US 20030195986A1

(19) **United States**

(12) **Patent Application Publication**

Densmore

(10) **Pub. No.: US 2003/0195986 A1**

(43) **Pub. Date: Oct. 16, 2003**

(54) **METHOD AND SYSTEM FOR DELIVERING DATA FROM A SERVER OBJECT TO A CLIENT OBJECT USING A NON-PROPRIETARY DATA TRANSFER PROTOCOL**

(76) Inventor: **Owen Densmore**, Palo Alto, CA (US)

Correspondence Address:
SONNENSCHEIN NATH & ROSENTHAL LLP
P.O. BOX 061080
WACKER DRIVE STATION, SEARS TOWER
CHICAGO, IL 60606-1080 (US)

(21) Appl. No.: **10/446,670**

(22) Filed: **May 29, 2003**

Related U.S. Application Data

(63) Continuation of application No. 09/107,536, filed on Jun. 30, 1998, now Pat. No. 6,591,305.

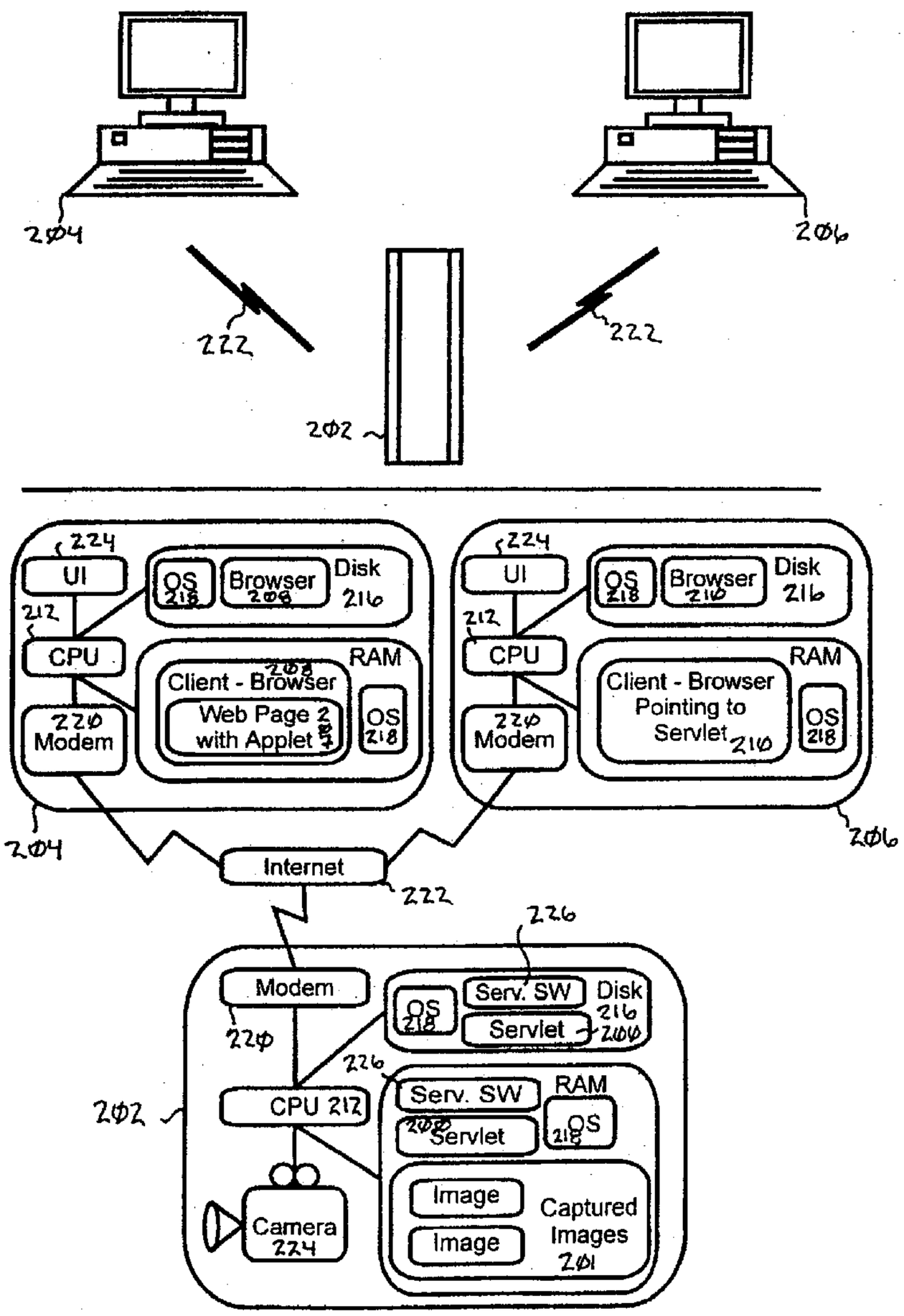
Publication Classification

(51) **Int. Cl.**[7] **G06F 15/16**

(52) **U.S. Cl.** **709/245**; 709/203

(57) **ABSTRACT**

The present invention is method and system capable of delivering video across the Internet using the hypertext transfer protocol. Client objects periodically request to view a uniform resource locator that is aliased to a servlet. For each request, the servlet retrieves and delivers a video image to the requesting client object. The invention can simultaneously deliver images to a variety of client objects, regardless of whether they are requesting the images at the same rate.



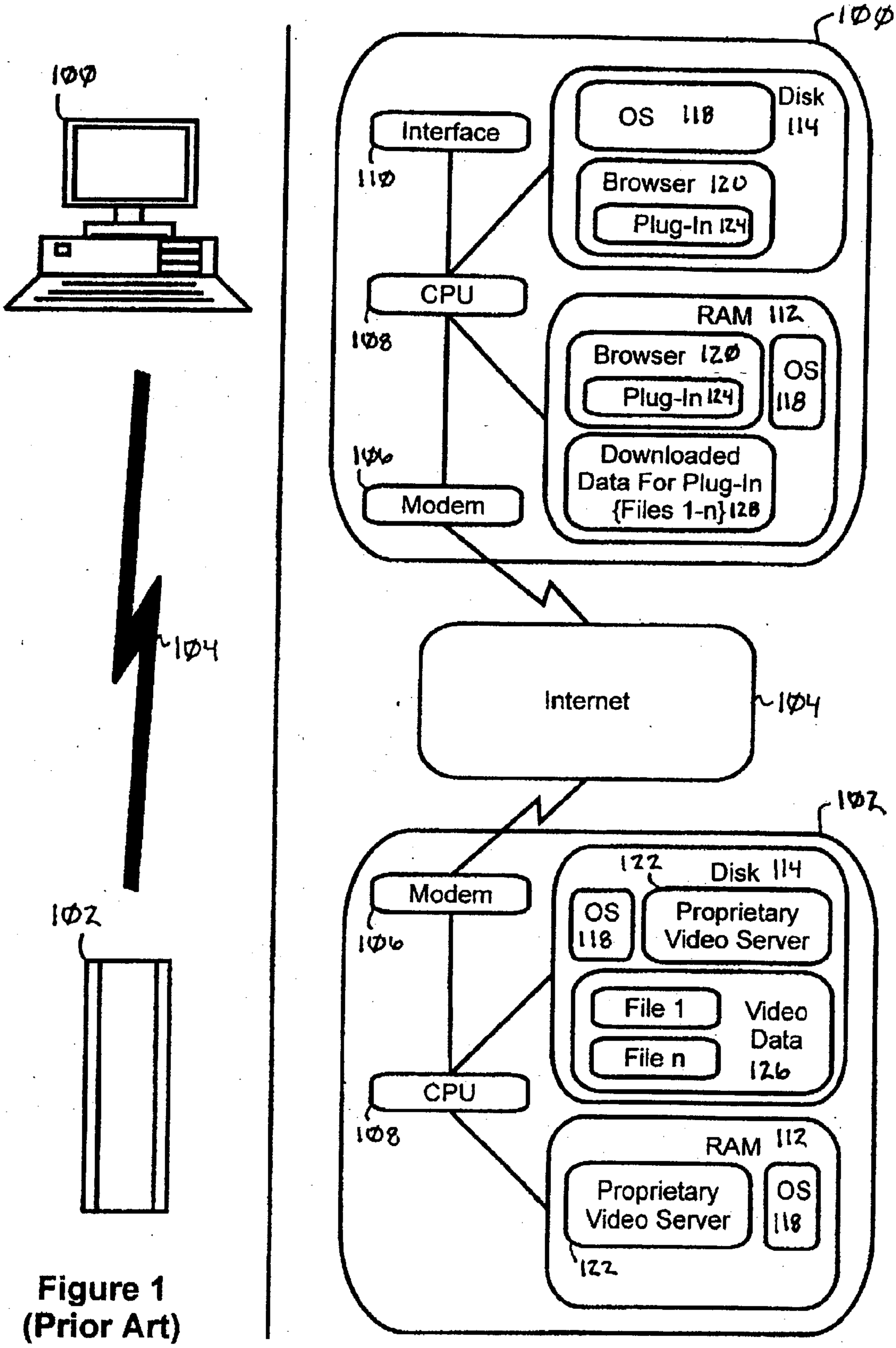


Figure 1 (Prior Art)

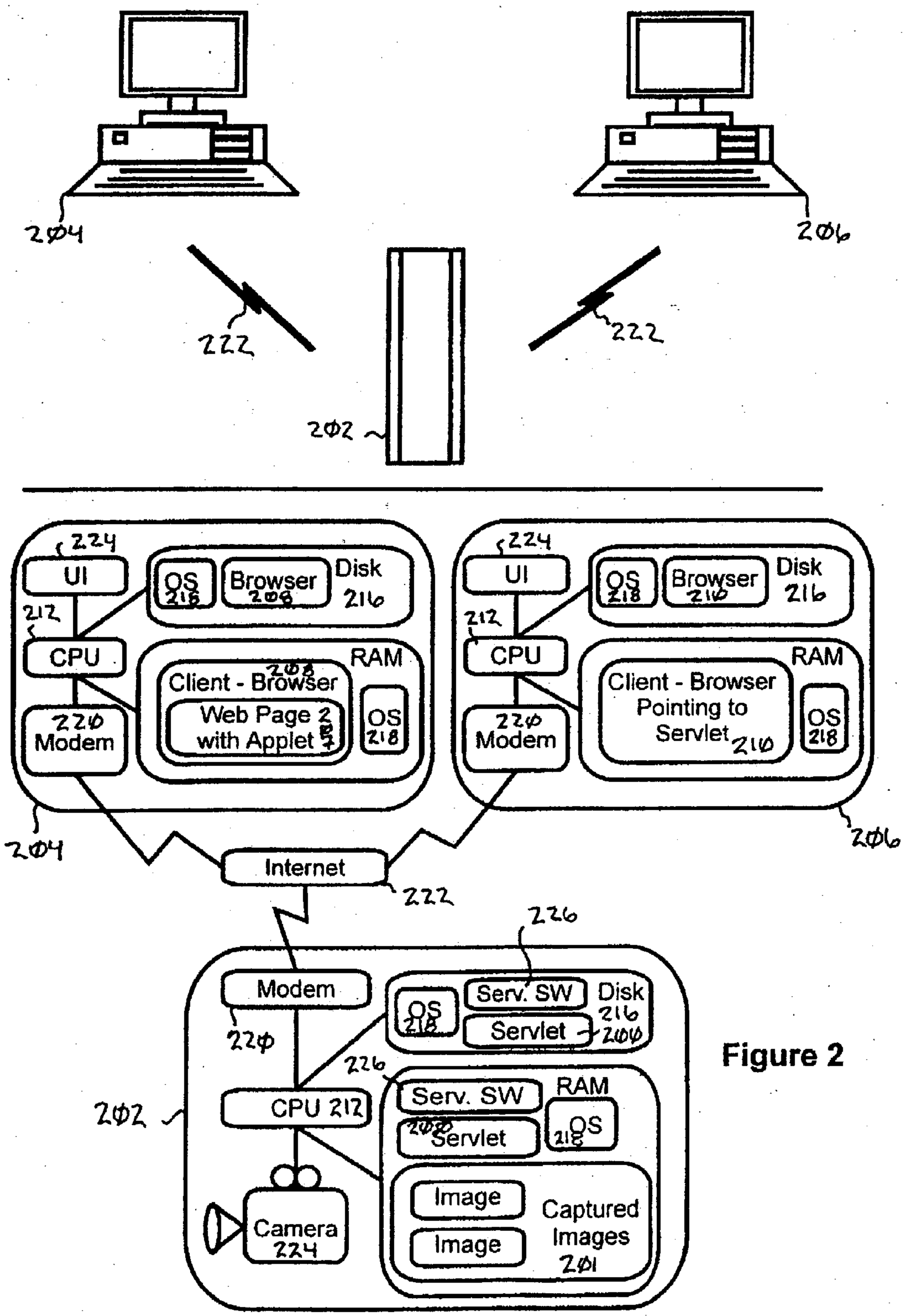

204
206
222
222
202
224
UI
212
CPU
220
Modem
OS
216
Browser
208
Disk
216
RAM
Client - Browser
208
Web Page 210
with Applet
OS
218
224
UI
212
CPU
220
Modem
OS
218
Browser
210
Disk
216
RAM
Client - Browser
Pointing to
Servlet 210
OS
218
204
Internet
222
226
Modem
220
OS
218
Serv. SW
Servlet
Disk
216
204
226
202
CPU 212
Serv. SW
208
Servlet
RAM
OS
218
Camera
224
Image
Image
Captured
Images
201

Figure 2

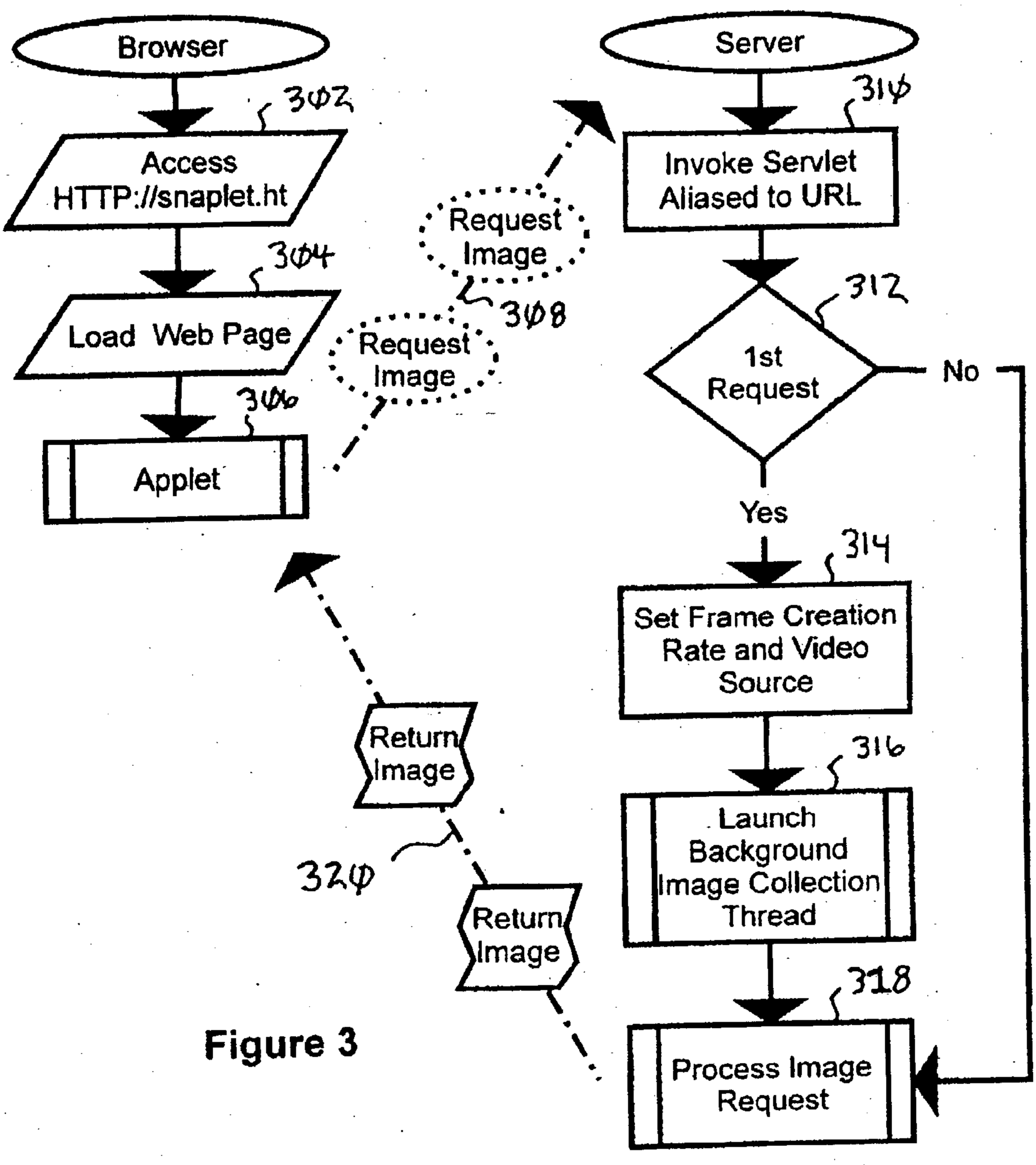

Browser
302
Access HTTP://snaplet.ht
304
Load Web Page
306
Applet
Request Image
308
Request Image
Server
310
Invoke Servlet Aliased to URL
312
1st Request
No
Yes
314
Set Frame Creation Rate and Video Source
316
Launch Background Image Collection Thread
318
Process Image Request
320
Return Image
Return Image

Figure 3

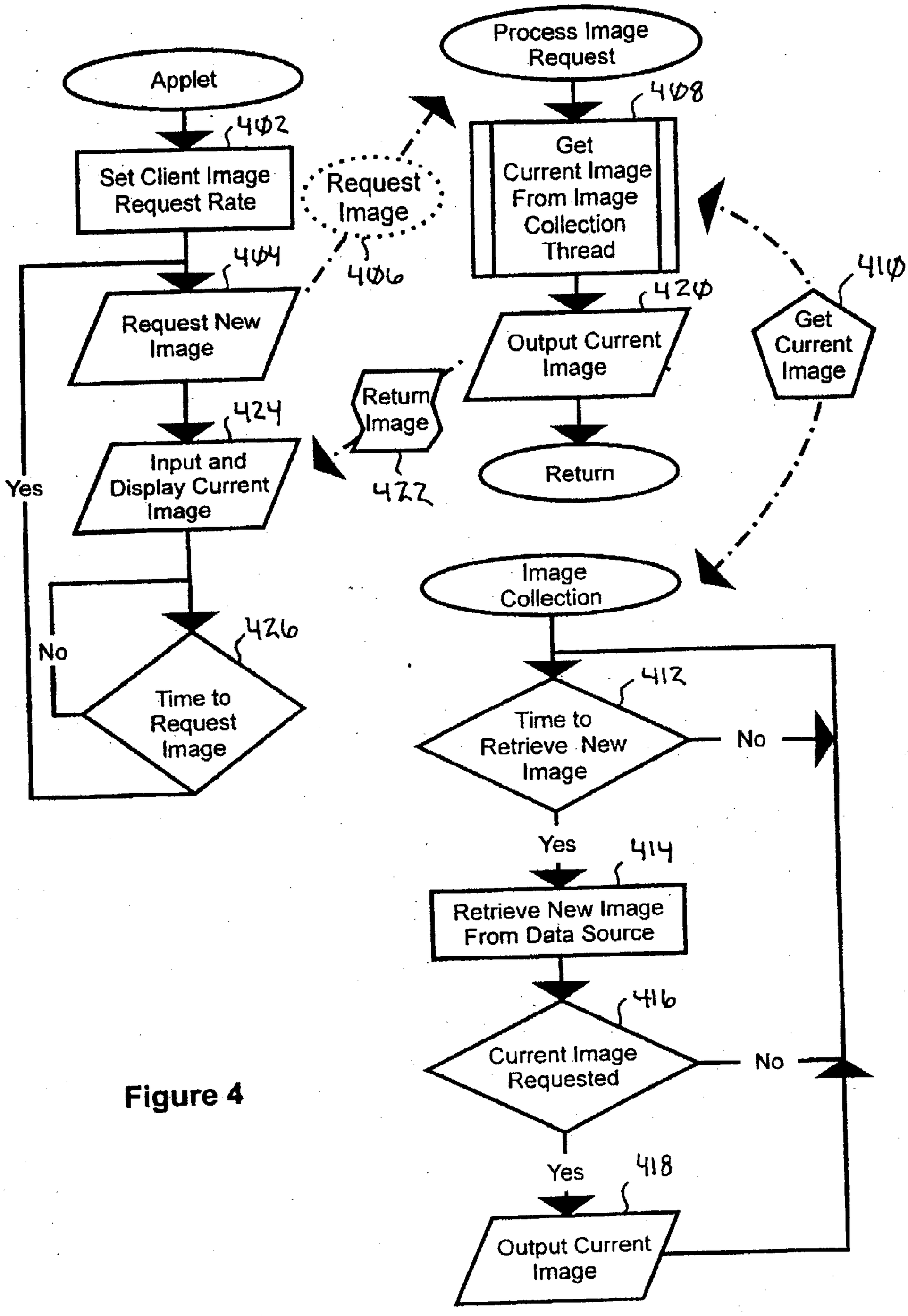

Applet
402
Set Client Image Request Rate
404
Request New Image
424
Input and Display Current Image
426
Time to Request Image
No
Yes
Request Image
406
Return Image
422
Process Image Request
408
Get Current Image From Image Collection Thread
420
Output Current Image
Return
Get Current Image
410
Image Collection
412
Time to Retrieve New Image
No
Yes
414
Retrieve New Image From Data Source
416
Current Image Requested
No
Yes
418
Output Current Image

Figure 4

METHOD AND SYSTEM FOR DELIVERING DATA FROM A SERVER OBJECT TO A CLIENT OBJECT USING A NON-PROPRIETARY DATA TRANSFER PROTOCOL

BACKGROUND OF THE INVENTION

[0001] 1. Field of the Invention

[0002] The present invention relates generally to the computerized delivery of data from a server object to a client object, and more specifically to a method and system for sequentially delivering video data to network clients while adhering to the hypertext transfer protocol.

[0003] 2. Related Art

[0004] With the increased popularity of the Internet, a strong need has developed to provide a method and system for delivering video data across the World Wide Web. Developing such technology is difficult, however, because the hypertext transfer protocol, which defines the standard for delivering Web data, does not provide a protocol for the sequential delivery of images. As a result, a number of proprietary protocols have been developed. But from these, no single protocol has emerged as the de facto standard.

[0005] The reasons for this are many, but perhaps most important is that the computers connected to the Web subscribe to a variety of platforms. These differing platforms may include different operating systems, such as those promulgated and trademarked by UNIX, SUN MICROSYSTEMS (e.g., SOLARIS), IBM (e.g., 0S/2), MICROSOFT (e.g., WINDOWS NT, WINDOWS 95, WINDOWS 3.1) and APPLE (e.g., MacOS, OS8). These operating systems have varying application program interfaces. Thus, video delivery software must be individually tailored to each API to run on the associated operating system—a costly and time consuming process.

[0006] Another impediment to the development of a consistent method of video delivery is that the computers connected to the Web vary greatly in terms of processing power. This variance is problematic because a protocol that works well with powerful computers may yield unsatisfactory results with computers having more limited power.

[0007] The prior art has not even attempted to create a universal delivery system. Instead, the most common approaches cater to computers that have substantial processing power or that are run under the more common operating systems. One such approach requires the installation of a "plug-in." This is illustrated in **FIG. 1**, which depicts a client computer **100** connected to a server computer **102** over the Internet **104**. As shown, the computers connect to the Internet **104** via modems **106**, although those skilled in the art will recognize that other forms of communication are possible, such as wireless hookups.

[0008] In addition to the modem **106**, the client computer **100** has the standard array of components: a central processing unit **108**, a user interface **110**, which typically consists of a keyboard, a mouse, and a monitor; a primary memory **112**, such as a random access memory, for program execution; and a secondary memory **114**, such as a disk, for storing programs and data that are not immediately needed for execution. As illustrated, the server computer **102** contains the same array of basic computer components **108**, **110**, **114**, with the exception of the user interface **110**. Those skilled in the art will appreciate, however, that the server computer **102** could also possess such an interface.

[0009] The client and the server computers **100**, **102** also each have an operating system **118** that is stored in the secondary memory **114** and loaded into the primary memory **112** to manage the respective computer's resources. For example, with respect to the client computer **100**, the operating system **118** provides an application program interface that allows programs, such as the browser **120**, to interact with the modem **106**. so that communication may be made with the server computer **102**. Similarly, the programs of the server computer **102**, such as the proprietary video server **122**, access the server computer's operating system **118** to communicate to the client computer **100**.

[0010] The plug-in method requires the server computer **102** to install the proprietary video server **122** and requires the client computer **100** to install a proprietary plug-in **124**. The proprietary server **122** retrieves video data **126** from the disk **114** of the server computer **102** and downloads it to the client computer **100**. The plug-in **124** is then invoked on the client side to display the downloaded data **128**. Plug-ins can be installed as stand alone programs, or, as shown in **FIG. 1**, can be installed "in-line" in a web browser **120**. In advanced systems, the downloaded data **128** will be displayed using a process known as "streaming," which is a compression and buffering technology that allows the client computer **100** to display data in near real-time. If the server **102** is incapable of streaming data, the plug-in **124** displays the data **128** after it has been downloaded.

[0011] While the plug-in technique provides a capable method of delivering video data in certain environments, the technique has several disadvantages. First, it requires the purchase of the proprietary server **122** and associated development tools to even create the video data **126**. And because this data **126** is designed for use with a proprietary plug-in **124**, it often does not conform to formats that are viewable under the hypertext transfer protocol. Thus, only those who have purchased the plug-in **124** can view the created video **126**, **128**. This purchase cost frequently represents more than a one-time investment because the proprietary protocol may require constant upgrades. Also, a newly emerging protocol may displace an old one, requiring the user to start anew.

[0012] Even if purchase costs do not represent a significant impediment to broad deployment of the media, other obstacles do. The installation of the plug-in **124** is often a complex procedure, and one made particularly difficult if the data is to be delivered through a secure gateway, sometimes referred to as a "firewall." Plug-ins also frequently require an elaborate infrastructure, such as repeaters, and can require more processing power than the client computer **100** can deliver. When this problem arises, the results can be merely unsatisfactory or nearly catastrophic. For example, when the modem **106** provides a limited bandwidth, the time required to download the video data may be so great that the user will abort the process in frustration. When the CPU **108** operates at a relatively low processing speed, or when the client computer **100** contains limited RAM **112**, the video may appear as a disjointed sequence of images or may be displayed so slowly that the client computer **100** appears to have ceased its processing.

[0013] Attempts to improve upon the plug-in system have been only marginally successful. For example, one attempt

has the client object sequentially accessing image files stored on the server's disk. In one version of this approach, the client object requests the image files at a rate independent of that which the server creates them. In this scheme, the client computer could request image files at a rate of two frames per second while the server computer retrieved images from a camera and stored them to disk at a rate of three frames per second. It was hoped that this rate decoupling would provide for efficient delivery because the client could control its use of network bandwidth. Unfortunately, the technique introduced fatal timing errors: Because the client requested data at a rate independent of the server, requests for image files occurred while the data server was updating the needed image, e.g., when the server was storing or retrieving images to and from the disk. As a result, the data server would return a "File not Found" message that caused some servers to permanently suspend the exchange of data. In colloquial terms, this is known as "crashing" and, next to the destruction of data, is the most undesirable result that a communication error can produce.

[0014] To avoid such errors, some methods deliver video data using casting technologies, such as the "PUSH" hypertext transfer protocol extension that is proprietary to NETSCAPE. This extension allows for the delivery of data across a network channel at a speed dictated by server software. While this can be efficient for some systems, it is inadequate for those with limited processing power because the client computer cannot reduce the rate at which it must receive the data. This can be particularly problematic in cases where the receiving computer is connected to a local area network. In this case, the receiving computer can monopolize the network's bandwidth, which can cause other computers connected to the LAN to experience delays when accessing network resources. The PUSH approach is also not broadly deployable because, like the plug-in approach, it requires the purchase of proprietary software, i.e., proprietary server software. The deployability is further limited by the fact that PUSH and other webcasting technologies are proprietary HTTP extensions. As such, they are not supported by all web browsers, and therefore some browsers are simply unable to receive "pushed" data.

[0015] Thus, there has been a long standing need for technology that can reliably deliver video data to a client computers that operate on differing platforms and have varying degrees of processing power. There is also a compelling need to deliver the data using a manner that minimizes investments in proprietary software. The present invention satisfies these needs.

INVENTION SUMMARY

[0016] Embodiments of the invention deliver video data from a server object to numerous client objects without deviating from the standards set by the hypertext transfer protocol. These embodiments are therefore widely deployable because they do not require the client objects to install proprietary software. This deployability is further enhanced because the embodiments allow the client objects to control their use of network bandwidth by defining their respective data delivery rates. Thus, even client computers with limited processing power can efficiently utilize the invention.

[0017] One aspect of one of these embodiments is the decoupling of the client objects' data delivery rates from the server object's data collection rate. That is, while the server object delivers data at the rates requested by the client objects, it retrieves data at a rate independent from those delivery rates. By maintaining an inventory of the undelivered data in random access memory, the server object can quickly deliver the data to clients who are requesting it at differing rates.

[0018] An embodiment of the invention utilizes a servlet to extend the data delivery capabilities of a server computer. Client objects request to view data by accessing a Uniform Resource Locator that is aliased to the servlet. Because this URL is accessible to the client objects regardless of whether the servlet is retrieving new data or otherwise maintaining the undelivered data inventory, the invention provides for more reliable delivery than was before possible.

BRIEF DRAWING DESCRIPTIONS

[0019] The accompanying drawings illustrate embodiments of the invention that will enable those skilled in the art to make and utilize the invention. In fact, these drawings, when considered with the written disclosure, will enable those skilled in the art to modify these embodiments in a manner that does not depart from the spirit and scope of the invention.

[0020] **FIG. 1** illustrates a prior art server computer connected to a client computer via the Internet.

[0021] **FIG. 2** depicts a server computer that delivers data to two client computers in accordance with an embodiment of the present invention.

[0022] **FIG. 3** is a flow chart of the interactions between a browser program and a server object in an embodiment of the present invention.

[0023] **FIG. 4** is a flow chart of the interactions between an applet and two server threads in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

[0024] Embodiments of the invention will now be described with reference to the accompanying drawings. It is understood that the invention is not limited to these embodiments, as the teachings contained herein will enable others to modify these embodiments without departing from the spirit and scope of the invention. Accordingly, the invention encompasses all such modifications that are described by the appended claims and their equivalents.

[0025] One embodiment of the present invention is a method and system for delivering data from a server object to a client object. In **FIG. 2**, an embodiment is shown where the server object is a servlet **200** that delivers a series of video images **201** from a server computer **202** to two client computers **204**, **206**.

[0026] A servlet **200** is a dynamically loadable object that extends the functionality of the server computer **202**. The client computers **204**, **206** access the servlet **200** in related, but different ways. The first client computer **204** accesses the servlet **200** via an applet **207** that is embedded in a web page, which is viewed by a web browser **208**. The applet **207** is a program that is run within the browser **208**. Thus, the applet **207** is similar to the servlet **200** in that both extend the functionality of their host computers. The second client

computer **206** also accesses the servlet **200** via a web browser **210**, but does so by pointing the browser **210** to a uniform resource locator that is aliased to the servlet **200**. Although the client computers **204**, **206** depicted here utilize web browsers, those skilled in the art will appreciate that the invention could be deployed with any browsing technology objects, such as dynamically linked libraries that contain browsing like functions.

[**0027**] The server and client computers **202**, **204**, **206** each have the standard computer components for storing data and executing programs: namely, central processing units **212**, primary memory (e.g., RAM) **214**, secondary memory (e.g., a disk) **216**, and operating systems **218**. Those skilled in the art will appreciate that the present invention is not limited to any particular CPU **212** or processing technology. Nor is the invention limited to any particular operating system **218**, as the invention could be utilized in any operating environment, such as WINDOWS 95, WINDOWS 98, UNIX, MacOS, or any JAVA runtime environment. "JAVA runtime environment" refers to the operating environment typified by a JAVA virtual machine and associated JAVA class libraries. JAVA is a registered trademark of SUN MICROSYSTEMS, Inc.

[**0028**] The computers **202**, **204**, **206** also have modems **220** for communicating with each other over a network **222**. As pictured here, the network **222** is the Internet. Those skilled in the art will understand, however, that embodiments could be deployed across any wide or local area network. The clients have user interfaces **224** that are comprised of a keyboard, a mouse, and a monitor. Although not shown, those skilled in the art will realize that the server computer **202** could also be equipped with a user interface. The server computer **202** has a camera **224** that operates as a data source to produce the images **201** that are delivered by the servlet **200**. In another embodiment, a file of images stored in the secondary memory **216** of the server computer **202** constitutes the data source. The server computer **202** is also equipped with server software **226** that invokes the servlet **200**.

[**0029**] In one embodiment, the applet **207** and the servlet **200** are written in JAVA, to run in the JAVA runtime environment. The preferred server software **226** is the JAVASERVER, which like the JAVA runtime environment, was developed and trademarked by SUN MICROSYSTEMS. The JAVA runtime environment is preferable because of its platform independency: A compiled JAVA object can be executed on any system that runs the JAVA virtual machine, which is not dependent on any particular CPU, processing technology, or operating system. Consequently, a JAVA software developer can create code more efficiently than developers that code in other languages.

[**0030**] Most browsers run the JAVA virtual machine and therefore can execute JAVA applets. Thus, embodiments that utilize the JAVA applet **207** will be ideal for most systems. But this not to say that the applet **207** must be written in JAVA. On the contrary, because the servlet **200** is aliased as a URL, applets written in any language can access the servlet **200**.

[**0031**] The servlet **200** could be written in other languages as well, such as C++, or a language employing Common Gateway Interface Scripting. These alternatives to JAVA, however, are less than ideal due to their lack of portability and increased overhead. C++ for example tends to be more bug-ridden than JAVA, and CGI scripting tends to be more complex. CGI also requires the creation and maintenance of a CGI bin, which may require reconfiguration of the server computer. And for security reasons, access to the bin is often not granted to all server objects. Therefore, CGI scripting might not be a possibility for a particular developer. CGI scripting also imposes performance-based penalties that JAVA does not. More specifically, a JAVA servlet once launched is persistent: once it has been launched, it remains resident in memory to fulfill all of its requests. In contrast, CGI technologies generally require distinct processes to be built and released for each individual request. This repetitive construction greatly slows data delivery. And while some CGI extension technologies have attempted to eliminate this overhead, they are far less portable than JAVA servlets. This is because Java servlets are platform independent. The same cannot be said for CGI extensions, which generally must be separately compiled to run on differing platforms.

[**0032**] To illustrate the advantages of JAVA, consider the following experiment that was conducted using an embodiment of the invention. A video delivery servlet was created and installed on a standard desktop computer—not on a heavy duty server. Viewers from within SUN MICROSYSTEMS were asked to access the servlet's URL to receive video data, which the desktop captured from a video tuner at a rate of 5 frames per second. The clients used a maximum viewing rate of 5 frames per second; the frames comprised approximately 40 kilobytes. Under these conditions the desktop delivered over 100 kilobyte frames per second to more than ten clients. Of course, this delivery rate does not represent the pinnacle performance that the invention is capable of, given that a standard desktop computer was used as a server. Nonetheless, it does illustrate the invention's potential. Indeed, by utilizing the inventive properties disclosed herein, a 10-15 frame per second client viewing rate is possible. And as computers' processing power improves, this rate is likely to further increase.

[**0033**] Regardless of what language the servlet is written in or what server software **226** it interfaces with, the servlet **200** can be accessed with or without the applet **207**. And regardless of how it is accessed, the servlet **200** delivers images **201** to the client browsers **208**, **210** in a format compatible with the hypertext transfer protocol. Thus, this embodiment is widely deployable because it delivers video data to client objects without forcing them to purchase and install propriety viewing software. This embodiment also helps solve the problem of bandwidth adaptability while ensuring reliable data delivery. The fatal "File not Found" errors that plague other systems are avoided because the servlet **200** that the client objects **207**, **208**, **210** access to request video images **201** is aliased to a URL, which is a stable location indicator that can be reliably accessed. This is true regardless of whether the servlet **200** is creating a new image or otherwise updating the images **201**.

[**0034**] The interactions between the browser **208** and the server software **226** are shown in **FIG. 3**. The process begins when the browser accesses a webpage (i.e., snaplet.html) stored on the server computer **202** (**302**). The browser loads the web page, which contains the applet **207** (**304**). The browser then launches the applet **207**, which requests images by accessing the servlet's URL (**306**, **308**). In turn, the server software **226** invokes the servlet **200** to satisfy the

browser's **208** request **(310)**. If this is the first request, the server software designates the data source and the frame creation rate, or capture rate, before returning the current image to the browser **(312, 314, 316, 318, 320)**. On subsequent requests, the current image is returned immediately **(312, 318, 320)**. By making repeated requests to the servlet's URL, numerous images are delivered to the browser **(308, 318, 320)**.

[0035] The manner in which these images are returned is illustrated in detail in **FIG. 4**, which shows the interactions between the applet and the two server threads referenced by blocks **316** and **318**. Basically, the applet interfaces with a "Process Image Request" thread that in turn interfaces with a background "Image Collection" thread. More specifically, the Image Collection thread constantly retrieves new images from the data source **224**. The Image Collection thread delivers the current image to the Process Image Request thread upon demand. This latter thread then delivers the current image to the client applet. When more than one client requests images, a set of Process Image Request threads is created: Each thread of this set maintains current state information for its client. When all clients have viewed a particular image, the embodiment disposes of that image via JAVA's garbage collector.

[0036] The interactions between the applet **207** and the server threads begin when the applet **207** sets the rate at which it will request images from the servlet **(402)**. The applet **207** then requests an image by accessing the servlet's **200** URL **(404, 406)**. As explained with reference to **FIG. 3**, this first request launches the two server threads illustrated in **FIG. 4 (316, 318)**. The Process Image Request thread retrieves the current image by interfacing with the Image Collection thread **(408, 410)**. The Image Collection thread operates in the background, continuously retrieving current images from the date source at a rate defined by the image capture rate **(412, 414, 314)**. Images are output to the Process Image Request thread upon request **(416, 418)**. In turn, the Process Image Request thread returns the current image to applet **207 (420, 422)**. The image is then displayed on the client browser, and the process repeats itself **(424, 426)**. Because the images are passed in an HTTP format, a proprietary reader is not required.

[0037] An important aspect of an embodiment of the invention is the flexibility that is derived from the three independent image delivery rates that **FIG. 4** illustrates, i.e., the capture rate of the Image Collection thread, the service rate of the Process Image request thread, and fetch rate of the applet. Because the capture rate is independent of the fetch rates, the collection of images is not hampered by clients that lack processing power. Nor is it hampered by situations in which many clients are requesting images: the images are collected at the constant capture rate and are served to the clients at their various fetch rates.

[0038] Another advantage of an embodiment of the invention is that the images are maintained in server computer's RAM **202, 214**. By maintaining the images in RAM, the embodiment delivers images **201** two or three times faster than conventional systems that maintained images on disk.

[0039] An embodiment will now be described with reference to a particular implementation, i.e., that of a "VCR style" image delivery system. The implementation details that follow are not meant to limit the invention, but are included to provide the fullest possible disclosure.

[0040] As explained above, it is necessary to define the servlet's **200** image capture rate and data source **(314)**. The capture rate is typically set in the range of 2-4 frames per second, the chosen value being selected based on the server computer's **202** processing power and by the performance characteristics desired by the servlet's **200** developer. The data source specifies the location from which the servlet retrieves video frames. This parameter can be set to retrieve the images from a camera **224** or from a data file residing on the disk **216** of the server computer **202**. The rate and source parameters are specified in an initial arguments file that is referenced when the servlet is compiled. For example, a servlet compiled with the initialization arguments:

[0041] video=snapshots.JPEG

[0042] rate=4

[0043] would retrieve images from a file named "snapshots.JPEG" at a rate of 4 frames per second. Similarly, a servlet complied with the initialization arguments:

[0044] video=videocard|videoport|0.25|25

[0045] rate=2

[0046] would retrieve images from a videocard at the rate of 2 frames per second. In the above example, the parameters that follow the videoport are scale and image quality.

[0047] Once compiled, the servlet **200** is invoked by specifying its class name. For example, where the servlet **200** is named "SnapServlet," it would be invoked with the command:

[0048] http://server_host_name/servlet/SnapServlet

[0049] This URL can be invoked by simply pointing a browser **210** to the servlet **200**. Preferably, however, the URL is accessed by running an applet **207** from within a browser **208**. By invoking the Servlet in this fashion, images can be sequentially delivered with a minimal amount of user input: The browser **208** automatically executes the applet **207** when loading the web page **(304, 306, 308)**.

[0050] An example of applet syntax that causes the browser **208** to invoke the servlet in the "VCR style" is:

```
<applet code="Snapshot.class" width=400 height=300
    <param name=file        value=http://skunks.eng/servlet/SnapServlet|1.0|10>"
    <param name=rates       value=".25|.5|.75|1|2|4">
    <param name=scales      value=".5|1|1.5|2">
    <param name=controls    value="South">
    <param name=labels      value="Resume|Pause|Seconds|Scale">
</apptet>
```

[0051] In this example, the file parameter directly relates to the servlet. The remaining parameters configure the applet menus and buttons and specify where these buttons are placed relative to the video (i.e., South). For completeness, these parameters are explained here.

[0052] The "applet code" tag informs the browser **208** to invoke the applet denominated as "Snapshot.class." The width and height parameters define the size of the displayed image.

[0053] The "file" parameter specifies that the applet **207** is to be run using "http://skunks.eng/servlet/SnapServlet" as its file source. This file source is not a traditional image file, but a URL alias for the servlet **200**. This alias allows client computers **204**, **206** to access the servlet **200**, and hence the images **201**, even when the servlet **200** is updating the requested image. The numbers that follow the "file" parameter, "1" and "10," define the default image scale and the number of seconds between the client's image requests.

[0054] The "rates" parameter provides a list of user-selectable image viewing rates, in images per second. For example, the 0.25 rate sets the fetch rate as one frame for every four seconds. Thus, the client computer **204** is programmed to retrieve images **201** at a rate independent of that which the servlet **200** creates the images **201**. Because the rate is selectable, the client can control its use of network bandwidth.

[0055] The "scales" parameter provides a number of scaling sizes and is used to increase or decrease the size of the image. This feature provides flexibility while preserving bandwidth: small files that are transferred from the server can be enlarged by the browser. When the scales string is left empty, no scaling options are provided.

[0056] The "controls" parameter defines the location of "resume,""pause," and "scale" controls, as well as the seconds display. The "labels" parameter defines the labels for start, stop, rate, and scale buttons.

[0057] During execution, the applet **207** makes a series of calls to the servlet's URL (i.e., http.//skunks/.eng/servlet/SnapServlet) **(308, 406)**. The servlet **200** intercepts these requests, processes them, and returns the corresponding images **(318, 422)**. In this manner, the servlet **200** sequentially provides the browser **208** with images from the data source **224** (See **FIGS. 3 and 4**). The applet **207** and the servlet **200** continue their processing steps until the images are exhausted or until the user pauses or terminates the applet **207**.

[0058] The above description outlines the embodiment in which the browser **208** possesses JAVA capabilities. If the browser lacks these, it nevertheless can view the images **201** by pointing to the URL that is aliased to the servlet **200** (i.e., http://skunks.eng/servlet/SnapServlet). The servlet **200** intercepts the URL request and returns the next unviewed image to the browser **210**. Subsequent unviewed images are returned whenever the user depresses the browser's **210** refresh control, which triggers a new URL request.

[0059] Thus, with a single servlet **200**, the present invention provides a delivery system that can be used with all Web browsers, regardless of the client computers' restrictions in processing power and use of network bandwidth. Morever, because the servlet **200** creates a new Process Image Request thread for each client browser **208**, **210**, many browsers can simultaneously view the images **201**. The servlet **200** maintains in RAM **202**, **214** any image that has yet to be viewed by a browser **208**, **210**. When all browsers **208**, **210** have viewed a particular image, it is deallocated from the server's RAM **202**, **214**, preferably using JAVA's garbage collector.

[0060] As the inventive properties disclosed herein can be employed in a variety of contexts, the scope of the invention should not be limited to the particular embodiments described here. For example, although the above embodiments were described with reference to an image format that complies with the HTTP protocol, i.e., JPEG. Those skilled in the art will appreciate, however, that the invention can be employed with any data format that adheres to this protocol, e.g, HTML text file, Multi-purpose Internet Mail Extension, and Internet audio.

[0061] Similarly, although the servlet and server software were described as present in RAM and on a hard disk, those skilled in the art will recognize that the server objects could be stored and invoked from any media that can store data or have data read from it. Examples of such media include floppy disks, magnetic tapes, phase discs, carrier waves sent across a network, and various forms ROM, such as DVDs and CDs. Thus, the present invention anticipates the use of all computer readable media.

[0062] The inventive properties could also be employed to effectuate a video conference. Each party's local computer would remotely run an applet that references a servlet tied to the other party's data source, typically a camera.

[0063] Further, the invention could be used for security purposes. Surveillance over an area could be maintained by installing an camera at a web site and using the servlet to distribute data to a wide area.

[0064] Those skilled in the art will also appreciate that the above invention is not limited to the delivery of images or text that creates an illusion of movement. For example, the technology disclosed herein could also be used to create a facsimile server.

[0065] The above detailed description will enable those skilled in the art to make numerous modifications to the described embodiments without departing from the spirt and scope of the claimed invention. Indeed, the chosen embodiments were selected so others could best utilize the invention by making such modifications to tailor the invention to their particular needs. The description therefore should not be read as limiting the invention to the embodiments explained herein. Rather, the scope of the invention is intended to be defined by the appended claims and their equivalents.

What is claimed is:

1. In a computerized system, a method for serving, to a client, video data objects from a plurality of video data objects, the method comprising:

providing a uniform resource locator that is aliased to a servlet so that when the uniform resource locator is accessed, the servlet is invoked;

receiving, from the client, a first call to the uniform resource locator;

upon receipt of the first call, invoking the servlet to select a first video data object from the plurality of video data objects;

returning the selected first video data object to the client;

receiving, from the client, a second call to the uniform resource locator;

upon receipt of the second call, invoking the servlet to select a second video data object from the plurality of video data objects, the second video data object being ordered after the first video data object of the plurality of data objects; and

returning the selected second video data object to the client.

2. The method of claim 1 wherein the video data objects are of a format capable of being delivered using the hypertext transfer protocol.

3. The method of claim 1, further comprising:

retrieving the first video data object from a data source;

storing the retrieved first video data object in a random access memory;

retrieving the second video data object from the data source;

storing the retrieved second video data object in the random access memory; and

after storing the first and the second video data objects in the random access memory, receiving the first call from the client.

4. The method of claim 3, further comprising:

allowing the client to select a rate at which the client calls the uniform resource locator.

5. A computer system for serving data objects to a client object, the computer system comprising:

a server object that is accessible as a location indicator using a non-proprietary communications protocol, the server object being configured to create a grouping of data objects by retrieving a plurality of data objects from a data source at a rate that is independent of a rate at which the client object requests the plurality data objects, and the server object being further configured to sequentially deliver, to the client object, each of the plurality of data objects that comprise the grouping of data objects.

6. The computer system of claim 5 wherein the data objects are graphical images and the data source is a file comprised of a plurality of the graphical images.

7. The computer system of claim 5 wherein the data source is a video camera.

8. The computer system of claim 7 wherein each of the data objects is a graphical image.

9. The computer system of claim 5 wherein the non-proprietary communications protocol is a hypertext transfer protocol and wherein the location indicator of the server object is a uniform resource locator.

10. The computer system of claim 5 wherein the data objects represent facsimile pages.

11. The computer system of claim 5 wherein the server object is a JAVA servlet that is aliased a uniform resource locator.

12. The computer system of claim 5 wherein the client object is a JAVA applet.

13. The computer system of claim 5 wherein the client object is a web browser configured to access the server object by accepting the server object's location indicator as a uniform resource locator.

14. A computer-readable medium containing instructions for delivering images to a client object, the computer-readable medium comprising:

instructions which, when invoked, receive a call that the client object makes in an attempt to retrieve an image, the reception being accomplished by having a server process aliased to an location indicator of the image;

after intercepting the call, instructions which, when invoked, select an image from a plurality of images; and

instructions which, when invoked, transmit the selected image to the client object using a non-proprietary communications protocol.

15. The computer-readable medium of claim 14 wherein the images are sequentially retrieved and stored in a memory of a computer system at a rate that is independent of a rate at which the client object requests the images.

16. The computer-readable medium of claim 14 wherein the images are blocks of video data that are sequentially stored in a random access memory of a computer system.

17. The computer-readable medium of claim 16 wherein the blocks of video data are retrieved from a port of the computer system that is configured for use with a video camera.

18. The computer-readable medium of claim 16 wherein the blocks of video data are retrieved from a video file.

19. In a computer system, a method for delivering video data from a client object to a server object, the method comprising:

sending, from the client object to the server object, a request to view a video image from a series of video images; and

upon receipt of the request, launching a background image collection thread that retrieves the series of video images from a video source, and further launching a process image request thread that interfaces with the background image collection thread to deliver the series of video images to the client object.

20. The method of claim 19 wherein the client object's request is made by accessing a resource locator that is aliased to a servlet, the method further comprising:

when the resource locator is accessed, launching the servlet to select, from the series of video images, a video image that the client has not yet viewed; and

returning the selected video image to the applet.

21. The method of claim 20 wherein the client object sends a plurality of requests to the server object, each request asking to view a current video image from among the series of video images, wherein the background image collection thread retrieves each video image from the series of video images at a rate that is independent of a rate at which the client object sends its requests, and wherein the process image request thread responds the client object's requests by retrieving the current video image from the background image collection thread.

22. The method of claim 20 wherein the servlet stores the series of video images in a random access memory.

23. A computerized system for serving data objects to a client, the system comprising:

means for providing a location indicator that is aliased to a servlet so that when the location indicator is accessed, the servlet is invoked;

means for receiving, from the client, a call to the location indicator;

upon receipt of the call, means for invoking the servlet to select a data object; and

means for returning the selected data object to the client.

24. The system of claim 23 wherein the servlet selects the data object with the aid of a first processing thread and a second processing thread, the first processing thread collecting data objects from a data source, and the second processing thread interfacing with the first processing thread to determine the data object that is needed to be delivered to the client.

25. The system of claim 24 wherein the second processing thread also interfaces with the client by providing the means for receiving, from the client, the call to the location indicator.

26. The system of claim 23 wherein the data objects are of a format capable of being delivered using the hypertext transfer protocol.

27. The system of claim 23, further comprising:

means for receiving, from the client, a second call to the location indicator;

upon receipt of the second call, means for invoking the servlet to select a second data object, the second data object being ordered after the first data object from among a plurality of data objects; and

means for returning the selected second video data object to the client.

* * * * *