(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 January 2009 (29.01.2009)

PCT

(10) International Publication Number
WO 2009/014659 A1

(51) International Patent Classification:
G06F 15/16 (2006.01)

(21) International Application Number:
PCT/US2008/008825

(22) International Filing Date: 18 July 2008 (18.07.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/950,774 19 July 2007 (19.07.2007) US

(71) Applicant *(for all designated States except US)*: **EBAY INC.** [US/US]; 2145 Hamilton Avenue, San Jose, California 95125 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **CHOI, Gregory** [US/US]; 1751 Parkview Green Circle, San Jose, California 95131 (US). **LAGUNAS, Diego** [IT/US]; 347 Massol Avenue # 204, Los Gatos, California 95030 (US). **POTTAVATHINI, Sathishwar** [IN/US]; 4211 Norwalk Drive, Apt CC104, San Jose, California 95129 (US).

(74) Agents: **STEFFEY, Charles E.** et al.; Schwegman, Lundberg & Woessner, P.A., P.O. Box 2938, Minneapolis, Minnesota 55402 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL,

[Continued on next page]

(54) Title: WEB PAGE CACHE STATUS DETECTOR

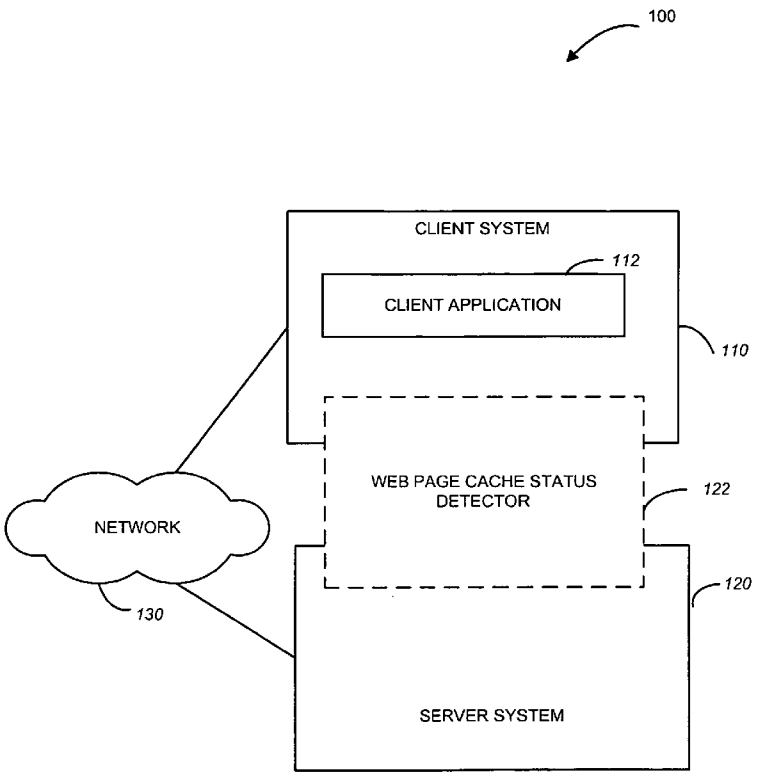

FIG. 1

(57) Abstract: A method and system to determine whether a web page has been cached is provided. An example system comprises a cookie generator, a cookie distributor, and a cookie evaluator. The cookie distributor may be configured to provide the code to a client system, in response to a request for web content from the client system. A value of the code to be updated at the client system in response to the client system initiating a request for the web content. The cookie evaluator may be configured to compare a value of the code to the default value. The cached status detector may be configured to use a result of the comparing to determine a cached status of the web content, the cached status to indicate whether the web content has been cached by the client system.

WO 2009/014659 A1

NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

WEB PAGE CACHE STATUS DETECTOR

RELATED APPLICATIONS

This application is related to and hereby claims the priority benefit of U.S. Provisional Patent Application No. 60/950,774 filed July 19, 2007 and entitled "METHOD AND SYSTEM TO DETECT WHETHER A WEB PAGE HAS BEEN CACHED", which application is incorporated herein by reference in its entirety.

TECHNICAL FIELD

This application relates to a method and system to detect whether a web page has been cached.

BACKGROUND

In the context of web application development and testing, it may be desirable to determine how fast can a web page be loaded in response to a user's request. For example a method for testing web-based applications may include measuring the response time of one or more web pages. Specifically, after the loading of a web page is initiated, an event is received indicating preparation to navigate to the web page and a timer mechanism is started. Another event is received indicating that the web page has completed loading and the timer mechanism is stopped and the elapsed time for the web page to load is determined by accessing the timer readings. This method does not distinguish between loading a web page for the first time and loading a web page that was previously cached at the client system associated with the requesting user. A web page sent from the server computer (server) typically behaves in the same manner as a cached web page does.

BRIEF DESCRIPTION OF DRAWINGS

Embodiments of the present invention are illustrated by way of example and not limitation in the figures of the accompanying drawings, in which like references indicate similar elements and in which:

Figure 1 is a block diagram showing a network environment within which a method and system to detect whether a web page has been cached may

be implemented;

Figure 2 is a block diagram illustrating a system to detect whether a web page has been cached, in accordance with an example embodiment;

Figure 3 is a flow chart illustrating a method to detect whether a web page has been cached, in accordance with an example embodiment; and

Figure 4 illustrates a diagrammatic representation of an example machine in the form of a computer system within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed.

DETAILED DESCRIPTION

A method and system to detect whether a web page has been cached is described. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of an embodiment of the present invention. It will be evident, however, to one skilled in the art that the present invention may be practiced without these specific details.

In one example scenario, in order to evaluate network latency, an approach may be pursued where the network latency is reflected in a value associated with the time duration it takes for a web page or any web content to load in response to a request. Example operations that may be utilized to perform this task are listed below.

1. Upon request, take the current time of the request (t1), and insert this time stamp as apart of the generated web page.
2. On the browser, take a time stamp at the beginning of the rendering of the web page (c1), and at the end of the rendering of the web page (c2).
3. After the page loads, send to the server, e.g., through an image tag, the values of t1 and (c2 – c1).
4. On the server, record the new current time (t2) and subtract the (t2) from (t1) to get the end to end time. Then perform a calculation as follows (t2 - t1) - (c2 - c1), which reflects network latency.

The approach described above may be beneficial in cases of normal web page execution. When a page that is being served is a cached web page, (e.g., the web page is being served in response to a user activating the "back" control

button on the browser), t1 represents the cached t1 time. This would affect the result in the end to end result calculation.

In order to determine where a page that has been loaded is a cached web page, an approached has been provided that uses web cookies, which is described below. Hypertext Transfer Protocol (HTTP) cookies, referred to as web cookies or merely cookies, are server generated identifiers stored on the computer of the person browsing the web, which are sent to the server with each request. In one example embodiment, a specified cookie (e.g., a code) may be set on the server to a default value for every request to access a particular web page. On the client, for every request to access the web page, the default value of the specified cookie is being modified. On the client, if the read cookie value is different from the default value stored on the server, it is concluded that the web page never hit the server, and therefore has been cached. Another way to describe this approach is as follows.

On the server for every request, we set a specified cookie to its default value and on the client we modify this value. On the client, if the read cookie value is not the default value, we know the page never hit the server, and therefore has been cached.

In one example embodiment, the method and system to determine whether a web page has been cached may be utilized as described below. Suppose an advertisement from a 3rd party is served up on a web given page. For every request for the web page, a unique identifier (e.g., generated on the client) may be added to the query string of the call in order to ensure that the advertisement call (ad request) is not cached. The result is that, even on cached web pages, the new advertisement call would have a new identifier, making it appear as if a new request has been made. which affects metrics. With the above solution, in one example embodiment, the identifier may be saved in the cookie. New requests would wipe out this value. If it is determined that this value is present, the cached identifier may
be used with the ad request so that the server side could identify which calls are new and which calls are cached.

Example system to detect whether a web page has been cached may be described with reference to a network environment 100 illustrated in Figure 1. The network environment 100 may include a client system (or client) 110 and a

server system (or server) 120. The client system 110 and the server system 120 may be in communications with each other via a network 130. The communications network 130 may be a public network (e.g., the Internet, a wireless network, a public switched telephone network (PSTN), etc.) or a private network (e.g., LAN, WAN, Intranet, etc.). Also shown in Figure 1 is a web page cache status detector 122. The web page cache status detector 122 may reside at the server system 120, at the client system 110, or be distributed between the server system 120 and the client system 110. The client system 110 is shown to host a client application (e.g., a web browser application) 112. The web page cache status detector 122 may be utilized to determine whether a web page that is a subject of information being gathered at the server system 120 has been cached by the client system 110. Example embodiment of a system to detect whether a web page has been cached may be described with reference to Figure 2.

Figure 2 is a block diagram illustrating a system 200 to determine whether a web page has been cached, in an example embodiment of the web page cache status detector 122 shown in Figure 2. The system 200 comprises a cookie generator 210 to generate HTML cookies, the cookie distributor 220 to store cookies at one or more client systems, and a so-called page loaded status detector 230 that determines whether a web page has been loaded at the client, and a cookie evaluator 240 to use cookies to determine whether a web page that has been loaded at the client was loaded from the server or from the client's cache. The system 200 further comprises a web page request detector 250 to receive a request for a web page from a client, and a response generator 260 to provide the requested web page to the client together with an instruction to update the value of the cookie stored at the client to the default value. In some embodiments, a default value of the cookie
may be updated by an update module 270. Example operations performed by the web page cache status detector 122 may be discussed with reference to Figure 3

Figure 3 is a flow chart of a method 300 to determine whether a web page has been cached, according to one example embodiment. The method 300 may be performed by processing logic that may comprise hardware (e.g., dedicated logic, programmable logic, microcode, etc.), software (such as run on

a general purpose computer system or a dedicated machine), or a combination of both. In one example embodiment, the processing logic resides at the server system 140 of Figure 1 and, specifically, at the system 200 shown in Figure 2.

As shown in Figure 3, the method 300 commences at operation 310, where the web page request detector 250 of Figure 2 received a request for web content from a client application. At operation 320, the cookie generator 210 of Figure 2 creates a cookie at a server system and sets the cookie to a default value. The method 300 may utilize the cookie distributor 220 of Figure 2 and the response generator 260 of Figure 2 send the response to the request for the web content and to store the cookie at the client system, at operation 330. The stored cookie is to be modified at the client system each time a request for a web page is initiated at the client. There are numerous ways in which the value of the cookie can be modified. For example, the value may be incrementally increased, decreased, or set to a randomly generated value.

At operation 340, the cookie is accessed at the client system. At operation 350, the cookie evaluator 240 of Figure 2, compares the value of the cookie received with the response to the request for the web content. If it is determined, at operation 380, that the two values match, the method 300 determines, at operation 370, that the web page has not been cached. Such determination may be made because, as described above, while each time a request for the web page is initiated, the value of the cookie at the client is updated, each time the web page is provided to the client from the server, the value of the cookie at the client is updated to a predetermined default value. If it is determined, at operation 360, that the two values are distinct from each other, the method 300 determines, at operation 380, that the web page has been cached, because it indicates that while the web page has been loaded, the client did not receive an instruction to update the value of the cookie.

Figure 4 shows a diagrammatic representation of machine in the example form of a computer system 400 within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed. In alternative embodiments, the machine operates as a standalone device or may be connected (e.g., networked) to other machines. In a networked deployment, the machine may operate in the capacity of a server or a client machine in server-client network environment, or as a peer machine in a

peer-to-peer (or distributed) network environment. The machine may be a personal computer (PC), a tablet PC, a set-top box (STB), a Personal Digital Assistant (PDA), a cellular telephone, a portable music player (e.g., a portable hard drive audio device such as an MP3 player), a web appliance, a network router, switch or bridge, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term "machine" shall also be taken to include any collection of machines that individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the methodologies discussed herein.

The example computer system 400 includes a processor 402 (e.g., a central processing unit (CPU), a graphics processing unit (GPU) or both), a main memory 404 and a static memory 406, which communicate with each other via a bus 408. The computer system 400 may further include a video display unit 440 (e.g., a liquid crystal display (LCD) or a cathode ray tube (CRT)). The computer system 400 also includes an alphanumeric input device 442 (e.g., a keyboard), a user interface (UI) navigation device 444 (e.g., a mouse), a disk drive unit 446, a signal generation device 448 (e.g., a speaker) and a network interface device 420.

The disk drive unit 446 includes a machine-readable medium 422 on which is stored one or more sets of instructions and data structures (e.g., software 424) embodying or utilized by any one or more of the methodologies or functions described herein. The software 424 may also reside, completely or at least partially, within the main memory 404 and/or within the processor 402 during execution thereof by the computer system 400, the main memory 404 and the processor 402 also constituting machine-readable media.

The software 424 may further be transmitted or received over a network 426 via the network interface device 420 utilizing any one of a number of well-known transfer protocols (e.g., HTTP).

While the machine-readable medium 422 is shown in an example embodiment to be a single medium, the term "machine-readable medium" should be taken to include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers) that store the one or more sets of instructions. The term "machine-readable medium" shall also be

taken to include any medium that is capable of storing, encoding or carrying a set of instructions for execution by the machine and that cause the machine to perform any one or more of the methodologies of the present invention, or that is capable of storing, encoding or carrying data structures utilized by or associated with such a set of instructions. The term "machine-readable medium" shall accordingly be taken to include, but not be limited to, solid-state memories, optical and magnetic media, and carrier wave signals. Such medium may also include, without limitation, hard disks, floppy disks, flash memory cards, digital video disks, random access memory (RAMs), read only memory (ROMs), and the like.

The embodiments described herein may be implemented in an operating environment comprising software installed on a computer, in hardware, or in a combination of software and hardware.

Although embodiments have been described with reference to specific example embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing from the broader spirit and scope of the invention. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A system comprising:

a cookie generator to:

create a code on the server system, and

set a value of the code to a default value;

a cookie distributor to provide the code to a client system, in response to a request for web content from the client system, a value of the code to be updated at the client system in response to the client system initiating a request for the web content;

a cookie evaluator to compare a value of the code to the default value; and

a cached status detector to use a result of the comparing to determine a cached status of the web content, the cached status to indicate whether the web content has been cached by the client system.

2. The system of claim 1, wherein the code is a web cookie.

3. The system of claim 1, further comprising:

a web content request detector to receive, from a client system, a request for the web content; and

a response generator to provide the requested web content to the client system together with a request to set the code at the client system to an updated value.

4. The system of claim 3, wherein the updated value is the default value.

5. The system of claim 3, further comprising an update module to update the default value to a new default value wherein the updated value is a new default value.

6. The system of claim 1, wherein the web content includes a third party content, the third party content is associated with an identifier, the identifier to be saved in the code.

7. The system of claim 6, comprising:

a web content request detector to receive, from a client system, a request for the web content; and

a response generator to provide the requested web content to the client system together with a request to erase the identifier from the code.

8. The system of claim 1, wherein the web content loaded status detector is to send to the client system a request to load the web content.

9. The system of claim 1, wherein the code evaluator is to:

determine that a value of the code matches the default value; and

conclude that the web content has not been cached by the client system.

10. The system of claim 1, wherein the code evaluator is to:

determine that a value of the code is distinct from the default value; and

conclude that the web content has been cached by the client system.

11. A method comprising:

creating a code on the server system;

setting a value of the code to a default value;

providing the code to a client system, in response to a request for web content from the client system, a value of the code to be updated at the client system in response to the client system initiating a request for the web content;

comparing a value of the code to the default value; and

using a result of the comparing to determine a cached status of the web content, the cached status to indicate whether the web content has been cached by the client system.

12. The method of claim 11, further comprising:
receiving a request for the web content; and
providing the web content to the client system.

13. The method of claim 11, wherein the updated value is the default value.

14. The method of claim 13, further comprising updating a value of the code on the server system to a new default value, wherein the updated value is the new default value.

15. The method of claim 11, wherein the web content includes a third party content, the third party content being associated with an identifier, the identifier to be saved in the code at the client system.

16. The method of claim 11, wherein the operation to determine a cached status of the web content comprises:
determining that a value of the code matches the default value; and
generating an indication that the web content has not been cached by the client system.

17. The method of claim 11, wherein the operation to determine a cached status of the web content comprises:
determining that a value of the code is distinct from the default value; and
generating an indication that the web content has cached by the client system.

18. The method of claim 11, wherein a value of the code to be updated at the client system by incrementally increasing the value of the code in response to the client system initiating a request for the web content.

19. The method of claim 11, wherein a value of the code to be updated at the client system by incrementally decreasing the value of the code in response to the client system initiating a request for the web content.

20. A machine-readable medium having instruction data to cause a machine to:

create a code on the server system;

set a value of the code to a default value;

provide the code to a client system, in response to a request for web content from the client system, a value of the code to be updated at the client system in response to the client system initiating a request for the web content;

compare a value of the code to the default value; and

use a result of the comparing to determine a cached status of the web content, the cached status to indicate whether the web content has been cached by the client system.

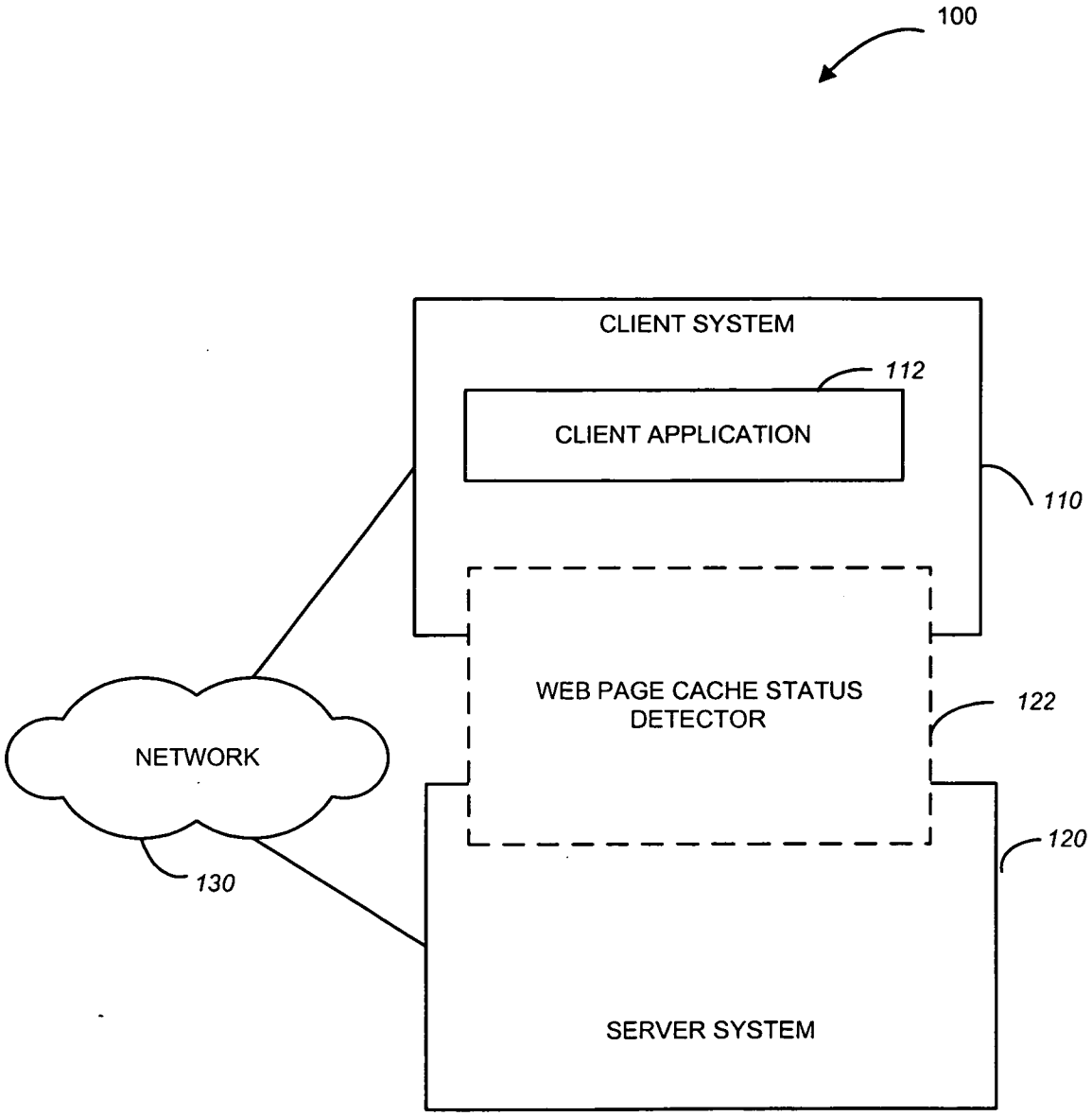
100
CLIENT SYSTEM
112
CLIENT APPLICATION
110
WEB PAGE CACHE STATUS DETECTOR
122
NETWORK
130
120
SERVER SYSTEM

FIG. 1

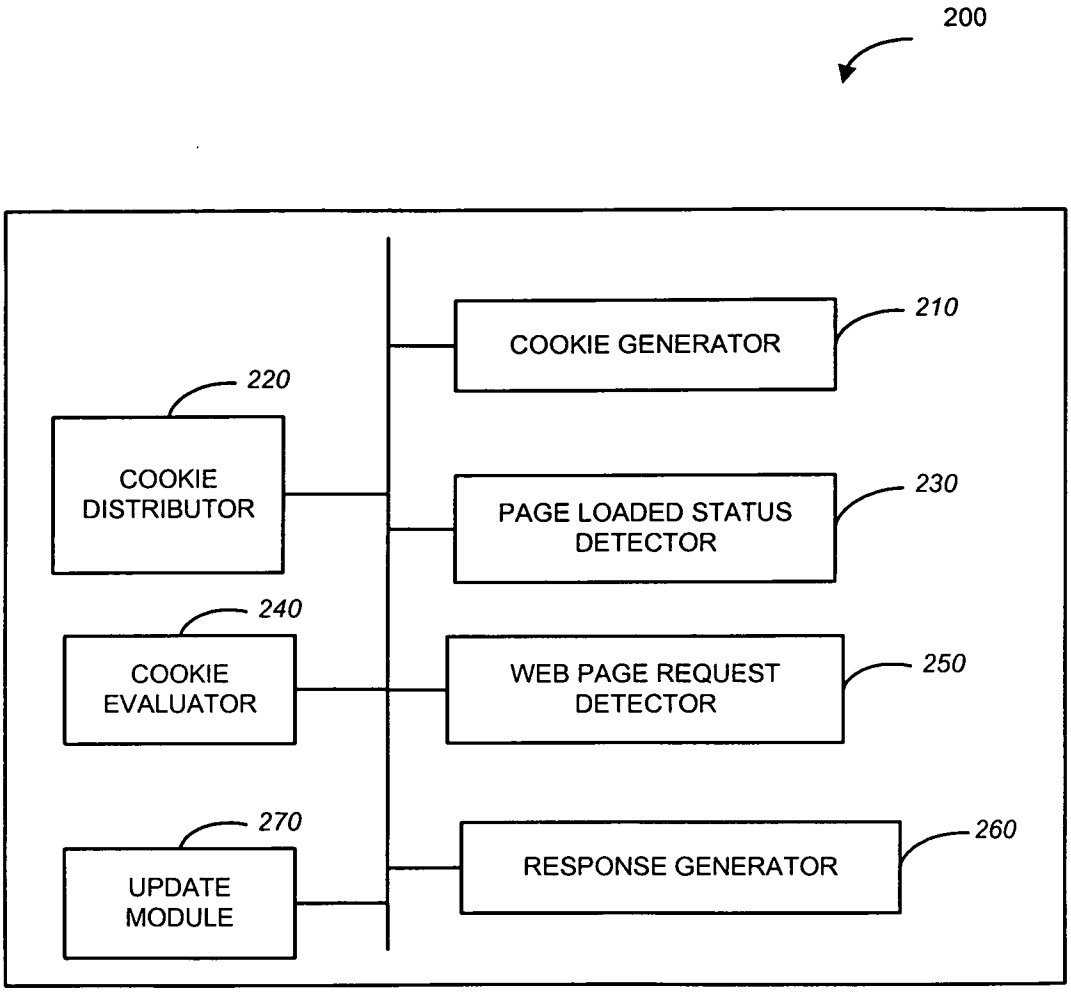
200
COOKIE GENERATOR
210
220
COOKIE DISTRIBUTOR
PAGE LOADED STATUS DETECTOR
230
240
COOKIE EVALUATOR
WEB PAGE REQUEST DETECTOR
250
270
UPDATE MODULE
RESPONSE GENERATOR
260

FIG. 2

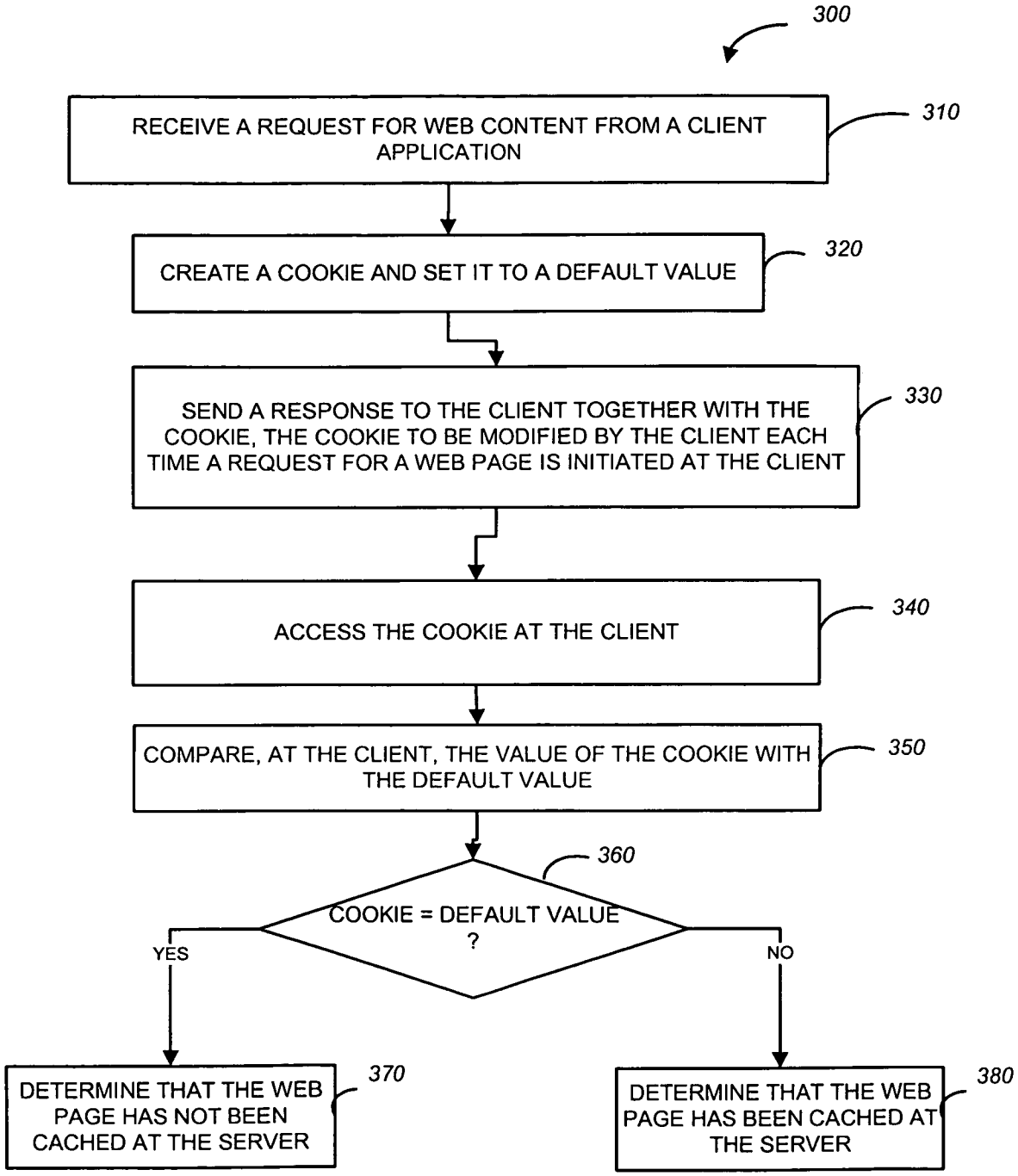
300
RECEIVE A REQUEST FOR WEB CONTENT FROM A CLIENT APPLICATION
310
CREATE A COOKIE AND SET IT TO A DEFAULT VALUE
320
SEND A RESPONSE TO THE CLIENT TOGETHER WITH THE COOKIE, THE COOKIE TO BE MODIFIED BY THE CLIENT EACH TIME A REQUEST FOR A WEB PAGE IS INITIATED AT THE CLIENT
330
ACCESS THE COOKIE AT THE CLIENT
340
COMPARE, AT THE CLIENT, THE VALUE OF THE COOKIE WITH THE DEFAULT VALUE
350
360
COOKIE = DEFAULT VALUE ?
YES
NO
DETERMINE THAT THE WEB PAGE HAS NOT BEEN CACHED AT THE SERVER
370
DETERMINE THAT THE WEB PAGE HAS BEEN CACHED AT THE SERVER
380

FIG. 3

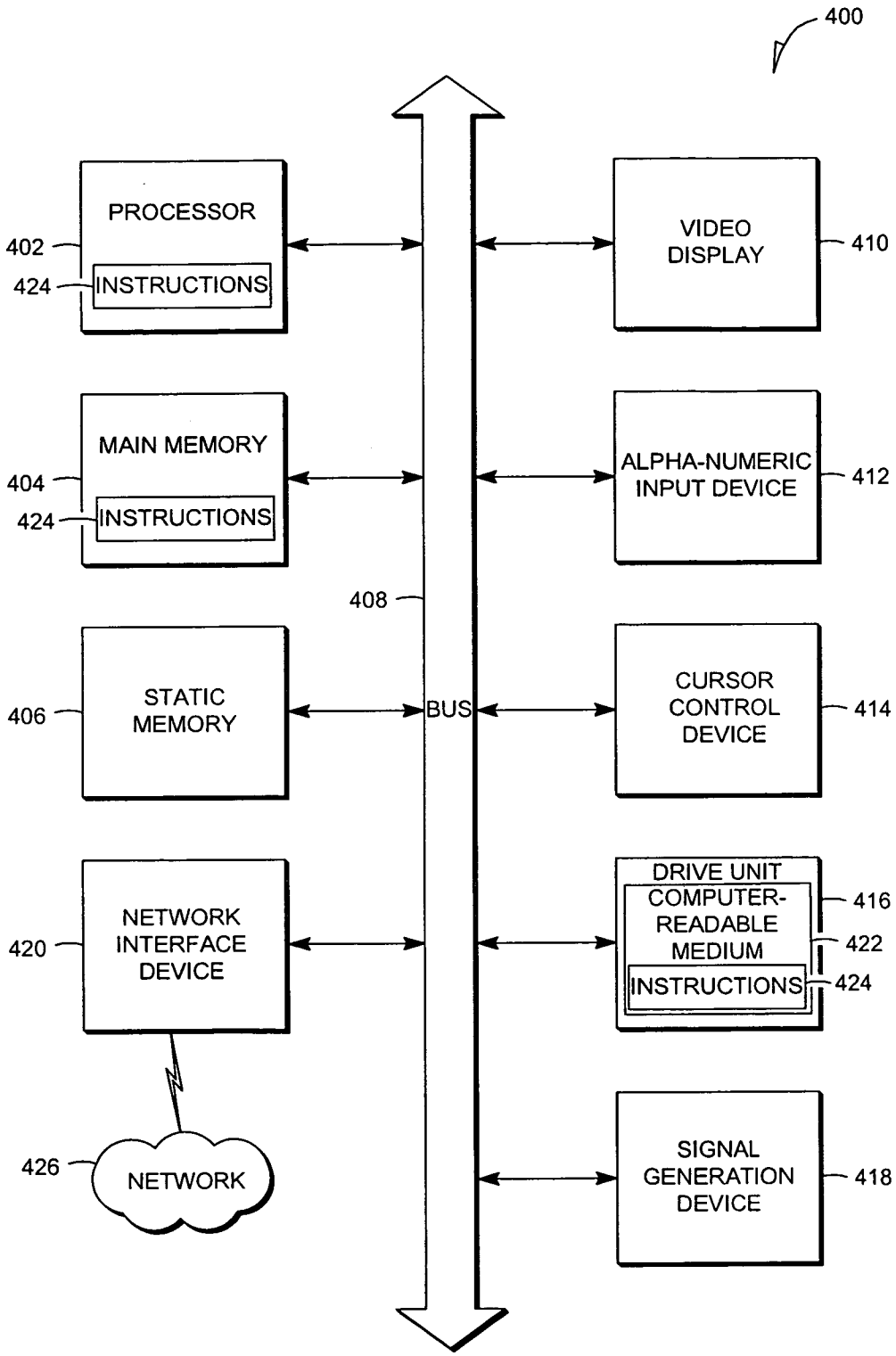
400
PROCESSOR
402
424
INSTRUCTIONS
MAIN MEMORY
404
424
INSTRUCTIONS
408
STATIC MEMORY
406
BUS
NETWORK INTERFACE DEVICE
420
426
NETWORK
VIDEO DISPLAY
410
ALPHA-NUMERIC INPUT DEVICE
412
CURSOR CONTROL DEVICE
414
DRIVE UNIT
COMPUTER-READABLE MEDIUM
416
422
INSTRUCTIONS
424
SIGNAL GENERATION DEVICE
418

FIGURE 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 08/08825

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 15/16 (2008.04)
USPC - 709/228
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
709/228

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
709/227,203,726/9,10,715/501.1,711/133

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Google; Google Scholar; Google Patents; PubWest(PGPB,USPT,USOC,EPAB,JPAB);
Search Terms Used: web page, cache, status, cookie generator, code, cookie distributor, cookie evaluator, cached status detector, update module

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2005/0044321 A1 (BIALKOWSKI et al.) 24 February 2005 (24.02.2005), para [0048]-[0052]	1-20
Y	US 2007/0150822 A1 (MANSOUR et al.) 28 June 2007 (28.06.2007), para [0090], para [0179], para [0215]-[0219]	1-20
A	US 6,931,439 B1 (HANMANN et al.) 16 August 2005 (16.08.2005)	1-20

☐ Further documents are listed in the continuation of Box C. ☐

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
31 October 2008 (31.10.2008)	14 NOV 2008

Name and mailing address of the ISA/US	Authorized officer:
Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774