A method and apparatus for transferring state information between a server computer system and a client computer system. In one embodiment of the method, an http client requests a file, such as an HTML document, on an http server, and the http server transmits the file to the http client. In addition, the http server transmits a state object, which describes certain state information, to the http client. The http client stores the state object, and will typically send the state object back to the http server when making later requests for files on the http server. In a typical embodiment, the state object includes a domain attribute which specifies a domain or network address, and the state object is transmitted from the http client to a server only when the http client makes an http request to the server and the server is within the domain. In one embodiment, the apparatus includes a processor and memory and a computer readable medium which stores program instructions. In the case of the client system, the instructions specify operations such as receiving and storing the state information; in the case of the server system, the instructions specify operations such as sending the state information to a client system.

28 Claims, 8 Drawing Sheets
### U.S. PATENT DOCUMENTS

<table>
<thead>
<tr>
<th>Patent Number</th>
<th>Date</th>
<th>Inventor(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4,528,643 A</td>
<td>7/1985</td>
<td>Freeny, Jr.</td>
</tr>
<tr>
<td>4,529,870 A</td>
<td>7/1985</td>
<td>Chaum</td>
</tr>
<tr>
<td>4,566,078 A</td>
<td>1/1986</td>
<td>Crabtree</td>
</tr>
<tr>
<td>4,578,530 A</td>
<td>3/1986</td>
<td>Zeidler</td>
</tr>
<tr>
<td>4,734,858 A</td>
<td>3/1988</td>
<td>Schlabique</td>
</tr>
<tr>
<td>4,759,063 A</td>
<td>7/1988</td>
<td>Chaum</td>
</tr>
<tr>
<td>4,759,064 A</td>
<td>7/1988</td>
<td>Chaum</td>
</tr>
<tr>
<td>4,775,935 A</td>
<td>10/1988</td>
<td>Yourick</td>
</tr>
<tr>
<td>4,795,890 A</td>
<td>1/1989</td>
<td>Goldman</td>
</tr>
<tr>
<td>4,827,608 A</td>
<td>5/1989</td>
<td>Shear</td>
</tr>
<tr>
<td>4,891,503 A</td>
<td>1/1990</td>
<td>Jewell</td>
</tr>
<tr>
<td>4,922,521 A</td>
<td>5/1990</td>
<td>Krikke et al.</td>
</tr>
<tr>
<td>4,926,480 A</td>
<td>5/1990</td>
<td>Chaum</td>
</tr>
<tr>
<td>4,941,089 A</td>
<td>7/1990</td>
<td>Fischer</td>
</tr>
<tr>
<td>4,947,028 A</td>
<td>8/1990</td>
<td>Gorog</td>
</tr>
<tr>
<td>4,947,430 A</td>
<td>8/1990</td>
<td>Chaum</td>
</tr>
<tr>
<td>4,949,230 A</td>
<td>8/1990</td>
<td>Chaum</td>
</tr>
<tr>
<td>4,977,595 A</td>
<td>12/1990</td>
<td>Ohta et al.</td>
</tr>
<tr>
<td>4,987,593 A</td>
<td>1/1991</td>
<td>Chaum</td>
</tr>
<tr>
<td>4,991,210 A</td>
<td>2/1991</td>
<td>Chaum</td>
</tr>
<tr>
<td>4,992,940 A</td>
<td>2/1991</td>
<td>Dworkin</td>
</tr>
<tr>
<td>4,996,711 A</td>
<td>2/1991</td>
<td>Chaum</td>
</tr>
<tr>
<td>5,056,712 A</td>
<td>10/1991</td>
<td>Enck</td>
</tr>
<tr>
<td>5,060,153 A</td>
<td>10/1991</td>
<td>Nakagawa</td>
</tr>
<tr>
<td>5,204,947 A</td>
<td>4/1993</td>
<td>Bernstein et al.</td>
</tr>
<tr>
<td>5,257,079 A</td>
<td>7/1993</td>
<td>Holloway</td>
</tr>
<tr>
<td>5,247,975 A</td>
<td>9/1993</td>
<td>Sprague et al.</td>
</tr>
<tr>
<td>5,276,736 A</td>
<td>1/1994</td>
<td>Chaum</td>
</tr>
<tr>
<td>5,305,195 A</td>
<td>4/1994</td>
<td>Murphy</td>
</tr>
<tr>
<td>5,309,437 A</td>
<td>5/1994</td>
<td>Perlman et al.</td>
</tr>
<tr>
<td>5,311,594 A</td>
<td>5/1994</td>
<td>Penzias</td>
</tr>
<tr>
<td>5,325,362 A</td>
<td>6/1994</td>
<td>Aziz</td>
</tr>
<tr>
<td>5,336,870 A</td>
<td>8/1994</td>
<td>Hughes</td>
</tr>
<tr>
<td>5,351,186 A</td>
<td>9/1994</td>
<td>Bullock et al.</td>
</tr>
<tr>
<td>5,351,293 A</td>
<td>9/1994</td>
<td>Mischener et al.</td>
</tr>
<tr>
<td>5,353,383 A</td>
<td>10/1994</td>
<td>Tsuchiyama</td>
</tr>
<tr>
<td>5,388,257 A</td>
<td>2/1995</td>
<td>Bauer</td>
</tr>
<tr>
<td>5,414,833 A</td>
<td>5/1995</td>
<td>Hershey et al.</td>
</tr>
<tr>
<td>5,457,738 A</td>
<td>10/1995</td>
<td>Sylvan</td>
</tr>
<tr>
<td>5,475,585 A</td>
<td>12/1995</td>
<td>Biash</td>
</tr>
<tr>
<td>5,491,820 A</td>
<td>2/1996</td>
<td>Belove et al.</td>
</tr>
<tr>
<td>5,521,631 A</td>
<td>5/1996</td>
<td>Budow et al.</td>
</tr>
<tr>
<td>5,530,849 A</td>
<td>6/1996</td>
<td>Haushevesky et al.</td>
</tr>
<tr>
<td>5,544,320 A</td>
<td>8/1996</td>
<td>Korad</td>
</tr>
<tr>
<td>5,544,322 A</td>
<td>8/1996</td>
<td>Cheng et al.</td>
</tr>
<tr>
<td>5,550,984 A</td>
<td>8/1996</td>
<td>Gell</td>
</tr>
<tr>
<td>5,557,516 A</td>
<td>9/1996</td>
<td>Hagan</td>
</tr>
<tr>
<td>5,557,518 A</td>
<td>9/1996</td>
<td>Rosen</td>
</tr>
<tr>
<td>5,557,798 A</td>
<td>9/1996</td>
<td>Skeen et al.</td>
</tr>
<tr>
<td>5,560,008 A</td>
<td>9/1996</td>
<td>Johnson et al. .................. 395/650</td>
</tr>
<tr>
<td>5,566,397 A</td>
<td>10/1996</td>
<td>Devarakonda et al.</td>
</tr>
<tr>
<td>5,583,996 A</td>
<td>12/1996</td>
<td>Tsuchiyama</td>
</tr>
<tr>
<td>5,590,197 A</td>
<td>12/1996</td>
<td>Chen et al.</td>
</tr>
<tr>
<td>5,592,378 A</td>
<td>1/1997</td>
<td>Cameron et al.</td>
</tr>
</tbody>
</table>

### FOREIGN PATENT DOCUMENTS

- **EP 054 298 A** 5/1993 Davis et al.
- **EP 046 920 A** 1/1991 Davis et al.
- **EP 054 298 A** 5/1993 Davis et al.
Van Mane, M.L.; “Putting the lid on Pandora’s cookie jar”; Aug. 19, 1996; PC Week magazine v. 13, No. 33, p. 8; Ziff-Davis Publishing Co.
Foster, E.; “Can mixing “cookies” with online marketing be a recipe for heartburn?”; Jul. 22, 1996; InfoWorld magazine v. 18, No. 30, p. 54; InfoWorld Publishing Company.
Netscape; “Persistent Client State HTTP Cookies”; retrieved from the Internet on Nov. 19, 1997 from webpage: http://home.netscape.com/newsref/std/cookiespec.html.
CMP Netguide magazine; “The Trouble With Cookies”; May 1, 1996; Issue 305, CMP Media.
Benners-Lee, Tim; webpage retrieved on Sep. 11, 2008: http://www.w3.org/People/Benners-Lee/5 pages.
Trubey, Phil, “Protocol Proposal based on SNPP,” e-mail dated Nov. 24, 1993, 7 pages.


Farber, David, “Interesting-people Message—RSA/NCSA/EIT Announcement on Secure Mosaic [also see note at very end], posting on Internet message board, Apr. 15, 1994, 4 pages.


Dan Kohn, “Prior Art on Open Market Patents,” posting on Internet message board, Mar. 9, 1998, 1 page.


Exhibit of emails relating to "wwworder" mailing list, dated May 31, 1994 to Jun. 28, 1994, 26 pages.


Hughes, Kevin, source code file, "getstats v1.0", Feb. 1, 1994, 64 pages.

McCartney, Todd, "Rhythm of The Pridelands Info," posting on Internet Newsgroup rec.arts.disney, Nov. 21, 1994, 1 page.


"Electronic In-Home Shopping: ‘Our Stores are Always Open,’" Chain Store Age Executive, Mar. 1985, 2 pages.

"Mail Offers a Holiday Treat for Hackers", Nov. 1985, Advertising Age, 1 page.


Montulli, Lou et al.; e-mail correspondence between members of HTTP Working Group ("http-wg"), Subject: cookies patent?, Feb. 4-6, 2000, 6 pages.

Kristol, David and Montulli, Lou.; e-mail correspondence between Lou Montulli and David Kristol; Subject: which release first had cookies?, Apr. 9, 2001, 3 pages.

Netscape’s First Amended Complaint (Dkt. 71), filed May 19, 2009, 8 pages.

ValueClick and FastClick’s Answer, Defenses, and Counterclaims/ Exhs. A-D (Dkt. 72), filed May 20, 2009, 64 pages.

ValueClick and FastClick’s Answers to First Amended Complaint/ Exhs. A-D (Dkt. 85), filed May 29, 2009, 68 pages.


Defendants’ Initial Markman Brief w/ Exhs. 1-4 (Dkt. 130), filed Jul. 16, 2009, 143 pages.


ValueClick and FastClick’s First Amended Answer w/ Exhs. A-D (Dkt. 168), filed Jul. 31, 2009, 76 pages.


Decl. of Sheila M. Costin in Support of Defendants’ Motion for Summary Judgment w/ Exhs. 3, 9, 13-14, 16-18, 20, 22, 23 (Dkt. 211), filed Sep. 4, 2009, 279 pages.


Defendants’ Opposition to Netscape’s Motion of Summary Judgment w/ Exhs. 11, 22-28 (Dkt. 231), filed Sep. 16, 2009, 81 pages.

Netscape’s Opposition to Defendants’ Motion of Summary Judgment (Redacted) w/ Exhs. 10, 24 (Dkt. 234), filed Sep. 16, 2009, 60 pages.

Defendants’ Reply to Netscape’s Motion of Partial Summary Judgment (Redacted) (Dkt. 247), filed Sep. 22, 2009, 38 pages.

Netscape’s Reply to Defendants’ Motion for Summary Judgment (Dkt. 248), filed Sep. 23, 2009, 38 pages.


Koen Holman, “Non-Persistent Cookie Proposal,” lists w3.org/Archives/Public/www-talk/msg01499.html, Aug. 11, 1995, VC.0000002-9, 8 pages.


Soverain’s Fourth Supplemental Responses to Amazon.com’s First Set of Interrogatories (Nos. 1-14), dated Mar. 21, 2005, VC.0006222-32, 11 pages.

Soverain’s Responses to Amazon.com’s First Set of Requests for Admission to Plaintiff Soverain Software (Nos. 1-100), dated Mar. 21, 2005, VC.0008574-621, 48 pages.

Soverain’s Responses to Interrogatory Nos. 22, 23, 26 and 36 of Amazon.com’s Third Set of Interrogatories (Nos. 17-38), dated Mar. 21, 2005, VC.0008548-56, 9 pages.

Soverain’s Second Supplemental Responses to Amazon.com’s First Set of Interrogatories (Nos. 1-14), dated Sep. 21, 2004, VC.0014476-89, 14 pages.


Third Supplement to Defendant Amazon.com’s Initial Disclosures, dated Mar. 4, 2005, VC.0008500-04, 8 pages.


Videotaped Deposition of Glenn Crocker with Exhibits (Mar. 10, 2005), VC.0013353-428, 76 pages.


Videotaped Deposition of Kevin Ming-Wei Kadaja Hughes with Exhibits (Mar. 21, 2005), VC.0013177-243, 127 pages.

Videotaped Deposition of Thomas Mark Levergood (Mar. 8, 2005), VC.0012607-814, Part 1 of 2, pp. 1-104.

Videotaped Deposition of Thomas Mark Levergood (Mar. 8, 2005), VC.0012607-814, Part 2 of 2, pp. 105-208.


Videotaped Deposition of Stephen Jeffrey Morris (Mar. 9, 2005), VC.0012154-211, 58 pages.
Foster, E.; “Can mixing ‘cookies’ with online marketing be a recipe for heartburn?”; InfoWorld magazine v18 n30 p. 54; InfoWorld Publishing Company, Jul. 22, 1996.

CMP Netguide magazine; “The Trouble With Cookies”; Issue 305; CMP Media, May 1, 1996.

* cited by examiner
FIG. 1B
One of the major features of the world-wide-web is the ability to retrieve images from sites distributed around the world. These images can be then be combined with a text document retrieved separately and presented as a single "virtual" document to an end-user.

This HTML document demonstrates this capability. This document was fetched from a server in northern California. However, this image was obtained from the University of Stockholm.

FIG. 2
<Title>Distributed Image Loading Example</Title>

One of the major features of the world-wide-web is the ability to retrieve images from sites distributed around the world. These images can then be combined with a text document retrieved separately and presented as a single "virtual" document to an end-user. This HTML document demonstrates this capability.

This document was fetched from a server in northern California. However, this image:

\[ <IMG align=middle src="http://www.nesa.uiue.edu/demoweb/al-small.gif"> \]

was obtained from Illinois.

And this one: \[ <IMG align=middle src="http://www.su.se/SUlogo.gif"> \] came from the \[ <A href="http://www.su.se/index.html"> University of Stockholm</A> \].

FIG. 3A
Distributed Image Loading Example

One of the major features of the world-wide-web is the ability to retrieve images from sites distributed around the world. These images can then be combined with a text document retrieved separately and presented as a single "virtual" document to an end-user. This HTML document demonstrates this capability.

This document was fetched from a server in Northern California. However, this image:

was obtained from Illinois. And this one:

was obtained from the University of Stockholm.
Client sends a URL to the server.

Server returns an HTML document and cookies.

Client sends cookies to the server.

FIG. 4
Begin

212 Browser Sends a Request to Merchant Server.

214 Browser Receives and Displays HTML Document.

215 Additional Browsing?

Yes

216 Browser selects a Particular Product.

No

218 Merchant Server Sends Form.

222 Browser Sends POST With Specific Information About Selected Product.

224 Merchant Server Generates and Sends Synthetic Page and Cookie Describing Selected Product.

226 Browser Stores Cookie That Describes Selected Product.

228 More Product Browsing?

Yes

230 Browser Accesses a Check-Out Page and Sends Cookie Describing Selected Products.

No

240 Server Processes the Transaction.

End

FIG. 5
METHOD AND APPARATUS FOR
MAINTAINING STATE INFORMATION ON
AN HTTP CLIENT SYSTEM IN RELATION
TO SERVER DOMAIN AND PATH
ATTRIBUTES

Matter enclosed in heavy brackets [ ] appears in
the original patent but forms no part of this reissue
specification; matter printed in italics indicates the
additions made by reissue.

Cette application est un division de U.S. patent application
Ser. No. 08/540,342, filed Oct. 6, 1995, qui est now U.S. Pat.
No. 5,774,670, which issued on June 30, 1998.

CROSS REFERENCE TO RELATED
APPLICATIONS

This application is a reissue application of U.S. Pat. No.
08/918,977 filed Aug. 27, 1997), which is a division of U.S.
Pat. No. 5,774,670 granted June 30, 1999 (application Ser.
No. 08/540,342, filed Oct. 6, 1995). Notice: More than one
reissue application has been filed for the reissue of U.S. Pat.
No. 6,134,592. The reissue applications of U.S. Pat. No.
6,134,592 are U.S. patent application Ser. No. 10/272,896
(the present application), as well as its divisional reissue
applications including U.S. patent application Ser. Nos.
11/737,043, 11/737,042 and 11/737,055 all of which were
filed on Apr. 18, 2007.

FIELD OF THE INVENTION

This invention relates to communication in a client-server
computer systems. Specifically, the invention relates to cli-
ent-server computer systems in which a server can send state
information to a client and the client stores the state
information for later retransmissions back to the server.

BACKGROUND OF THE INVENTION

An important use of computers is the transfer of informa-
tion over a network. Currently, the largest computer network
in existence is the Internet. The Internet is a worldwide
interconnection of computer networks that communicate
using a common protocol. Millions of computers, from low
end personal computers to high-end super computers are
coupled to the InterNet.

The InterNet grew out of work funded in the 1960s by the
U.S. Defense Department’s Advanced Research Projects
Agency. For a long time, InterNet was used by researchers in
universities and national laboratories to share information.
As the existence of the InterNet became more widely known,
many users outside of the academic/research community
e.g., employees of large corporations) started to use InterNet
to carry electronic mail.

In 1989, a new type of information system known as the
World-Wide-Web (“the Web”) was introduced to the Inter-
Net. Early development of the Web took place at CERN, the
European Particle Physics Laboratory. The Web is a wide-
area hypermedia information retrieval system aimed to give
wide access to a large universe of documents. At that time, the
Web was known to and used by the academic/research com-

In 1993, researchers at the National Center for Supercom-
puting Applications (NCSA) released a Web browser called
“Mosaic” that implemented a graphical user interface (GUI).
Mosaic’s graphical user interface was simple to learn yet
powerful. The Mosaic browser allows a user to retrieve docu-
ments from the World-Wide-Web using simple point-and-click
commands. Because the user does not have to be tech-
ically trained and the browser is pleasant to use, it has the
potential of opening up the InterNet to the masses.

The architecture of the Web follows a conventional client
server model. The terms “client” and “server” are used to
refer to a computer’s general role as a requester of data (the
client) or provider of data (the server). Under the Web envi-
ronment, Web browsers reside in clients and Web documents
reside in servers. Web clients and Web servers communicate
using a protocol called “HyperText Transfer Protocol” (HTTP).
A browser opens a connection to a server and initiates a
request for a document. The server delivers the
requested document, typically in the form of a text document
coded in a standard HyperText Markup Language (HTML)
format, and when the connection is closed in the above
interaction, the server serves a passive role, i.e., it accepts
commands from the client and cannot request the client to perform
any action.

The communication model under the conventional Web
environment provides a very limited level of interaction
between clients and servers. In many systems, increasing the
level of interaction between components in the systems often
makes the systems more robust, but increasing the interaction
increases the complexity of the interaction and typically
slows the rate of the interaction. Thus, the conventional Web
environment provides less complex, faster interactions
because of the Web’s level of interaction between clients and
servers.

In the conventional Web environment, clients do not retain
information of a session after the session is closed. In many
systems, the ability to retain information after the systems
become inactive is crucial to the functioning of the systems.
Thus, it is desirable to allow clients to have this ability.

SUMMARY OF THE INVENTION

The present invention involves a client-server system on a
network in which a server can send state information to a
client and the client stores the state information. The stored
state information can later be sent back to the server at appro-
riate times. In this manner, the state of a client can be
maintained in the client-server system where no state inher-
ently exists in such a system.

One embodiment of the present invention is a network
system for communicating documents containing informa-
tion such as text and one or more images. The system com-
prises a first computer (i.e., a server) capable of sending such
documents over a network such as the InterNet. The system
also has a second computer (i.e., a client) which can request
these documents or files from the server. After the requested
documents are received, the client can display the documents.
In accordance with the present invention, the server can send
state information to the client when a document is sent. The
client then stores the state information, which is typically in
the form of a state object. In a subsequent request for docu-
ments to the server, the client can send the stored state infor-
mation to the server.

In an embodiment of the invention, the server uses a hyper-
text transfer protocol (“http”) to communicate over the net-
work with clients; such clients also communicate with the
server using the hyperertext transfer protocol. This server and
these clients are referred to as an http server and http clients respectively. The server typically will include a server processor and a memory and a computer readable medium, such as a magnetic ("hard disk") or optical mass storage device, and the computer readable medium of the server contains computer program instructions for transmitting the file from the server system to the client system and for transmitting the state object to the client system. The client typically will include a client processor and a memory and a computer readable medium, such as a magnetic or optical mass storage device, and the computer readable medium of the client contains computer program instructions for receiving the state object, which specifies the state information, from the server and for storing the state object at the client. The state object, in a typical embodiment, will include a name attribute, such as a domain attribute.

One of the applications of the present invention is an online shopping system. A customer can browse information delivered by a merchant server using a browser running on a client. The customer can also select products to be placed in a virtual shopping basket. The server then sends state information related to the selected products to the browser on the client for storage. When the customer wants to purchase the products in the virtual shopping basket, the browser sends the corresponding state information to a specified check-out Web page for processing.

Another application of the present invention is an "on-line" information service, such as a newspaper’s Web server which includes articles or other information from the newspaper’s subscription services. In one example, a newspaper or publishing company may have several different publications, each requiring a separate subscription fee which may differ among the different publications. A user of the information service may browse the different publications by making http requests, from the client’s/user’s computer system, to the publisher’s Web server which responds with the requested publication and state information specifying the user’s identification, and other subscription information (e.g., user registration and billing information) which allows the user to view the contents of the publication; this information is typically provided by the user at least once in a conventional log-on process. Thereafter, this information is included in the state information which is exchanged between the client and the server in the process of the invention. Accordingly, when the user, during the browsing process, desires to view another publication (e.g., from the same or different publisher) this state information will be transmitted back to the Web server to provide the necessary subscription information (thereby entitling the user to view the publication) without requiring the user to re-enter the necessary subscription information. In this manner, a user may browse from publication to publication on the Web server or a different Web server in the domain without having to re-enter, when seeking a new publication, the necessary subscription information.

These and other features of the present invention will be disclosed in the following description of the invention together with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features, and advantages of the present invention will be apparent from the following detailed description of the preferred embodiment of the invention with references to the following drawings.

FIG. 1A is a pictorial diagram of a computer network used in the present invention.
InterNet which is an interconnection of networks. A different request from computer 102 may be for a file that resides in server B. In this case, the data is transmitted from server B through the network to server A and, finally, to computer 102. The distance between servers A and B may be very long, e.g., across continents, or very short, e.g., within the same city. Further, in traversing the network the data may be transferred through several intermediate servers and many routing devices, such as bridges and routers.

The World-Wide-Web ("The Web") uses the client-server model to communicate information between clients and servers. Web Servers are coupled to the InterNet and respond to document requests from Web clients. Web clients (also known as Web "browsers") are programs that allow a user to simply access Web documents located on Web Servers.

FIG. 1B shows, in more detail, an example of a client-server system interconnected through the InterNet 100. In this example, a remote server system 122 is interconnected through the InterNet to client system 120. The client system 120 includes conventional components such as a processor 124, memory 125 (e.g., RAM), a bus 126 which couples the processor 124 and memory 125, a mass storage device 127 (e.g., a magnetic hard disk or an optical storage disk) coupled to the processor and memory through an I/O controller 128 and a network interface 129, such as a conventional modem.

The server system 122 also includes conventional components such as a processor 134, memory 135 (e.g., RAM), a bus 136 which couples the processor 134 and memory 135, a mass storage device 137 (e.g., a magnetic or optical disk) coupled to the processor 134 and memory 135 through an I/O controller 138 and a network interface 139, such as a conventional modem. It will be appreciated from the description below that the present invention may be implemented in software which is stored as executable instructions on a computer readable medium on the client and server systems, such as mass storage devices 127 and 137 respectively, or in memories 125 and 135 respectively.

InterNet Addresses

As discussed in the background, the InterNet consists of a worldwide computer network that communicates using well defined protocol known as the InterNet Protocol (IP). Computer systems that are directly connected to the InterNet each have an unique InterNet address. An InterNet address consists of four numbers where each number is less than 256. The four numbers of an InterNet address are commonly written out separated by periods such as 192.168.0.1.

To simplify InterNet addressing, the "Domain Name System" was created. The domain name system allows users to access InterNet resources with a simpler alphanumeric naming system. An InterNet Domain name consists of a series of alphanumeric names separated by periods. For example, the name "drizzle.stanford.edu" is the name for a computer in the physics department at Stanford University. Read from left to right, each name defines a subset of the name immediately to the right. In this example, "drizzle" is the name of a workstation in the "stanford" domain. Furthermore, "stanford" is a subset of the "edu" domain. When a domain name is used, the computer accesses a "Domain Name Server" to obtain the explicit number InterNet address.

Uniform Resource Locators

To further define the addresses of resources on the InterNet, the Uniform Resource Locator (URL) is a descriptor that specifies a type of InterNet resource and its location. URLs have the following format:

```
resource_type://domain.address/path_name
```

Where "resource_type" defines the type of internet resource.

Web documents are identified by the resource type "http" which indicates that the hypertext transfer protocol should be used to access the document. Other resource types include "ftp" (file transmission protocol) and "telnet". The "domain.address" defines the domain name address of the computer that the /resource is located on. Finally, the "path_name" defines a directory path within the file system of the server that identifies the resource. The right most name on the path name portion is usually the name of an actual file. Web pages are identified by the resource type "http". By convention, most Web pages end with the suffix "html" that suggests the file is a HyperText Markup Language document.

An example of a URL for a Web document is:

```
http://info.cern.ch/hypertext/DataSources/WWW/Geographical.html
```

This URL indicates that by using the HTTP (Web) protocol to reach a server called "info.cern.ch", there is a directory "hypertext/DataSources/WWW" that contains a hypertext document named "Geographical.html". Resources on the Internet are uniquely addressable by their URL.

Browsing the World-Wide-Web

To access an initial Web document, the user enters the URL for a Web document into a Web browser program. The Web browser then sends an http request to the server that has the Web document using the URL. The Web server responds to the http request by sending the requested HTTP object to the client. In most cases, the HTTP object is a plain text (ASCII) document containing text (in ASCII) that is written in HyperText Markup Language (HTML). The HTML document usually contains hyperlinks to other Web documents. The Web browser displays the HTML document on the screen for the user and the hyperlinks to other Web documents are emphasized in some fashion such that the user can select the hyperlink.

FIG. 2 illustrates the retrieval of remote text and images and their integration in a Web page by a client computer 130. In FIG. 2, server A contains a text document coded in a standard HTML format. Server B contains an image file called Image 1 and server C contains another image file called Image 2. Each of these servers is remotely located from the other servers and the client 130. The transfer of data is via the Internet. It should be appreciated that the text and image files could be located in the same server which is remote from client 130.

FIG. 3A shows an example of an HTML document. FIG. 3B shows the corresponding integrated document (Web page) as it would appear on a display screen of a client computer. The first line of the document in FIG. 3A reads "<title>Distributed Image Loading Example</title>". In this case, the tags "<title>" and "</title>" are HTML delimiters corresponding to the beginning and ending, respectively, of text that is designated as the title of the HTML document. The title could be used for various purposes, such as listing of the document in an automatically generated index.

The second line of the HTML document of FIG. 3A reads "<h1>Distributed Image Loading Example</h1>". The "<h1>" and "</h1>" are HTML delimiters for a header that is to be displayed in a largest font. The browser software running on the client computer 130 interprets the header tags and thus displays the text between the header tags in a largest font size on the client's display screen.
One of the major features... capability. At the end of the text paragraph is another HTML tag shown as <p>. This is a tag indicating the end of a paragraph.

To continue with the second paragraph of the HTML document, the text reads: "This document... image: <!--IMG align-middle src=http://www.uu.se/SUlogo.gif--> was obtained... ."

The text in angle brackets defines an image to be placed in the text. Specifically, the "IMG" tag indicates that an image is being defined. The "align-middle" tag indicates that the image should be aligned in the middle of the current line of text. Finally, the "src="" tag indicates that the source image file can be located using the URL "http://www.uu.se/SUlogo.gif".

The line continues with the phrase "from the <A href='http://www.uu.se/index.html'>University of Stockholm</A>". This phrase defines "University of Stockholm" as a link to another Web document. Specifically, the "A" tag defines the beginning of a link. The "href='"" tag defines that the link is to a Web page that can be located using the URL "http://www.uu.se/index.html". Next, the text "University of Stockholm" is the text that will be the link. Finally, the "/A" tag defines the end of the link definition. As illustrated in FIG. 3B, the text "University of Stockholm" is displayed with underlining that indicates it is a link to another document.

If the user selects the underlined text "University of Stockholm", then the browser will send out a http request for the Web page at the URL address "http://www.uu.se/index.html".

It can be seen from the above example that the HTML document contains all information a browser needs for displaying a Web page. Thus, the only responsibility of a Web server is to provide the requested document, and there is no need for the server to request a client to do anything else. However, this role of a server also limits the utility of the Web environment.

Adding State Information to the HyperText Transfer Protocol

The present invention provides an extension to the prior art HTTP protocol. Using the teachings of the present invention, when a server responds to an http request by returning an HTTP object to a client, the server may also send a piece of state information that the client system will store. In an embodiment of the present invention, the state information is referred to as a "cookie". Included in the state information (the cookie) is a description of a range of URL's for which that state information should be repeated back to. Thus, when the client system sends future HTTP requests to servers that fall within the range of defined URL's, the requests will include a transmission of the current value of the state object. By adding the ability to transfer state information back and forth, Web servers can then play an active role in transactions between clients and servers. The term state object is also used herein to refer to the state information.

FIG. 4 is a drawing showing schematically the flow of information between a client system and a server system. At a time indicated by numeral 174, the client system sends an http request to the Web server. In response to the http request, the server returns an HTML document together with a header, which is typically separate from the HTML documents, at a time indicated by numeral 174. The header may contain one or more cookies. Upon receiving the HTML document and the header, the client system displays an HTML document on the screen and stores the cookies in a memory such as a storage medium. The client may switch and work on other tasks, or may be shut down completely. This is indicated by a dash line 176. At a time indicated by numeral 178, the client system may access a Web server that is specified in the received cookie, such that the client system transmits the cookies to the server, thus providing state information about the client system to the server system.

This extension to the http protocol provides a powerful new tool which enables a large number of new types of applications to be written for a Web-based environment. Examples of new applications include on-line shopping that stores information about items currently selected by consumers, on-line services that can send back registration information and thus free users from retyping a user-id on next connection, and Web sites that can store per-user preferences on the client system and have the client supply those preferences every time the site is later accessed.

Server Behavior

A particular embodiment of the state information is described below in order to provide an example according to the present invention. It will be appreciated that alternative formats may be used in accordance with the principles of the present invention. As stated above, the extension to the HTTP protocol adds a new piece of state information to the HTTP header as part of an HTTP response from a Web server. Typically, the state information is generated by a common gateway interface ("CGI") script. The state information is stored by the receiving client system in the form of a "cookie list" for later use. The syntax of the new data, in one embodiment, is:

Set-Cookie: NAME=VALUE; expires=DATE; path=PATH;
domain=DOMAIN_NAME; secure

The capitalized terms can be set by the server system. The first attribute is "NAME=VALUE". This attribute serves to identify a cookie. The "NAME" attribute is a name for the cookie. The "NAME" attribute is the only required attribute on the "Set-Cookie" header in one embodiment. The "VALUE" is a value assigned to the previously defined name. The "VALUE" is a string of characters excluding, in one embodiment, semicolon, comma, and white spaces. If there is a need to place these characters in the VALUE, standard encoding methods, such as URL's type % XX encoding, can be used.

The "expires" attribute specifies a data string that defines the valid life time of the corresponding cookie. Once the expiration date has been reached, the cookie will no longer be stored in the client system. Thus, the client system will no longer respond to Web servers with the cookie. Many coding schemes for designating time can be used. In a preferred embodiment, the "expires" attribute is formatted as: Wdy, DD-Mon-YY HH:MM:SS GMT in the this format, "Wdy" designates the day of a week, "DD-Mon-YY" designates the day, month and year, and "HH:MM:SS GMT" designates the hour, minute and second, in GMT time zone. Note that the "expires" attribute lets a client know when it is safe to purge a cookie, however, the client is not required to delete the cookie. If an expires attribute is not provided by the server, then the cookies expires when the user's session ends. This can be implemented by storing the cookie only in volatile memory.

The "domain=DOMAIN_NAME" attribute defines a domain for which the cookie is valid. The domain attribute is usually set using the domain name of the sending Web server. Client systems examine the domain attribute when making later http requests. If the server that the client system is accessing falls within the defined DOMAIN_NAME, then the cookie may be sent to the server when making the http request. (The "path" must also be examined as will be explained later.) When a server system falls within the defined
US RE42,892 E

DOMAIN_NAME, this is referred to as a “tail match.” Note that a domain name that defines a subset of a domain is
due to match a larger enclosing domain. For example, the host names “arvil.acme.com” and “shipping.acme.com”
fall within the “acme.com” domain.

Only hosts within the specified domain can set a cookie for a domain. The value of the “domain” attribute must have at
least two periods in them to prevent accepting values of the form “.com” and “.edu”. If no domain name is specified, then
the default value of the “domain” attribute is the domain name of the server that generated the cookie header.

The “path” attribute is used to specify a subset of file system directories in a domain for which the cookie is valid.
If a cookie has already passed “domain” matching, then the path name of the URL for a requested document is compared
with the “path” attribute. If there is a match, the cookie is considered valid and is sent along with the http request.
All the characters of the defined path must match, however there may be additional characters on the path name. Thus, further
defined subdirectories will match a path to the parent directory. For example, the path “foo/bar” would match “/foo/bar/bar.html”,
and every “foo/bar”, but “/foo” will not match the path “/”. Note that the path “/” is the most general path since it
will match any path. If no path is specified when a cookie is created, then the default path will be the same path as the
document that was sent with the header which contains the cookie.

The last element of the cookie definition is the optional label of “secure.” If a cookie is marked “secure,” then the
cookie will only be retransmitted if there is a secure communication channel to the server system. In a preferred
embodiment of the present invention, this means that the cookie will only be sent to HTTPS servers. (HTTP over SSL.) If the
“secure” attribute is not specified, a cookie is considered safe to be sent over unsecured channels.

The defined extension to the HTTP protocol allows multiple set-cookie headers to be issued in a single HTTP
response. Each set-cookie header should follow the conventions of the above described format.

Client Behavior

As previously described, when a client receives a set-cookie command in a header, the client system stores the
cookie in some type of storage. In order not to place too much burden on client systems, each client system is expected to be
able to store only a limited number of cookies. In one embodiment, the storage requirements for the client systems are:

1. 300 total cookies;
2. 4 kilobytes per cookie; and
3. 20 cookies per server or domain (note that this rule
treats completely specified hosts and domains which are
different as separate entities, and each entity has a 20
cookies limitation).

Servers should not expect clients to be able to exceed these limits. When the 300 total cookies or the 20 cookie per server
limit is exceeded, clients may delete the least recently used
cookie even if the cookie’s “expires” time has not passed.

If a cookie is received that matches the “NAME”, “domain” and “path” attributes of a previously received
cookie, then the previously received cookie will be overwritten.

Using this technique, it is possible for a server to delete a cookie previously sent to a client. Specifically, a server that
wishes to delete a previous cookie sends a cookie having “expires” time which is in the past that matches the “NAME”,
“domain”, and “path” attributes of cookie to be deleted. Since the new overwritten cookie contains a expires time that has
passed, the cookie will be deleted by the client system. Note “NAME”, “domain” and “path” attributes of the expired

cookie must match exactly those of the valid cookie. Since a system must be within the domain that is specified in the
domain attribute, it is difficult for any server other than the
originator of a cookie to delete or change a cookie.

When a client system that implements the present invention
wishes to send an http request to a particular Web server, the
client system first examines its cookie list to see if the
cookie list contains any matching cookies that need to be sent
to the particular Web server. Specifically, before the client
sends an http request to a Web server, the client compares the
URL of the requested Web document against all of the stored
cookies. If any of the cookies in the cookie list matches the
requested URL then information containing the name/value
pairs of the matching cookies will be sent along with the
HTTP request. The format of the line is:

Cookie: NAME1=value1; NAME2=value2; . . .

When a client sends cookies to a server, all cookies with a
more specific path mapping should be sent before cookies
with less specific path mappings. For example, a cookie
“name1=foo” with a path mapping of “/bar” should be sent
before a cookie “name2=foo” with a path mapping of “/”
if they are both to be sent since the path “/bar” is more specific
than the global matching path “/”.

Paths having a higher-level value do not override more
specific path mappings. If there are multiple matches for a
given cookie name, but with separate paths, all the matching
cookies will be sent. Thus, both the cookie “name=foo” with
a path mapping of “/bar” and the cookie “name=foo” with a
path mapping of “/” should be sent since they have different
path names.

Some clients access Web servers on the Internet through
firewall systems that are designed to prevent unwanted Internet
traffic from affecting a local area network coupled to the
Internet. Firewall systems often implement “proxy servers”
that filter traffic to and from the Internet. It is important that
proxy servers not cache Set-cookie commands when caching
HTTP information. Thus, if a proxy server receives a
response that contains a Set-cookie header, the proxy server
should immediately propagate the Set-cookie header to the
client. Similarly, if a client system request contains a

“Cookie:” header, the cookie header should be forwarded
through a proxy even if a conditional “If-modified-since”
request is being made.

To further describe the present invention, the following
examples describe a set of Web transactions operating in
accordance with the present invention:

EXAMPLE 1

A client system requests a Web document from the Web
server “telemarking.acme.com” and receives in response:

Set-Cookie: CUSTOMER=WILE_E_COYOTE; path=/;
expires=Wednesday, Nov. 9, 1999 23:12:40

The client system stores this cookie in a local (client-side)
storage unit (e.g. mass storage 127 or memory 125). Since no
domain name was specifically identified, the domain will be
set to “telemarking.acme.com” since that is the domain name
of the server that generated the cookie. When the client later
makes an http request for a document in any path (since the
path is “/”) of a server system in the telemarking.acme.com
domain, the client sends:

Cookie: CUSTOMER=WILE_E_COYOTE

Assuming the client system makes another request to the
telemarking.acme.com domain, the client might receive
another cookie from the server such as:

Set-Cookie: PART_NUMBER=ROCKET_LAUNCHER;
path=/
US RE42,892 E

The client will locally store this additional cookie. Again, no domain name was identified, such that the default domain, "telemarking.acme.com" will be stored. Now, if the client makes yet another request to the "telemarking.acme.com" domain, the client will send all the cookies it has for that domain. Specifically, the client sends:

Cookie: CUSTOMER=WILE_E_COYOTE;
PART_NUMBER=ROCKET_LAUNCHER;

Assuming, the client continues transactions with the "telemarking.acme.com" server, it may receive the following cookie from the server:

Set-Cookie: SHIPPING=FEDEX; path=/foo

Then, if the client requests a document in path "/" on the "telemarking.acme.com" server, the client will send two cookies as state information:

Cookie: CUSTOMER=WILE_E_COYOTE;
PART_NUMBER=ROCKET_LAUNCHER;

Note that the cookie "SHIPPING=FEDEX" was not sent because the path "/" does not match the path "/foo". On the other hand, when the client requests a document on the "telemarking.acme.com" server in path "/foo" on this server, then the client will send three cookies as state information:

Cookie: CUSTOMER=WILE_E_COYOTE;
PART_NUMBER=ROCKET_LAUNCHER;
SHIPPING=FEDEX

EXAMPLE 2

Assume that all of the transactions of Example 1 have been cleared. A client system then requests a Web document from the Web server "telemarking.acme.com" and receives in response:

Set-Cookie: PART_NUMBER=ROCKET_LAUNCHER_1;

The client stores this cookie in a local (client-side) storage unit. Since no domain name was specifically identified, the domain will be set to "telemarking.acme.com". When the client later makes a request to a document in any path on a system in the telemarking.acme.com domain, the client sends back the following data as information:

Cookie: PART_NUMBER=ROCKET_LAUNCHER_1

Assuming the client continues to access the "telemarking.acme.com" server, the client may later receive from the server:

Set-Cookie: PART_NUMBER=RIDING_ROCKET_23;

The new cookie has the same name (PART_NUMBER) as an old cookie stored on the client system. Note that the old cookie is not overwritten since the new cookie has a different path attribute. Now, if the client makes a request for a document in the path "/ammno" on the telemarking.acme.com server, the client should send the following two cookies as state information:

Cookie: PART_NUMBER=RIDING_ROCKET_23;
PART_NUMBER=ROCKET_LAUNCHER_1

Both cookies are sent since the path of the requested document ("ammno") matches both the "/" path of the first cookie and the "/ammno" path of the second cookie. Note that the cookie PART_NUMBER=RIDING_ROCKET_23 is sent first since it has a more specific path ("ammno") than the global path ("/") associated with the cookie PART_NUMBER=ROCKET_LAUNCHER_1.

An On-line Shopping System Application

To illustrate one possible use of the state information system of the present invention, an implementation of an on-line shopping system will be described. The on-line shopping system allows customers to shop in one or more stores that are implemented as Web servers on the Internet. A customer can browse information on the Web servers that describe products available from the stores. When a desired product is found, the user can place the product into a virtual shopping basket. This virtual shopping basket is implemented as a set of cookies that are sent to the client computer system and stored on the client computer system. At check-out time, the customer pays for the selected products using some type of payment system such as a credit card. After payment is received, the on-line shopping system notifies the stores to ship the selected products to the customer. FIG. 5 is a flow chart showing the operation of the merchant system during an on-line shopping "trip" by a customer. The customer can run a browser on a client computer system, such as computer system 140 shown in FIG. 6 or client system 120 shown in FIG. 1B. The computer system 140 of FIG. 6 includes a display device 142 (such as a monitor), a display screen 144, a cabinet 146 (which encloses components typically found in a computer, such as CPU, RAM, ROM, video card, hard drive, sound card, serial ports, etc.), a keyboard 148, a mouse 150, and a modem 152. Mouse 150 have one or more buttons, such as buttons 154. The computer needs to have some type of communication device such as that Modem 152 allows computer system 140 to be connected to the Internet. Other possible communication devices include ethernet network cards.

The customer uses Web browser software to access an on-line "merchant" server that is operated by a merchant having products to sell. This merchant server is a server computer system such as server system 122 shown in FIG. 1B. Specifically, the browser software sends an http request for the home Web page of a merchant Web server (step 212). The merchant Web server responds to the request with an HTML document that is displayed by the browser (step 214). The home Web page contains information about the merchant and its products (e.g., shoes, hats, shirts, etc.). The home Web page can implement a set of linked Web pages that describe the products that are available from the merchant. Each product may be associated with its own HTML document that fully describes the product. Products can be described using text, images, sounds, video clips, and any other communication form supported by Web browsers. The user can continue browsing through Web pages of the merchant server by repeating steps 212, 214, and 215.

After browsing through the Web pages provided by the server, the customer may select a product (step 216) by, for example, "clicking" (in the conventional manner) on an image of a product that causes the browser to request a Web page that fully describes the product. If the customer wishes to buy shoes from the merchant, the customer could click on a "buy it" button. The merchant server then sends an HTML form document that requests the customer to send necessary details for the purchase (step 218). For example, the customer may select a quantity, a desired style, and size of the product as requested by the form document. The browser then sends a POST command under HTTP, which transmits the data entered into the form to the merchant server (step 222). The data on the submitted form (e.g., quantity, size, style, etc.) is analyzed by the server and the transaction is processed. The server then generates a synthetic page and sends it to the browser running on the client system. This synthetic page preferably contains a thank you note along with confirmation information. Cookies containing information describing the selected product are also sent at this time (step 224).

The browser software running on the client system stores the cookies describing the selected products within the client
computer system (step 226). The stored cookies include an identification of the contents of a virtual shopping basket that contains the products selected by the consumer. In an embodiment of the present invention, the cookies are stored in a file located in a storage medium (such as a hard disk) of client computer system 140. The time interval for storing the cookies that describe the selected products can be set to any desired length. In one embodiment of the present invention, the cookies are deleted when the customer exits from the browser. This can be accomplished by not setting the “expires” attribute of the product description cookies. In another embodiment of the present invention, the cookies are kept valid (prior to their expiration) even after the customer exits from the browser and turns off computer 140. This can be accomplished by setting the “expires” attribute of the product description cookies to a later date.

After selecting a product, the customer may do additional shopping (e.g., buy a hat) from the same store or other stores (step 228). In this case, steps 212, 214, 215, 216, 218, 222, 224 and 226 need to be performed for the additional products. Each selection of a product in step 222 will result in the transmission of a cookie from the server to the client, which cookie identifies the selected product. The customer may also exit from the merchant system at any time.

When the customer desires to buy the products, the customer accesses a link that identifies a “check-out” Web page. The check-out Web page causes the browser to send all the product description cookies (230). Thus, the check-out Web page empties out the virtual shopping basket. The merchant server generates a total bill for all the products in the virtual shopping basket. The server may then request billing information (e.g., credit card number) and shipping (e.g., address) information from the customer using a form. In a preferred embodiment the transaction of credit card information is transmitted using a secure medium. The transaction server then performs a real-time credit card authorization. Once the transaction is authorized, transaction server sends messages to individual merchants to fulfill the order (step 240).

Other functions could be added to the above described merchant system. For example, several persons could use the same browser for shopping. In this case, the browser identifies the person doing the shopping, and assigns product description cookies to the appropriate person. Thus, each person would have their own virtual shopping basket.

The invention has been described with reference to specific exemplary embodiments thereof and various modifications and changes may be made thereto without departing from the broad spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense; the invention is limited only by the following claims.

What is claimed is:

1. A method for subscribing to an on-line information service, said method comprising the steps of:
   requesting a first information service from an http server;
   transmitting a state object from a client computer system to said http server, said state object being stored on said client system and specifying user information to said http server.
   [2. A method as in claim 1 wherein said state object specifies user identification information and billing information.
   [3. A method as in claim 1 further comprising:
   requesting a second information service from said http server or an alternative http server;
   transmitting said object to said http server or said alternative http server.]

4. A method as in claim 3 wherein a user of said on-line information service browses from said first information service to said second information service without having to enter user information.

5. A method of claim 3 wherein said user information comprises subscription information required by a first publisher of said first information service.

6. A method as in claim 5 wherein said user information comprises subscription information required by a second publisher of said second information service.

7. A computer-implemented method performed by a hardware server system, comprising:
   receiving, at the server system, a hypertext transfer protocol (HTTP) request from a client;
   responding to the HTTP request by transmitting an HTTP response to the client wherein the HTTP response includes an HTTP header, the HTTP header including at least one set-cookie instruction specified by a “Set-Cookie” text string, wherein the set-cookie instruction includes:
   a name-value pair, the name-value pair specifying an assignment of a particular value to a particular name and being specified in the set-cookie instruction by a text string in a “NAME=VALUE” format; and
   attribute information, wherein the attribute information specifies criteria to enable the client to determine whether to return the name-value pair to the server system with a subsequent HTTP request and wherein the attribute information includes:
   a domain attribute that specifies a domain for which the name-value pair is valid, the domain being specified in the set-cookie instruction as a text string in a “domain=DOMAIN” format;
   a path attribute specifying a range of Uniform Resource Locators (URLs), in a domain of the server system, for which the name-value pair is valid, the path attribute being specified in the set-cookie instruction as a text string in a “path=PATH” format;
   and
   an expiration attribute that specifies a valid life time for the name-value pair, the valid life time specifying the persistent storage of the name-value pair across one or more browser sessions, each browser session corresponding to a period during which a browser application is running on the client, and terminating on a specified date, the expiration attribute being specified in the set-cookie instruction as a text string in a “expires=DATE” format.

8. The method of claim 7, further comprising: receiving a subsequent HTTP request from the client, wherein the subsequent HTTP request includes the name-value pair, and using the received name-value pair to identify a user.

9. The method of claim 8, wherein the HTTP request is received by a first server in the server system within a domain; and wherein the subsequent HTTP request is received by a second server in the server system within the domain, the second server being a different server from the first server.

10. The method of claim 7, wherein the HTTP header additionally includes a “secure” label indicating that the client should only send the name-value pair over a secure communication channel.

11. The method of claim 7, wherein the name-value pair includes a user identifier.
12. The method of claim 7, wherein the name-value pair includes information used by the server system to determine user preference information.

13. A computer storage device storing a computer program that embodies the method of claim 7.

14. The method of claim 7, wherein the name-value pair includes subscription information used by the server system to determine whether a user is authorized to access restricted content.

15. The method of claim 7, wherein the name-value pair includes information used by the server system to associate a user with one or more items selected for purchase.

16. The method of claim 7, wherein the HTTP response includes HTML content.

17. A computer-implemented server system, for use in a communications network, comprising:
   a processing system comprising one or more processors; and
   a memory comprising one or more computer readable media, wherein the memory stores computer instructions that, when executed by the processing system, cause the server system to perform the operations of:
   receiving, in response to the HTTP request, a hypertext transfer protocol (HTTP) request;
   sending, in response to the HTTP request, an HTTP response, wherein the HTTP response includes an HTTP header that includes at least one set-cookie instruction specified by a “Set-Cookie” text string, wherein the set-cookie instruction includes:
   a name-value pair, the name-value pair specifying an assignment of a particular value to a particular name and being specified in the set-cookie instruction by a text string in a “NAME=VALUE” format; and
   attribute information, wherein the attribute information specifies criteria to enable the client to determine whether to return the name-value pair to the server system with a subsequent HTTP request and wherein the attribute information includes:
   a domain attribute that specifies a domain for which the name-value pair is valid, the domain being specified in the set-cookie instruction as a text string in a “domain=DOMAIN” format;
   a path attribute that specifying a range of uniform resource locators (URLs) for which the name-value pair is valid in a domain of the server system, the path being specified in the set-cookie instruction as a text string in a “path=PATH” format; and
   an expiration attribute that specifies a valid life time for the first name-value pair; the valid life time specifying the persistent storage of the name-value pair across one or more browser sessions, each browser session corresponding to a period during which a browser application is running on the client, and terminating on a specified date, the expiration attribute being specified in the set-cookie instruction as a text string in an “expires=DATE” format.

18. The server system of claim 17, wherein the memory further stores computer instructions for performing the operations of:
   receiving a subsequent HTTP request from the client, wherein the subsequent HTTP request includes the name-value pair; and
   using the received name-value pair to identify a user.

19. The server system of claim 18, wherein the HTTP request is received by a first server in the server system within a domain; and wherein the subsequent HTTP request is received by a second server in the server system within the domain, the second server being a different server from the first server.

20. The server system of claim 17, wherein the HTTP header in the HTTP response further includes a secure attribute that specifies that the name-value pair should be returned by the client in a subsequent HTTP request only if the subsequent HTTP request is made using a secure channel.

21. The server system of claim 17, wherein the name-value pair includes a user identifier.

22. The server system of claim 17, wherein the name-value pair includes subscription information used by the server system to determine whether a user is authorized to access restricted content.

23. The server system of claim 17, wherein the name-value pair includes information used by the server system to associate a user with one or more items selected for purchase.

24. The server system of claim 17, wherein the name-value pair includes information used by the server system to determine a user’s preferences.

25. The server system of claim 17, wherein the HTTP response includes HTML content.

26. A computer-implemented method performed by a hardware client system, the method comprising:
   sending a first hypertext transfer protocol (HTTP) request to a server system during a first browsing session, the first browsing session corresponding to a period of time during which a browser application is running on the client system;
   receiving an HTTP response from the server system, wherein the HTTP response includes an HTTP header, the HTTP header specifying a “Set-Cookie” text string and including:
   a name-value pair;
   a domain attribute that specifies a domain for which the name-value pair is valid,
   a path attribute that specifies a range of uniform resource locators (URLs) for which the name-value pair is valid, in the domain, and
   an expiration attribute that specifies a valid life time for the name-value pair;
   storing the name-value pair on the client system such that the name-value pair is related to at least the domain attribute and the path attribute;
   subsequently, determining whether the name-value pair is valid for a URL of a second HTTP request by the client system made during a second browsing session, the second browsing session corresponding to a period of time during which a browser application is running on the client system and differing from the first browsing session, wherein determining whether the name-value pair is valid comprises comparing the URL to the domain attribute and the path attribute, and determining whether the second HTTP request is made at a time within the valid life time; and
   when the name-value pair is determined to be valid, transmitting the name-value pair within an HTTP header in the second HTTP request according to a “Cookie: NAME=VALUE” format.

27. The method of claim 26, wherein the HTTP header in the HTTP response additionally includes a “secure” label that specifies to the client system that the name-value pair should only be transmitted over a secure communication channel.

28. The method of claim 26, further comprising, on the client system:
subsequent to storing the name-value pair on the client system, receiving a second HTTP header from the server system, the second HTTP header specifying a second name-value pair, a second domain attribute, and a second path attribute; determining whether three conditions are met: (1) a name portion of the second name-value pair matches a name portion of the stored name-value pair; (2) the second domain attribute matches the domain attribute of the stored name-value pair; and (3) the second path attribute matches the path attribute of the stored name-value pair; and when the three conditions are met, overwriting the stored name-value pair on the client system with the second name-value pair.

29. A non-transitory computer-readable medium that stores a browser program which embodies the method of claim 26.

30. The method of claim 26, further comprising: determining whether the date specified by the expiration attribute is before a current date and deleting the name-value pair from memory when the date specified by the expiration attribute is before a current date.

31. A client system, comprising:
   - a processing system comprising one or more processors;
   - a memory comprising one or more computer-readable media, the memory containing computer instructions that, when executed by the processing system, cause the client system to perform the operations of:
     - sending a first hypertext transfer protocol (HTTP) request to a server system during a first browsing session, the first browsing session corresponding to a period of time during which a browser application is running on the client system;
     - receiving, in response to the first HTTP request, an HTTP response from the server system, wherein the HTTP response includes an HTTP header that specifies a "Set-Cookie:" text string and includes:
       - a name-value pair;
       - a domain attribute that specifies a domain for which the name-value pair is valid,
       - a path attribute that specifies a range of uniform resource locators for which the name-value pair is valid in the domain, and
       - an expiration attribute that specifies a valid life time for the name-value pair;
     - storing, in memory, the name-value pair;
     - sending a second HTTP request to the server system, during a second browsing session, the second browsing session corresponding to a period of time during which a browser application is running on the client system and differing from the first browsing session, wherein the second HTTP request specifies a domain and a resource; and
     - including the name-value pair in an HTTP header in the second HTTP request according to a "Cookie: NAME=VALUE;" format only if:
       - the domain specified by the second HTTP request is within the domain specified by the domain attribute,
       - the resource specified by the second HTTP request is within the path specified by the path attribute, and
       - unspecified by the expiration attribute.

32. The client system of claim 31, wherein the memory further includes instructions for performing the operation of determining whether the date specified by the expiration attribute is before a current date and deleting the name-value pair from memory when the date specified by the expiration attribute is before a current date.

33. The client system of claim 31, wherein:
   - the HTTP header in the HTTP response further includes a secure attribute that specifies that the name-value pair should be returned by the client system in a subsequent HTTP request only if the subsequent HTTP request is made using a secure channel; and
   - wherein sending the second HTTP request to the server system further comprises:
     - including the name-value pair in the HTTP header in the second HTTP request only if the second HTTP request is made using a secure channel.

34. A computer-implemented method performed by a hardware server system, comprising:
   - receiving, at the server system, a hypertext transfer protocol (HTTP) request from a client for an HTML document;
   - responding to the HTTP request by transmitting an HTTP response to the client, wherein the HTTP response includes the requested HTML document and an HTTP header, the HTTP header including at least one set-cookie instruction specified by a "Set-Cookie:" text string, wherein the set-cookie instruction includes:
     - a name-value pair, the name-value pair specifying an assignment of a particular value to a particular name and being specified in the set-cookie instruction by a text string in a "NAME=VALUE" format, wherein the name-value pair includes information descriptive of the requested HTML document, and
     - attribute information, wherein the attribute information specifies criteria to enable the client to determine whether to return the name-value pair to the server system with a subsequent HTTP request and wherein the attribute information includes:
       - a domain attribute that specifies a domain for which the name-value pair is valid, the domain being specified in the set-cookie instruction as a text string in a "domain=DOMAIN" format;
       - a path attribute specifying a range of Uniform Resource Locators (URLs), in a domain of the server system, for which the name-value pair is valid, the path attribute being specified in the set-cookie instruction as a text string in a "path=PATH" format; and
       - an expiration attribute that specifies a valid life time for the name-value pair, the valid life time specifying the persistent storage of the name-value pair across one or more browser sessions, each browser session corresponding to a period during which a browser application is running on the client, and terminating on a specified date, the expiration attribute being specified in the set-cookie instruction as a text string in an "expires=DATE" format.