” “proxy” server, and “user preferences” including “quality/speed tradeoff.” Tso, 7:15-8:9. The one or more selection criteria “dictate which of transcode providers 24 are invoked” to “transcode[] the data stream appropriately.” <i>Id.</i>; Tso, 10:24-49. However, as discussed in §IX.A.2, Tso’s broad teachings do not require a specific technique for specifying or including such criteria in a request. Madisetti, ¶106. As further discussed in §IX.A.2, a POSITA would have been motivated to modify Lafer’s teaching of “user session information” in a “tag” with Tso’s selection criteria teachings by embedding Tso’s selection criteria as the “session information” substitution tokens within Lafer’s tags within the tag-enhanced HTML page that Lafer uses to generate requests for personalized content. Tso’s selection criteria specifying, <i>e.g.</i>, “network characteristics,” “network client,” and “proxy” characteristics describe types of “user session information” as taught by Lafer. Adding Tso’s transcoding and caching teachings to Lafer by using these substitution tokens to specify selection criteria for transcoding and caching content advantageously implements Lafer’s teachings of using cached data to personalize webpages in a manner that allows personalized transcoding of content for individual users while simultaneously eliminating unnecessary re-transcoding and re-caching of content. <i>E.g.</i>, Lafer, 4:24-31; Tso, 7:15-8:9. Modifying Lafer with Tso’s teachings therefore yields a system in which the substitution token information of Lafer’s tags is used to specify how to transcode and generate (and cache) Web media as taught by Tso—not only rendering obvious, but affirmatively disclosing this element. Lafer, 4:24-31; Madisetti, ¶107. • 6:66-7:3 (“<i>Such selection criterion may comprise</i>, for example, information contained in a header portion of a data packet received by transcoding server 34, such as a MIME type, <i>a URL (Uniform Resource Locator)</i>, a last modified time indicator and so on.”)

'575 Claim	Lafer in view of Tso
	• 7:15-8:9 (“The following discussion provides still more examples of the types of <i>information which may be used to dictate which of transcode service providers 24 are invoked</i>. ... The <i>predetermined selection criterion may comprise</i>: (1) <i>network client 12</i>, such as a display dimension, <i>resolution, number of colors</i>, processor type, memory/disk configuration, modem or network interface type, installed add-in boards (for example, hardware compression/decompression) ... (4) <i>network characteristics, including</i> best-case, worst-case and average latency, <i>bandwidth</i> and/or error rates (for example, for wireless communications) between network client 12 and a proxy, and/or between a proxy and a server (this may be predetermined for guaranteed bandwidth links like ATM (Asynchronous Transfer Mode), or dynamically measured/predicted for so-called "best effort" links like many IP (Internet Protocol) links); (5) <i>proxy characteristics, including system load</i>, available storage, physical location/proximity, and identity (owner); (6) <i>user preferences, including preferred content quality/speed tradeoff</i>”) • 8:10-13 (“<i>Applying</i> the above-listed selection criteria, or <i>combinations thereof</i>, embodiments of the present invention <i>may be used to provide a virtually limitless range of dynamic transcoding services</i>.”) • 9:60-65 (“As is well known in the art, <i>browser 32 may actually transmit a plurality of additional HTTP requests corresponding to each of various distinct hypertext objects that may be embedded in the Web page</i>. In such a case, transcoding <i>server 34 may process each such request in the manner described below</i>.”) • 10:24-49 (“Upon being invoked, <i>parser 22</i> first calls cache interface 28 with the requested hypertext object to <i>determine whether a copy of the required version already resides in</i>

'575 Claim	Lafer in view of Tso
	server-side cache memory 30. For purposes of illustration, assume <i>no entry exists in server-side cache memory 30 for the requested hypertext object</i>. HTTP remote proxy 36 then invokes a call to retrieve the hypertext object from Internet 18 over server/network communications link 16. ... If parser 22 detects a match for a predetermined selection criterion, the HTTP stream handle is given to the appropriate transcode service provider 24. Transcode service provider 24 then transcodes the data stream appropriately, and HTTP remote proxy 26 transmits the transcoded data stream to network client 12.”) • 14:1-7 (“[P]rocessing begins when a user on network client 12 requests a hypertext object from browser 32 (Step 100). This could be in the form of a request for a specific Web page, in which case a plurality of hypertext objects will likely be displayed to the user, or in the form of a click on an image already being displayed to the user.”) • 16:39-44 (“In another particular implementation, information identifying users to be given VIP treatment may be passed to transcoding server 34 in the form of a Web page. Upon receipt of such a Web page, the proxy may subsequently allow servicing threads to perform work for requests generated by VIP users first.”) • See also 3:43-60, 4:1-5, 4:45-50, 4:62-66, 5:36-42, 6:9-50, 6:64-66, 10:24-32, 10:53-11:3, 11:45-49, 13:36-44, 14:1-7, 14:21-63, 14:37-63.
[1.g] placing said generated Web media in a	Lafer discloses placing said generated Web media in a media cache on said network system (e.g., “Content DB” holds content “data” for “content objects”). Lafer teaches that an “object cache holds content objects” and that “[t]he data for these objects are maintained in respective databases

'575 Claim	Lafer in view of Tso
media cache on said network system;	including a “Content DB.” Lafer, 3:37-43. Lafer also teaches that this content “is constructed from information in the ... databases.” Lafer, 4:46-54. As discussed for element [1.a], a POSITA would have understood that the underlying “data” for content stored in the “content DB” is “media,” including media generated as discussed in [1.f]. The generated content is cached in the “content DB.” Lafer, 3:23-42, 4:46-54; Madisetti, ¶¶108-09, 111. • 3:23-42 (see [1.pre]) • 4:46-54 (see [1.f]) To the extent additional disclosure of this element is required, Tso further discloses placing said generated Web media in a media cache on said network system (e.g., “caching of an original version of the retrieved hypertext object” and “transcoded versions of content” in “cache” on the network system (see [1.a])). <i>E.g., Tso:</i> <i>See [1.a], [1.f].</i> Tso also discloses that “[s]erver-side cache memory 30 may be used to store original and transcoded versions of content for later transmission to network client 12.” Tso, 4:1-5. This includes “original version[s] ... retrieved” after initially “determin[ing] that the requested hypertext object [such as displayed images] does not exist in the server-side cache memory 30.” Tso, 14:1-7, 14:37-46. As further discussed for [1.f] and in §IX.A.2, a POSITA would have been motivated to modify Lafer with this teaching to advantageously reduce the processing required to re-transcode data, yielding a system in which both “original” and “transcoded” versions of content are stored in the “Content DB.” Madisetti, ¶¶110-12.



'575 Claim	Lafer in view of Tso
	Figure 9 <pre> graph TD B((B)) --> 240{TRANSCODE OBJECT?} 240 -- Y --> 250[TRANSCODE OBJECT] 240 -- N --> 260[SEND OBJECT TO HTTP LOCAL PROXY] 250 --> 260 260 --> 270[STORE OBJECT IN CLIENT-SIDE CACHE] 270 --> 280{SUPPORTED DATA TYPE?} 280 -- Y --> 290[BROWSER DISPLAYS OBJECT] 280 -- N --> 300[DECODE OBJECT IF NEEDED] 300 --> 310{SUPPORTED DATA TYPE?} 310 -- Y --> 320[BROWSER DISPLAYS OBJECT] 310 -- N --> 330[ADD-IN DISPLAYS OBJECT] </pre> FIG.9 4:1-5 (see [1.pre]) 4:62-5:7 (“Unlike most cache memories, server-side cache interface 28 and server-side cache memory 30 enable maintenance of multiple representations of a given cached object, with descriptive information about each representation included in server-side cache memory 30.”) 14:1-7 (see [1.f])

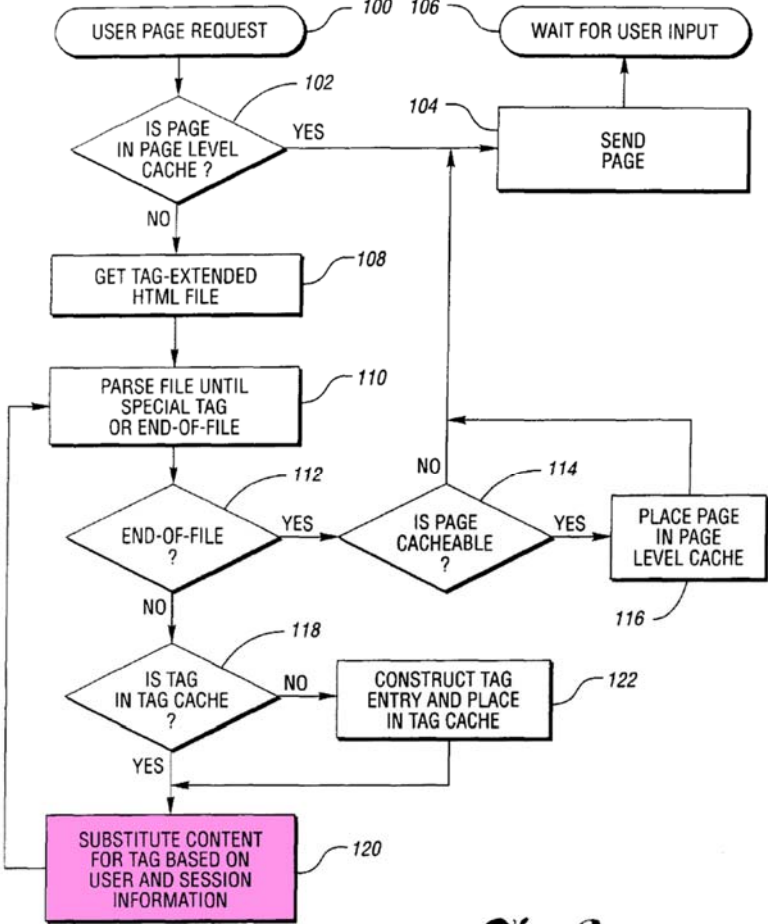
'575 Claim	Lafer in view of Tso
	• <i>See also</i> 4:38-48, 6:9-61, 7:15-65, 14:37-46.
[1.h] converting said at least one media tag to at least one standard HTML equivalent tag that refers to said media in said cache; and	Lafer discloses converting said at least one media tag to at least one standard HTML equivalent tag that refers to said media in said cache (<i>e.g.</i>, “a tag-content pair is constructed from information in the object cache” that “map[s] HTML fragments [including standard HTML tags] into the tag” to allow “HTML data” to be “substituted” for the tag). <u>E.g., Lafer:</u> <i>See</i> [1.d]-[1.e], [1.g]. As discussed for [1.e], Lafer discloses that the “tag,” which is not “part of the standard HTML tag library,” is linked to a standard “HTML fragment” that refers to “content data” in the “Content DB.” Lafer, 4:11-13, 3:32-43, Fig. 1; Madisetti, ¶¶113-15. Thus, when a file is parsed, the “specialized tags” in the file are replaced with, and thus converted into, “content information in the form of HTML fragments” such as standard HTML tags (as discussed in §IX.A.1), by “substitut[ing]” the special tag with the HTML fragment. Lafer, 1:54-58, Abstract. And, at minimum, it would have been obvious to do so (<i>see</i> §IX.A.1). Madisetti, ¶116.

'575 Claim	Lafer in view of Tso
	Figure 2 <pre> graph TD 100([USER PAGE REQUEST]) --> 102{IS PAGE IN PAGE LEVEL CACHE?} 102 -- YES --> 104[SEND PAGE] 104 --> 106([WAIT FOR USER INPUT]) 106 --> 102 102 -- NO --> 108[GET TAG-EXTENDED HTML FILE] 108 --> 110[PARSE FILE UNTIL SPECIAL TAG OR END-OF-FILE] 110 --> 112{END-OF-FILE?} 112 -- YES --> 114{IS PAGE CACHEABLE?} 114 -- YES --> 116[PLACE PAGE IN PAGE LEVEL CACHE] 116 --> 104 114 -- NO --> 106 112 -- NO --> 118{IS TAG IN TAG CACHE?} 118 -- YES --> 120[SUBSTITUTE CONTENT FOR TAG BASED ON USER AND SESSION INFORMATION] 120 --> 104 118 -- NO --> 122[CONSTRUCT TAG ENTRY AND PLACE IN TAG CACHE] 122 --> 114 </pre> <i>Fig. 2</i> Abstract (“... when a request for distribution of the HTML page is received from a user, replacing each tag with content information in the form of HTML fragments previously created based on user preference information.”) 3:23-42 (see [1.pre]) 4:46-54 (see [1.f]) See also 1:49-2:3, 3:22-25, 4:32-35.
[1.i] storing said at	Lafer discloses storing said at least one media tag and said at least one standard HTML equivalent tag in said database (e.g.,

'575 Claim	Lafer in view of Tso
least one media tag and said at least one standard HTML equivalent tag in said database;	“tag-content pair,” linking the “specialized tag” and “HTML fragment,” is stored in “tag cache”). <u>E.g., Lafer:</u> See [1.b], [1.e]-[1.h]. As discussed for [1.h], Lafer also discloses that “[t]he tag cache is composed of a plurality of tag-content pairs” “constructed from information in the object cache” “to map HTML fragments into the tag.” Lafer, 3:23-42, 4:46-54. As discussed for [1.f], the “tag” is not a standard HTML tag, while a “HTML fragment” not only renders obvious, but discloses a standard HTML equivalent tags that reference content in the “Content DB.” Lafer, 4:11-13, 3:23-42. The “tag-content pair” is constructed and stored in the tag cache. Lafer, 4:46-54; Madisetti, ¶¶117-19.

'575 Claim	Lafer in view of Tso
	• Figure 2 <pre> graph TD 100([USER PAGE REQUEST]) --> 102{IS PAGE IN PAGE LEVEL CACHE?} 102 -- YES --> 104[SEND PAGE] 104 --> 106([WAIT FOR USER INPUT]) 106 --> 100 102 -- NO --> 108[GET TAG-EXTENDED HTML FILE] 108 --> 110[PARSE FILE UNTIL SPECIAL TAG OR END-OF-FILE] 110 --> 112{END-OF-FILE?} 112 -- YES --> 114{IS PAGE CACHEABLE?} 114 -- YES --> 116[PLACE PAGE IN PAGE LEVEL CACHE] 116 --> 104 114 -- NO --> 122[CONSTRUCT TAG ENTRY AND PLACE IN TAG CACHE] 112 -- NO --> 118{IS TAG IN TAG CACHE?} 118 -- YES --> 120[SUBSTITUTE CONTENT FOR TAG BASED ON USER AND SESSION INFORMATION] 120 --> 110 118 -- NO --> 122 122 --> 114 </pre> <i>Fig. 2</i> • 3:23-42 (see [1.pre]) • 4:46-54 (see [1.f]) • See also 5:36-39.
[1.j] modifying said HTML document or browser	Lafer discloses modifying said HTML document or browser language by replacing said at least one media tag by said at least one standard HTML equivalent tag (e.g., “substituting ... predetermined HTML fragments for each particular tag embedded in

'575 Claim	Lafer in view of Tso
language by replacing said at least one media tag by said at least one standard HTML equivalent tag;	a requested HTML file based on the stored user preference information”). <u>E.g., Lafer:</u> See '575 [1.d]-[1.e], [1.h]-[1.i]. As discussed for [1.e] and [1.h]-[1.i], Lafer also discloses “substituting one or more predetermined HTML fragments,” (standard HTML equivalent tags (see [1.f])), “for each particular tag” “embedded in a requested HTML file” to “construct[] ... personalized HTML pages.” Madisetti, ¶¶120-22.

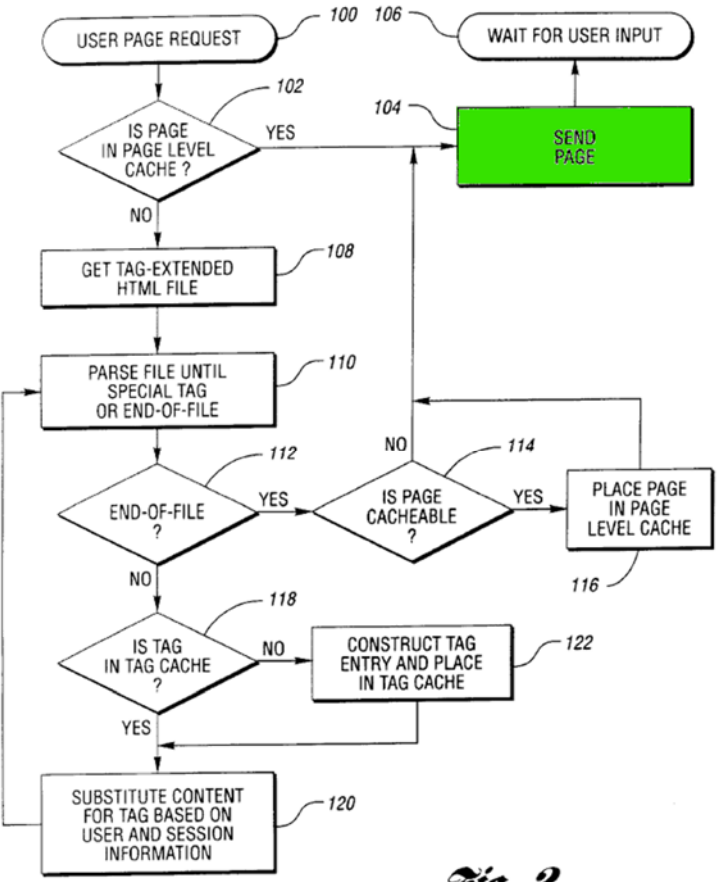
'575 Claim	Lafer in view of Tso
	Figure 2  <i>Fig. 2</i> 2:15-33 (see [1.pre]) 4:8-13 (“<i>If, while parsing the file and before the end-of-file is reached, a particular tag is encountered, the tag is replaced with one or more HTML fragments associated with the tag.</i> More specifically, a tag in accordance with the present invention does not constitute a part of the standard HTML tag library.”) - See also 4:32-45, 4:55-61.

'575 Claim	Lafer in view of Tso
[1.k] delivering said modified HTML document or browser language from said network system to said Web server; and	Lafer discloses delivering said modified HTML document or browser language from said network system to said Web server (<i>e.g.</i>, the network system (<i>see</i> [1.a]) returns the processed “page” to the web server (<i>see</i> [1.b]) after “end-of-file is encountered” and the “page is placed in the page cache”). <u>E.g., Lafer:</u> <i>See</i> [1.a]-[1.b], [1.j]. Lafer also discloses that, once the modified HTML document is generated as discussed for [1.j] as part of the “page formation mapping process” 24, “a check is made to determine if the page should be cached” and “[i]f so, ... the page is placed in the page cache.” Lafer, 3:62-4:5. As discussed for [1.b], the “page cache” is part of the Web server. Madisetti, ¶¶123-25.

'575 Claim	Lafer in view of Tso
	• Figure 1 Figure 1 illustrates a system architecture for a web application. The system includes multiple users (USER1, USERn) interacting with a Web Server (2) and a Network System (4). The Web Server (2) contains: FRONT-END LOGIC (12): Includes user session information (28). Page Cache (16): Stores HTML Pages (18, 20, 22). The Network System (4) contains: Tag Cache (30): Stores tag-content pairs (tag1/content1, tag2/content2, tag3/content3, ..., tagn/contentn). Tag Formation Mapping (24): Maps tags to content. Tag-Extended HTML File (26): Stores HTML files. Object Cache (48): Stores various objects (content object, topic list object, topic to content map object, user information session). Databases (58, 60, 62, 64): Content DB, Topic DB, Topic/Content DB, and User DB. Arrows indicate data flow between components, showing how user requests are processed through the web server and network system to retrieve content from the databases and caches.

'575 Claim	Lafer in view of Tso
	• Figure 2 <pre> graph TD 100([USER PAGE REQUEST]) --> 102{IS PAGE IN PAGE LEVEL CACHE?} 102 -- YES --> 104[SEND PAGE] 104 --> 106([WAIT FOR USER INPUT]) 102 -- NO --> 108[GET TAG-EXTENDED HTML FILE] 108 --> 110[PARSE FILE UNTIL SPECIAL TAG OR END-OF-FILE] 110 --> 112{END-OF-FILE?} 112 -- YES --> 114{IS PAGE CACHEABLE?} 114 -- YES --> 116[PLACE PAGE IN PAGE LEVEL CACHE] 116 --> 104 114 -- NO --> 122[CONSTRUCT TAG ENTRY AND PLACE IN TAG CACHE] 112 -- NO --> 118{IS TAG IN TAG CACHE?} 118 -- NO --> 122 118 -- YES --> 120[SUBSTITUTE CONTENT FOR TAG BASED ON USER AND SESSION INFORMATION] 120 --> 110 122 --> 114 </pre> <i>Fig. 2</i> • 3:62-4:7 (see [1.c]) • See also 2:50-55, 2:65-67, 3:8-10, 4:3-7.
[1.i] delivering said modified HTML document or browser language from said Web	Lafer discloses delivering said modified HTML document or browser language from said Web server to said customer browser for said customer to view (e.g., the Web server (see [1.b]) sends the “page” to a customer’s “browser” (see [1.c]) used to “view the HTML pages”). <u>E.g., Lafer:</u> See [1.b]-[1.c], [1.j]-[1.k].

'575 Claim	Lafer in view of Tso
server to said customer browser for said customer to view.	Lafer also discloses that once the “page is placed in the page cache,” “the server processor ... distribute[s] the constructed HTML files to the user” via “front-end logic [that] is responsible for interfacing with users U1 to Un through the communication medium 2.” Lafer, 3:62-4:5, 2:32-33, 2:65-67. As discussed for [1.b], the “front-end logic” is part of the Web server. The user views the HTML page using a standard “browser.” Lafer, 2:62-64, Fig. 1; Madisetti, ¶¶126-28. Figure 1 The diagram illustrates a web server architecture. At the top, two user systems are shown: USER1 (U1) with browser B1 and USERn (Un) with browser Bn. They are connected via a communication medium (2) to a Web Server (indicated by a green dashed box). Inside the Web Server, the Front-End Logic (12) contains user session information (14) and is connected to a Page Cache (16) which stores HTML Pages (18, 20, 22). The Front-End Logic is also connected to a Tag Cache (14) which stores tag-content pairs (30-44). The Tag Cache is connected to a Page Formation Mapping (24) block, which in turn connects to a Tag-Extended HTML File (26). Below the Tag Cache is a Tag Cache Entry Mapping (46) block. The Object Cache (48) is connected to the Tag Cache Entry Mapping (46) and contains content objects (50), topic list objects (52), topic to content map objects (54), and user information sessions (56). These objects are mapped to four databases: Content DB (58), Topic DB (60), Topic/Content DB (62), and User DB (64).

'575 Claim	Lafer in view of Tso
	• Figure 2  <pre> graph TD 100([USER PAGE REQUEST]) --> 102{IS PAGE IN PAGE LEVEL CACHE?} 102 -- YES --> 104[SEND PAGE] 104 --> 106([WAIT FOR USER INPUT]) 106 --> 100 102 -- NO --> 108[GET TAG-EXTENDED HTML FILE] 108 --> 110[PARSE FILE UNTIL SPECIAL TAG OR END-OF-FILE] 110 --> 112{END-OF-FILE?} 112 -- YES --> 114{IS PAGE CACHEABLE?} 112 -- NO --> 118{IS TAG IN TAG CACHE?} 118 -- YES --> 120[SUBSTITUTE CONTENT FOR TAG BASED ON USER AND SESSION INFORMATION] 120 --> 110 118 -- NO --> 122[CONSTRUCT TAG ENTRY AND PLACE IN TAG CACHE] 122 --> 110 114 -- YES --> 116[PLACE PAGE IN PAGE LEVEL CACHE] 116 --> 102 114 -- NO --> 104 </pre> <i>Fig. 2</i> • 2:15-33 (see [1.pre]) • 2:62-64 (see [1.c]) • 2:65-3:22 (see [1.b]) • 3:67-4:7 (see [1.c])

B. Ground 2: Lafer in View of Tso and Cao (Claim 9)

1. Overview of Cao and Motivation to Modify Lafer (in view of Tso) with Cao

Cao teaches various approaches to address “the problem of cache consistency,” and “updat[ing]” “cached copies... when the originals change.” Cao, 445; Madisetti, ¶129. In approaches that provide “strong consistency” in the cache, the system “sends an ‘if-modified-since’ request” when “a request for a document hits in the cache.” Cao, 445, 447; Madisetti, ¶¶130-31.² In response, the server with the original asset “checks whether the document has been modified since the timestamp.” Cao, 447; Madisetti, ¶131. “If the server replies with ... ‘document unmodified,’” “the cached copy is returned to the user.” Cao, 449, 447; Madisetti, ¶131. Conversely, “[i]f the server replies with the document” that contains “new data,” “the cached copy is deleted.” Cao, 449, 447; Madisetti, ¶131.

Lafer, **Tso**, and **Cao** are all analogous prior art and in the same field as the ’575—networked media caching and delivery—and are reasonably pertinent to the

² The “polling-every-time” approach sends the “if-modified-since” request each time a user requests cached content, while the “invalidation” approach uses the same “if-modified-since” requests to handle cache server failures and to avoid scalability issues. Cao, 448-49, 455; Madisetti, ¶130.

alleged problems identified in the '575 of updating original assets that are stored in cache. '575, 4:57-60 (“original assets need to be updated”), 3:59-60, 13:49-58, 14:9-13; Lafer, 3:37-43 (“objects are maintained in respective databases”); Tso, 7:20-8:9 (“predetermined selection criterion may comprise ... cache restriction of prioritization parameters such as update frequency or replacement preferences”); Cao, 445 (“cache consistency must be maintained, that is, cached copies should be updated when the originals change”), 445-49; Madisetti, ¶132.

While **Lafer** and **Tso** provide broad disclosures of caching media objects, **Cao** provides details about how to maintain cache consistency such that the cached objects remain up-to-date. For example, in addition to **Lafer’s** teaching of “maintain[ing]” a database” and **Tso’s** teaching that the “cache [may] restrict[] ... prioritization parameters such as update frequency or replacement preferences,” **Cao** provides details about maintaining “strong” cache consistency by issuing an if-modified-since command to verify that content has not changed in response to a request for that content. Lafer, 3:37-43; Tso, 7:20-8:9; Cao, 445, 449. For multiple independent reasons, a POSITA would have been motivated to further modify **Lafer** in view of **Tso** with **Cao’s** teachings, yielding a system in which, when a request for content is received, the system verifies that the original assets have not changed

(using an “if-modified-since request”) and, if the original assets have changed, deletes cached information. Cao, 447; Madisetti, ¶133.

First, Cao discloses advantageously applying its teachings to caching systems such as those disclosed in **Lafer** and **Tso** to ensure that only updated content is served during networked content distribution. Cao, 445; Madisetti, ¶134. **Lafer** recognizes and seeks to accommodate the “dramatic growth in the amount of information which can be distributed from a Web server to a user.” Lafer, 1:13-18. **Tso** similarly recognizes that “network characteristics,” such as “latency, bandwidth and/or error rates,” impact how content should be transcoded. Tso, 4:1-5, 7:33-43. **Cao** teaches that the disclosed cache consistency methods advantageously address challenges that arise from the “Web continu[ing] to explode in size” and by better allowing the caching of content in such a network to advantageously improve “both network traffic and document access latency” while ensuring that “no stale copy of the modified document will ever be returned to the user.” Cao, 445; Madisetti, ¶134.

Second, Lafer and **Tso** both recognize the need for cache consistency, but leave it to a POSITA to determine how to maintain cache consistency. **Lafer** broadly provides that the databases in its system must be “maintained.” Lafer, 3:37-43. **Tso** likewise discloses that selection criterion might include “cache restriction or prioritization parameters such as update frequency or replacement preferences,”

without specifying a specific cache consistency protocol to implement using those parameters. Tso, 7:60-65. A POSITA therefore would have been motivated to look to **Cao's** cache consistency protocol to implement **Tso's** and **Lafer's** goal of maintaining the cache in a manner that advantageously avoids providing users with out-of-date content because, otherwise, "the lack of strong consistency reduces the effectiveness of client caches as a way to improve the scalability of the Web." Cao, 445; Madisetti, ¶135.

Third, the disclosed process of modifying content not only renders obvious, but affirmatively discloses that the tag itself is also removed because the content associated with the tag-content pair has changed. Madisetti, ¶136. **Lafer** discloses that each tag stores "attributes" such as "content data" for the associated content, which are used to generate the tag-content pair. Lafer, 3:22-30, 4:8-31. **Cao** discloses that when content is modified, the information about that content, including the "content data" attributes, is no longer valid and therefore is deleted. Cao, 449. Indeed, **Cao** equates deleting the "cached copy" of content with deleting an "entry" for the content, and also teaches deleting "all information" when a document is invalidated. Cao, 449, 456; Madisetti, ¶136. A POSITA therefore would have been motivated to modify **Lafer** with **Cao's** teachings by "deleting" both the content, and the tag-content pair, when content changes, ensuring that tag values in the "tag

cache,” including content data associated the tag-content pairs, are up-to-date. Lafer, 3:22-30; Cao, 449; Madisetti, ¶136; *see also* Ex. 1019 (Sabharwal and Bhatia), 1867-68, 1875.

In light of the above teachings, a POSITA would have had a reasonable expectation of success in applying **Cao’s** teachings of maintaining cache consistency to Lafer’s object and page caches by issuing an if-modified-since request to verify that the cached version of original version of content requested by a user is up-to-date. Indeed, **Lafer** and **Tso** both already teach systems that maintain objects in databases, and **Cao** discloses that the disclosed methods are already implemented in different file systems and web browsers. Lafer, 3:37-43 (“objects are maintained in respective databases”); Tso, 7:60-65 (“update frequency or replacement preferences”); Cao, 445-47 (similar teachings implemented in, *e.g.*, “Netscape browser,” “Sprite” file systems, “callbacks in AFS”); *see also* Ex. 1016 (Barbará), 1:9-3:4 (describing field of “cache memory management strategies” and proposing similar solutions); Ex. 1017, 119-120 (HTTP protocol specification for “If-Modified-Since” request). Madisetti, ¶137. Applying **Cao’s** teachings to implement **Lafer** would have been a straightforward application of those teachings in a manner that advantageously improves the efficacy of the system. Madisetti, ¶137.

2. Claim Chart

'575 Claim	Lafer in View of Tso and Cao
[9.pre] A process for creating a media associated with an HTML equivalent tag and associated with a media tag for a system, comprising the steps of:	Lafer discloses a process for creating a media associated with an HTML equivalent tag (<i>e.g.</i>, a process for creating “content data” associated with an “HTML fragment” including standard HTML tags (<i>see</i> [1.h])) and associated with a media tag (<i>e.g.</i>, associated, through a “tag-content pair,” with a media tag (<i>see</i> [1.b])) for a system (<i>e.g.</i>, for either a “user” computer or a system including “page formation mapping” components). <u>E.g., Lafer:</u> <i>See</i> '575 [1.pre]-[1.b], [1.e]-[1.j]. Lafer also discloses that “content data,” which is media data, is stored in a “Content DB.” Lafer, 3:37-43, Fig. 1, Madisetti, ¶¶138-39, 141. As discussed for [1.f], the system generates a “tag-content pair” that associates the “specialized tag” with the “HTML fragment,” and these are associated with the “content data” because the “HTML fragment” refers to the “content data.” Lafer, 3:37-43. And as discussed in §IX.A.1, not only renders obvious, but discloses an HTML fragment including an HTML equivalent tag. Madisetti, ¶141. Lafer discloses that this process is performed for at least two different systems that meet the claimed “system.” First, media is created for “user” computers used to view HTML pages, which include media, on a browser. Lafer, 2:62-64, Fig. 1. These user computers are a claimed “system” (“User System”). Madisetti, ¶142. Second, Lafer also discloses that this process is performed for a “system” comprising components used to form a “Page Formation Mapping” process 24 as shown in Fig. 1. Lafer, Fig. 1, 3:8-14. As discussed below for [9.b] and [9.d.2], this system comprises at least the “front-end logic” and “page cache.” During page formation, the system “parses” a “tag-extended” HTML file for “specialized” tags and requests “HTML fragments,” which are

'575 Claim	Lafer in View of Tso and Cao
	HTML equivalent tags that are created for the system. Lafer, 4:8-13. This “system” comprises “hardware” and “software” used to form one or more “logical blocks” in Figure 1—e.g., one or more of the “front-end logic,” “page cache,” and “storage medium” for “tag-extended HTML files.” Lafer, 3:8-21, 4:58-5:2. Lafer teaches “combin[ing]” or “further subdivid[ing]” the components shown in Fig. 1 used to implement the features of the “page formation mapping” function implemented by the claimed system,” such that it teaches the claimed “system” that includes front-end logic (“Front-End System”). <i>See id.</i>; Madisetti, ¶143. • Figure 1 The diagram illustrates a system architecture. At the top, multiple user devices are shown, each consisting of a browser (B1, Bn) and a user session (U1, Un). These devices are connected to a central system. The system is divided into several functional blocks: Front-End Logic (12) which handles user session information (14); a Page Cache (16) storing HTML pages (18, 20, 22); a Tag Cache (30) containing tag-content pairs (32, 34, 36, 38, 40, 42, 44); Page Formation Mapping (24) which maps tags to content; and a Tag-Extended HTML File (26). The system also includes an Object Cache (48) which stores various objects (50, 52, 54, 56) and a Tag Cache Entry Mapping (46). At the bottom, there are four databases: Content DB (58), Topic DB (60), Topic/Content DB (62), and User DB (64). Arrows indicate the flow of data and interactions between these components.

'575 Claim	Lafer in View of Tso and Cao
	• Figure 2 <pre> graph TD 100([USER PAGE REQUEST]) --> 102{IS PAGE IN PAGE LEVEL CACHE?} 102 -- YES --> 104[SEND PAGE] 104 --> 106([WAIT FOR USER INPUT]) 106 --> 102 102 -- NO --> 108[GET TAG-EXTENDED HTML FILE] 108 --> 110[PARSE FILE UNTIL SPECIAL TAG OR END-OF-FILE] 110 --> 112{END-OF-FILE?} 112 -- YES --> 114{IS PAGE CACHEABLE?} 114 -- YES --> 116[PLACE PAGE IN PAGE LEVEL CACHE] 116 --> 104 114 -- NO --> 122[CONSTRUCT TAG ENTRY AND PLACE IN TAG CACHE] 112 -- NO --> 118{IS TAG IN TAG CACHE?} 118 -- YES --> 120[SUBSTITUTE CONTENT FOR TAG BASED ON USER AND SESSION INFORMATION] 120 --> 110 118 -- NO --> 122 122 --> 120 120 --> 110 </pre> <i>Fig. 2</i> • 2:62-64 (see [1.c]) • 2:65-3:22 (see [1.b]) • 3:23-42 (see [1.pre]) • 4:8-13 (see [1.j]) • 4:46-54 (see [1.f]) • 4:58-5:2 (see [1.a])

'575 Claim	Lafer in View of Tso and Cao
	• See also Abstract, 3:22-30, 3:65-67. To the extent additional disclosure is required, Tso discloses a process for creating a media (<i>e.g.</i>, “retriev[ing] the hypertext object” and “transcod[ing] the data stream appropriately,” <i>see</i> [1.f]) for a system (<i>e.g.</i>, for the content to be “transmit[ted] the transcoded data stream to network client 12” (<i>see</i> [1.f])). <i>See</i> [1.f]. In addition to Lafer’s teaching of constructing a “tag-content pair” that is associated with a standard HTML fragment for content when the tag does not exist in the tag cache, Tso further discloses a process in which, when content requested by a user is not cached, an original version of the content is retrieved, and then transcoded, to generate content that is cached and then sent to the user. Tso, 6:28-44, 4:1-5, 10:24-49. As further discussed in §IX.A.2, a POSITA would have been motivated to modify Lafer with this teaching, yielding a system in which, when a specialized tag does not yet exist, the system creates the media content, associates it with the HTML fragment used to reference the content, and associates the content with a specialized “tag.” Lafer, 1:54-58, 4:46-54; Tso, 6:28-44, 4:1-5, 10:24-49; Madisetti, ¶¶14044.
[9.a] receiving a request from said system for said HTML equivalent tag;	Lafer discloses receiving a request from said system for said HTML equivalent tag (<i>e.g.</i>, “request” from the system (<i>see</i> [9.pre]) for a “page” causes “tag cache” to be “checked” for associated “HTML fragments” (<i>see</i> [1.c]), which include HTML equivalent tags). <u>E.g., Lafer:</u> <i>See</i> ’575 [1.c], [1.h]-[1.l]. As discussed for [1.c], Lafer discloses that a user issues a request for HTML content. Lafer, 1:15-18. For the <u>User System</u>, the request for HTML is a request for a page including the HTML

'575 Claim	Lafer in View of Tso and Cao
	equivalent tag, and is serviced by providing the user with an HTML page that includes that tag (<i>see</i> [1.j]-[1.l]). Lafer, 3:62-67. Madisetti, ¶¶145-47. For the <u>Front-End System</u>, Lafer discloses that “the tag-extended file is ... parsed until either a particular tag or the end-of-file is encountered.” Lafer, 3:62-67. “If a particular tag is encountered prior to the end-of-file, the tag cache is checked” for “HTML fragments that are substituted for tag values.” Lafer, 4:32-35, 3:22-27. This “check” is a request from the <u>Front-End System</u> for an HTML equivalent tag (<i>see</i> [1.h]-[1.j]). Madisetti, ¶148.

'575 Claim	Lafer in View of Tso and Cao
	• Figure 1 Figure 1 illustrates a system architecture for content delivery. At the top, two users, USER1 and USERn, are shown interacting with the system via their respective browsers (B1, Bn) and computers (U1, Un). The system (2) is a central processing unit that manages content delivery. It includes a FRONT-END LOGIC (12) which handles user session information (14). A Page Cache (16) stores HTML pages (18, 20, 22). A Tag Cache (28) stores tags (30, 32, 34, 36, 38, 40, 42, 44) and their corresponding content (32, 36, 40, 44). A Page Formation Mapping (24) maps tags to content, resulting in a Tag-Extended HTML File (26). An Object Cache (48) stores various objects (50, 52, 54, 56) used in the system. These objects are mapped to a Tag Cache Entry Mapping (46). The system also includes several databases: Content DB (58), Topic DB (60), Topic/Content DB (62), and User DB (64). Arrows indicate the flow of data and interactions between these components.

'575 Claim	Lafer in View of Tso and Cao
	• Figure 2 <pre> graph TD 100([USER PAGE REQUEST]) --> 102{IS PAGE IN PAGE LEVEL CACHE?} 102 -- YES --> 104[SEND PAGE] 104 --> 106([WAIT FOR USER INPUT]) 106 --> 102 102 -- NO --> 108[GET TAG-EXTENDED HTML FILE] 108 --> 110[PARSE FILE UNTIL SPECIAL TAG OR END-OF-FILE] 110 --> 112{END-OF-FILE?} 112 -- YES --> 114{IS PAGE CACHEABLE?} 114 -- YES --> 116[PLACE PAGE IN PAGE LEVEL CACHE] 116 --> 104 114 -- NO --> 122[CONSTRUCT TAG ENTRY AND PLACE IN TAG CACHE] 112 -- NO --> 118{IS TAG IN TAG CACHE?} 118 -- YES --> 120[SUBSTITUTE CONTENT FOR TAG BASED ON USER AND SESSION INFORMATION] 118 -- NO --> 122 120 --> 104 122 --> 104 </pre> <i>Fig. 2</i> • 2:15-33 (see [1.pre]) • 3:44-55, 3:62-4:7 (see [1.c]) • 4:32-35 (see [1.e]) • <i>See also</i> 3:2-3, 3:22-27, 3:32-36, 4:8-10.
[9.b] combining said media tag with bandwidth information;	Lafer discloses combining said media tag with information (e.g., incorporating “session information” during “dynamic substitution of content for tag values” using a “substitution token” in the “tag”).

'575 Claim	Lafer in View of Tso and Cao
	<u>E.g., Lafer:</u> As discussed for [1.f], Lafer discloses that the “specialized tags” include “tag attributes.” Lafer, 4:24-31; <i>see also</i> §IX.A.3([1.f]). These attributes include “text values that can contain substitution tokens” that are “replaced by values” including “user session information.” <i>Id.</i> This “user session information” is collected by the “front-end logic” responsible for “servic[ing] the user.” Lafer, 2:65-3:5. Replacing the “substitution token” with “user session information” discloses combining the media tag with user session information. Madisetti, ¶¶149-50, 152. • 2:65-3:22 (<i>see</i> [1.b]) • 3:15-21 (<i>see</i> [1.a]) • 4:24-31 (<i>see</i> [1.f]) • <i>See also</i> 3:37-50. Tso further discloses bandwidth information (<i>e.g.</i>, “predetermined selection criterion” specifying “network client,” “network” and “proxy” characteristics including “bandwidth,” and “user preferences” for “content quality/speed tradeoff”). Lafer leaves it to a POSITA to select the “user session information” to be included. Tso further discloses that when a request for content is made, “selection criteria” transmitted as part of the request are used to determine how to transcode content in response to the request—but does not require a specific way of transmitting the selection criteria. Tso, 6:64-7:3. Those selection criteria specify features that a POSITA would have recognized describe information about a user session communicating with a server, including “network characteristics,” including “bandwidth,” as well as information about the “network client,” “proxy” server, and “user preferences” including “quality/speed tradeoff.” Tso, 7:15-8:9. As further discussed in §IX.A.2, a

'575 Claim	Lafer in View of Tso and Cao
	POSITA would have been motivated to modify Lafer's teaching of combining "user session information" in a "tag" with Tso's selection criteria teachings describing types of session information, by using the "session information" substitution token in Lafer's tag to carry these selection criteria from Tso. Lafer, 4:24-31; Tso, 7:15-8:9. Modifying Lafer with Tso's teachings yields a system in which Lafer's "substitution tokens" within a tag's "attributes" include bandwidth information about the user session, disclosing this element. Lafer, 4:24-31; Madisetti, ¶¶151-53. <u>E.g., Tso:</u> • 7:15-8:9, 8:10-13 (see [1.f]) • See also 3:8-17, 6:64-7:3, 10:37-49, 17:9-15.
[9.c] determining if a media tag entry associated with said media tag exists in a media tag database;	Lafer discloses determining if a media tag entry associated with said media exists in a media tag database (e.g., "check[ing]" a "tag cache" implemented in a "hash table" for a "particular tag," associated with "content data," see [1.e]-[1.f]). <u>E.g., Lafer:</u> See '575 [1.e]-[1.f]. As discussed above in §IX.A.3([1.e]-[1.f]), Lafer discloses that "[i]f a particular tag is encountered prior to the end-of-file, the tag cache is checked" to determine "[i]f the tag is already associated with contents in the tag cache." Lafer, 4:32-35. Such "tags" are associated with "content" stored in a "Content DB" because the "HTML fragment" associated with the tag identifies content data in the "Content DB." Lafer, 3:22-30, Fig. 1; see [1.f]; Madisetti, ¶¶154-56. • 3:23-42, 4:55-61 (see [1.pre]) • 4:32-35 (see [1.e])

'575 Claim	Lafer in View of Tso and Cao
[9.d] if said media tag entry exists in said media database, determining if at least one original asset used to create said media has changed;	Lafer discloses [determining] if said media tag entry exists in said media database (<i>e.g.</i>, “determining” that a “tag is already associated with contents in the tag cache,” <i>see</i> [1.e]-[1.f]). <u>E.g., Lafer:</u> <i>See</i> ’575 [1.e]-[1.f], [9.c]. Claim 9 recites process steps for how to handle three cases as part of the process: (1) when a “media tag entry exists” and the “original asset” “<u>has not</u> changed” (<i>see</i> [9.d]-[9.d.2]); (2) when a “media tag entry exists” and the “original asset” “<u>has</u> changed” (<i>see</i> [9.e]-[9.e.5]); and (3) when the “media tag entry does not exist” (<i>see</i> [9.f]-[9.f.4]). Each case ends with “return[ing] said HTML equivalent tag to said system.” <i>See</i> [9.d.2], [9.e.5], [9.f.4]. Although all three are met by the art as discussed below, as discussed in §VIII.A, only one of these three cases must be met to render obvious the claim. Madisetti, ¶157. For this element, as discussed above in §§IX.A.3([1.e]-[1.f]), IX.B.2([9.c]), Lafer discloses determining if the “tag” is associated with a content HTML fragment in the tag cache. Lafer, 4:32-35. If the tag already exists, then the cached entry is used for substituting content with the HTML fragment in response to the request. Lafer, 3:23-42, 4:32-45; Madisetti, ¶158-59, 161. • 3:23-42 (<i>see</i> [1.pre]) • 4:32-35 (<i>see</i> [1.e]) • <i>See also</i> 5:36-39. Cao discloses if said media entry exists in said media database (<i>e.g.</i>, if “there is a cached copy”), determining if at least one original asset used to create said media has changed (<i>e.g.</i>, “check[ing] whether the document has been modified”). <u>E.g., Cao:</u>

'575 Claim	Lafer in View of Tso and Cao
	As discussed above in §IX.A.3([1.f]), Lafer teaches determining whether a tag appears in cache; if it does, then the content associated with the tag is retrieved. Lafer, 4:32-35, 3:23-42. Lafer further teaches that content that is no longer stored in the cache must be “regenerat[ed] and restor[ed]”—disclosing that content in a cache changes and must be updated—but leaves it to a POSITA to determine how to confirm that content being served is up-to-date. Lafer, 5:36-39. Cao provides additional and complementary teachings of how to perform this confirmation. When a request for content “hits in the cache,” Cao teaches using “an ‘if-modified-since’ request” to confirm that the content item being requested is up to date. Cao, 445, 447. Madisetti, ¶¶160-62. As further discussed in §IX.B.1, a POSITA would have been motivated to modify Lafer with this teaching to, <i>e.g.</i>, ensure that clients are not provided with out-of-date content, yielding a system in which, when a request is made using a “media tag,” the system checks whether the associated underlying original asset used to generate the media associated with the media tag (<i>see</i> [9.pre]) has changed. Madisetti, ¶163. • 445 (“In the polling-every-time approach, <i>every time the user requests a document and there is a cached copy, the cache first contacts the Web server to validate the cached copy</i>, then returns the copy to the user. In this approach, a write is complete when the modification is registered in the server’s file system.”) • 447 (“In addition, each client can send an ‘if-modified-since’ request, containing the URL of the document and a timestamp, to the Web server. <i>Upon receiving the request, the server checks whether the document has been modified since the timestamp. If so, the server returns the status code ‘200’ and the new data; otherwise, the server returns the code ‘304,’ which stands for ‘document unmodified.’</i>”)

'575 Claim	Lafer in View of Tso and Cao
	• 448 (“Polling-every-time guarantees that, <i>when the user requests a document, the returned document is up-to-date at the time when the server processes the ‘if-modified-since’ message</i>. If a server or network failure happens when the user requests a document, polling-every-time can inform the user of the failure.”) • <i>See also 447, 449.</i>
[9.d.1] if said at least one original asset used to create said media has not changed, retrieving said HTML equivalent tag from said database; and	Lafer discloses that if said at least one original asset used to create said media has not changed (<i>e.g.</i>, “the tag is already associated with contents in the tag cache”), retrieving said HTML equivalent tag from said database (<i>e.g.</i>, “the contents are substituted for the tag”). <u>E.g., Lafer:</u> As discussed for [1.h] and [1.j], cached entries of content allow “HTML data” to be substituted for a special tag. Lafer discloses that “[i]f the tag is already associated with contents in the tag cache, the contents are substituted for the tag as in block 120.” Lafer, 4:32-35 (<i>see</i> [1.e], [1.h], [1.j]); Madisetti, ¶¶164-65, 167. To the extent further disclosure of this element is required, Cao further discloses that if said at least one original asset used to create said media has not changed (<i>e.g.</i>, “the server returns the code ‘304,’ which stands for ‘document unmodified’”), retrieving said cached copy from said database (<i>e.g.</i>, “the cached copy is returned to the user”). <u>E.g., Cao:</u> As discussed for [9.d], Lafer discloses that when an entry exists in cache for a tag (<i>i.e.</i>, there is a cache hit for the tag), the tag is substituted for the HTML fragment in the file. Cao discloses that, when a cache hit occurs, the server confirms that the content is unmodified by returning the code ‘304,’ which stands for ‘document unmodified’” in response to “an ‘if-modified-since’ request.” Cao, 447. “If the server replies with status code 304, the

'575 Claim	Lafer in View of Tso and Cao
	cached copy is returned to the user.” Cao, 449; Madisetti, ¶¶166-68. As further discussed for [9.d] and in §IX.B.1, a POSITA would have been motivated to modify Lafer with Cao’s teachings to avoid serving out-of-date content by verifying that the content is unmodified, yielding a system in which, when the system verifies that the original asset has not changed, the cached version of the tag is used to retrieve the associated HTML fragment, which not only renders obvious, but affirmatively discloses an HTML equivalent tag, and place this equivalent tag in the web page. <i>Id.</i>; Madisetti, ¶169. • 447 (see [9.d]) • 449 (“For polling-every-time, we simply <i>send an ‘if-modified-since’ request to the server every time a cache hit happens. If the server replies with status code 304, the cached copy is returned to the user.</i> If the server replies with the document, the cached copy is deleted, the new copy is put in the cache and returned to the user.”)
[9.d.2] return said HTML equivalent tag to said system;	Lafer discloses returning said HTML equivalent tag to said system (e.g., returning from “[a] tag cache ... HTML fragments that are substituted for tag values,” see [1.k]-[1.l]). <u>E.g., Lafer:</u> See ’575 [1.h], [1.k]-[1.l]. Lafer also discloses that “[i]f a particular tag is encountered prior to the end-of-file, the tag cache is checked.” Lafer, 4:32-35. “If the tag is already associated with contents in the tag cache, the contents are substituted for the tag.” <i>Id.</i>; Madisetti, ¶¶170-72. With respect to the <u>Front-End System</u>, as discussed for [1.k]-[1.l], the modified HTML document, including the HTML equivalent tag, is cached in the “page cache” and sent to the “front-end

'575 Claim	Lafer in View of Tso and Cao
	logic” for returning to the user computer—disclosing that the HTML equivalent tag is returned to the system. Lafer, 3:22-31, 4:32-35; <i>see</i> §IX.A.3([1.k]-[1.l]); Madisetti, ¶173. Similarly, for the <u>User System</u>, the HTML equivalent tag is embedded in the HTML page that is returned to the user computer after it is “constructed” as discussed in [1.l]. Lafer, 2:15-33; <i>see</i> §IX.A.3([1.l]); Madisetti, ¶174. • 2:15-33, 3:22-42 (<i>see</i> [1.pre]) • 4:32-35 (<i>see</i> [1.e])
[9.e] if said at least one original asset used to create said media has changed, removing said media tag entry from said database;	Lafer and alternatively Lafer in view of Tso not only renders obvious, but affirmatively discloses removing said media tag entry from said database (<i>e.g.</i>, removing a “tag” such that it must be “regenerat[ed] and restor[ed]”). <i>See</i> [1.pre], [1.f]. Lafer discloses that tags must be “regenerat[ed]” and “restor[ed]”—disclosing that previously-existing tags have been removed from cache. Lafer, 5:36-39 (<i>see</i> [1.f]). And Tso further discloses content “data” that is media such as image data (<i>see</i> [1.pre]). Madisetti, ¶¶175-76, 178. Cao discloses if said at least one original asset used to create said media has changed (<i>e.g.</i>, if “the document has been modified since the timestamp”), removing said entry from said database (<i>e.g.</i>, “the cached copy is deleted”). <u>E.g., Cao:</u> Cao discloses that if “the document has been modified since the timestamp,” “the server returns the ... new data.” Cao, 447. Cao refers to documents in the cache as managed using cache “entries.” <i>See</i> Cao, 449, 455. “If the server replies with the document” that contains the new data, “the cached copy is deleted.” Cao, 449. Cao further teaches that when the server

'575 Claim	Lafer in View of Tso and Cao
	responds to an update request with an updated version of content, the cached copy is removed by deleting the “entry” in cache—disclosing that both the content itself and a cached copy are deleted when out of date. Cao, 449; Madisetti, ¶¶177-79. As further discussed in §IX.B.1, a POSITA would have been motivated to modify Lafer with Cao’s teachings of verifying that cache entries are up-to-date to ensure that clients are served with current versions of data, yielding a system in which the system “regenerat[es] and restor[es]” out-of-date data as taught by Lafer by deleting all cached information, including the media tag, and replacing it with new data when the cached copy is out-of-date, rendering obvious this element. Cao, 447, 449; Madisetti, ¶180. • 447 (<i>see</i> [9.d]) • 449 (“Every time a cache hit happens, the cache entry’s time-to-live is checked. If it expires, <i>the original implementation deletes the entry</i> and sends a regular GET request for the document. Our implementation keeps the cached file and sends an “if-modified-since” request instead. ... For polling-every-time, we simply <i>send an ‘if-modified-since’ request to the server every time a cache hit happens</i>. If the server replies with status code 304, the cached copy is returned to the user. <i>If the server replies with the document, the cached copy is deleted, the new copy is put in the cache and returned to the user.</i>”)
[9.e.1] create said media using said media tag;	Lafer in view of Tso renders obvious creating said media using said media tag (<i>see</i> §IX.A.3([1.f])). Madisetti, ¶¶181-83.
[9.e.2] storing said	Lafer and, alternatively, Lafer in view of Tso discloses storing said media in a media cache (<i>see</i> §IX.A.3([1.g])).

'575 Claim	Lafer in View of Tso and Cao
media in a media cache;	Madisetti, ¶¶184-86.
[9.e.3] generating an HTML document referring to said generated media;	Lafer discloses generating an HTML document referring to said media (<i>see</i> [1.j]). Tso discloses said generated media (<i>see</i> [1.j]). <u>E.g., Lafer:</u> As disclosed for [1.j] (§IX.A.3), Lafer discloses modifying an HTML document by replacing at least one media tag with at least one standard HTML equivalent tag. Madisetti, ¶¶187-88. Modifying Lafer with Tso's teachings as discussed for [1.f]-[1.g], [9.e.1]-[9.e.2], those HTML equivalent tags refer to said generated media in order to “personalize[] HTML pages.” Lafer, 1:54-58; §IX.B.1 (motivation to combine); Madisetti, ¶¶189-90. • 1:54-58 (<i>see</i> [1.d]) • See also 1:59-2:1, 4:55-61.
[9.e.4] placing said media tag and said HTML equivalent tag in said media database; and	Lafer discloses placing said media tag and said HTML equivalent tag in said media database (<i>see</i> [1.i]). <u>E.g., Lafer:</u> Lafer discloses this element for the reasons explained for [1.i] (<i>see</i> §IX.A.3). Madisetti, ¶¶191-93.
[9.e.5] return said HTML equivalent tag to said system;	Lafer discloses returning said HTML equivalent tag to said system. <i>See</i> [9.d.2]. Madisetti, ¶¶194-95.

'575 Claim	Lafer in View of Tso and Cao
[9.f] if said media tag entry does not exist in said media database, create said media using said media tag;	Lafer in view of Tso renders obvious if said media tag entry does not exist in said media database, create said media (see [1.f], [9.e.1]). Lafer modified with Tso's teachings renders obvious this element for the reasons discussed in [1.f] and [9.e.1]. Madisetti, ¶¶196-98.
[9.f.1] storing said media in a media cache;	Tso discloses storing said media in a media cache. See [9.e.2]. Madisetti, ¶¶199-201.
[9.f.2] generating HTML or browser language referring to said generated media;	Lafer discloses generating HTML or browser language referring to said media. Tso discloses said generated media. See [9.e.3]. Lafer in view of Tso renders obvious generating “HTML or browser language” for the same reasons it renders obvious generating “an HTML document.” Madisetti, ¶¶202-05.
[9.f.3] placing said media tag and said HTML equivalent tag in said media database; and	Lafer discloses placing said media tag and said HTML equivalent tag in said media database. See [9.e.4]. Madisetti, ¶¶206-08.

'575 Claim	Lafer in View of Tso and Cao
[9.f.4] return said HTML equivalent tag to said system.	Lafer discloses returning said HTML equivalent tag to said system. See [9.e.5], [9.d.2]. Madisetti, ¶¶209-11.

C. Grounds 3-4: Lafer in view of Tso, [Cao for Claim 9,] and Moore

Lafer in view of Tso renders obvious claim 1 and in further view of Cao renders obvious claim 9, including Lafer's teachings of HTML fragments that not only render obvious, but affirmatively disclose elements reciting "HTML equivalent tag[s]." See §§IX.A.1-IX.A.2. Nonetheless, to the extent required for the recitation of an "HTML equivalent tag" in [1.h]-[1.j], or [9.pre]-[9.a], [9.d.1]-[9.d.2], [9.e.3]-[9.e.5] or [9.f.2]-[9.f.4] ("HTML Limitations"), Moore (Ex. 1007) provides explicit teachings of an HTML fragment that is a standard HTML "IMG" tag; Madisetti, ¶212.

Moore teaches that webpages are generated using HTML source code for rendering, *e.g.*, "graphics" and "pictures." Moore, 3:40-53. For example, including an "IMG command," which takes the form of an "HTML tag," causes "an image to display" on a page. Moore, 4:5-35; 5:29-30. Moore also uses the term "fragment" consistent with its understood meaning to refer to a set of lines of code, and therefore not only renders obvious, but discloses a "fragment" that is a "tag." Thus, Moore

discloses a standard “HTML equivalent tag” (*e.g.*, “IMG command” “HTML tag”) as recited in the HTML Limitations. Madisetti, ¶213.

Moore is analogous prior art and in the same field as the ’575 and **Lafer** (and **Cao** and **Tso**)—networked media caching and delivery—because it is directed to “methods and apparatus for enhancing the performance of displaying images on Internet browsers.” Moore, 1:11-14. While **Lafer** provides broad teachings of using “HTML fragments” with underlying data maintained in a database, **Lafer** leaves it to a POSITA to implement the mapping between such fragments and the underlying data. Lafer, 3:32-43. **Moore** teaches the specific standard HTML tag—an “IMG” tag—used to cause an image to display in a page. Moore, 4:5-35, 5:26-29. **Moore** uses the term “fragment” (consistent with the meaning to a POSITA) to refer to an “HTML code sequence”—a POSITA further would have understood an image tag to be an example of such “sequences.” Moore, 4:5-35, 6:59-7:12; Madisetti, ¶214. A POSITA therefore would have been motivated to modify **Lafer** (in view of **Tso** (and **Cao**)) with **Moore’s** teachings of including image “content” in a page because **Moore** teaches a well-known, standard HTML tag used to include such images in a page. *Id.* And because such well-known tags were standard, a POSITA would have had a reasonable expectation of success using them in implementing **Lafer’s**

teachings (in view of **Tso** and **Cao**)—rendering obvious the elements reciting a “[standard] HTML equivalent tag.” Madisetti, ¶214.

X. SECONDARY CONSIDERATIONS

There is no evidence in the ’575’s prosecution history or elsewhere supporting any secondary considerations arguments, or evidence of nexus to any challenged Claim. *See generally* ’575 FH; Madisetti, ¶215. Indeed, as demonstrated by the prior art referenced herein, any purported solutions to problems or unexpected results in the ’575 were already well known. Madisetti, ¶215. To the extent PO asserts the existence of any secondary considerations in its responses, Petitioner reserves the right to address any such evidence.

XI. CONCLUSION

Substantial, new, and noncumulative technical teachings have been presented for ’575 Claims 1 and 9, which are obvious for the reasons set forth above. Madisetti, ¶¶216-21. There is a reasonable likelihood that Petitioner will prevail as to the Claims. *Inter partes* review of claims 1 and 9 is accordingly requested.

Respectfully submitted,

Dated: January 10, 2023

By: /James. L. Davis, Jr./
Name: James. L. Davis, Jr.
Counsel for Petitioner

CERTIFICATE OF COMPLIANCE

Pursuant to §42.24(d), the undersigned certifies that the foregoing Petition for *Inter Partes* Review of U.S. Patent No. 6,792,575 contains, as measured by the word-processing system used to prepare this paper, 13,988 words. This word count does not include the items excluded by §42.24 as not counting towards the word limit.

Respectfully submitted,

Dated: January 10, 2023

By: /Daniel W. Richards/
Name: Daniel W. Richards
Counsel for Petitioner

CERTIFICATE OF SERVICE

I hereby certify that on January 10, 2023, I caused a true and correct copy of the foregoing Petition for *Inter Partes* Review of U.S. Patent No. 6,792,575 and supporting exhibits to be served via express mail on the Patent Owner at the following correspondence address of record as listed on PAIR:

22862 - GLENN PATENT GROUP
c/o Perkins Coie LLP
P.O. Box 1247
Seattle, WA 98111-1247
UNITED STATES

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

Jason R. Bartlett (jbartlett@mkwllp.com)
Jason A. Crotty (jcrotty@mkwllp.com)
Marc J. Pernick (mpernick@mkwllp.com)
MAURIEL KAPOUYTIAN WOODS LLP
450 Sansome Street, Suite 1005
San Francisco, CA 94111
Tel: (415) 738-6228

Steven Callahan (scallahan@ccrglaw.com)
Christopher T. Bovenkamp (cbovenkamp@ccrglaw.com)
CHARHON CALLAHAN ROBSON &
GARZA, PLLC
3333 Lee Parkway, Suite 460
Dallas, TX 75219
Tel: (214) 521-6400
David E. Moore (#3983)
Bindu A. Palapura (#5370)
Brandon R. Harper (#6418)

Carson R. Bartlett (#6750)
Hercules Plaza, 6th Floor
1313 N. Market Street
Wilmington, DE 19801
Tel: (302) 984-6000
dmoore@potteranderson.com
bpalapura@potteranderson.com
bharper@potteranderson.com
cbartlett@potteranderson.com

Dated: January 10, 2023

By: /Jonathan Bradford /
Name: Jonathan Bradford
ROPES & GRAY LLP