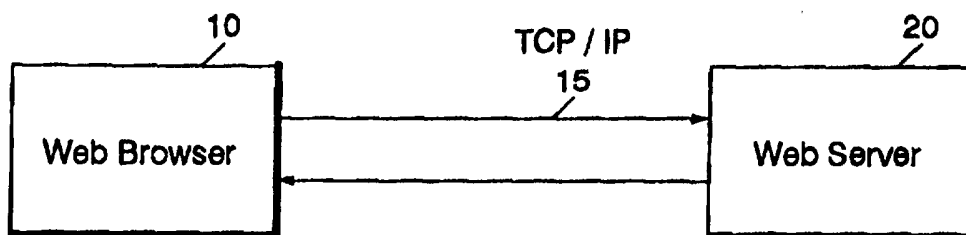




## INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<b>(51) International Patent Classification <sup>6</sup> :</b> <b>H04L 29/06</b>	<b>A2</b>	<b>(11) International Publication Number:</b> <b>WO 97/30538</b> <b>(43) International Publication Date:</b> 21 August 1997 (21.08.97)
<b>(21) International Application Number:</b> PCT/US96/11551 <b>(22) International Filing Date:</b> 11 July 1996 (11.07.96) <b>(30) Priority Data:</b> 08/601,804                      15 February 1996 (15.02.96)      US <b>(71) Applicant (for all designated States except US):</b> IBM CORPORATION [US/US]; 972/B656, P.O. Box 12195, Research Triangle Parc, NC 27709 (US). <b>(71)(72) Applicants and Inventors:</b> BITTINGER, Reed, Richard [US/US]; 2712 Salisbury Plain, Raleigh, NC 27613 (US). FRAENKEL, Michael, Levi [US/US]; 3013-23 Inland Trail, Raleigh, NC 27613 (US). HOUSEL, Barron, Cornelius [US/US]; 702 Kensington Drive, Chapel Hill, NC 27514 (US). LINDQUIST, David, Bruce [US/US]; 4001 Lake Springs Court, Raleigh, NC 27613 (US). <b>(74) Agent:</b> HERNDON, Jerry, W.; IBM Corporation, 972/B656, P.O. Box 12195, Research Triangle Park, NC 27709 (US).	<b>(81) Designated States:</b> BR, CA, CN, CZ, ES, HU, JP, KR, PL, US, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).  <b>Published</b> <i>Without international search report and to be republished upon receipt of that report.</i>	

**(54) Title:** CLIENT/SERVER COMMUNICATION SYSTEM

**(57) Abstract**

A method, apparatus and program product for increasing the performance of a client/server system having a client application resident on a first computer and communicating with a server application resident on a second computer remote from the first computer. The client application and the server application utilize a client/server independent communication protocol for communication between the client and the server and at least one segment of the communication between the client application in the first computer and the server application in the second computer occurs over an external communication link. The method, apparatus and program product intercept communications in the client/server independent communication protocol originated by the remote client prior to transmission of the communications on the external communication link and convert the communication originated by the client to a second client/server specific communication protocol. The converted communication is transmitted over the external communication link and received from the communication transmitted over the external communication link. The communication received over the external communication link is converted from the client server specific communication protocol to the client/server independent communication protocol and the communication originated by the remote client is provided to the server in the client/server independent communication protocol. Differencing, caching or protocol reduction techniques increase performance over the external communication link. The applications may be a web browser and a web server and may communicate over a wireless communication link.

**FOR THE PURPOSES OF INFORMATION ONLY**

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	GB	United Kingdom	MW	Malawi
AT	Austria	GE	Georgia	MX	Mexico
AU	Australia	GN	Guinea	NE	Niger
BB	Barbados	GR	Greece	NL	Netherlands
BE	Belgium	HU	Hungary	NO	Norway
BF	Burkina Faso	IE	Ireland	NZ	New Zealand
BG	Bulgaria	IT	Italy	PL	Poland
BJ	Benin	JP	Japan	PT	Portugal
BR	Brazil	KE	Kenya	RO	Romania
BY	Belarus	KG	Kyrgyzstan	RU	Russian Federation
CA	Canada	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic	KR	Republic of Korea	SE	Sweden
CG	Congo	KZ	Kazakhstan	SG	Singapore
CH	Switzerland	LI	Liechtenstein	SI	Slovenia
CI	Côte d'Ivoire	LK	Sri Lanka	SK	Slovakia
CM	Cameroon	LR	Liberia	SN	Senegal
CN	China	LT	Lithuania	SZ	Swaziland
CS	Czechoslovakia	LU	Luxembourg	TD	Chad
CZ	Czech Republic	LV	Latvia	TG	Togo
DE	Germany	MC	Monaco	TJ	Tajikistan
DK	Denmark	MD	Republic of Moldova	TT	Trinidad and Tobago
EE	Estonia	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	UG	Uganda
FI	Finland	MN	Mongolia	US	United States of America
FR	France	MR	Mauritania	UZ	Uzbekistan
GA	Gabon			VN	Viet Nam

## Client/Server Communication System

### Field of the Invention

The present invention relates to communications between client server applications such as a web browser and a web server. More particularly the present invention relates to communications over a low-speed or wireless communication link between two computers, one running a client application and the other running a server application.

### Background of the Invention

The recent publicity and emphasis on the "information superhighway" has increased awareness and acceptance of the Internet as a mass communication media. This broad based recognition of the Internet as a viable media for communication and interaction across multiple networks has also created a large established user base built upon the Internet standardized protocols for interaction between computer networks.

The paradigm for the Internet is that of a client-server relationship where Internet clients (browsers) communicate with Internet servers. To provide greater access to the Internet the communication protocols and languages utilized by the clients and servers have become standardized. These protocols include the Hyper-Text Transfer Protocol (HTTP), which is the communication protocol used for communications between clients and servers, and the Transfer Control Protocol/Internet Protocol (TCP/IP) the TCP portion of which is the transport specific protocol for communication between computers or applications. Also standardized is the language in which clients and servers communicate which is called Hyper-Text Markup Language (HTML). Because these protocols and language are machine independent, and utilize a connection-less best-efforts protocol to sending information, each transaction is fully self contained. Thus, for example,

each message from a client contains information about the capabilities of the browser and is independent of any other communications for the communication to be completed. This self-contained nature of the communications between a client and a server may be referred to as "stateless" communications and increases the amount of data which must be transferred between a client and a server for a given communication.

In the context of the World Wide Web client/server applications the client may be a web browser which acts as the user interface. The web browser sends user requests to the appropriate web server and formats and displays the HTML data returned from the web server. The web browser also evaluates the HTML data to determine if there are any embedded hyper-link statements in the HTML data which would require subsequent browser requests which would then be initiated by the browser. A web server acts as the server for the client and processes the web browsers requests and returns the requested response as an HTML data portion of a HTTP data stream.

As an example of a typical world wide web communication the case of a web browser initiating a request for a "home page" from the web server illustrates the basic relationship between HTTP, HTML, TCP and the web browser and server. When the user of the web browser requests information from a specific web site, the web browser initiates communication with the web server by sending a "get" request to the web server specifying the Universal Resource Locator (URL) of the desired web site which, for purposes of this example, may be a "home page." The URL acts as the address of the web site and is unique throughout the Internet. The web server would then obtain and supply the web browser with the HTML data corresponding to the home page specified by the URL. This operation may involve further communications on the Internet by the Internet web server or the URL may specify the server which is in the

local network to which the browser is attached. The web browser would then evaluate the HTML data received as an HTTP data stream from the web server to see if there were any embedded hyper-links such as an icon or an image and, if such a hyper-link exists would initiate requests specifying the URL of the hyper-link to obtain the specified data. This data would then be incorporated into the home page and displayed to the user. As is seen in this simple example, a single user input request by a web browser may result in multiple additional requests which are automatically carried out by the web browser in response to the receipt of the HTML data corresponding to the user input request.

The basic communication structure for an Internet based system is depicted in **Figure 1**. In **Figure 1** a web browser 10 communicates with a web server 20 over a communication link 15. This communication link is typically a local area network connection, wide area network connection a connection over telephone lines or a combination of connection methods. The web browser 10 communicates with the web server 20 using TCP/IP. For the majority of Internet communications a web browser communicates with a web server using the generic communication protocol HTTP which is transmitted between the web browser and the web server over the TCP/IP link between the web browser and the web server. The actual data transferred between the web browser 10 and the web server 20 are HTTP data objects (e.g. HTML data) as described above. The web server 20 may be a proxy which receives web browser communications from a number of web browsers and routes them to the appropriate server.

The popularity of the web browser/web server and their common information and transport protocols, HTML and HTTP, has lead to rapid acceptance of web technology as a universal interface for network access to information. Furthermore, because the protocols and language for communication between web browsers and web servers are

standardized the communication protocols and language will be the same whether a user is using Netscape Navigator™, NCSA Mosaic™, WebExplorer™ or any other web browser as their web browser to access network information. Therefore, the large installed user base for web browsers combined with the connectivity of the Internet and the ease of writing web application servers using the HTTP defined Common Gateway Interface (CGI) make web technology very attractive for a large class of forms-based applications.

At the same time that the Internet was growing in popularity and acceptance, mobile computing was also increasing in popularity. The use of laptops, notebooks, Personal Digital/Communication Assistants (PDAs/PCAs) and other portable devices has led to an increase in demands for wireless communications. Wireless wide area networks, cellular communications and packet radio, however, suffer from common limitations if used in a web context. The high cost per byte of communications, slow response time, low bandwidth and unreliability all hamper use of wireless technology for the stateless communication protocol of the World Wide Web. Also, because the web protocol is stateless the amount of data per request and the number of requests transferred over the wireless connection are larger than would be necessary if the communication were not self contained. Thus, combining wireless technology, or any low-speed communication technology, with web technology seems impractical as the strength of the web technology in its universal nature exacerbates the weaknesses of the wireless technology.

#### Objects and Summary of the Invention

In view of the above limitations it is one object of the present invention to take advantage of the installed user base of World Wide Web technology in a low speed communication environment such as wireless communications.

It is a further object of the present invention to use existing communication protocols and languages in a low

speed or wireless communication system without requiring modification of web browser or web server applications.

It is an additional object of the present invention to provide a method of communicating across an external  
5 communication link which reduces the amount of communication required and thereby enhances performance of the communication system.

In view of these and other objects, the present  
10 invention provides a method of increasing the performance of a web browser application resident on a first computer and communicating using the Hyper-Text Transfer Protocol (HTTP) with a web server application resident on a second computer remote from the first computer. At least one segment of the  
15 communication between the web browser application in the first computer and the web server application in the second computer occurs over an external communication link. One embodiment of the method of the present invention includes intercepting the HTTP data stream corresponding to a communication originated by the web browser prior to  
20 transmission of the HTTP data stream on the external communication link. The intercepted HTTP data stream originated by the web browser is converted from the HTTP protocol to a client/server specific communication protocol and the converted web browser originated communication is  
25 transmitted to the second computer over the external communication link as a client/server specific data stream. The second computer receives the client/server specific data stream transmitted over the external communication link and reconstructs the HTTP data stream corresponding to the  
30 communication from the web browser from the client/server specific data stream received over the external communication link by converting the client/server specific data stream received in the client/server specific communication protocol to an HTTP data stream. The  
35 communication originated by the web browser is provided to the web server as an HTTP data stream.

In a further embodiment of the present invention the converted web browser originated communication is transmitted over a wireless communication link.

5 An additional aspect of the present invention utilizing a cache resident in the first computer further includes storing a HTTP data stream to be received by the web browser in response to a web browser originated communication with the web server in a cache resident in the first computer to create a client cache entry corresponding to the web browser  
10 originated communication with the web server. Web browser originated communications are interrogated to determine if a client cache entry exists corresponding to the web browser originated communication. If a client cache entry exists corresponding to the web browser originated communication  
15 then the client cache entry is supplied to the web browser as an HTTP data stream in response to the web browser originated communication.

A further aspect of the present invention includes storing user defined information associated with a specific  
20 web browser information request so as to provide the user defined information resident in the first computer. Web browser originated communications are interrogated to determine if the information requested by the web browser corresponds to a request for which user defined information  
25 is stored and the user defined stored information is provided to the web browser as an HTTP data stream in response to the web browser originated communication if the web browser communication corresponds to a request for which user defined information is stored.

30 Additional embodiments of the present invention utilizing time coherent caching include storing the time of creation of a client cache entry to create a client cache entry time record and evaluating the client cache entry time record to determine if the client cache entry corresponding  
35 to the web browser originated communication was created within a predetermined client coherency time interval prior



to the web browser requesting the information. The client cache entry is provided to the web browser as an HTTP data stream in response to the web browser originated communication if the client cache entry was created within a  
5 predetermined client coherency time interval prior to the web browser requesting the information. Client cache entries may also be maintained across multiple instances of the web browser.

Additional embodiments of the present invention include  
10 a server cache where the HTTP data stream received from the web server in response to a browser originated communication is stored in a cache resident in the second computer to create a server request cache entry. The web browser originated communication is interrogated to determine if a  
15 server request cache entry corresponding to the browser originated communication has been previously stored in the cache. If a server cache entry exists then the server cache entry associated with the browser originated communication is converted to a client/server specific communication  
20 protocol and sent to the first computer over the external communication link as a client/server specific data stream. The first computer acquires the client/server specific data stream transmitted over the external communication link and reconstructs the HTTP data stream corresponding to the  
25 server request cache entry from the client/server specific data stream received over the external communication link by converting the client/server specific data stream received in the client/server specific communication protocol to an HTTP data stream. The server request cache entry is  
30 corresponding to the information requested by the web browser is provided to the web browser as an HTTP data stream.

A further embodiment of the caching aspect of the present invention includes determining if a server request  
35 cache entry corresponding to the web browser originated communication was created within a predetermined client

coherency time interval prior to the second computer receiving the web browser originated communication and transforming the server request cache entry which corresponds to the web browser originated communication to a client/server specific communication protocol if the server request cache entry was created within the predetermined client coherency time interval. A further embodiment would include determining if a client cache entry exists corresponding to the web browser originated communication which is identical to a server cache entry corresponding to the web browser originated communication. The interval of time between when the second computer received the web browser originated communication and when the server request cache entry corresponding to the web browser originated communication was created is calculated to provide entry age data. A coherent entry response is sent to the first computer which includes the entry age data for the server cache entry corresponding to the web browser originated communication over the external communication link if the two cache entries are identical. The client cache entry time record corresponding to the web browser originated communication is updated by subtracting from the current time of the first computer the entry age data received from the coherent entry response.

In an embodiment of the present invention which includes differencing to reduce data on the external communication link, the web browser originated communication corresponds to a Common Gateway Interface (CGI) request. The intercepted CGI request is interrogated to determine if a client base cache entry corresponding to the intercepted CGI request exists to provide a client CGI base form. The intercepted CGI request is also interrogated to determine if a server base cache entry corresponding to the intercepted CGI request exists to provide a server CGI base form. A HTTP data stream received from the web server in response to a browser originated communication which is a request to a

Common Gateway Interface (CGI) is stored in a cache resident in the second computer to create a server base cache entry and a HTTP data stream to be provided to the web browser in response to a browser originated communication which is a request to a CGI is stored in a cache resident in the first computer to create a client base cache entry. The HTTP data stream corresponding to the communication originated by the web server in response to the intercepted CGI request is intercepted prior to transmission of the HTTP data stream on the external communication link. The intercepted CGI response is compared to the server CGI base form to provide CGI difference data corresponding to the difference between the intercepted CGI response and the server CGI base form and the CGI difference data is sent to the web browser of the first computer over the external communication link as a client/server specific data stream. The client/server specific data stream transmitted over the external communication link by the second computer is acquired by the first computer and the HTTP data stream corresponding to the communication from the web server is reconstructed from the client/server specific data stream received over the external communication link by combining the client CGI base form with the CGI difference data received over the external communication link to create an HTTP data stream corresponding to the intercepted CGI response. The communication originated by the web server is provided to the web browser as an HTTP data stream.

A rebasing embodiment of the differencing aspect of the present invention determines if the server CGI base form is identical to the client CGI base form and transmits the server CGI base form and the CGI difference data to the web browser over the external communication link as a client/server specific data stream. The HTTP data stream corresponding to the communication from the web server is reconstructed from the client/server specific data stream received over the external communication link by combining

the server CGI base form with the CGI difference data received over the external communication link to create an HTTP data stream corresponding to the intercepted CGI response. The client CGI base form corresponding to the CGI request is updated by storing the received server CGI base form as the client base cache entry corresponding to the CGI request.

The server CGI base form may also be updated in a further aspect of the present invention by determining if the difference between the server CGI base form and the CGI response is greater than a predefined difference threshold. The server CGI base form corresponding to the CGI request is updated by storing the CGI response received from the web server as the server base cache entry corresponding to the CGI request if the CGI difference data and the comparison then utilizes the updated server CGI base form which is also sent to the first computer.

In a protocol reduction aspect of the present invention a persistent connection is established between the first computer and the second computer over the external communication link. The persistent connection is maintained until all web browser originated communications are completed. A plurality of web browser originated communications are intercepted and multiplexed onto the external communication link while the persistent connection is maintained. The client/server specific data stream may then be demultiplexed to create a plurality of HTTP data streams and the plurality of HTTP data streams are provided to the web server.

Header reduction aspects of the present invention include providing to the second computer over the external communication link computer specific information corresponding to predefined characteristics of the first computer and storing the computer specific information corresponding to the predefined characteristics of the first computer to provide browser header information. The first

computer removes the computer specific information from the web browser originated communication and the second computer combines the stored browser header information with the client/server specific data stream to create an HTTP data stream.

5

In a further aspect of the present invention the HTTP data stream corresponding to a communication originated by the web server is captured prior to transmission of the HTTP data stream on the external communication link and converted from the HTTP protocol to a client/server specific communication protocol. The converted web server originated communication is sent to the web browser over the external communication link as a client/server specific data stream. The first computer acquires the client/server specific data stream transmitted over the external communication link and rebuilds the HTTP data stream corresponding to the communication from the web server from the client/server specific data stream received over the external communication link by converting the client/server specific data stream received in the client/server specific communication protocol to an HTTP data stream. The communication originated by the web server is furnished to the web browser as an HTTP data stream.

10

15

20

A further embodiment of the header reduction aspect of the present invention includes providing to the first computer over the external communication link computer specific information corresponding to predefined characteristics of the second computer and storing the computer specific information corresponding to the predefined characteristics of the second computer to provide server header information. The second computer removes the computer specific information from the web server originated communication and the first computer combines the server header information with the client/server specific data stream to create an HTTP data stream.

30

35

In a further embodiment of the protocol reduction aspect of the present invention, a persistent connection between the first computer and the second computer is established over the external communication link. The persistent connection is maintained until all web server originated communications are completed. A plurality of web server originated communications are intercepted and multiplexed onto the external communication link while the persistent connection is maintained. The client/server specific data stream may be demultiplexed to create a plurality of HTTP data streams and the plurality of HTTP data streams provided to the web server.

A further embodiment of the present invention provides a method of increasing the performance of a client/server system having a client application resident on a first computer and communicating with a server application resident on a second computer remote from the first computer. The client application and the server application utilize a client/server independent communication protocol (*i.e.* stateless) for communication between the client and the server and at least one segment of the communication between the client application in the first computer and the server application in the second computer occurs over an external communication link. The method comprises intercepting communications in the client/server independent communication protocol originated by the remote client prior to transmission of the communications on the external communication link. The intercepted communications originated by the client are converted to a second client/server specific communication protocol and transmitted over the external communication link. The second computer receives the communication transmitted over the external communication link and converts the communication received over the external communication link from the client server specific communication protocol to the client/server independent communication protocol. The

communication originated by the remote client is provided to the server in the client/server independent communication protocol. A further aspect of the present invention incorporates intercepting communications in the client/server independent communication protocol originated by the server prior to transmission of the communications on the external communication link. The intercepted communications are converted to a second client/server specific communication protocol and transmitted over the external communication link. The communications are received from the external communication link by the first computer and converted from the client/server specific communication protocol to the client/server independent communication protocol. The communication originated by the server is provided to the remote client in the client/server independent communication protocol.

As will be appreciated by those of skill in this art, the above described aspects of the present invention may also be provided as apparatus or computer readable program means.

#### Brief Description of the Drawings

**Figure 1** is a block diagram of a typical web browser/web server system;

**Figure 2** is a block diagram of a web browser/web server system according to one embodiment of the present invention utilizing a client intercept and a server intercept;

**Figure 3** is a flow diagram depicting operations carried out by a client-side intercept module in a preferred embodiment of the present invention implementing a coherent caching system;

**Figure 4** is a flow diagram depicting operations carried out by a client-side intercept module in a preferred embodiment of the present invention implementing a coherent caching system;

**Figure 5** is a flow diagram depicting operations carried out by a server-side intercept module in a preferred embodiment of the present invention implementing a coherent caching system;

5 **Figure 6** is a flow diagram depicting operations carried out by a server-side intercept module in a preferred embodiment of the present invention implementing a coherent caching system;

10 **Figure 7** is a flow diagram depicting operations carried out by a client-side intercept module in a preferred embodiment of the present invention implementing a differencing data transfer system;

15 **Figure 8** is a flow diagram depicting operations carried out by a client-side intercept module in a preferred embodiment of the present invention implementing a differencing data transfer system;

20 **Figure 9** is a flow diagram depicting operations carried out by a server-side intercept module in a preferred embodiment of the present invention implementing a differencing data transfer system;

**Figure 10** is a flow diagram depicting operations carried out by a server-side intercept module in a preferred embodiment of the present invention implementing a differencing data transfer system;

25 **Figure 11** is a block diagram of one aspect of the present invention utilizing virtual sockets;

30 **Figure 12** is a block diagram of a client-side intercept module and a server-side intercept module according to one embodiment of the present invention utilizing virtual sockets;

35 **Figure 13** is a flow diagram depicting operations carried out by a socket manager of either the client-side intercept module or the server-side intercept module according to one embodiment of the present invention utilizing virtual sockets;



**Figure 14** is a flow diagram depicting operations carried out by a client-side intercept function in one embodiment of the present invention utilizing virtual sockets;

5 **Figure 15** is a flow diagram depicting operations carried out by a server-side intercept function in one embodiment of the present invention utilizing virtual sockets;

10 **Figure 16-1** is a flow diagram depicting the virtual create operation according to one embodiment of the present invention utilizing virtual sockets;

**Figure 16-2** is a flow diagram depicting the virtual send operation according to one embodiment of the present invention utilizing virtual sockets;

15 **Figure 16-3** is a flow diagram depicting the virtual receive operation according to one embodiment of the present invention utilizing virtual sockets;

20 **Figure 16-4** is a flow diagram depicting the virtual select operation according to one embodiment of the present invention utilizing virtual sockets;

**Figure 17-1** is a flow diagram depicting the virtual flush operation according to one embodiment of the present invention utilizing virtual sockets; and

25 **Figure 17-2** is a flow diagram depicting the virtual close operation according to one embodiment of the present invention utilizing virtual sockets.

#### Detailed Description

30 The present invention now will be described more fully hereinafter with reference to the accompanying drawings, in which preferred embodiments of the invention are shown. This invention may, however, be embodied in many different forms and should not be construed as limited to the embodiments set forth herein; rather, these embodiments are provided so that this disclosure will be thorough and  
35 complete, and will fully convey the scope of the invention

to those skilled in the art. Like numbers refer to like elements throughout.

5           **Figures 3 to 10 and 13 to 17-2** are flowchart illustrations of methods and systems according to the invention. It will be understood that each block of the flowchart illustrations, and combinations of blocks in the flowchart illustrations, can be implemented by computer program instructions. These computer program instructions may be loaded onto a computer or other programmable apparatus to produce a machine, such that the instructions which execute on the computer or other programmable apparatus create means for implementing the functions specified in the flowchart block or blocks. These computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instruction means which implement the function specified in the flowchart block or blocks. The computer program instructions may also be loaded onto a computer or other programmable apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide steps for implementing the functions specified in the flowchart block or blocks.

10           Accordingly, blocks of the flowchart illustrations support combinations of means for performing the specified functions and combinations of steps for performing the specified functions. It will also be understood that each block of the flowchart illustrations, and combinations of blocks in the flowchart illustrations, can be implemented by special purpose hardware-based computer systems which perform the specified functions or steps, or combinations of special purpose hardware and computer instructions.

Figure 2 illustrates one embodiment of the present invention. As seen in Figure 2, a web browser 10 communicates with a client-side intercept module 30. The web server 20 communicates with a server-side interrupt module 40. The client-side interrupt module 30 then communicates with the server-side interrupt module 40 over the communication link 35. The web browser 10 and the client-side intercept module 30 may be contained in a first computer 5. The server-side intercept module 40 and the web server 20 may be contained in a second computer 6. The first computer 5 and the second computer 6 communicate over external communication link 35.

Preferably, the web browser 10 is a Internet web browser utilizing hypertext transfer protocol (HTTP) and hypertext markup language (HTML) to communicate with an Internet web server 20 which also uses HTTP and HTML. In operation, the web browser 10 would output an HTTP data stream which is intercepted by the client-side intercept module 30. The intercept of the HTTP data stream by the client-side intercept module 30 may be accomplished through the use of the TCP/IP loop-back feature where the client side intercept module 30 resides at an IP address having a network number of 127, such as 127.0.0.1. The client-side intercept module 30 then converts or transforms the HTTP data stream into a client/server specific protocol and transmits the client/server specific data stream onto the external communication link 35. The server-side intercept module 40 receives the client/server specific data stream and reconstructs the original HTTP data stream corresponding to the web browser originated communication. This reconstructed HTTP data stream is then transferred to the web server 20. The web server 20 responds to the HTTP data stream in the normal manner of an Internet web server. As will be appreciated by one of skill in the art, the web server 20 may also be a proxy which allows multiple browsers to connect to the Internet.

When information is received by the web server 20 for transmission to the web browser 10, for example, in response to a browser request for a specific URL home page, the web server 20 outputs an HTTP data stream corresponding to the communication to be sent to the web browser 10. This web server originated communication is intercepted by the server-side intercept module 40 and transformed by a client/server specific data stream. The client/server specific data stream corresponding to the web server originated communication is then sent on the external communication link 35 from the second computer to the first computer. The client/server specific data stream is received by the client-side intercept module 30 and the original HTTP data stream corresponding to the web server originated communication is rebuilt and provided to the web browser 10.

In a particular embodiment of the present invention, the external communication link 35 is a wireless communication link. In such a case, in order to obtain system performance which is acceptable to users, it is desirable to reduce the amount of communication over the external communication link 35 both in the frequency of the communications and in the amount of information which must be transferred over the communication link 35. Accordingly, the present invention utilizes caching, differencing, and protocol reduction techniques to minimize the amount of communication required over the external communication link 35. These techniques are accomplished by converting the stateless or stochastic protocols of HTTP into a client/served specific protocol which utilizes information specific to the client and the server to reduce the amount and frequency of communications.

While the present invention has and will be described with respect to a single web browser application and a single web server application, as will be appreciated by those of skill in this art, the benefits and advantages of

the present invention may also be achieved with multiple web browsers associated with a single web server. Thus, the methods, apparatus and program products of the present invention in connection with multiple browsers each  
5 communicating with a client-side intercept module and these client side intercept modules would then communicate with the server-side intercept module of the web server or web proxy.

In one embodiment of the present invention, both the  
10 client-side intercept module 30 and the server-side intercept module 40 have cache storage capabilities. The client cache resident in the first computer stores HTTP data streams to be received by the web browser in response to a web browser originated communication. The server cache  
15 resident in the second computer stores HTTP data streams which are received from the web server in response to a browser originated communication.

As will be appreciated by one of skill in the art, the cache resident in the first computer or the second computer  
20 may be of any size based upon the specific hardware configurations of the computers. These caches store information for each communication including, the URL of the communication, a unique identifier based on the communications contents such as a cyclic redundancy check  
25 (CRC) of the data of the communication, the store date time (SDT) indicating the time when the cache entry was created or refreshed and the data of the communication. Thus, a directory of cache entries may be created for each communication stored in the cache. Furthermore, because of  
30 the limited resources available in any given hardware configuration, any number of caching techniques known to one of skill in the art for maintaining the caches resident in the first computer and the second computer may be utilized. Thus, for example, the cache may invalidate the oldest  
35 directory entry if a user defined cache size would be exceeded by the addition of a new entry and then the new

entry added in place of the invalidated entry. Furthermore, cache entries may be maintained over multiple instances of the web browser or web server applications or even power-on cycles of the first or second computers to create a  
5 persistent cache.

The operation of the caching structure according to one aspect of the present invention will be now be described with reference to **Figures 3** through **6**, which are flowcharts describing the operation of the client-side intercept module  
10 **30** and the server-side intercept module **40**.

Specifically with reference to **Figure 3**, block **100** indicates that the client-side intercept module **30** has received a request from the web browser **10**. This request may take the form of an HTTP data stream. The client-side  
15 intercept module **30** checks the uniform resource locator (URL) of the incoming request as is reflected in block **105**. The client-side intercept module **30** determines from the URL if the information corresponding to the web browser originated request has previously been stored in the client  
20 cache resident in the first computer.

If the information corresponding to the URL has not been previously stored in the client cache, then the operations depicted in block **106** are carried out by the client-side intercept module. The client-side intercept  
25 module **30** transmits a request on the external communication link **35** to the server-side intercept module **40**.

If, however, upon interrogating the web browser originated communication as depicted in block **105** a client cache entry exists which corresponds to the web browser  
30 originating communication, then in the simplest embodiment this information would be provided to the web browser as an HTTP data stream. However, as reflected in **Figure 3**, the preferred embodiment of the present invention performs what is referred to herein as a coherency interval check on the  
35 cache entry corresponding to the web browser originated

communication. This operation is reflected in block 110 of **Figure 3**.

5 The coherency interval for a client-side intercept module may be user defined and is the length of time which a cache entry may exist before it becomes stale and, even if present, must be refreshed by requesting the information corresponding to the web browser originated communication from the web server. The coherency interval check reflected in block 110 may be carried out by comparing the current  
10 date and time to the sum of the SDT of the cache entry corresponding to the web browser originated communication and the coherency interval specified by the user. If the current date and time is greater than this sum then the information stored in the cache corresponding to the web  
15 browser originated communication has become stale and the "No" branch of block 110 is taken. However, if the current date and time is less than the sum of the SDT plus the user defined coherency interval, then the "Yes" branch of block 110 is taken and, as reflected in block 111, the cache entry is supplied to the browser as an HTTP data stream. Thus  
20 completing the browser originated communication received by the client-side intercept module 30 in block 100 of **Figure 3**.

25 If the coherency interval check reflected in block 110 determines that the cache entry resident in the first computer is stale, then a request is made to the server-side intercept module 40 to check the coherency of the cache entry resident in the second computer. This operation is reflected in block 112 of **Figure 3**. This is accomplished by  
30 supplying across the external communication link 35 to the server-side intercept module 40 the coherency interval for the particular client-side intercept module 30 the HTTP request originated by the web browser 10 and a unique indicia of the contents of the client cache corresponding to  
35 the URL of the web browser originated communication. In a

preferred embodiment, this unique indicia is the results of a cyclic redundancy check or CRC for the cache entry.

Turning now to **Figure 5** which reflects the server-side intercept module operations in response to information received over the external communication link 35 from the client-side intercept module 30. When the server-side intercept module 40 receives a request from the client-side intercept module, the server-side intercept module 40 receives the predetermined client coherency time interval, the CRC value for the client cache entry, and the HTTP request originated by the web browser. The receipt of this information is reflected in block 120 of **Figure 5**.

After receiving the information from the client-side intercept module 30, the server-side intercept module 40 checks its server cache resident in the second computer to determine if a server cache entry exists corresponding to the URL of the HTTP request originated by the web browser. If, after interrogating the web browser originated communication as reflected in block 125, the server-side intercept module 40 determines that a cache entry does exist corresponding to the information requested by the web browser originated communication the "Yes" branch of block 125 is taken. The server-side intercept module 40 then compares the current date and time of the SSI module 40 to the sum of the SDT of the server cache entry corresponding to the information requested by the web browser originated communication and the predetermined client coherency time interval received from the client-side intercept module.

If the current date and time is less than the sum of the SDT for the server cache entry and the coherency interval, then the "Yes" path of block 130 of **Figure 5** is taken. The server-side intercept module 40 then compares the CRC of the server cache entry to the CRC of the client cache entry to determine if the two cache entries are identical. If the two cache entries are identical, then the "Yes" path of block 135 is taken and, as reflected in block



136, a "coherent" response is sent to the client-side intercept module 30.

5 If the conditional of block 135 determines that the CRC's are not equal, then the information contained in the client cache and the server cache are not identical and, as reflected in block 137, the server-side intercept module sends the server cache entry to the first computer over the external communication link. In sending the server cache entry to the client-side intercept module 30, the server-side intercept module converts the entry to a client  
10 specific communication protocol which includes the CRC of the server cache entry, the server cache entry data, and the age of the server cache entry. The age of the server cache entry is calculated by subtracting from the current date and time the SDT of the cache entry.  
15

Finally, with respect to **Figure 5**, if either the sum of the SDT plus the predetermined client coherency time interval is less than the current date and time or if no cache entry exists corresponding to the URL of the web browser originated communication, then the "No" path of  
20 block 130 or the "No" path of block 125, respectively, will be taken. Thus, the operations of block 126 will be carried out and the server-side intercept module 40 will send to the server the web browser originated communication as an HTTP data stream. If the server-side intercept module 40 must  
25 send the web browser originated communication to the server as an HTTP data stream, then the server-side intercept module 40 will execute the operations of **Figure 6**.

30 As seen in **Figure 6** block 140, in response to the web browser originated communication, the server-side intercept module will receive an HTTP data stream from the web server 20. Upon receipt of the HTTP data stream, the server-side intercept module 40 will calculate the CRC for the HTTP data stream and temporarily store the HTTP data stream. Then, as  
35 reflected in block 145, the server-side intercept module interrogates the HTTP data stream and determines if a server

cache entry corresponding the URL of the HTTP data stream exists. If such an entry does exist, then the "Yes" path of block 145 is carried out. The server-side intercept module 40 then compares the recently calculated CRC of the HTTP data stream received from the web server 20 with the CRC of the server cache entry corresponding to the URL of the web server originated response communication as reflected in block 150. If the CRC's are the same, then the "Yes" branch of block 150 is carried out. The server-side intercept module 40 updates the SDT entry for the server cache entry as reflected in block 151 and empties from temporary storage the HTTP data stream received by the web server 20 as shown in block 152.

If the results of the CRC comparisons indicate that the server cache entry is different than the HTTP data stream received from the web server 20, then the "No" path of block 150 is carried out. The server-side intercept module 40 removes from the server cache the existing data as reflected in block 153 and then, as reflected in block 154, updates the server cache with the newer information. As seen in block 154, this update includes storing in the server cache the CRC of the web server communication storing as part of the cache entry the current date and time as the SDT for the cache entry and storing the HTTP data stream. In either case, whether a server cache entry is updated or whether the server cache entry is found to be identical to the HTTP data stream received from the web server 20, the server-side intercept module then determines if the server cache entry is identical to the client cache entry corresponding to the web browser originating communication. This operation is reflected in block 155.

If the server-side intercept module 40 determines that a cache entry does not exist corresponding to the response received from the web server 20, then the "No" path of block 145 is taken. A server cache entry is created as reflected in block 146 by storing the URL of the response from the web

server, by storing the CRC of the response from the web server, by storing the HTTP data stream, and by storing as the SDT the current date and time. After creating a cache entry corresponding to the web browser originated  
5 communication, the server-side intercept module 40 then again compares the CRC of this server cache entry to the CRC of the corresponding client cache entry as reflected in block 155.

10 If the results of the comparison of the server cache entry to the client cache entry indicate that the cache entries are identical, then the "Yes" branch of block 155 is taken and the operations of block 156 are carried out. In block 156 it is seen that the server-side intercept module 40 sends a coherent response to the client-side intercept  
15 module 30. The server-side intercept module 40 transforms the server request cache entry to a client/server specific data stream by sending the coherent response and sending an age of zero to the client-side intercept module.

20 If the server-side intercept module 40 determines that the client cache entry is not identical to the server cache entry corresponding to the web browser originated communication, then the "No" branch of block 155 is taken and the operations of block 157 are carried out. As reflected in block 157, the server-side intercept module  
25 40 converts or transforms the server cache entry into a client/server specific data stream. The data stream includes the CRC of the server cache entry, the server cache entry HTTP data stream, and the age of the cache entry which is set to zero. This client/server specific communication  
30 is then transmitted over the external communication link 35 to the client-side intercept module 30.

The functions of the client-side intercept module 30 upon receipt of a communication from the server-side intercept module will now be described with respect to  
35 **Figure 4**. As seen in block 160, the client-side intercept module 30 receives or acquires the client/server specific

data stream which was transmitted over the external communication link 35. The client-side intercept module then determines what type of response was received from the server-side intercept module 40 as reflected in block 165. If the server-side intercept module 40 indicates that the client cache entry is coherent, i.e. the server cache entry and the client cache entry are identical, then the operations reflected in block 166 are carried out. As seen in block 166, the client-side intercept module 30 updates the SDT of the client cache entry corresponding to the web browser originated communication with the difference between the current date and time and the age received from the server-side intercept module 40. Thus, without synchronizing the two clocks of the first computer 5 and the second computer 6, the present invention has revised the coherency time of the cache entry of the first computer to reflect the newer data of the second computer. After updating the SDT for the client cache entry corresponding to the web browser originated communication, the client-side intercept module 30 transfers the client cache entry to the web browser 10 as an HTTP data stream. This operation is shown in block 174.

If, however, the client-side intercept module 30 determines that the response type is a data or data stream response, then the "stream" path out of block 165 is taken and the operations of block 167 are carried out. The client-side intercept module 30 receives the HTTP data stream and temporarily stores this data. Then, as reflected in block 170 of **Figure 4**, the client-side intercept module 30 determines if a cache entry exists corresponding to the web browser originated communication. If a cache entry exists, then the "Yes" path of block 170 is taken and, as reflected in block 171, the existing cache entry is flushed. The client-side intercept module then updates the client cache entry corresponding to the web browser originated communication by storing the CRC of the HTTP data stream

received from the server-side intercept module 40, by storing as the SDT the difference between the current date and time and the age received from the server-side intercept module 40, and by storing the HTTP data stream. This operation is reflected in block 172.

If no cache entry exists corresponding to the web browser originated communication, then the "No" path of block 170 is taken. A client cache entry is created by carrying out the operations reflected in block 173. As seen in block 173, the client-side intercept module 30 creates a client cache entry by storing the URL of the HTTP data stream received from the server-side intercept module 40 by storing the CRC of the HTTP data stream received from the server-side intercept module 40 and by storing the HTTP data stream. The client-side intercept module 30 also updates the SDT or stores the SDT by subtracting from the current date and time the age received over the external communication link 35 from the server-side intercept module 40.

However, a client cache entry is created whether through the operations of blocks 166, 172, or 173, the client-side intercept module transfers or provides the client cache entry to the web browser 10 as an HTTP data stream. These operations are reflected in block 174 of **Figure 4**.

As will be appreciated by one of skill in the art, the client cache and the server cache may be implemented with memory or with mass storage such as hard disks, read/write CD-ROMS, optical disks, or other storage technologies. Furthermore, as will be appreciated by one of skill in the art, the client-side intercept module and the server-side intercept module may be implemented through software, hardware, or a combination thereof.

While references made to caches being resident in a particular first or second computer, as will be appreciated by one of skill in the art, the benefits of the present

invention may be achieved even though the cache is not resident in the first computer but is simply on the same side of the external communication link as the computer. Thus, a hardware cache could be implemented external to the first computer that serves as a client cache and connected to the first computer by high speed communications and yet, as long as the cache is on the same side of the external communication link as the first computer, the benefits of the present invention will be achieved.

In an alternative embodiment of the present invention, the server-side intercept module 40 does not maintain a copy of the HTTP data stream received from the web server 20 but simply maintains a directory entry for the communication. The directory entry would include the URL of the communication, the CRC calculated for the HTTP data stream and the time when the HTTP data stream was received from the web server and the SDT for the communication which may be set to the time when the CRC was calculated. In such a case when the client-side intercept module 30 sends a request to the server-side intercept module 40 for a communication which corresponds to a URL for which the server-side intercept module has maintained a CRC and SDT, then the server-side intercept module checks the CRC received from the client-side intercept module 30 to determine if it corresponds to the CRC of the latest HTTP data stream for the specified URL. If there is a match, then a coherent response is sent to the client-side intercept module. If there is not a match, then the server-side intercept module sends the HTTP data stream received from the client-side intercept module to the web server 20 and returns to the client-side intercept module 30 the response received from the web server 20.

Figure 7, 8, 9, and 10 reflect the operations carried out by the client-side intercept module 30 and the server-side intercept module 40 in another aspect of the present invention which utilizes differencing to reduce the data

transferred over the external communication link 35. As seen in **Figure 7**, block 200 illustrates the receipt by the client-side intercept module 30 of an HTTP request from the web browser 10. As reflected in block 205, the client-side intercept module 30 interrogates the intercepted HTTP request from the web browser 10 to determine if the request is to a common gateway interface (CGI). If the request is not to a common gateway interface, then the client-side intercept module 30 passes the request to the server-side intercept module as reflected in **Figures 3** through **6** and is illustrated by block 206 of **Figure 7**.

If, however, the web browser originated communication corresponds to a CGI request, then the "Yes" path out of block 205 is taken. As reflected in block 210, the client/server intercept module 30 determines if a client base cache entry exists corresponding to the HTTP data stream which was previously to be provided to the web browser in response to a corresponding CGI request. This interrogation of the CGI request may be accomplished by comparing the URL of the web browser originated communication to the URLs stored in the client base cache.

The client base cache may be initialized by storing the first HTTP data stream received by the client-side intercept module 30 which is to be provided to the web browser 10 for a given URL. This base cache entry may be maintained over numerous instances or sessions of the web browser 10. The client base cache entries may be updated as reflected in **Figures 7, 8, 9, and 10**. If a client base cache entry exists corresponding to the URL for the web browser originated communication, then the CRC to be sent to the server-side intercept module 40 over the external communication link 35 is set equal to the CRC for the client base cache entry as reflected in block 211 of **Figure 7**. If no client base cache entry exists, then the "No" path out of block 210 of **Figure 7** is taken and the CRC for the request to be sent over the external communication link 35 to the

server-side intercept module 40 is nulled. This operation is reflected in block 212 of **Figure 7**.

5 Block 213 illustrates the operations of sending the CGI request to the server-side intercept module 40 over the external communication link. As reflected in block 213, the client-side intercept module 30 transmits the HTTP request and the request CRC which has either been set to null if no client base cache entry exists for the URL of the CGI request or has been set to the CRC of the client base cache entry if an entry does exist. Thus, the client-side intercept module has converted the CGI request to a client/server specific protocol, transmitted the client/server specific communication over the external communication link to be received by the server-side intercept module 40.

10 The actions taken by the server-side intercept module when a CGI request is received are reflected in **Figure 9**. The receipt of the CGI request by the server-side intercept module 40 is shown in block 220. When the server-side intercept module 40 receives the CGI request, it saves a copy of the CRC value and the HTTP request. As seen in block 221, the server-side intercept module 40 passes the HTTP request to the web server 20.

20 When the server-side intercept module 40 receives a response to the HTTP request corresponding to the web browser originated communication or CGI request, the server-side intercept module 40 receives this response as an HTTP data stream as reflected in block 230 of **Figure 10**. As seen in block 230, the server-side intercept module 40 saves the HTTP data stream and computes a CRC value for the HTTP data stream received from the web server 20. The server-side intercept module 40 also nulls the difference value to initialize the difference data. The server-side intercept module then determines if the response received as a web server originated communication is a response to a CGI request as shown in block 235. If the answer is no, then



the "No" path out of block 235 of **Figure 10** is taken and the operations of block 236 are executed to send the HTTP data stream to the client-side intercept module. As reflected in block 236, this operation may involve the caching operations described in **Figures 3** through 6. If the response received in block 230 is a response to a CGI request, then the "Yes" path out of block 235 is taken and the server-side intercept module then determines if a server base cache entry exists for the CGI response as reflected in block 240.

A server base cache entry may be created the first time the server-side intercept module 40 receives a response to a CGI request. In this instance the result of the conditional reflected in block 240 will cause the "No" path to be taken out of block 240. The server-side intercept module 40 will then create a server base cache entry corresponding to the CGI request by storing the URL for the CGI, the HTTP data stream for the response to the CGI request, and the CRC for the HTTP data stream. This operation is reflected in block 241. To be compatible with the coherent cache system described in **Figures 3** through 6, the server base cache entry may also include the SDT. As used herein, the term server CGI base form refers to the server base cache entry corresponding to the CGI request received from the web browser 10.

If a server base cache entry exists corresponding to the CGI request then the "Yes" path out of block 240 is taken. The server-side intercept module compares the CRC of the server base cache entry to the CRC of the response received from the web server 20. These operations are reflected in block 245 of **Figure 10**. If the CRCs are the same, then the server-side intercept module determines if the CRC for the server base cache entry corresponds to the CRC for the client base cache entry. If these two CRC values are the same, then the client base cache entry, the server base cache entry, and the response received from the web server 20 all contain the same HTTP data stream. The

comparison of the server base cache entry to the client base cache entry is reflected in block 250.

5 If the two base cache entries are the same, then the server-side intercept module need not send the base cache entry to the client-side intercept module 30 and so, as reflected in block 251, the HTTP data stream data to be transferred to the client-side intercept module 30 is nulled. The server-side intercept module 40 then converts the HTTP data stream received from the web server 20 to a client/server specific communication protocol by transmitting the CRC of the HTTP data stream stored in the server base cache corresponding to the CGI request, the nulled HTTP data stream data and the nulled difference data to indicate that the response to the CGI request was identical to the client base cache entry, as illustrated in block 252.

10 Returning to block 245, if the CRC for the server base cache entry corresponding to the CGI request is different than the CRC for the response received from the web server in response to the CGI request originated by the web browser, then the "No" path out of block 245 is taken. The server-side intercept module 40 then carries out the operations reflected in block 246. The server-side intercept module 40 compares the intercepted CGI response to the server base cache entry corresponding to the intercepted CGI request or the server CGI base form. This comparison of the intercepted CGI response to the server CGI base form provides CGI difference data which corresponds to the difference between the intercepted CGI response and the server CGI base form.

20 The differencing may be performed by any method known to those of skill in the art for determining the difference between a base form and a modified form. One method of differencing suitable for use in the present invention is described in "a Cross-Platform Binary Diff" by Coppieters, Dr. Dobb's Journal, May 1995, pp. 32-36, the disclosure of

which is incorporated herein by reference as if set forth fully. Other methods which may be used in determining the difference data include those described in *IBM Technical Disclosure Bulletin*, Vol. 22, No. 8A, January 1980 which is also incorporated herein by reference as if set forth fully.

The server-side intercept module 40 then determines if the server CGI base form requires updating as reflected in block 247. This determination may be made by determining if the average difference data between the intercepted CGI response and the server CGI base form is over a predefined threshold. Other methods of determining if the server base cache entry corresponding to the CGI request should be updated may include time coherency such as that described in **Figures 3** through **6** or other methods known to those with skill in the art to determine if the difference data has increased to such an extent that rebasing to create a new base cache entry would improve system performance.

If rebasing of the server is not required, then the "No" path out of block 247 is taken and the server-side intercept module 40 carries out the operations of block 250 to determine if the CRC of the client base cache entry is the same as that of the server base cache entry or the server CGI base form is identical to a client CGI base form which are the base cache entries of the server and the client which correspond to the particular CGI request of the web browser originated communication. If the base forms are the same, then the client does not need to be rebased and the HTTP data stream information is nulled, as reflected in block 251. The server-side intercept module 40 then sends the difference response to the client-side intercept module 30 by sending the CRC of the server base cache entry corresponding to the CGI request (*i.e.* the CRC of the server CGI base form), by sending the nulled HTTP data stream which would correspond to the base data and by sending the difference data determined in block 246. These operations are again reflected as block 252 of **Figure 10**.

If the server-side intercept module 40 determines that the CRCs are not the same for the client CGI base form and the server CGI base form, then the client needs to be rebased. The client rebasing operation consists of sending the server CGI base form to the client-side intercept module 30. To perform this operation, the server-side intercept module sets the HTTP data stream data to be sent to the client-side intercept module 30 equal to the server CGI base form. This operation is reflected in block 253. The server-side intercept module 40 then converts the HTTP data stream received from the web server to a client/server specific protocol by sending the CRC of the server CGI base form, the HTTP data stream data corresponding to the server CGI base form, and sending the difference data between the CGI base form and the response received from the web server as seen in block 252. This information is then transmitted over the external communication link 35 to the client-side intercept module 30.

Returning to block 247, if server rebasing is required then the "yes" path out of block 247 is taken. As reflected in block 248 the server side intercept module updates the server base cache entry corresponding to the browser originated communication with the HTTP data stream received from the web server. The CRC of the response is also updated and the CGI difference data is nulled. The server side intercept module then compares the CRC of the new server side cache entry as reflected in block 250 and completes the transfer as described above.

The operations of the client-side intercept module upon receipt of a response from the server-side intercept module 40 are shown in Figure 8. The receipt of the response from the server-side intercept module 40 by the client-side intercept module 30 is reflected in block 260. As seen in block 265, the client-side intercept module 30 determines if the response is a response to a CGI request. If the response is not to a CGI request, then the client-side

intercept module carries out the operations of block 267 which may include the cache operations reflected in **Figures 3 through 6**. If, however, the response is to a CGI request, then the "Yes" path out of block 265 is taken. The client-side intercept module 30 saves the HTTP data stream data, the difference data, and the CRC acquired from the client/server specific data stream transmitted over the external communication link. These operations are reflected in block 266 of **Figure 8**.

The client-side intercept module 30 then determines if a client base cache entry corresponding to the intercepted CGI request exists which would contain a client CGI base form. This interrogation is shown in block 270 and may be carried out by examining the URL of the HTTP request or HTTP response. If a client CGI base form exists, then the "Yes" path out of block 270 is taken. The client-side intercept module 30 then compares the CRC received over the external communication link to that of the CRC of the client CGI base form as shown in block 275. If they are different, then the "No" path of block 275 is taken and the client rebases by updating the CGI base form by replacing the client base cache entry corresponding to the URL of the CGI request of the web browser originated communication with the HTTP data stream data received over the external communication link 35 from the server side intercept module 40. The client base cache entry also is updated with respect to the CRC for the HTTP data stream. These operations are reflected in block 276 of **Figure 8**.

If the CRC received over the external communication link 35 is the same as the CRC of the CGI base form, then the server-side intercept module server CGI base form is the same as the client-side intercept module client CGI base form and the "Yes" path out of block 275 is taken.

Whether the base forms are the same or the client is rebased, the operations reflected in block 277 are carried out by the client-side intercept module 30. Block 277

reflects the client-side intercept module 30 reconstructing the HTTP data stream corresponding to the communication from the web server 20 from the client/server specific data stream received over the external communication link 35 by  
5 combining the client CGI base form with the CGI difference data received over the external communication link 35 to create an HTTP data stream corresponding to the intercepted CGI response. As seen in block 278, this response is then provided to the web browser 10 as an HTTP data stream.

10 If no CGI base form exists in the client corresponding to the URL of the CGI request, then the "No" path out of block 270 of **Figure 8** is taken. As seen in block 271, the client-side intercept module 30 creates a client base cache entry corresponding to the URL of the CGI request by storing  
15 the URL, the CRC of the HTTP data stream received over the external communication link from the server-side intercept module 40, and the actual HTTP data stream data. Storing this information creates a client base cache entry corresponding to the intercepted CGI request and thus  
20 creates a client CGI base form. The client-side intercept module may then carry out the operations of block 277 by reconstructing the HTTP data stream by combining or merging the client CGI base form with the CGI difference data which may have been nulled.

25 The present differencing techniques may also be applied to non-CGI data. In such an instance, the server-side intercept module 40 would need to keep multiple generations of server base cache entries to allow for the possibility that client-side intercept modules of web browsers connected  
30 to the web server may have different base forms. The server-side intercept module could then compare the CRC received from the client-side intercept module with the CRC of each of the prior generations of server base forms until a match was obtained. The server-side intercept module 40  
35 may then optionally rebase the client-side intercept module 30 or simply provide the difference data to the client-side

intercept module 30. Thus, the differencing methodologies described herein with respect to the CGI request may apply equally to any HTTP request and response.

While the above system of maintaining multiple generations of base forms may allow for the use of differencing with non-CGI requests, this methodology is more memory or storage intense and does not fully exploit the caching capabilities described above. To reduce memory or storage requirements and exploit the caching methods described above, the following preferred method of using differencing for non-CGI requests may be utilized. In this preferred implementation the server side intercept module calculates the difference between the a server base form corresponding to the request and the HTTP data stream of the response from the web server. This difference data is then stored by the server side intercept module. The server base form is then updated by replacing the base form with the new response from the web server, including updating the CRC of the base form. However, rather than discarding the old CRC, the CRC's for previous base forms are stored as is the difference data. The prior generations of difference data and CRCs are then selectively transmitted to the client side intercept module based upon the CRC of the client base form corresponding to the non-CGI request.

As an example of the non-CGI differencing method, if the server side intercept module receives a non-CGI request this request would also be accompanied by the CRC of the base form resident in the client side intercept module corresponding to the URL of the non-CGI request. When the server side intercept module received the response from the web server it would calculate the CRC of the response. The server side intercept module would then calculate the difference between the response and the server base form for the URL and save this difference data. The server side intercept module would update the server base form with the response data and archive the CRC of the previous base form

and the difference data between the response and the old base form. The server side intercept module would then compare the CRC of the client base form with the server base form CRC and any stored or archived CRCs to determine if a match is found. If no match is found the response is simply sent to the client side intercept module.

If a match is found then the difference data corresponding to the CRC match and any subsequent difference data up to and including the current difference data is sent to the client side intercept module. The client side intercept module then applies the difference data to the client base form to reconstruct the response. Thus, if the CRC match occurred with a CRC for a base form which was three generations old then three sets of difference data would be sent to the client side intercept module and the construction of the response would be accomplished by applying three successive difference data sets to the client base form. If however, the number of difference data sets or the sizes of the difference data sets required to reconstruct the response is so great that sending the actual response would require less data transfer than the response itself may be sent by the server side intercept module. In any event, after reconstructing or receiving the response the client side intercept module would update the client base form for the URL of the request with the response data and update the CRC with the CRC for the response. Because the client base form is updated each time a response is received for a particular URL, the client cache described above may be utilized as the cache for the client base form, thereby eliminating the need for a separate cache of the client base forms if differencing is utilized on non-CGI requests.

In a further aspect of the present invention, additional communication savings may be achieved based upon the redundancy of a stateless communication protocol such as HTTP. In such a protocol, the client transmits information



about itself to the server each time a communication is initiated. Likewise, the server communicates specific information about itself to the client each time a response is initiated.

5           In one alternative embodiment of the present invention, the first computer 5 communicates to the second computer 6 the computer specific information corresponding to the predefined characteristics of the first computer. The second computer stores this computer specific information.  
10 The first computer then removes the computer specific information from subsequent web browser originated communications prior to transmission on the external communication link 35. The second computer 6 then rebuilds the original web browser originated communication by  
15 combining the stored computer specific information with the subsequent communication received over the external communication link 35 to create an HTTP data stream.

          In addition to removing the computer specific information from communications originated by the web  
20 browser, this computer specific information may also be removed from communications originated by the web server. In such a case, the second computer 6 of **Figure 2** provides to the first computer 5 over the external communication link 35 the computer specific information corresponding to the  
25 predefined characteristics of the second computer 6. The first computer 5 stores the computer specific information to provide server header information. On subsequent communications, the second computer 6 removes the computer specific information from the web server originated  
30 communication and transmits the remaining portion of the web server originated communication on the external communication link 35. The first computer 5 receives the communication over the external communication link and rebuilds the original web server originated communication by  
35 combining the server header information with the client/server specific data stream received over the

external communication link to create an HTTP data stream. In both instances, the operations of removing the computer specific information and storing the information to create either server header information or client header  
5 information are carried out by the client-side intercept module 30 or the server-side intercept module 40, depending upon whether the operation takes place in the first computer 5 or the second computer 6.

In one embodiment of the present invention, the web  
10 browser 10 communicates to the client-side intercept module 30 using the Transmission Control Protocol/Internet Protocol (TCP/IP). TCP may also be used for a communication between the client-side intercept module 30 and the server-side intercept module 40 over the external communication link 35.  
15 Finally, TCP may be used for communication between the server-side intercept module 40 and the web server 20. While TCP may be used for communications between the various components that make up the system of the present invention, the HTTP protocol does not provide the most efficient means  
20 for communication over an external communication link. To increase the performance of the external communication link 35, one embodiment of the present invention creates what are referred to herein as "virtual sockets" which are utilized in the connection between the web browser and the client-  
25 side intercept module 30 and between the server-side intercept module 40 and the web server 20. The operation of these virtual sockets will now be described with reference to **Figures 11** through **17**.

**Figure 11** is a block diagram of one possible  
30 implementation of the present invention utilizing the concept of virtual sockets. As seen in **Figure 11**, the first computer 5 and the second computer 6 are connected over the external communication link 35. The web browser 10 has a plurality of real sockets which connect the web browser 10  
35 to the client-side intercept module 30. As seen in **Figure 11**, the first real socket is depicted as **65a** on the web

browser 10 and the corresponding socket is 65b on the client-side intercept module 30. This first real socket is the TCP socket over which the web browser 10 requests further connections from the client-side intercept module 30.

When the web browser 10 requests a new TCP connection, a communication occurs over the real socket 65a which is received at the real socket 65b. The client-side intercept module 30 will then create another real socket for communication with the web browser 10. As seen in Figure 11, a plurality of real sockets are created on the web browser 10 with a corresponding real socket being created on the client-side intercept module 30. These real sockets are depicted as 60a through 64a on the web browser 10 and 60b through 64b on the client-side intercept module 30. These real sockets are the means through which the web browser 10 communicates with the client-side intercept module 30. After creating the real sockets 60a through 64a and 60b through 64b, communications over these sockets are multiplexed onto a real socket 36a which provides access for the client-side intercept module 30 to the external communication link 35. Real sockets 36a and 36b are created when a request is sent over real socket 37a of computer 5 to real socket 37b of computer 6. Upon receipt of the connection request by real socket 37b, real sockets 36a and 36b are created. Sockets 37a and 37b act as the first real sockets for communication between the client side intercept module and the server side intercept module and may only be utilized for establishing the connection between the two modules reflected by sockets 36a and 36b. Each of these real sockets operates under standard TCP/IP protocols. When communications are received by the second computer 6 over the external communication link 35, they are received at real socket 36b. The server-side intercept module 40 then demultiplexes the communications received at socket 36b and provides them to the appropriate socket for transmission to

the web server 20. Thus, for example, a communication over socket 60a to socket 60b for a request of information from a specific URL would be multiplexed onto socket 36a, received by socket 36b, demultiplexed by the server-side intercept module 40, and transmitted from socket 60c to socket 60d on the web server 20. Likewise, communications occurring over socket 61a are received by socket 61b, multiplexed by the client-side intercept module 30, and transmitted from socket 36a to socket 36b where the server-side intercept module 40 demultiplexes the communication and transmits it over socket 61c to socket 61d. Thus, communications over socket 60a and 60b, 61a and 61b, 62a and 62b, 63a and 63b, and 64a and 64b are transmitted over the respective corresponding sockets between the server-side intercept module 40 and the web server 20 of socket 60c and socket 60d, socket 61c and 61d, socket 62c and socket 62d, socket 63c and socket 63d, and socket 64c and 64d.

In a similar manner, responses to requests from web browser 10 by the web server 20 are also transmitted over sockets connecting the web server 20 to the server-side intercept module 40 and over the external communication link 35 to the client-side intercept module 30, and then to the web browser 10. Thus, for example, a response originated by web server 20 could be sent over socket 60d to socket 60c and multiplexed by the server-side intercept module 40 onto socket 36b where it is transmitted over the external communication link 35 to socket 36a. The client-side intercept module 30 then demultiplexes the communication and provides it to socket 60b for transmission to socket 60a on the web browser 10. A similar communication path is established for each socket being utilized by the web browser 10 or the web server 20. As will be appreciated by one of skill in the art, while the present invention has been described with respect to 4 socket connections between the web browser 10 and the web server 20, any number of

sockets may be opened for providing communication access between the web browser 10 and the web server 20.

5 **Figure 12** is a block diagram illustrating the implementation of the virtual socket system in the client-side intercept module 30 and the server-side intercept module 40. External to these modules the real sockets between the client-side intercept module 30 and the web browser 10 and the server-side intercept module 40 and the web server 20 function as normal TCP/IP sockets. Thus, the use of virtual sockets is transparent to the web browser 10 and the web server 20.

10 A particular embodiment of the present invention will be described with respect to the block diagram **Figure 12** and the flow diagrams of **Figures 13** through 17. **Figure 13** is a flow chart for the socket manager depicted as block 68 in **Figure 12**. Referring to **Figure 13**, block 300 reflects the creation of the real socket manager 68 of the client-side intercept module 30. After the real socket manager 68 is created, it creates a first real socket shown as socket 65b in **Figure 12**. The creation of this first real socket is reflected as block 301 of **Figure 13**. After creating the first real socket 65b, the socket manager 68, resident in the client-side intercept module 30, also referred to herein as the client socket manager, waits for an event on the first real socket 65b as is seen in block 302 of **Figure 13**. When an event is received on the first real socket 65b, the real socket manager 68 examines the event and, based upon that examination, takes one of five paths as reflected in block 305 of **Figure 13**.

15 20 25 30 35 If a real socket is created in response to a communication request received at the first real socket 65b, then, as reflected in the path from block 305 to block 306 of **Figure 13**, the real socket manager 68 adds the real socket to the real event list. The real socket manager then creates a simplex virtual socket as indicated in block 307. In the case of the client-side intercept module, the real

socket manager initiates an application function which carries out functions of the client-side intercept module for the virtual socket created as reflected in block 308 of **Figure 13**.

5 As used herein, the term "simplex socket" or "simplex virtual socket" refers to a socket which connects directly to either a single socket or a single application. As used herein, "multiplex socket" refers to a socket which connects to a plurality of other sockets. Thus, the multiplex socket carries out a multiplexing or demultiplexing function and the simplex socket preforms a one-to-one connection. Thus, for example, in carrying out the functions of blocks 306 through 308 of **Figure 13**, the client socket manager 68 would, in response to the first connection request received by the first real socket 65b, create real socket 60b, simplex virtual socket 70, and initiate the client-side intercept function in an application 80. Similarly for subsequent events where a real socket is created, the real socket manager would create real sockets 61b, 62b, 63b, or 64b and simplex virtual sockets 71, 72, 73, or 74, and initiate a CSI function corresponding to the created real and virtual sockets depicted as blocks 81, 82, 83, or 84 of **Figure 12**.

25 The operation of the client-side intercept function will now be described with reference to the real socket 60b, the simplex virtual socket 70, and the client-side intercept function 80 reflected in **Figure 12**. Block 325 of **Figure 14** reflects the creation of the client-side intercept function 80. Upon creation, the client-side intercept function 80 waits for an event on the simplex virtual socket 70 as indicated in block 326. This wait operation is carried out by performing the virtual select function which is described in **Figure 16-4**. Upon receipt of an event, the event is examined as reflected in block 330. If the event is a virtual socket close, then the client-side intercept function 80 deletes the simplex virtual socket 70 as

reflected in block 349 and terminates as reflected in block 350 of Figure 14.

5 If the event is the receipt of data, then the path from block 330 to block 331 is taken and the client-side intercept function 80 receives the browser originated communication from the simplex virtual socket 70 by executing the virtual receive operation described herein with reference to Figure 16-3. The client-side intercept function then carries out the function of the client-side intercept module as described above (see for example Figures 10 3 and 7), which is reflected in block 332. The client-side intercept function 80 then creates a multiplex virtual socket 90 which is connected to the real socket 36a in the client-side intercept module 30. Real socket 36a is 15 connected to real socket 36b on the server-side intercept module 40. The creation of the multiplex virtual socket is reflected in block 333 of Figure 14 and carried out by performing the virtual create operation described herein with reference to Figure 16-1. Block 334 reflects the operation of sending the information received from the web 20 browser over the real socket 60b and the simplex virtual socket 70 after the client-side intercept function 80 is carried out for the web browser originated communication. This communication is queued to the multiplex virtual socket 25 90 by performing the virtual send operation described herein with reference to Figure 16-2. The client-side intercept function 80, after queuing the request to the multiplex virtual socket 90, flushes the data queued in the multiplex virtual socket 90 as reflected in block 335 of Figure 14, and then waits for an event on the multiplex virtual socket 30 as reflected in block 336. The virtual flush function is carried out by performing the virtual flush operation described herein with reference to Figure 17-1 which takes the data from the multiplexed virtual socket queue and 35 provides the data to the real socket 36a. The wait operation may be carried out by performing the virtual

select function described in **Figure 16-4**. At this point, the client-side intercept module has intercepted the web browser originated communication and transferred the communication to the server-side intercept module over the external communication link 35.

Returning to **Figure 13**, which reflects the flowchart for the socket manager in either the server-side intercept module 40 or the client-side intercept module 30. The real socket manager in the server-side intercept module or the server socket manager, shown as block 69 in **Figure 12**, carries out the same function as the client socket manager shown as block 68. In creating a first real socket as shown in block 301, the server-side intercept module 30 creates a "well known port" 37b for receiving requests for sockets from the client-side intercept module 30 associated with the server-side intercept module 40. When a real event occurs on the real socket 36b of the server-side intercept module 40, the event is examined as reflected in block 305. In the present case, the event is the receipt of data from real socket 36a and so the path from block 305 to block 320 of **Figure 13** is taken. The data received on real socket 36b is examined and, in our present example, because the data is a web browser originated communication transmitted by the client-side intercept module, a new virtual socket must be created in the server-side intercept module 40. Thus the path from block 320 to block 321 of **Figure 13** is taken. The server socket manager 69 then carries out the operations reflected in block 321, block 322, block 323, and block 324 of **Figure 13**. The server socket manager 69 creates a multiplex virtual socket 95, as shown in block 321, cancels the multiplex socket activity timer as reflected in block 322 and initiates an application of the server-side intercept function as reflected in block 323 of **Figure 13** and shown as block 85 in **Figure 12**. The data received at the real socket 36b is then queued to the multiplex virtual socket 95 and a virtual event is signaled.



5 The creation of the server-side intercept function, as reflected in block 323, is shown as block 360 of **Figure 15**. After creation of the server-side intercept function 85, the function receives the data from the multiplex virtual socket 95 which was sent from the client-side intercept module 30 and corresponds to the web browser originated communication. This operation is reflected as block 361 of **Figure 15**. After receiving the data from the client-side intercept module, the server-side intercept function 85 processes the data as described above for the server-side intercept module. The carrying out of the server-side functions is reflected in block 362 (see for example Figures 5 and 9). After processing the information, the server-side intercept function 85 creates a simplex virtual socket 75 by performing a virtual create, the operation of which is described herein with reference to **Figure 16-1**. This operation is reflected in block 363 of **Figure 15**. The server-side intercept function 85 then sends the web browser originated communication to the simplex virtual socket 75 as shown in block 364 by performing a virtual send, the operation of which is describe herein with reference to **Figure 16-2**. The server-side intercept function 85 then performs a virtual flush to flush data queued in the simplex virtual socket 75 to the real socket 60c and waits for an event on the simplex virtual socket 75. The virtual flush operation is described herein with reference to **Figure 17-1**. The send and flush operations are shown in blocks 364 and 365 of **Figure 15**. The wait operation may be carried out by performing the virtual select function described in **Figure 16-4**. When the server-side intercept function 85 created the simplex virtual socket 75, a corresponding real socket 60c was also created. By sending the web browser originated communication to the simplex virtual socket 75, the server-side intercept function 85 transferred the web browser originated communication to the web server.

When the server-side intercept module 40 receives the response from the web server on the real socket 60c, a real event occurs and the server socket manager 69 exits block 302 of **Figure 13** and examines the event which occurred on real socket 60c as reflected in block 305. In the present case, it will be data for an existing virtual socket and the path from block 320 of **Figure 13** to block 324 will be taken. The data received on the real socket 60c is queued to the virtual socket 75 and a virtual event is signaled. When the virtual event is signaled, the virtual-side intercept function 85 exits block 366 of **Figure 15** and examines the event as shown in block 370. If the event is a socket closed, then an error condition occurs and an error message is constructed as the response as shown in block 375 of **Figure 15**. However, if the event is the receipt of data, then the path from block 370 to block 371 is taken and the server-side intercept function 85 performs a virtual receive, as described herein with reference to **Figure 16-3**, to obtain the server response from the simplex virtual socket 75 as shown in block 371. The server-side intercept function 85 then performs a virtual close of the simplex virtual socket 75 as reflected in block 372 and described herein with reference to **Figure 17-2** and processes the response as described above for the server-side intercept module and shown in block 373 (see for example Figures 6 and 10).

Whether the exit path of block 370 of **Figure 15** is the error path to block 375 or the data path to block 371, at block 374 the simplex virtual socket 75 is deleted. The server-side intercept function then performs a virtual send operation to the multiplex virtual socket 95 to transmit the web server originated communication to the client-side intercept module 30, as shown in block 376. The server-side intercept function 85 then performs a virtual flush operation to flush the data queued in the multiplex virtual socket 95. These operations are shown in block 377. The

server-side intercept function 85 then performs a virtual close operation to close the multiplex virtual socket 95 as shown in block 378 of **Figure 15**. Finally, the server-side intercept function 85 deletes the multiplex virtual socket and terminates, as reflected in blocks 379 and 380.

5 The server-side intercept function performs the virtual send and flush operations to the multiplex virtual socket 95. These trigger events on the real socket 36a and the client socket manager 68 exits block 302 and examines the event, as shown in block 305, because the data is received on real socket 36a, the path from block 305 to block 320 of **Figure 13** is taken and the data is queued to multiplex virtual socket 90. Therefore, when real sock 36a receives the web server response from real socket 36b over the external communication link 35, this information is demultiplexed and provided to the appropriate multiplex virtual socket. The receipt of the data causes a virtual event to occur as shown in block 324 of **Figure 13** and block 336 of **Figure 14** would be exited and the client-side intercept function 80 would examine the event as reflected in block 340 of **Figure 14**.

10 If the event is a socket closed response, then the path from block 340 to block 345 of **Figure 14** is taken and the client-side intercept function 80 creates an error message response and proceeds to block 344 of **Figure 14**. If the event is data received, as would be the case in the present example, then the path from block 340 to block 341 of **Figure 14** is taken and the client-side intercept function 80 performs a virtual receive operation to receive the response from the multiplex virtual socket 90. This receive operation is reflected in block 341 of **Figure 14**. After receiving the data from the multiplex virtual socket 90, the client-side intercept function 80 performs a virtual close operation to close the multiplex virtual socket 90 as reflected in block 342. The client-side intercept function 80 then processes the response as described above for the

client-side intercept module as reflected in block 343 (see for example Figures 4 and 8).

The operations of block 344 are then carried out whichever path is taken exiting block 340. The client-side intercept function 80 deletes the multiplex virtual socket as shown in block 344 and then performs the virtual send operation to send the response to the browser via the simplex virtual socket 70 as shown in block 346. When the virtual send operation completes, the client-side intercept function 80 performs a virtual flush operation to flush the data queued in the simplex virtual socket as shown in block 347 to the real socket 60b and then performs a virtual close operation to close the simplex virtual socket as shown in block 348. After closing the simplex virtual socket to the client-side intercept function the simplex virtual socket is deleted and the client-side intercept function terminates as shown in blocks 349 and 350 of **Figure 14**.

As will be appreciated by one of skill in the art, the present invention has been described with respect to one particular instance of the creation of simplex and multiplex virtual sockets and client-side intercept and server-side intercept functions, however, a plurality of these functions may be created within a single client-side intercept module or server-side intercept module. Accordingly, a client-side intercept module and server-side intercept module according to the present invention may create a TCP/IP connection between the client-side intercept module 30 and a server-side intercept module 40 and then multiplex on the TCP/IP connection plurality of web browser or web server originated communications while maintaining the TCP/IP connection.

The remaining functions of the client socket manager and the server socket manager may best be understood with reference to **Figures 16-1 through 16-4** and **Figures 17-1 and 17-2** which describe the operations carried out by the client-side intercept module and the server-side intercept module when a virtual create, a virtual send, a virtual

5 receive, a virtual select, a virtual flush, or a virtual  
close operation is executed as reflected in flowcharts of  
**Figure 14** and **Figure 15**. When a virtual create operation is  
performed, such as shown in block 333 of **Figure 14** and block  
10 363 of **Figure 15**, the operations beginning with block 400 of  
**Figure 16-1** are carried out. The socket manager then  
determines if a real socket is required as shown in block  
405. If a real socket already exists, such as when create  
creates a multiplex virtual socket which is to be connected  
15 to an existing real socket, then the "No" path out of block  
405 is taken and the virtual socket is connected to the real  
socket as shown in block 409. If, however, a real socket is  
required, then the "Yes" path of block 405 is taken. As  
seen in block 406, a real socket is created. The real  
20 socket is then added to the event list as shown in block 408  
for monitoring as reflected in block 302 of **Figure 13**.  
After creating a real socket and establishing a connection,  
the virtual socket is then connected to the real socket as  
shown in block 409 and create operation is completed as  
shown in block 410.

For carrying out the virtual send operation reflected  
in blocks 334 and 346 of **Figure 14**, or blocks 364 and 376 of  
**Figure 15**, the operations beginning with block 420 of **Figure**  
25 **16-2** are carried out. The data is added to the virtual  
socket queue as shown in block 427 and when complete, the  
send operation terminates as shown in block 428.

The virtual receive operation reflected in blocks 331  
and 341 of **Figure 14** and blocks 361 and 371 of **Figure 15** are  
performed by carrying out the operations beginning at block  
30 430 of **Figure 16-3**. As shown in block 435, the virtual  
socket queue is evaluated to determine if any data is  
present on the virtual socket queue. If data is present on  
the virtual socket queue, then the "Yes" path of block 435  
is taken and the data is returned to the function calling  
35 the receive operation as shown in block 436. If there is no  
data on the virtual socket queue and the socket is not

marked as closing, then the "No" path of decision block 440 is taken and nothing is returned as shown in block 441. However, if there is no data on the queue and the socket is marked as closing, then the "Yes" path of block 440 is taken and the socket is marked closed as shown in block 442 and the closed socket response is returned to the operation requesting the receive as shown in block 443.

The virtual select operation carried out in blocks 326 and 336 of Figure 14 and block 366 of Figure 15 is performed by carrying out the operations beginning with block 445 of Figure 16-4. As seen in block 446 it is first determined if data or a virtual close operation is pending for the selected virtual socket. If no data or virtual close are pending then the "no" path out of block 446 is taken and the process waits for a virtual event on the selected virtual socket as reflected in block 447 and terminates after receiving such an event as reflected in block 448. If data or a virtual close is pending for the selected virtual socket the a virtual event has already occurred and the "yes" path out of block 446 is taken and the process terminates as reflected in block 448.

The virtual flush operation referred to in blocks 335 and 347 of Figure 14 and blocks 365 and 377 of Figure 15 is performed by carrying out the operations beginning with block 450 of Figure 17-1. When called, the virtual flush operation determines if there is any data in the virtual socket queue to be flushed as reflected in the decision block 455. If there is no data in the virtual socket queue, then the flush operation simply terminates and returns to the calling function as reflected by the "No" path of block 455. If, however, there is data in the queue, then the "Yes" path of block 455 is taken and it is determined if the virtual socket queue is for a multiplex socket as shown in block 460. If it is a multiplex socket, then the socket header, which consists of three bytes reflecting a unique identifier for the socket and the amount of data in the

transfer, is added to the real socket buffer as reflected in block 461. In either case, if it is a multiplex socket or a simplex socket, the data for the real socket is then moved to the real socket buffer as shown in block 462. If the  
5 real socket buffer is full, then the "Yes" path of block 465 is taken and the data from the real socket buffer is sent on the real socket as shown in block 466. If the real buffer is not full, then the "No" path of block 465 is taken. The virtual flush function then tests to determine if there is  
10 any other data on any other multiplex virtual socket queue which is to be sent to the real socket. If the answer is Yes, then the "yes" path of block 470 is taken and the data in the real socket buffer is not sent until the virtual flush operation is called again to flush one of the other  
15 virtual socket queues. If there is no other data or after adding the data from the other multiplex virtual sockets, then the operation of block 466 is carried out and the data in the real socket buffer is sent on the real socket. After all the data in the virtual socket queue corresponding to  
20 the function which called the virtual flush operation is sent to the real socket, then the virtual flush operation terminates as reflected in block 467.

The virtual close operation shown in blocks 342 and 348 of Figure 14 and blocks 372 and 378 of Figure 15 is carried  
25 out by performing the operations beginning with block 480 of Figure 17-2. When the virtual close operation is called, the operation first tests to determine if the virtual close is of a multiplex virtual socket as reflected in block 485. If it is a multiplex virtual socket, then the "Yes" path of  
30 block 485 is taken and the "close" operation indicator is added to the virtual socket queue. Whether the virtual close is of a multiplex virtual socket or not, the virtual close operation calls the virtual flush operation as shown in block 487 and then disconnects from the real socket as  
35 shown in block 488. The operation then tests to see if the virtual close is of a simplex virtual socket as shown in

block 490, and if not, the "No" path is taken to block 495. Because the close is of a multiplex virtual socket, block 495 tests to determine if it's the last multiplex virtual socket and, if it is the last multiplex virtual socket, sets the multiplex activity timer as shown in block 496. If it is not the last multiplex virtual socket, then block 496 is skipped.

Returning to block 490, if the virtual close is of a simplex virtual socket, then the corresponding real socket is removed from the event list as shown in block 491 and the real socket is closed and deleted as shown in block 492. Whether the socket is simplex or multiplex virtual socket, the virtual socket is marked as closed in block 497 and the close operation terminates in block 498.

Figure 13 will now be described as it relates to Figures 16-1 through 16-4 and Figures 17-1 and 17-2. When a real event occurs, block 302 of Figure 13 is exited and the socket manager examines the event based upon how the event was generated. If the event is the timing out of the multiplex socket activity timer which was set in block 496 of Figure 17-2, then the path from block 305 to block 312 is taken in Figure 13. As shown in Figure 13, the operations of block 312 and 313 are then carried out by the socket manager to close the multiplex real socket and delete the multiplex real socket which corresponds to the socket which connects the client-side intercept module to the server-side intercept module. The socket manager then waits for the next real event. This multiplex event timer is reset by the creation of a multiplex virtual socket as shown in block 322.

If the event occurring on the real socket is a real socket close such as the web server performing a close operation on the socket connections between the web server and the server-side intercept module, then the path from block 305 to block 309 of Figure 13 is taken. The socket manager removes the real socket from the real event list as



shown in block 309 and disconnects the virtual socket or sockets in the case of multiple multiplex sockets from the real socket or sockets as shown in block 310. The socket manager then marks the virtual socket as closing and signals a virtual event. This operation is reflected in block 311 and when all data is emptied from the virtual socket queue, the virtual socket will close. After marking the virtual socket as closing, the socket manager then determines whether or not the real socket, which is to be closed, is a simplex socket as shown in decision block 315. If the real socket closing is a simplex socket, then the real socket is closed and deleted as reflected in block 316. The socket manager then waits for the next real event as shown in block 302.

If it is not a simplex real socket which is being closed, then the "No" path of block 315 is taken and the socket manager then waits for the next real event. Thus, the multiplex real socket or the socket connecting the client-side intercept module and the server-side intercept module can only be closed by the timeout of the multiplex socket activity timer. This allows for the maintenance of the connection between the client-side intercept module and the server-side intercept module even after the last communication between the modules has occurred for a user specified predetermined time. In the event of a subsequent connection request from the browser prior to the timing out of the multiplex socket activity timer, the communication could be carried out without reestablishing the connection between the client-side intercept module and the server-side intercept module and thereby eliminating the need for the overhead of reestablishing such a connection.

The final path to be described of **Figure 13** is when a real event occurs and the event is the receipt of data on the multiplex real socket or sockets 36a or 36b in **Figure 12**. When data is received on the multiplex real sockets, this data is examined and in the event the data includes the

close operation indicator such as that added to a virtual queue in block 486 of **Figure 17-2**, then a virtual close operation is performed and the path from block 320 to block 310 is taken. The socket manager disconnects from the real socket the multiplex virtual socket identified in the data received on the real socket as shown in block 310 and then marks the virtual socket as "closing" and signals a virtual event as shown in block 311. Because the close is the close of a multiplex virtual socket, the "No" path out of block 315 is taken and the socket manager waits for another real event as shown in block 302.

Through carrying out the operations described in **Figures 13 through 17** a particular aspect of the present invention establishes a persistent connection between the first computer and the second computer over the external communication link. The persistent connection is maintained until all web browser originated communications are completed and a plurality of web browser originated communications are intercepted and multiplexed them onto the external communication link while the persistent connection is maintained. The client/server specific data stream may then be demultiplexed to create a plurality of HTTP data streams and the plurality of HTTP data streams are provided to the web server. The persistent connection is also maintained until all web server originated communications are completed. A plurality of web server originated communications are intercepted and multiplexed onto the external communication link while the persistent connection is maintained. Furthermore, the client/server specific data stream may be demultiplexed to create a plurality of HTTP data streams and the plurality of HTTP data streams provided to the web server.

In the drawings and specification, there have been disclosed typical preferred embodiments of the invention and, although specific terms are employed, these terms are used in a generic and descriptive sense only and not for

purposes of limitation, the scope of the invention being set forth in the following claims.

## THAT WHICH IS CLAIMED IS:

1. A method of increasing the performance of a web browser application resident on a first computer and communicating using the Hyper-Text Transfer Protocol (HTTP) with a web server application resident on a second computer remote from said first computer where at least one segment of the communication between the web browser application in the first computer and the web server application in the second computer occurs over an external communication link, the method comprising the following steps:

intercepting the HTTP data stream corresponding to a communication originated by the web browser prior to transmission of the HTTP data stream on the external communication link;

converting the HTTP data stream originated by the web browser from the HTTP protocol to a client/server specific communication protocol;

transmitting the converted web browser originated communication to the second computer over the external communication link as a client/server specific data stream;

receiving the client/server specific data stream transmitted over the external communication link;

reconstructing the HTTP data stream corresponding to the communication from the web browser from the client/server specific data stream received over the external communication link by converting the client/server specific data stream received in the client/server specific communication protocol to an HTTP data stream; and

providing the communication originated by the web browser to the web server as an HTTP data stream.

2. A method according to claim 1 wherein said transmitting step further comprises the step of transmitting the converted web browser originated communication over a wireless communication link.

3. A method according to claim 1 further comprising the following steps of:

storing an HTTP data stream to be received by the web browser in response to a web browser originated  
5 communication with the web server in a cache resident in the first computer to create a client cache entry corresponding to the web browser originated communication with the web server;

interrogating the web browser originated communication  
10 to determine if a client cache entry exists corresponding to the web browser originated communication; and

wherein the following step is substituted for said converting, transmitting, receiving, reconstructing and providing steps if a client cache entry exists corresponding  
15 to the web browser originated communication:

supplying the client cache entry to the web browser as an HTTP data stream in response to the web browser originated communication.

4. A method according to Claim 1 further comprising the steps of:

storing user defined information associated with a specific web browser information request so as to provide  
5 the user defined information resident in the first computer;

interrogating the web browser originated communication to determine if the information requested by the web browser corresponds to a request for which user defined information is stored;

wherein said providing step comprises providing the user defined stored information to the web browser as an HTTP data stream in response to the web browser originated communication if said interrogating step determines that the web browser communication corresponds to a request for which user defined information is stored; and  
10

wherein said interrogating step is substituted for said transmitting, said receiving, and said reconstructing steps  
15

if the user defined stored information is provided to the web browser.

5. A method according to Claim 4 further comprising the step of receiving the user defined information from the first computer across the communication link.

6. A method according to Claim 3 wherein said storing step further comprises the step of storing the time of creation of a client cache entry to create a client cache entry time record;

5 wherein said interrogating step further comprises the step of evaluating the client cache entry time record to determine if the client cache entry corresponding to the web browser originated communication was created within a predetermined client coherency time interval prior to the  
10 web browser requesting the information; and

wherein said providing step comprises providing the client cache entry to the web browser as an HTTP data stream in response to the web browser originated communication if said determining step determines that a client cache entry  
15 was created within a predetermined client coherency time interval prior to the web browser requesting the information.

7. A method according to Claim 6 further comprising the step of maintaining client cache entries across multiple instances of the web browser.

8. The method of Claim 1 further comprising the steps of:

5 storing the HTTP data stream received from the web server in response to a browser originated communication in a cache resident in the second computer to create a server request cache entry;

interrogating the web browser originated communication to determine if a server request cache entry corresponding to the browser originated communication has been previously stored in the cache; and

transforming the server request cache entry associated with the browser originated communication to a client/server specific communication protocol;

sending the transformed server request cache entry to the first computer over the external communication link as a client/server specific data stream;

acquiring the client/server specific data stream transmitted over the external communication link;

wherein said reconstructing step comprises reconstructing the HTTP data stream corresponding to the server request cache entry from the client/server specific data stream received over the external communication link by converting the client/server specific data stream received in the client/server specific communication protocol to an HTTP data stream; and

wherein said providing step comprises providing the server request cache entry to the information requested by the web browser to the web browser as an HTTP data stream.

9. The method of Claim 6 further comprising the steps of:

storing the HTTP data stream received from the web server in response to a browser originated communication in a cache resident in the second computer to create a server request cache entry;

evaluating the web browser originated communication to determine if a server request cache entry corresponding to the web browser originated communication exists; and

converting the server request cache entry corresponding to the browser originated communication to a client/server specific communication protocol;

15 sending the server request cache entry to the first  
computer over the external communication link by  
transmitting on the external communication link a  
client/server specific data stream;

acquiring the client/server specific data stream sent  
over the external communication link by the second computer;

20 wherein said reconstructing step comprises  
reconstructing the HTTP data stream corresponding to server  
cache entry from the client/server specific data stream  
received over the external communication link by converting  
the client/server specific data stream received in the  
client/server specific communication protocol to an HTTP  
25 data stream ; and

wherein said providing step comprises transferring the  
server request cache entry corresponding to the information  
requested by the web browser to the web browser as an HTTP  
data stream.

10. The method of Claim 9 further comprising the steps  
of:

5 determining if a server request cache entry  
corresponding to the web browser originated communication  
was created within a predetermined client coherency time  
interval prior to the second computer receiving the web  
browser originated communication; and

10 wherein said converting step comprises transforming the  
server request cache entry which corresponds to the web  
browser originated communication to a client/server specific  
communication protocol if said determining step determines  
that the server request cache entry was created within the  
predetermined client coherency time interval.

11. The method according to Claim 9 further comprising  
the steps of:

determining if a client cache entry exists  
corresponding to the web browser originated communication



5 which is identical to a server cache entry corresponding to the web browser originated communication; and

wherein said converting step comprises calculating the interval of time between when the second computer received the web browser originated communication and when the server request cache entry corresponding to the web browser originated communication was created to provide entry age data;

10 wherein said sending step comprises transmitting as the client/server specific protocol a coherent entry response to the first computer which includes the entry age data for the server cache entry corresponding to the web browser originated communication over the external communication link;

15 wherein said reconstructing step comprises updating the client cache entry time record corresponding to the web browser originated communication by subtracting from the current time of the first computer the entry age data received from the coherent entry response; and

20 wherein said providing step comprises transferring the HTTP data stream stored as the client cache entry corresponding to the web browser originated communication to the web browser as an HTTP data stream if a coherent entry response is received from the external communication link.

12. The method of Claim 1 wherein said web browser originated communication corresponds to a Common Gateway Interface (CGI) request, said method further comprising the steps of:

5 interrogating the intercepted CGI request to determine if a client base cache entry corresponding to the intercepted CGI request exists to provide a client CGI base form;

10 interrogating the intercepted CGI request to determine if a server base cache entry corresponding to the

intercepted CGI request exists to provide a server CGI base form;

15 storing a HTTP data stream received from the web server in response to a browser originated communication which is a request to a Common Gateway Interface (CGI) in a cache resident in the second computer to create a server base cache entry;

20 storing a HTTP data stream to be provided to the web browser in response to a browser originated communication which is a request to a CGI in a cache resident in the first computer to create a client base cache entry;

25 intercepting the HTTP data stream corresponding to the communication originated by the web server in response to the intercepted CGI request prior to transmission of the HTTP data stream on the external communication link;

comparing the intercepted CGI response to the server CGI base form to provide CGI difference data corresponding to the difference between the intercepted CGI response and the server CGI base form;

30 sending the CGI difference data to the web browser to the first computer over the external communication link as a client/server specific data stream;

35 acquiring the client/server specific data stream transmitted over the external communication link sent by the second computer;

40 wherein said reconstructing step comprises reconstructing the HTTP data stream corresponding to the communication from the web server from the client/server specific data stream received over the external communication link by combining the client CGI base form with the CGI difference data received over the external communication link to create an HTTP data stream corresponding to the intercepted CGI response; and

wherein said providing step comprises providing the communication originated by the web server to the web browser as an HTTP data stream.

13. The method of Claim 12 further comprises the steps of:

determining if the server CGI base form is identical to the client CGI base form;

5 wherein said sending step comprises transmitting the server CGI base form and transmitting the CGI difference data to the web browser over the external communication link as a client/server specific data stream;

10 wherein said reconstructing step comprises reconstructing the HTTP data stream corresponding to the communication from the web server from the client/server specific data stream received over the external communication link by combining the server CGI base form with the CGI difference data received over the external communication link to create an HTTP data stream

15 corresponding to the intercepted CGI response; and updating the client CGI base form corresponding to the CGI request by storing the received server CGI base form as the client base cache entry corresponding to the CGI request.

14. The method of Claim 12 further comprising the steps of:

5 determining if the difference between the server CGI base form and the CGI response is greater than a predefined difference threshold;

updating the server CGI base form corresponding to the CGI request by storing the CGI response received from the web server as the server base cache entry corresponding to the CGI request if the CGI difference data; and

10 wherein said comparing step and said sending step utilizes the updated server CGI base form.

15. The method of Claim 1 further comprising the steps of:

establishing a persistent connection between the first  
computer and the second computer over the external  
5 communication link;

maintaining the persistent connection until all web  
browser originated communications are completed; and

wherein said intercepting step comprises intercepting a  
plurality of web browser originated communications and  
10 multiplexing them onto the external communication link while  
the persistent connection is maintained.

16. The method of Claim 15:

wherein said reconstructing step comprises  
demultiplexing the client/server specific data stream to  
create a plurality of HTTP data streams; and

5 wherein said providing step comprises providing said  
plurality of HTTP data streams to the web server.

17. The method according to Claim 1 further comprising  
the step of:

providing to the second computer over the external  
communication link computer specific information  
5 corresponding to predefined characteristics of the first  
computer;

storing the computer specific information corresponding  
to the predefined characteristics of the first computer to  
provide browser header information;

10 wherein said transforming step comprises the step of  
removing the computer specific information from the web  
browser originated communication; and

wherein said reconstructing step comprises combining  
the browser header information with the client/server  
15 specific data stream to create an HTTP data stream.

18. A method according to claim 1 further comprising  
the steps of:

capturing the HTTP data stream corresponding to a  
communication originated by the web server prior to  
5 transmission of the HTTP data stream on the external  
communication link;

converting the HTTP data stream originated by the web  
server from the HTTP protocol to a client/server specific  
communication protocol;

10 sending the converted web server originated  
communication to the web browser over the external  
communication link as a client/server specific data stream;  
acquiring the client/server specific data stream  
transmitted over the external communication link;

15 rebuilding the HTTP data stream corresponding to the  
communication from the web server from the client/server  
specific data stream received over the external  
communication link by converting the client/server specific  
data stream received in the client/server specific  
20 communication protocol to an HTTP data stream; and

furnishing the communication originated by the web  
server to the web browser as an HTTP data stream.

19. The method according to Claim 18 further  
comprising the step of:

providing to the first computer over the external  
communication link computer specific information  
5 corresponding to predefined characteristics of the second  
computer;

storing the computer specific information corresponding  
to the predefined characteristics of the second computer to  
provide server header information;

10 wherein said converting step comprises the step of  
removing the computer specific information from the web  
server originated communication; and

wherein said rebuilding step comprises combining the  
server header information with the client/server specific  
data stream to create an HTTP data stream.

20. The method of Claim 18 further comprising the steps of:

5 establishing a persistent connection between the first computer and the second computer over the external communication link;

maintaining the persistent connection until all web server originated communications are completed; and

10 wherein said capturing step comprises intercepting a plurality of web server originated communications and multiplexing them onto the external communication link while the persistent connection is maintained.

21. The method of Claim 20:

wherein said rebuilding step comprises demultiplexing the client/server specific data stream to create a plurality of HTTP data streams; and

wherein said furnishing step comprises providing said plurality of HTTP data streams to the web server.

22. A method of increasing the performance of a web browser application resident on a first computer and communicating using the Hyper-Text Transfer Protocol (HTTP) with a web server application resident on a second computer remote from said first computer where at least one segment of the communication between the web browser application in the first computer and the web server application in the second computer occurs over an external communication link, the method comprising the following steps:

10 intercepting the HTTP data stream corresponding to a communication originated by the web browser prior to transmission of the HTTP data stream on the external communication link;

15 converting the HTTP data stream originated by the web browser from the HTTP protocol to a client/server specific communication protocol; and

transmitting the converted web browser originated communication to the web server over the external communication link as a client/server specific data stream.

23. A method according to claim 22 wherein said transmitting step further comprises the step of transmitting the converted web browser originated communication over a wireless communication link.

24. A method according to claim 22 further comprising the following steps of:

storing an HTTP data stream to be received by the web browser in response to a web browser originated communication with the web server in a cache resident in the first computer to create a client cache entry corresponding to the web browser originated communication with the web server;

interrogating the web browser originated communication to determine if a client cache entry exists corresponding to the web browser originated communication; and

wherein the following step is substituted for said converting and said transmitting, steps if a client cache entry exists corresponding to the web browser originated communication:

supplying the client cache entry to the web browser as an HTTP data stream in response to the web browser originated communication.

25. A method according to Claim 22 further comprising the steps of:

storing user defined information associated with a specific web browser information request so as to provide the user defined information resident in the first computer;

interrogating the web browser originated communication to determine if the information requested by the web browser

corresponds to a request for which user defined information is stored; and

10

providing the user defined stored information to the web browser as an HTTP data stream in response to the web browser originated communication if said interrogating step determines that the web browser communication corresponds to a request for which user defined information is stored; and

wherein said interrogating step and said providing step are substituted for said transmitting step if the user defined stored information is provided to the web browser.

26. A method according to Claim 25 further comprising the step of receiving the user defined information from the first computer across the communication link.

27. A method according to Claim 24 wherein said storing step further comprises the step of storing the time of creation of a client cache entry to create a client cache entry time record;

5

wherein said interrogating step further comprises the step of evaluating the client cache entry time record to determine if the client cache entry corresponding to the web browser originated communication was created within a predetermined client coherency time interval prior to the web browser requesting the information; and

10

wherein said supplying step comprises providing the client cache entry to the web browser as an HTTP data stream in response to the web browser originated communication if said determining step determines that a client cache entry was created within a predetermined client coherency time interval prior to the web browser requesting the information.

28. A method according to Claim 27 further comprising the step of maintaining client cache entries across multiple instances of the web browser.



29. The method of Claim 22 wherein said web browser originated communication corresponds to a Common Gateway Interface (CGI) request, said method further comprising the steps of:

5           interrogating the intercepted CGI request to determine if a client base cache entry corresponding to the intercepted CGI request exists to provide a client CGI base form;

10           storing a HTTP data stream to be provided to the web browser in response to a browser originated communication which is a request to a CGI in a cache resident in the first computer to create a client base cache entry;

15           reconstructing the HTTP data stream corresponding to a communication from the web server from the client/server specific data stream received over the external communication link by combining the client CGI base form with CGI difference data received over the external communication link to create an HTTP data stream; and

          providing the communication originated by the web server to the web browser as an HTTP data stream.

30. The method of Claim 29 further comprising the steps of:

          wherein said reconstructing step comprises reconstructing the HTTP data stream corresponding to a communication from the web server from the client/server specific data stream received over the external communication link by combining the a server CGI base form received over the external communication link with the CGI difference data received over the external communication link to create an HTTP data stream corresponding to the intercepted CGI response; and

5           reconstructing the HTTP data stream corresponding to a communication from the web server from the client/server specific data stream received over the external communication link by combining the a server CGI base form received over the external communication link with the CGI difference data received over the external communication link to create an HTTP data stream corresponding to the intercepted CGI response; and

10           reconstructing the HTTP data stream corresponding to a communication from the web server from the client/server specific data stream received over the external communication link by combining the a server CGI base form received over the external communication link with the CGI difference data received over the external communication link to create an HTTP data stream corresponding to the intercepted CGI response; and

          updating the client CGI base form corresponding to the CGI request by storing the received server CGI base form as the client base cache entry corresponding to the CGI request.

31. The method of Claim 22 further comprising the steps of:

establishing a persistent connection between the first computer and the second computer over the external

5 communication link;

maintaining the persistent connection until all web browser originated communications are completed; and

wherein said intercepting step comprises intercepting a plurality of web browser originated communications and multiplexing them onto the external communication link while the persistent connection is maintained.

32. The method of Claim 31 further comprising the steps of:

receiving a client/server specific data stream over the external communication link from the second computer;

5 demultiplexing the client/server specific data stream to create a plurality of HTTP data streams to provide the plurality of HTTP data streams to the web browser; and

providing the plurality of HTTP data streams to the web browser.

33. The method according to Claim 22 further comprising the step of:

providing to the second computer over the external communication link computer specific information

5 corresponding to predefined characteristics of the first computer; and

wherein said transforming step comprises the step of removing the computer specific information from the web browser originated communication.

34. A method of increasing the performance of a web browser application resident on a first computer and communicating using the Hyper-Text Transfer Protocol (HTTP) with a web server application resident on a second computer

5 remote from said first computer where at least one segment of the communication between the web browser application in the first computer and the web server application in the second computer occurs over an external communication link, the method comprising the following steps:

10 intercepting the HTTP data stream corresponding to a communication originated by the web server prior to transmission of the HTTP data stream on the external communication link;

15 converting the HTTP data stream originated by the web server from the HTTP protocol to a client/server specific communication protocol; and

transmitting the converted web server originated communication to the web browser over the external communication link as a client/server specific data stream.

35. The method of Claim 34 further comprising the steps of:

5 storing the HTTP data stream received from the web server in response to a browser originated communication in a cache resident in the second computer to create a server request cache entry;

10 interrogating the web browser originated communication to determine if a server request cache entry corresponding to the browser originated communication has been previously stored in the cache; and

wherein said converting step comprises transforming the server request cache entry associated with the browser originated communication to a client/server specific communication protocol; and

wherein said transmitting step comprises sending the transformed server request cache entry to the first computer over the external communication link as a client/server specific data stream.

36. The method of Claim 35 further comprising the steps of:

determining if a server request cache entry corresponding to the web browser originated communication was created within a predetermined client coherency time interval prior to the second computer receiving the web browser originated communication; and

wherein said transforming step comprises transforming the server request cache entry which corresponds to the web browser originated communication to a client/server specific communication protocol if said determining step determines that the server request cache entry was created with the predetermined client coherency time interval.

37. The method according to Claim 35 further comprising the steps of:

determining if a client cache entry exists corresponding to the web browser originated communication which is identical to a server cache entry corresponding to the web browser originated communication; and

wherein said transforming step comprises calculating the interval of time between when the second computer received the web browser originated communication and when the server request cache entry corresponding to the web browser originated communication was created to provide entry age data; and

wherein said transmitting step comprises transmitting as the client/server specific protocol a coherent entry response to the first computer which includes the entry age data for the server cache entry corresponding to the web browser originated communication over the external communication link.

38. The method of Claim 34 wherein said web browser originated communication corresponds to a Common Gateway

Interface (CGI) request, said method further comprising the steps of:

5           interrogating the intercepted CGI request to determine if a server base cache entry corresponding to the intercepted CGI request exists to provide a server CGI base form;

10           storing a HTTP data stream received from the web server in response to a browser originated communication which is a request to a Common Gateway Interface (CGI) in a cache resident in the second computer to create a server base cache entry;

15           wherein said intercepting step comprises intercepting the HTTP data stream corresponding to the communication originated by the web server in response to the intercepted CGI request prior to transmission of the HTTP data stream on the external communication link;

20           comparing the intercepted CGI response to the server CGI base form to provide CGI difference data corresponding to the difference between the intercepted CGI response and the server CGI base form; and

          wherein said transmitting step comprises sending the CGI difference data to the web browser over the external communication link as a client/server specific data stream.

39. The method of Claim 38 further comprises the steps of:

          determining if the server CGI base form is identical to a client CGI base form corresponding to the CGI request; and

          wherein said sending step comprises transmitting the server CGI base form and transmitting the CGI difference data to the web browser over the external communication link as a client/server specific data stream.

40. The method of Claim 38 further comprising the steps of:

determining if the difference between the server CGI  
base form and the CGI response is greater than a predefined  
5 difference threshold;

updating the server CGI base form corresponding to the  
CGI request by storing the CGI response received from the  
web server as the server base cache entry corresponding to  
the CGI request if the CGI difference data; and

wherein said comparing step and said sending step  
utilizes the updated server CGI base form.

41. The method of Claim 34 further comprising the  
steps of:

establishing a persistent connection between the first  
computer and the second computer over the external  
5 communication link;

maintaining the persistent connection until all web  
browser originated communications are completed; and

wherein said intercepting step comprises intercepting a  
plurality of web server originated communications and  
10 multiplexing them onto the external communication link while  
the persistent connection is maintained.

42. A method of increasing the performance of a  
client/server system having a client application resident  
on a first computer and communicating with a server  
application resident on a second computer remote from said  
5 first computer wherein said client application and said  
server application utilize a client/server independent  
communication protocol for communication between the client  
and the server and where at least one segment of the  
communication between the client application in the first  
10 computer and the server application in the second computer  
occurs over an external communication link, the method  
comprising the following steps:

intercepting communications in the client/server  
independent communication protocol originated by the remote

15 client prior to transmission of the communications on the  
external communication link;  
converting the communication originated by the client  
to a second client/server specific communication protocol;  
transmitting the converted communication over the  
20 external communication link;  
receiving the communication transmitted over the  
external communication link;  
converting the communication received over the external  
communication link from the client server specific  
25 communication protocol to the client/server independent  
communication protocol; and  
providing the communication originated by the remote  
client to the server in the client/server independent  
communication protocol.

43. The method of claim 42 further comprising:  
intercepting communications in the client/server  
independent communication protocol originated by the server  
prior to transmission of the communications on the external  
5 communication link;  
converting the communication originated by the server  
to a second client/server specific communication protocol;  
transmitting the converted communication over the  
external communication link;  
10 receiving the communication transmitted over the  
external communication link;  
converting the communication received over the external  
communication link from the client/server specific  
communication protocol to the client/server independent  
15 communication protocol; and  
providing the communication originated by the server to  
the remote client in the client/server independent  
communication protocol.

44. An apparatus for increasing the performance of a web browser application resident on a first computer and communicating using the Hyper-Text Transfer Protocol (HTTP) with a web server application resident on a second computer remote from said first computer where at least one segment of the communication between the web browser application in the first computer and the web server application in the second computer occurs over an external communication link, the apparatus comprising:

means for intercepting the HTTP data stream corresponding to a communication originated by the web browser prior to transmission of the HTTP data stream on the external communication link;

means for converting the HTTP data stream originated by the web browser from the HTTP protocol to a client/server specific communication protocol;

means for transmitting the converted web browser originated communication to the second computer over the external communication link as a client/server specific data stream;

means for receiving the client/server specific data stream transmitted over the external communication link;

means for reconstructing the HTTP data stream corresponding to the communication from the web browser from the client/server specific data stream received over the external communication link by converting the client/server specific data stream received in the client/server specific communication protocol to an HTTP data stream; and

means for providing the communication originated by the web browser to the web server as an HTTP data stream.

45. An apparatus according to claim 44 wherein said means for transmitting further comprises means for transmitting the converted web browser originated communication over a wireless communication link.



46. An apparatus according to claim 44 further comprising:

5 means for storing an HTTP data stream to be received by the web browser in response to a web browser originated communication with the web server in a cache resident in the first computer to create a client cache entry corresponding to the web browser originated communication with the web server;

10 means for interrogating the web browser originated communication to determine if a client cache entry exists corresponding to the web browser originated communication; and

means for supplying the client cache entry to the web browser as an HTTP data stream in response to the web browser originated communication.

47. An apparatus according to Claim 44 further comprising:

5 storing user defined information associated with a specific web browser information request so as to provide the user defined information resident in the first computer;

means for interrogating the web browser originated communication to determine if the information requested by the web browser corresponds to a request for which user defined information is stored; and

wherein said means for providing comprises means for providing the user defined stored information to the web browser as an HTTP data stream in response to the web browser originated communication if said means for interrogating determines that the web browser communication corresponds to a request for which user defined information is stored.

48. An apparatus according to Claim 46 further comprising means for receiving the user defined information from the first computer across the communication link.

49. An apparatus according to Claim 44 wherein said means for storing further comprises means for storing the time of creation of a client cache entry to create a client cache entry time record;

5 wherein said means for interrogating further comprises means for evaluating the client cache entry time record to determine if the client cache entry corresponding to the web browser originated communication was created within a predetermined client coherency time interval prior to the  
10 web browser requesting the information; and

wherein said means for providing comprises means for providing the client cache entry to the web browser as an HTTP data stream in response to the web browser originated communication if said means for determining determines that  
15 a client cache entry was created within a predetermined client coherency time interval prior to the web browser requesting the information.

50. An apparatus according to Claim 49 further comprising means for maintaining client cache entries across multiple instances of the web browser.

51. An apparatus according to Claim 44 further comprising:

means for storing the HTTP data stream received from the web server in response to a browser originated  
5 communication in a cache resident in the second computer to create a server request cache entry;

means for interrogating the web browser originated communication to determine if a server request cache entry corresponding to the browser originated communication has  
10 been previously stored in the cache; and

means for transforming the server request cache entry associated with the browser originated communication to a client/server specific communication protocol;

15 means for sending the transformed server request cache entry to the first computer over the external communication link as a client/server specific data stream;

means for acquiring the client/server specific data stream transmitted over the external communication link;

20 wherein said means for reconstructing comprises means for reconstructing the HTTP data stream corresponding to the server request cache entry from the client/server specific data stream received over the external communication link by converting the client/server specific data stream received in the client/server specific communication protocol to an  
25 HTTP data stream ; and

wherein said means for providing comprises means for providing the server request cache entry to the information requested by the web browser to the web browser as an HTTP data stream.

52. An apparatus according to Claim 49 further comprising:

5 means for storing the HTTP data stream received from the web server in response to a browser originated communication in a cache resident in the second computer to create a server request cache entry;

10 means for evaluating the web browser originated communication to determine if a server request cache entry corresponding to the web browser originated communication exists; and

means for converting the server request cache entry corresponding to the browser originated communication to a client/server specific communication protocol;

15 means for sending the server request cache entry to the first computer over the external communication link by transmitting on the external communication link a client/server specific data stream;

means for acquiring the client/server specific data stream sent over the external communication link by the second computer;

wherein said means for reconstructing comprises means for reconstructing the HTTP data stream corresponding to server cache entry from the client/server specific data stream received over the external communication link by converting the client/server specific data stream received in the client/server specific communication protocol to an HTTP data stream ; and

wherein said means for providing comprises means for transferring the server request cache entry corresponding to the information requested by the web browser to the web browser as an HTTP data stream.

53. An apparatus according to Claim 52 further comprising:

means for determining if a server request cache entry corresponding to the web browser originated communication was created within a predetermined client coherency time interval prior to the second computer receiving the web browser originated communication; and

wherein said means for converting comprises means for transforming the server request cache entry which corresponds to the web browser originated communication to a client/server specific communication protocol if said means for determining determines that the server request cache entry was created within the predetermined client coherency time interval.

54. An apparatus according to Claim 52 further comprising:

means for determining if a client cache entry exists corresponding to the web browser originated communication which is identical to a server cache entry corresponding to the web browser originated communication; and

wherein said means for converting comprises means for calculating the interval of time between when the second computer received the web browser originated communication and when the server request cache entry corresponding to the web browser originated communication was created to provide entry age data;

wherein said means for sending comprises means for transmitting as the client/server specific protocol a coherent entry response to the first computer which includes the entry age data for the server cache entry corresponding to the web browser originated communication over the external communication link;

wherein said means for reconstructing comprises means for updating the client cache entry time record corresponding to the web browser originated communication by subtracting from the current time of the first computer the entry age data received from the coherent entry response; and

wherein said means for providing comprises means for transferring the HTTP data stream stored as the client cache entry corresponding to the web browser originated communication to the web browser as an HTTP data stream if a coherent entry response is received from the external communication link.

55. An apparatus according to Claim 44 wherein the intercepted web browser originated communication is an intercepted CGI request further comprising:

means for interrogating the intercepted CGI request to determine if a client base cache entry corresponding to the intercepted CGI request exists to provide a client CGI base form;

means for interrogating the intercepted CGI request to determine if a server base cache entry corresponding to the intercepted CGI request exists to provide a server CGI base form;

means for storing a HTTP data stream received from the web server in response to a browser originated communication which is a request to a Common Gateway Interface (CGI) in a cache resident in the second computer to create a server base cache entry;

means for storing a HTTP data stream to be provided to the web browser in response to a browser originated communication which is a request to a CGI in a cache resident in the first computer to create a client base cache entry;

means for intercepting the HTTP data stream corresponding to the communication originated by the web server in response to the intercepted CGI request prior to transmission of the HTTP data stream on the external communication link;

means for comparing the intercepted CGI response to the server CGI base form to provide CGI difference data corresponding to the difference between the intercepted CGI response and the server CGI base form;

means for sending the CGI difference data to the web browser to the first computer over the external communication link as a client/server specific data stream;

means for acquiring the client/server specific data stream transmitted over the external communication link sent by the second computer;

wherein said means for reconstructing comprises means for reconstructing the HTTP data stream corresponding to the communication from the web server from the client/server specific data stream received over the external communication link by combining the client CGI base form with the CGI difference data received over the external communication link to create an HTTP data stream corresponding to the intercepted CGI response; and

wherein said means for providing comprises means for providing the communication originated by the web server to the web browser as an HTTP data stream.

56. An apparatus according to Claim 55 further comprising:

means for determining if the server CGI base form is identical to the client CGI base form;

5 wherein said means for sending comprises means for transmitting the server CGI base form and transmitting the CGI difference data to the web browser over the external communication link as a client/server specific data stream;

10 wherein said means for reconstructing comprises means for reconstructing the HTTP data stream corresponding to the communication from the web server from the client/server specific data stream received over the external communication link by combining the server CGI base form with the CGI difference data received over the external communication link to create an HTTP data stream

15 corresponding to the intercepted CGI response; and means for updating the client CGI base form corresponding to the CGI request by storing the received server CGI base form as the client base cache entry

20 corresponding to the CGI request.

57. An apparatus according to Claim 56 further comprising:

means for determining if the difference between the server CGI base form and the CGI response is greater than a predefined difference threshold;

5 means for updating the server CGI base form corresponding to the CGI request by storing the CGI response received from the web server as the server base cache entry corresponding to the CGI request if said determining means

10 determines that the difference between the server CGI base form and the CGI response is greater than the predefined difference threshold; and

wherein said means for comparing and said means for sending utilize the updated server CGI base form.

58. An apparatus according to Claim 44 further comprising:

means for establishing a persistent connection between the first computer and the second computer over the external communication link;

means for maintaining the persistent connection until all web browser originated communications are completed; and

wherein said means for intercepting comprises means for intercepting a plurality of web browser originated communications and multiplexing them onto the external communication link while the persistent connection is maintained.

59. An apparatus according to Claim 58 wherein said means for reconstructing comprises means for demultiplexing the client/server specific data stream to create a plurality of HTTP data streams and wherein said means for providing comprises means for providing said plurality of HTTP data streams to the web server.

60. An apparatus according to Claim 44 further comprising:

means for providing to the second computer over the external communication link computer specific information corresponding to predefined characteristics of the first computer;

means for storing the computer specific information corresponding to the predefined characteristics of the first computer to provide browser header information;

wherein said means for transforming comprises means for removing the computer specific information from the web browser originated communication; and

wherein said means for reconstructing comprises means for combining the browser header information with the client/server specific data stream to create an HTTP data stream.



61. An apparatus according to Claim 44 further comprising:

means for capturing the HTTP data stream corresponding to a communication originated by the web server prior to transmission of the HTTP data stream on the external communication link;

means for converting the HTTP data stream originated by the web server from the HTTP protocol to a client/server specific communication protocol;

means for sending the converted web server originated communication to the web browser over the external communication link as a client/server specific data stream;

means for acquiring the client/server specific data stream transmitted over the external communication link;

means for rebuilding the HTTP data stream corresponding to the communication from the web server from the client/server specific data stream received over the external communication link by converting the client/server specific data stream received in the client/server specific communication protocol to an HTTP data stream ; and

means for furnishing the communication originated by the web server to the web browser as an HTTP data stream.

62. An apparatus according to Claim 61 further comprising:

means for providing to the first computer over the external communication link computer specific information corresponding to predefined characteristics of the second computer;

means for storing the computer specific information corresponding to the predefined characteristics of the second computer to provide server header information;

wherein said means for converting comprises means for removing the computer specific information from the web server originated communication; and

15 wherein said means for rebuilding comprises means for  
combining the server header information with the  
client/server specific data stream to create an HTTP data  
stream.

63. An apparatus according to Claim 61 further  
comprising:

5 means for establishing a persistent connection between  
the first computer and the second computer over the external  
communication link;

means for maintaining the persistent connection until  
all web server originated communications are completed; and

10 wherein said means for capturing comprises means for  
intercepting a plurality of web server originated  
communications and multiplexing them onto the external  
communication link while the persistent connection is  
maintained.

64. An apparatus according to Claim 63 wherein said  
means for rebuilding comprises means for demultiplexing the  
client/server specific data stream to create a plurality of  
HTTP data streams, and wherein said means for furnishing  
5 comprises means for providing said plurality of HTTP data  
streams to the web server.

65. An apparatus for increasing the performance of a  
web browser application resident on a first computer and  
communicating using the Hyper-Text Transfer Protocol (HTTP)  
with a web server application resident on a second computer  
5 remote from said first computer where at least one segment  
of the communication between the web browser application in  
the first computer and the web server application in the  
second computer occurs over an external communication link,  
comprising:

10 means for intercepting the HTTP data stream  
corresponding to a communication originated by the web

browser prior to transmission of the HTTP data stream on the external communication link;

15 means for converting the HTTP data stream originated by the web browser from the HTTP protocol to a client/server specific communication protocol; and

means for transmitting the converted web browser originated communication to the web server over the external communication link as a client/server specific data stream.

66. An apparatus according to Claim 65 wherein said means for transmitting comprises means for transmitting the converted web browser originated communication over a wireless communication link.

67. An apparatus according to claim 65 further comprising:

5 means for storing an HTTP data stream to be received by the web browser in response to a web browser originated communication with the web server in a cache resident in the first computer to create a client cache entry corresponding to the web browser originated communication with the web server;

10 means for interrogating the web browser originated communication to determine if a client cache entry exists corresponding to the web browser originated communication; and

15 means for supplying the client cache entry to the web browser as an HTTP data stream in response to the web browser originated communication.

68. An apparatus according to Claim 65 further comprising:

5 means for storing user defined information associated with a specific web browser information request so as to provide the user defined information resident in the first computer;

means for interrogating the web browser originated communication to determine if the information requested by the web browser corresponds to a request for which user defined information is stored; and

means for providing the user defined stored information to the web browser as an HTTP data stream in response to the web browser originated communication if said means for interrogating determines that the web browser communication corresponds to a request for which user defined information is stored.

69. An apparatus according to Claim 68 further comprising means for receiving the user defined information from the first computer across the communication link.

70. An apparatus according to Claim 67 wherein said means for storing further comprises means for storing the time of creation of a client cache entry to create a client cache entry time record;

wherein said means for interrogating further comprises means for evaluating the client cache entry time record to determine if the client cache entry corresponding to the web browser originated communication was created within a predetermined client coherency time interval prior to the web browser requesting the information; and

wherein said means for supplying comprises means for providing the client cache entry to the web browser as an HTTP data stream in response to the web browser originated communication if means for determining determines that a client cache entry was created within a predetermined client coherency time interval prior to the web browser requesting the information.

71. An apparatus according to Claim 70 further comprising means for maintaining client cache entries across multiple instances of the web browser.

72. An apparatus according to Claim 65 wherein said web browser originated communication corresponds to a Common Gateway Interface (CGI) request, said apparatus further comprising:

5 means for interrogating the intercepted CGI request to determine if a client base cache entry corresponding to the intercepted CGI request exists to provide a client CGI base form;

10 means for storing a HTTP data stream to be provided to the web browser in response to a browser originated communication which is a request to a CGI in a cache resident in the first computer to create a client base cache entry;

15 means for reconstructing the HTTP data stream corresponding to a communication from the web server from the client/server specific data stream received over the external communication link by combining the client CGI base form with CGI difference data received over the external communication link to create an HTTP data stream; and

20 means for providing the communication originated by the web server to the web browser as an HTTP data stream.

73. An apparatus according to Claim 72 further comprising:

5 wherein said means for reconstructing comprises means for reconstructing the HTTP data stream corresponding to a communication from the web server from the client/server specific data stream received over the external communication link by combining the a server CGI base form received over the external communication link with the CGI difference data received over the external communication link to create an HTTP data stream corresponding to the intercepted CGI response; and

10 means for updating the client CGI base form corresponding to the CGI request by storing the received

server CGI base form as the client base cache entry  
15 corresponding to the CGI request.

74. An apparatus according to Claim 65 further  
comprising:

means for establishing a persistent connection between  
the first computer and the second computer over the external  
5 communication link;

means for maintaining the persistent connection until  
all web browser originated communications are completed; and

wherein said means for intercepting comprises means  
for intercepting a plurality of web browser originated  
10 communications and multiplexing them onto the external  
communication link while the persistent connection is  
maintained.

75. An apparatus according to Claim 74 further  
comprising:

means for receiving a client/server specific data  
stream over the external communication link from the second  
5 computer;

means for demultiplexing the client/server specific  
data stream to create a plurality of HTTP data streams to  
provide the plurality of HTTP data streams to the web  
browser; and

10 means for providing the plurality of HTTP data streams  
to the web browser.

76. An apparatus according to Claim 65 further  
comprising:

means for providing to the second computer over the  
external communication link computer specific information  
5 corresponding to predefined characteristics of the first  
computer; and

wherein said means for transforming comprises means for removing the computer specific information from the web browser originated communication.

77. An apparatus for increasing the performance of a web browser application resident on a first computer and communicating using the Hyper-Text Transfer Protocol (HTTP) with a web server application resident on a second computer remote from said first computer where at least one segment of the communication between the web browser application in the first computer and the web server application in the second computer occurs over an external communication link, the apparatus comprising:

means for intercepting the HTTP data stream corresponding to a communication originated by the web server prior to transmission of the HTTP data stream on the external communication link;

means for converting the HTTP data stream originated by the web server from the HTTP protocol to a client/server specific communication protocol; and

means for transmitting the converted web server originated communication to the web browser over the external communication link as a client/server specific data stream.

78. An apparatus according to Claim 77 further comprising:

means for storing the HTTP data stream received from the web server in response to a browser originated communication in a cache resident in the second computer to create a server request cache entry;

means for interrogating the web browser originated communication to determine if a server request cache entry corresponding to the browser originated communication has been previously stored in the cache; and

wherein said means for converting comprises means for transforming the server request cache entry associated with the browser originated communication to a client/server specific communication protocol; and

15 wherein said means for transmitting comprises means for sending the transformed server request cache entry to the first computer over the external communication link as a client/server specific data stream.

79. An apparatus according to Claim 78 further comprising:

5 means for determining if a server request cache entry corresponding to the web browser originated communication was created within a predetermined client coherency time interval prior to the second computer receiving the web browser originated communication; and

10 wherein said means for transforming comprises means for transforming the server request cache entry which corresponds to the web browser originated communication to a client/server specific communication protocol if said means for determining determines that the server request cache entry was created with the predetermined client coherency time interval.

80. An apparatus according to Claim 78 further comprising:

5 means for determining if a client cache entry exists corresponding to the web browser originated communication which is identical to a server cache entry corresponding to the web browser originated communication; and

10 wherein said means for transforming comprises means for calculating the interval of time between when the second computer received the web browser originated communication and when the server request cache entry corresponding to the web browser originated communication was created to provide entry age data; and



wherein said means for transmitting comprises means for transmitting as the client/server specific protocol a coherent entry response to the first computer which includes the entry age data for the server cache entry corresponding to the web browser originated communication over the external communication link.

81. An apparatus according to Claim 77 wherein said web browser originated communication corresponds to a Common Gateway Interface (CGI) request, said apparatus further comprising:

means for interrogating the intercepted CGI request to determine if a server base cache entry corresponding to the intercepted CGI request exists to provide a server CGI base form;

means for storing a HTTP data stream received from the web server in response to a browser originated communication which is a request to a Common Gateway Interface (CGI) in a cache resident in the second computer to create a server base cache entry;

wherein said means for intercepting comprises intercepting the HTTP data stream corresponding to the communication originated by the web server in response to the intercepted CGI request prior to transmission of the HTTP data stream on the external communication link;

means for comparing the intercepted CGI response to the server CGI base form to provide CGI difference data corresponding to the difference between the intercepted CGI response and the server CGI base form; and

wherein said means for transmitting comprises means for sending the CGI difference data to the web browser over the external communication link as a client/server specific data stream.

82. An apparatus according to Claim 81 further comprising:

means for determining if the server CGI base form is identical to a client CGI base form corresponding to the CGI request; and

wherein said means for transmitting comprises means for transmitting the server CGI base form and transmitting the CGI difference data to the web browser over the external communication link as a client/server specific data stream.

83. An apparatus according to Claim 81 further comprising:

means for determining if the difference between the server CGI base form and the CGI response is greater than a predefined difference threshold;

means for updating the server CGI base form corresponding to the CGI request by storing the CGI response received from the web server as the server base cache entry corresponding to the CGI request if said means for determining determines that the difference between the server CGI base form and the CGI response is greater than a predefined difference threshold; and

wherein said means for comparing and said means for sending utilizes the updated server CGI base form.

84. An apparatus according to Claim 77 further comprising:

means for establishing a persistent connection between the first computer and the second computer over the external communication link;

means for maintaining the persistent connection until all web browser originated communications are completed; and

wherein said means for intercepting comprises means for intercepting a plurality of web server originated communications and multiplexing them onto the external communication link while the persistent connection is maintained.

85. An apparatus for increasing the performance of a client/server system having a client application resident on a first computer and communicating with a server application resident on a second computer remote from said first computer wherein said client application and said server application utilize a client/server independent communication protocol for communication between the client and the server and where at least one segment of the communication between the client application in the first computer and the server application in the second computer occurs over an external communication link, the apparatus comprising:

means for intercepting communications in the client/server independent communication protocol originated by the remote client prior to transmission of the communications on the external communication link;

means for converting the communication originated by the client to a second client/server specific communication protocol;

means for transmitting the converted communication over the external communication link;

means for receiving the communication transmitted over the external communication link;

means for converting the communication received over the external communication link from the client server specific communication protocol to the client/server independent communication protocol; and

means for providing the communication originated by the remote client to the server in the client/server independent communication protocol.

86. An apparatus according to claim 85 further comprising:

means for intercepting communications in the client/server independent communication protocol originated

5 by the server prior to transmission of the communications on  
the external communication link;  
means for converting the communication originated by  
the server to a second client/server specific communication  
10 protocol;  
means for transmitting the converted communication over  
the external communication link;  
means for receiving the communication transmitted over  
the external communication link;  
means for converting the communication received over  
15 the external communication link from the client/server  
specific communication protocol to the client/server  
independent communication protocol; and  
means for providing the communication originated by the  
server to the remote client in the client/server independent  
20 communication protocol.

87. A computer program product for increasing the  
performance of a web browser application resident on a first  
computer and communicating using the Hyper-Text Transfer  
Protocol (HTTP) with a web server application resident on a  
5 second computer remote from said first computer where at  
least one segment of the communication between the web  
browser application in the first computer and the web server  
application in the second computer occurs over an external  
communication link comprising:

10 A computer-readable storage medium having computer-  
readable program code means embodied in said medium, said  
computer-readable program code means comprising:  
computer-readable program code means for intercepting  
the HTTP data stream corresponding to a communication  
15 originated by the web browser prior to transmission of the  
HTTP data stream on the external communication link;  
computer readable program code means for converting the  
HTTP data stream originated by the web browser from the HTTP  
protocol to a client/server specific communication protocol;

20 computer readable program code means for transmitting  
the converted web browser originated communication to the  
second computer over the external communication link as a  
client/server specific data stream;

25 computer readable program code means for receiving the  
client/server specific data stream transmitted over the  
external communication link;

30 computer readable program code means for reconstructing  
the HTTP data stream corresponding to the communication from  
the web browser from the client/server specific data stream  
received over the external communication link by converting  
the client/server specific data stream received in the  
client/server specific communication protocol to an HTTP  
data stream; and

35 computer readable program code means for providing the  
communication originated by the web browser to the web  
server as an HTTP data stream.

88. A computer program product according to Claim 87  
wherein said computer readable program code means for  
transmitting further comprises computer readable program  
code means for transmitting the converted web browser  
5 originated communication over a wireless communication link.

89. A computer program product according to claim 87  
further comprising:

5 computer readable program code means for storing an  
HTTP data stream to be received by the web browser in  
response to a web browser originated communication with the  
web server in a cache resident in the first computer to  
create a client cache entry corresponding to the web browser  
originated communication with the web server;

10 computer readable program code means for interrogating  
the web browser originated communication to determine if a  
client cache entry exists corresponding to the web browser  
originated communication; and

computer readable program code means for supplying the client cache entry to the web browser as an HTTP data stream in response to the web browser originated communication.

90. A computer program product according to Claim 87 further comprising:

storing user defined information associated with a specific web browser information request so as to provide the user defined information resident in the first computer;

computer readable program code means for interrogating the web browser originated communication to determine if the information requested by the web browser corresponds to a request for which user defined information is stored; and

wherein said computer readable program code means for providing comprises computer readable program code means for providing the user defined stored information to the web browser as an HTTP data stream in response to the web browser originated communication if said computer readable program code means for interrogating determines that the web browser communication corresponds to a request for which user defined information is stored.

91. A computer program product according to Claim 90 further comprising computer readable program code means for receiving the user defined information from the first computer across the communication link.

92. A computer program product according to Claim 87 wherein said computer readable program code means for storing further comprises computer readable program code means for storing the time of creation of a client cache entry to create a client cache entry time record;

wherein said computer readable program code means for interrogating further comprises computer readable program code means for evaluating the client cache entry time record to determine if the client cache entry corresponding to the

10 web browser originated communication was created within a  
predetermined client coherency time interval prior to the  
web browser requesting the information; and

wherein said computer readable program code means for  
providing comprises computer readable program code means for  
15 providing the client cache entry to the web browser as an  
HTTP data stream in response to the web browser originated  
communication if said computer readable program code means  
for determining determines that a client cache entry was  
created within a predetermined client coherency time  
20 interval prior to the web browser requesting the  
information.

93. A computer program product according to Claim 92  
further comprising computer readable program code means for  
maintaining client cache entries across multiple instances  
of the web browser.

94. A computer program product according to Claim 87  
further comprising:

computer readable program code means for storing the  
HTTP data stream received from the web server in response to  
5 a browser originated communication in a cache resident in  
the second computer to create a server request cache entry;

computer readable program code means for interrogating  
the web browser originated communication to determine if a  
server request cache entry corresponding to the browser  
10 originated communication has been previously stored in the  
cache; and

computer readable program code means for transforming  
the server request cache entry associated with the browser  
originated communication to a client/server specific  
15 communication protocol;

computer readable program code means for sending the  
transformed server request cache entry to the first computer

over the external communication link as a client/server specific data stream;

20 computer readable program code means for acquiring the client/server specific data stream transmitted over the external communication link;

wherein said computer readable program code means for reconstructing comprises computer readable program code  
25 means for reconstructing the HTTP data stream corresponding to the server request cache entry from the client/server specific data stream received over the external communication link by converting the client/server specific data stream received in the client/server specific  
30 communication protocol to an HTTP data stream ; and

wherein said computer readable program code means for providing comprises computer readable program code means for providing the server request cache entry to the information requested by the web browser to the web browser as an HTTP  
35 data stream.

95. A computer program product according to Claim 92 further comprising:

computer readable program code means for storing the HTTP data stream received from the web server in response to  
5 a browser originated communication in a cache resident in the second computer to create a server request cache entry;

computer readable program code means for evaluating the web browser originated communication to determine if a server request cache entry corresponding to the web browser  
10 originated communication exists; and

computer readable program code means for converting the server request cache entry corresponding to the browser originated communication to a client/server specific communication protocol;

15 computer readable program code means for sending the server request cache entry to the first computer over the



external communication link by transmitting on the external communication link a client/server specific data stream;

computer readable program code means for acquiring the client/server specific data stream sent over the external communication link by the second computer;

wherein said computer readable program code means for reconstructing comprises computer readable program code means for reconstructing the HTTP data stream corresponding to server cache entry from the client/server specific data stream received over the external communication link by converting the client/server specific data stream received in the client/server specific communication protocol to an HTTP data stream ; and

wherein said computer readable program code means for providing comprises computer readable program code means for transferring the server request cache entry corresponding to the information requested by the web browser to the web browser as an HTTP data stream.

96. A computer program product according to Claim 95 further comprising:

computer readable program code means for determining if a server request cache entry corresponding to the web browser originated communication was created within a predetermined client coherency time interval prior to the second computer receiving the web browser originated communication; and

wherein said computer readable program code means for converting comprises computer readable program code means for transforming the server request cache entry which corresponds to the web browser originated communication to a client/server specific communication protocol if said computer readable program code means for determining determines that the server request cache entry was created within the predetermined client coherency time interval.

97. A computer program product according to Claim 95 further comprising:

computer readable program code means for determining if a client cache entry exists corresponding to the web browser originated communication which is identical to a server cache entry corresponding to the web browser originated communication; and

wherein said computer readable program code means for converting comprises computer readable program code means for calculating the interval of time between when the second computer received the web browser originated communication and when the server request cache entry corresponding to the web browser originated communication was created to provide entry age data;

wherein said computer readable program code means for sending comprises computer readable program code means for transmitting as the client/server specific protocol a coherent entry response to the first computer which includes the entry age data for the server cache entry corresponding to the web browser originated communication over the external communication link;

wherein said computer readable program code means for reconstructing comprises computer readable program code means for updating the client cache entry time record corresponding to the web browser originated communication by subtracting from the current time of the first computer the entry age data received from the coherent entry response; and

wherein said computer readable program code means for providing comprises computer readable program code means for transferring the HTTP data stream stored as the client cache entry corresponding to the web browser originated communication to the web browser as an HTTP data stream if a coherent entry response is received from the external communication link.

98. A computer program product according to Claim 87 wherein the intercepted web browser originated communication is an intercepted CGI request further comprising:

5 computer readable program code means for interrogating the intercepted CGI request to determine if a client base cache entry corresponding to the intercepted CGI request exists to provide a client CGI base form;

10 computer readable program code means for interrogating the intercepted CGI request to determine if a server base cache entry corresponding to the intercepted CGI request exists to provide a server CGI base form;

15 computer readable program code means for storing a HTTP data stream received from the web server in response to a browser originated communication which is a request to a Common Gateway Interface (CGI) in a cache resident in the second computer to create a server base cache entry;

20 computer readable program code means for storing a HTTP data stream to be provided to the web browser in response to a browser originated communication which is a request to a CGI in a cache resident in the first computer to create a client base cache entry;

25 computer readable program code means for intercepting the HTTP data stream corresponding to the communication originated by the web server in response to the intercepted CGI request prior to transmission of the HTTP data stream on the external communication link;

30 computer readable program code means for comparing the intercepted CGI response to the server CGI base form to provide CGI difference data corresponding to the difference between the intercepted CGI response and the server CGI base form;

35 computer readable program code means for sending the CGI difference data to the web browser to the first computer over the external communication link as a client/server specific data stream;

computer readable program code means for acquiring the client/server specific data stream transmitted over the external communication link sent by the second computer;

40 wherein said computer readable program code means for reconstructing comprises computer readable program code means for reconstructing the HTTP data stream corresponding to the communication from the web server from the client/server specific data stream received over the external communication link by combining the client CGI base form with the CGI difference data received over the external communication link to create an HTTP data stream

45 corresponding to the intercepted CGI response; and

50 wherein said computer readable program code means for providing comprises computer readable program code means for providing the communication originated by the web server to the web browser as an HTTP data stream.

99. A computer program product according to Claim 98 further comprising:

5 computer readable program code means for determining if the server CGI base form is identical to the client CGI base form;

10 wherein said computer readable program code means for sending comprises computer readable program code means for transmitting the server CGI base form and transmitting the CGI difference data to the web browser over the external communication link as a client/server specific data stream;

15 wherein said computer readable program code means for reconstructing comprises computer readable program code means for reconstructing the HTTP data stream corresponding to the communication from the web server from the client/server specific data stream received over the external communication link by combining the server CGI base form with the CGI difference data received over the external communication link to create an HTTP data stream corresponding to the intercepted CGI response; and

20 computer readable program code means for updating the client CGI base form corresponding to the CGI request by storing the received server CGI base form as the client base cache entry corresponding to the CGI request.

100. A computer program product according to Claim 99 further comprising:

computer readable program code means for determining if the difference between the server CGI base form and the CGI response is greater than a predefined difference threshold;

5 computer readable program code means for updating the server CGI base form corresponding to the CGI request by storing the CGI response received from the web server as the server base cache entry corresponding to the CGI request if  
10 said determining computer readable program code means determines that the difference between the server CGI base form and the CGI response is greater than the predefined difference threshold; and

15 wherein said computer readable program code means for comparing and said computer readable program code means for sending utilize the updated server CGI base form.

101. A computer program product according to Claim 87 further comprising:

computer readable program code means for establishing a persistent connection between the first computer and the second computer over the external communication link;

5 computer readable program code means for maintaining the persistent connection until all web browser originated communications are completed; and

wherein said computer readable program code means for intercepting comprises computer readable program code means for intercepting a plurality of web browser originated communications and multiplexing them onto the external communication link while the persistent connection is maintained.

102. A computer program product according to Claim 101 wherein said computer readable program code means for reconstructing comprises computer readable program code means for demultiplexing the client/server specific data stream to create a plurality of HTTP data streams and  
5 wherein said computer readable program code means for providing comprises computer readable program code means for providing said plurality of HTTP data streams to the web server.

103. A computer program product according to Claim 87 further comprising:

computer readable program code means for providing to the second computer over the external communication link  
5 computer specific information corresponding to predefined characteristics of the first computer;

computer readable program code means for storing the computer specific information corresponding to the predefined characteristics of the first computer to provide  
10 browser header information;

wherein said computer readable program code means for transforming comprises computer readable program code means for removing the computer specific information from the web browser originated communication; and

15 wherein said computer readable program code means for reconstructing comprises computer readable program code means for combining the browser header information with the client/server specific data stream to create an HTTP data stream.

104. A computer program product according to Claim 87 further comprising:

5 computer readable program code means for capturing the HTTP data stream corresponding to a communication originated by the web server prior to transmission of the HTTP data stream on the external communication link;

computer readable program code means for converting the HTTP data stream originated by the web server from the HTTP protocol to a client/server specific communication protocol;

10 computer readable program code means for sending the converted web server originated communication to the web browser over the external communication link as a client/server specific data stream;

15 computer readable program code means for acquiring the client/server specific data stream transmitted over the external communication link;

20 computer readable program code means for rebuilding the HTTP data stream corresponding to the communication from the web server from the client/server specific data stream received over the external communication link by converting the client/server specific data stream received in the client/server specific communication protocol to an HTTP data stream; and

25 computer readable program code means for furnishing the communication originated by the web server to the web browser as an HTTP data stream.

105. A computer program product according to Claim 104 further comprising:

5 computer readable program code means for providing to the first computer over the external communication link computer specific information corresponding to predefined characteristics of the second computer;

10 computer readable program code means for storing the computer specific information corresponding to the predefined characteristics of the second computer to provide server header information;

wherein said computer readable program code means for converting comprises computer readable program code means for removing the computer specific information from the web server originated communication; and

15            wherein said computer readable program code means for  
rebuilding comprises computer readable program code means  
for combining the server header information with the  
client/server specific data stream to create an HTTP data  
stream.

106. A computer program product according to Claim 104  
further comprising:

5            computer readable program code means for establishing a  
persistent connection between the first computer and the  
second computer over the external communication link;

            computer readable program code means for maintaining  
the persistent connection until all web server originated  
communications are completed; and

10            wherein said computer readable program code means for  
capturing comprises computer readable program code means for  
intercepting a plurality of web server originated  
communications and multiplexing them onto the external  
communication link while the persistent connection is  
maintained.

107. A computer program product according to Claim 106  
wherein said computer readable program code means for  
rebuilding comprises computer readable program code means  
for demultiplexing the client/server specific data stream to  
5            create a plurality of HTTP data streams, and wherein said  
computer readable program code means for furnishing  
comprises computer readable program code means for providing  
said plurality of HTTP data streams to the web server.

108. A computer program product for increasing the  
performance of a web browser application resident on a first  
computer and communicating using the Hyper-Text Transfer  
Protocol (HTTP) with a web server application resident on a  
5            second computer remote from said first computer where at  
least one segment of the communication between the web



browser application in the first computer and the web server application in the second computer occurs over an external communication link, comprising:

10 A computer-readable storage medium having computer-readable program code means embodied in said medium, said computer-readable program code means comprising:

computer readable program code means for intercepting the HTTP data stream corresponding to a communication  
15 originated by the web browser prior to transmission of the HTTP data stream on the external communication link;

computer readable program code means for converting the HTTP data stream originated by the web browser from the HTTP protocol to a client/server specific communication protocol;  
20 and

computer readable program code means for transmitting the converted web browser originated communication to the web server over the external communication link as a client/server specific data stream.

109. A computer program product according to Claim 108 wherein said computer readable program code means for transmitting comprises computer readable program code means for transmitting the converted web browser originated  
5 communication over a wireless communication link.

110. A computer program product according to claim 108 further comprising:

computer readable program code means for storing an HTTP data stream to be received by the web browser in  
5 response to a web browser originated communication with the web server in a cache resident in the first computer to create a client cache entry corresponding to the web browser originated communication with the web server;

computer readable program code means for interrogating  
10 the web browser originated communication to determine if a

client cache entry exists corresponding to the web browser originated communication; and

15 computer readable program code means for supplying the client cache entry to the web browser as an HTTP data stream in response to the web browser originated communication.

111. A computer program product according to Claim 108 further comprising:

5 computer readable program code means for storing user defined information associated with a specific web browser information request so as to provide the user defined information resident in the first computer;

10 computer readable program code means for interrogating the web browser originated communication to determine if the information requested by the web browser corresponds to a request for which user defined information is stored; and

15 computer readable program code means for providing the user defined stored information to the web browser as an HTTP data stream in response to the web browser originated communication if said computer readable program code means for interrogating determines that the web browser communication corresponds to a request for which user defined information is stored.

112. A computer program product according to Claim 111 further comprising computer readable program code means for receiving the user defined information from the first computer across the communication link.

5 113. A computer program product according to Claim 110 wherein said computer readable program code means for storing further comprises computer readable program code means for storing the time of creation of a client cache entry to create a client cache entry time record;

wherein said computer readable program code means for interrogating further comprises computer readable program

code means for evaluating the client cache entry time record to determine if the client cache entry corresponding to the web browser originated communication was created within a predetermined client coherency time interval prior to the web browser requesting the information; and

wherein said computer readable program code means for supplying comprises computer readable program code means for providing the client cache entry to the web browser as an HTTP data stream in response to the web browser originated communication if computer readable program code means for determining determines that a client cache entry was created within a predetermined client coherency time interval prior to the web browser requesting the information.

114. A computer program product according to Claim 113 further comprising computer readable program code means for maintaining client cache entries across multiple instances of the web browser.

115. A computer program product according to Claim 108 wherein said web browser originated communication corresponds to a Common Gateway Interface (CGI) request, said computer program product further comprising:

computer readable program code means for interrogating the intercepted CGI request to determine if a client base cache entry corresponding to the intercepted CGI request exists to provide a client CGI base form;

computer readable program code means for storing a HTTP data stream to be provided to the web browser in response to a browser originated communication which is a request to a CGI in a cache resident in the first computer to create a client base cache entry;

computer readable program code means for reconstructing the HTTP data stream corresponding to a communication from the web server from the client/server specific data stream received over the external communication link by combining

the client CGI base form with CGI difference data received  
over the external communication link to create an HTTP data  
stream; and

computer readable program code means for providing the  
communication originated by the web server to the web  
browser as an HTTP data stream.

116. A computer program product according to Claim 115  
further comprising:

wherein said means for reconstructing comprises  
computer readable program code means for reconstructing the  
HTTP data stream corresponding to a communication from the  
web server from the client/server specific data stream  
received over the external communication link by combining  
the a server CGI base form received over the external  
communication link with the CGI difference data received  
over the external communication link to create an HTTP data  
stream corresponding to the intercepted CGI response; and

computer readable program code means for updating the  
client CGI base form corresponding to the CGI request by  
storing the received server CGI base form as the client base  
cache entry corresponding to the CGI request.

117. A computer program product according to Claim 108  
further comprising:

computer program code means for establishing a  
persistent connection between the first computer and the  
second computer over the external communication link;

computer readable program code means for maintaining  
the persistent connection until all web browser originated  
communications are completed; and

wherein said computer readable program code means for  
intercepting comprises computer readable program code means  
for intercepting a plurality of web browser originated  
communications and multiplexing them onto the external

communication link while the persistent connection is maintained.

118. A computer program product according to Claim 117 further comprising:

5 computer readable program code means for receiving a client/server specific data stream over the external communication link from the second computer;

computer readable program code means for demultiplexing the client/server specific data stream to create a plurality of HTTP data streams to provide the plurality of HTTP data streams to the web browser; and

10 computer readable program code means for providing the plurality of HTTP data streams to the web browser.

119. A computer program product according to Claim 108 further comprising:

5 computer readable program code means for providing to the second computer over the external communication link computer specific information corresponding to predefined characteristics of the first computer; and

10 wherein said computer readable program code means for transforming comprises computer readable program code means for removing the computer specific information from the web browser originated communication.

120. A computer program product for increasing the performance of a web browser application resident on a first computer and communicating using the Hyper-Text Transfer Protocol (HTTP) with a web server application resident on a  
5 second computer remote from said first computer where at least one segment of the communication between the web browser application in the first computer and the web server application in the second computer occurs over an external communication link comprising:

10 A computer-readable storage medium having computer-readable program code means embodied in said medium, said computer-readable program code means comprising:

15 computer readable program code means for intercepting the HTTP data stream corresponding to a communication originated by the web server prior to transmission of the HTTP data stream on the external communication link;

20 computer readable program code means for converting the HTTP data stream originated by the web server from the HTTP protocol to a client/server specific communication protocol; and

computer readable program code means for transmitting the converted web server originated communication to the web browser over the external communication link as a client/server specific data stream.

121. A computer program product according to Claim 120 further comprising:

5 computer readable program code means for storing the HTTP data stream received from the web server in response to a browser originated communication in a cache resident in the second computer to create a server request cache entry;

10 computer readable program code means for interrogating the web browser originated communication to determine if a server request cache entry corresponding to the browser originated communication has been previously stored in the cache; and

15 wherein said computer readable program code means for converting comprises computer readable program code means for transforming the server request cache entry associated with the browser originated communication to a client/server specific communication protocol; and

wherein said computer readable program code means for transmitting comprises computer readable program code means for sending the transformed server request cache entry to

20 the first computer over the external communication link as a  
client/server specific data stream.

122. A computer program product according to Claim 121  
further comprising:

5 computer readable program code means for determining if  
a server request cache entry corresponding to the web  
browser originated communication was created within a  
predetermined client coherency time interval prior to the  
second computer receiving the web browser originated  
communication; and

10 wherein said computer readable program code means for  
transforming comprises computer readable program code means  
for transforming the server request cache entry which  
corresponds to the web browser originated communication to a  
client/server specific communication protocol if said  
computer readable program code means for determining  
15 determines that the server request cache entry was created  
with the predetermined client coherency time interval.

123. A computer program product according to Claim 121  
further comprising:

5 computer readable program code means for determining if  
a client cache entry exists corresponding to the web browser  
originated communication which is identical to a server  
cache entry corresponding to the web browser originated  
communication; and

10 wherein said computer readable program code means for  
transforming comprises computer readable program code means  
for calculating the interval of time between when the second  
computer received the web browser originated communication  
and when the server request cache entry corresponding to the  
web browser originated communication was created to provide  
entry age data; and

15 wherein said computer readable program code means for  
transmitting comprises computer readable program code means

for transmitting as the client/server specific protocol a coherent entry response to the first computer which includes the entry age data for the server cache entry corresponding to the web browser originated communication over the external communication link.

124. A computer program product according to Claim 120 wherein said web browser originated communication corresponds to a Common Gateway Interface (CGI) request, said computer program product further comprising:

computer readable program code means for interrogating the intercepted CGI request to determine if a server base cache entry corresponding to the intercepted CGI request exists to provide a server CGI base form;

computer readable program code means for storing a HTTP data stream received from the web server in response to a browser originated communication which is a request to a Common Gateway Interface (CGI) in a cache resident in the second computer to create a server base cache entry;

wherein said computer readable program code means for intercepting comprises intercepting the HTTP data stream corresponding to the communication originated by the web server in response to the intercepted CGI request prior to transmission of the HTTP data stream on the external communication link;

computer readable program code means for comparing the intercepted CGI response to the server CGI base form to provide CGI difference data corresponding to the difference between the intercepted CGI response and the server CGI base form; and

wherein said computer readable program code means for transmitting comprises computer readable program code means for sending the CGI difference data to the web browser over the external communication link as a client/server specific data stream.



125. A computer program product according to Claim 124 further comprising:

5 computer readable program code means for determining if the server CGI base form is identical to a client CGI base form corresponding to the CGI request; and

10 wherein said computer readable program code means for transmitting comprises computer readable program code means for transmitting the server CGI base form and transmitting the CGI difference data to the web browser over the external communication link as a client/server specific data stream.

126. A computer program product according to Claim 124 further comprising:

5 computer readable program code means for determining if the difference between the server CGI base form and the CGI response is greater than a predefined difference threshold;

10 computer readable program code means for updating the server CGI base form corresponding to the CGI request by storing the CGI response received from the web server as the server base cache entry corresponding to the CGI request if said computer readable program code means for determining determines that the difference between the server CGI base form and the CGI response is greater than a predefined difference threshold; and

15 wherein said computer readable program code means for comparing and said computer readable program code means for sending utilizes the updated server CGI base form.

127. A computer program product according to Claim 120 further comprising:

5 computer readable program code means for establishing a persistent connection between the first computer and the second computer over the external communication link;

computer readable program code means for maintaining the persistent connection until all web browser originated communications are completed; and

10 wherein said computer readable program code means for  
intercepting comprises computer readable program code means  
for intercepting a plurality of web server originated  
communications and multiplexing them onto the external  
communication link while the persistent connection is  
maintained.

5 128. A computer program product for increasing the  
performance of a client/server system having a client  
application resident on a first computer and communicating  
with a server application resident on a second computer  
remote from said first computer wherein said client  
application and said server application utilize a  
client/server independent communication protocol for  
communication between the client and the server and where at  
least one segment of the communication between the client  
10 application in the first computer and the server application  
in the second computer occurs over an external communication  
link, the computer program product comprising:

15 A computer-readable storage medium having computer-  
readable program code means embodied in said medium, said  
computer-readable program code means comprising:

20 computer readable program code means for intercepting  
communications in the client/server independent  
communication protocol originated by the remote client prior  
to transmission of the communications on the external  
communication link;

computer readable program code means for converting the  
communication originated by the client to a second  
client/server specific communication protocol;

25 computer readable program code means for transmitting  
the converted communication over the external communication  
link;

computer readable program code means for receiving the  
communication transmitted over the external communication  
link;

30 computer readable program code means for converting the  
communication received over the external communication link  
from the client server specific communication protocol to  
the client/server independent communication protocol; and  
35 computer readable program code means for providing the  
communication originated by the remote client to the server  
in the client/server independent communication protocol.

129. A computer program product according to claim 128  
further comprising:

5 computer readable program code means for intercepting  
communications in the client/server independent  
communication protocol originated by the server prior to  
transmission of the communications on the external  
communication link;

10 computer readable program code means for converting the  
communication originated by the server to a second  
client/server specific communication protocol;

computer readable program code means for transmitting  
the converted communication over the external communication  
link;

15 computer readable program code means for receiving the  
communication transmitted over the external communication  
link;

20 computer readable program code means for converting the  
communication received over the external communication link  
from the client/server specific communication protocol to  
the client/server independent communication protocol; and

computer readable program code means for providing the  
communication originated by the server to the remote client  
in the client/server independent communication protocol.

FIG. 1  
PRIOR ART

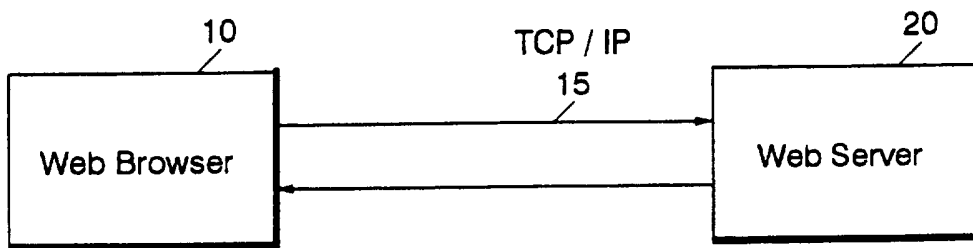
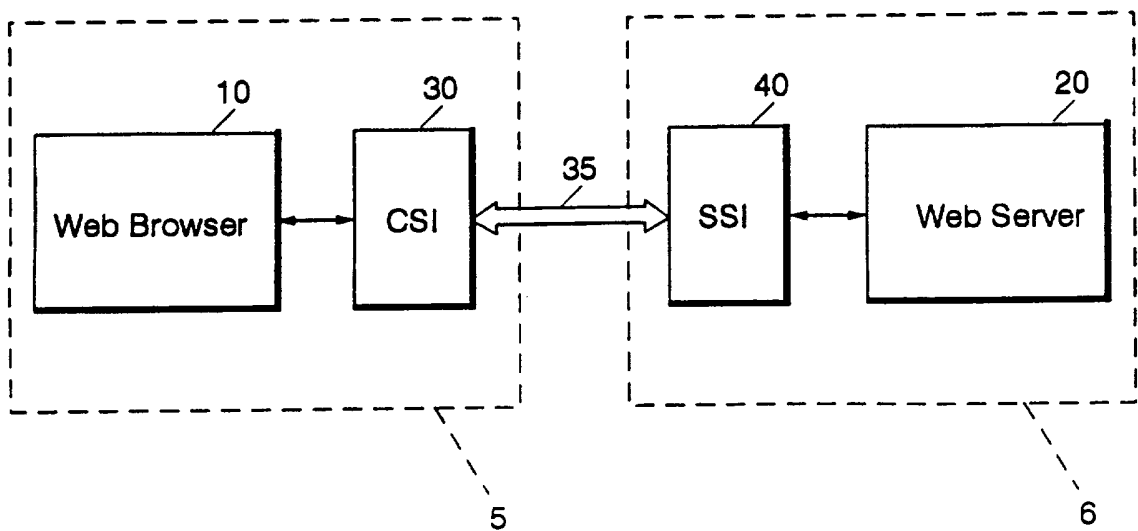


FIG. 2



2/19

FIG. 3

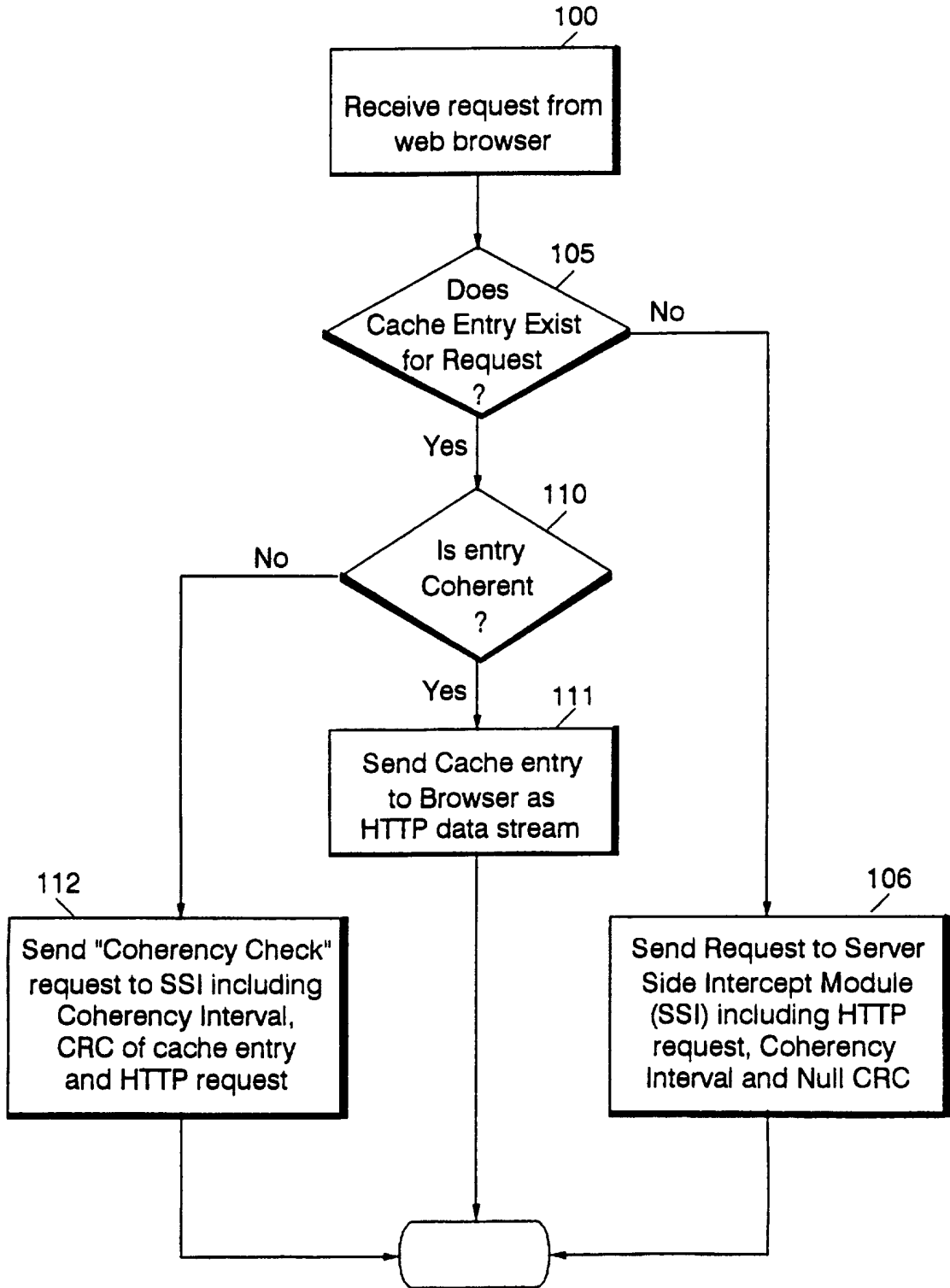


FIG. 4

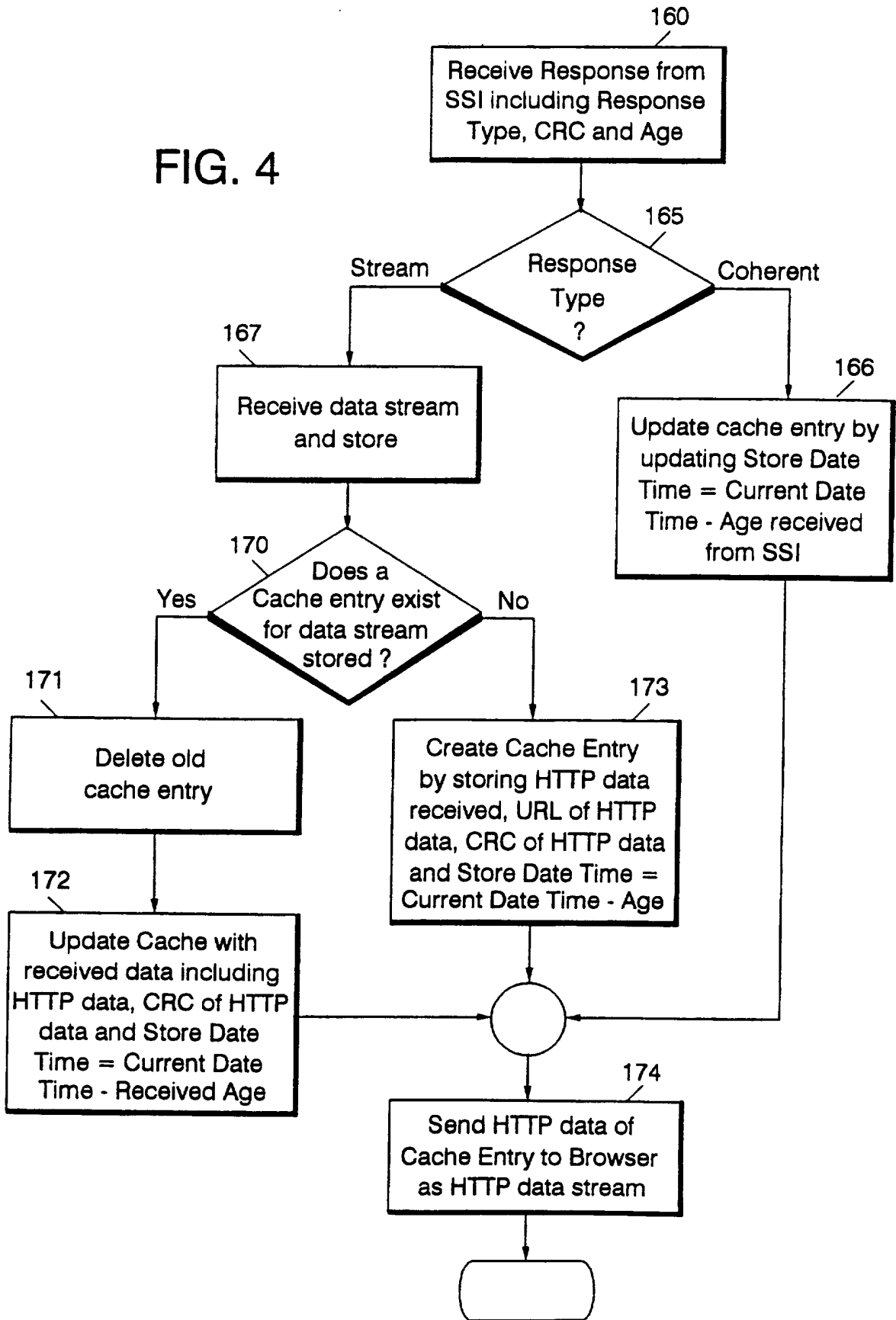
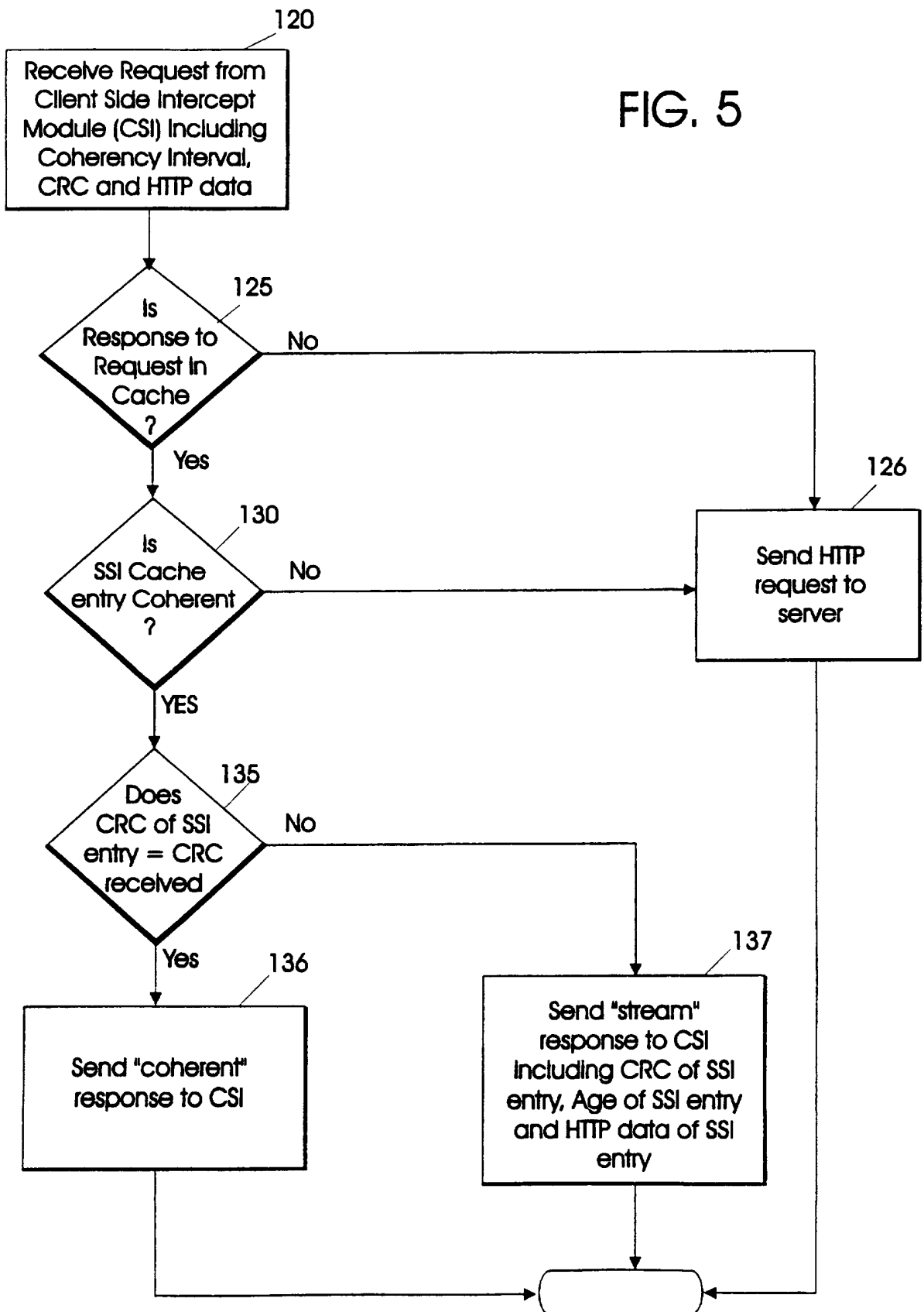


FIG. 5



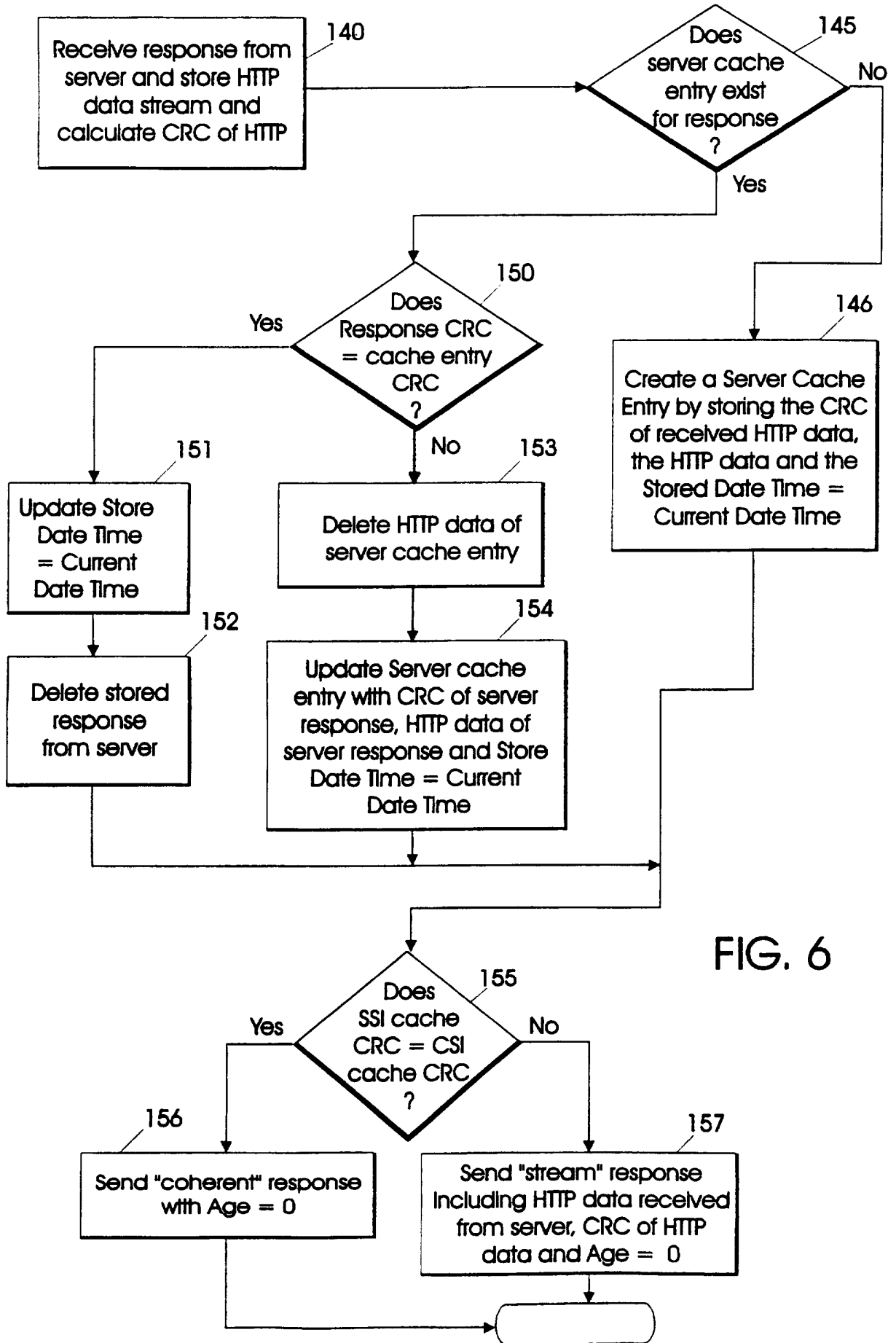


FIG. 6



FIG. 7

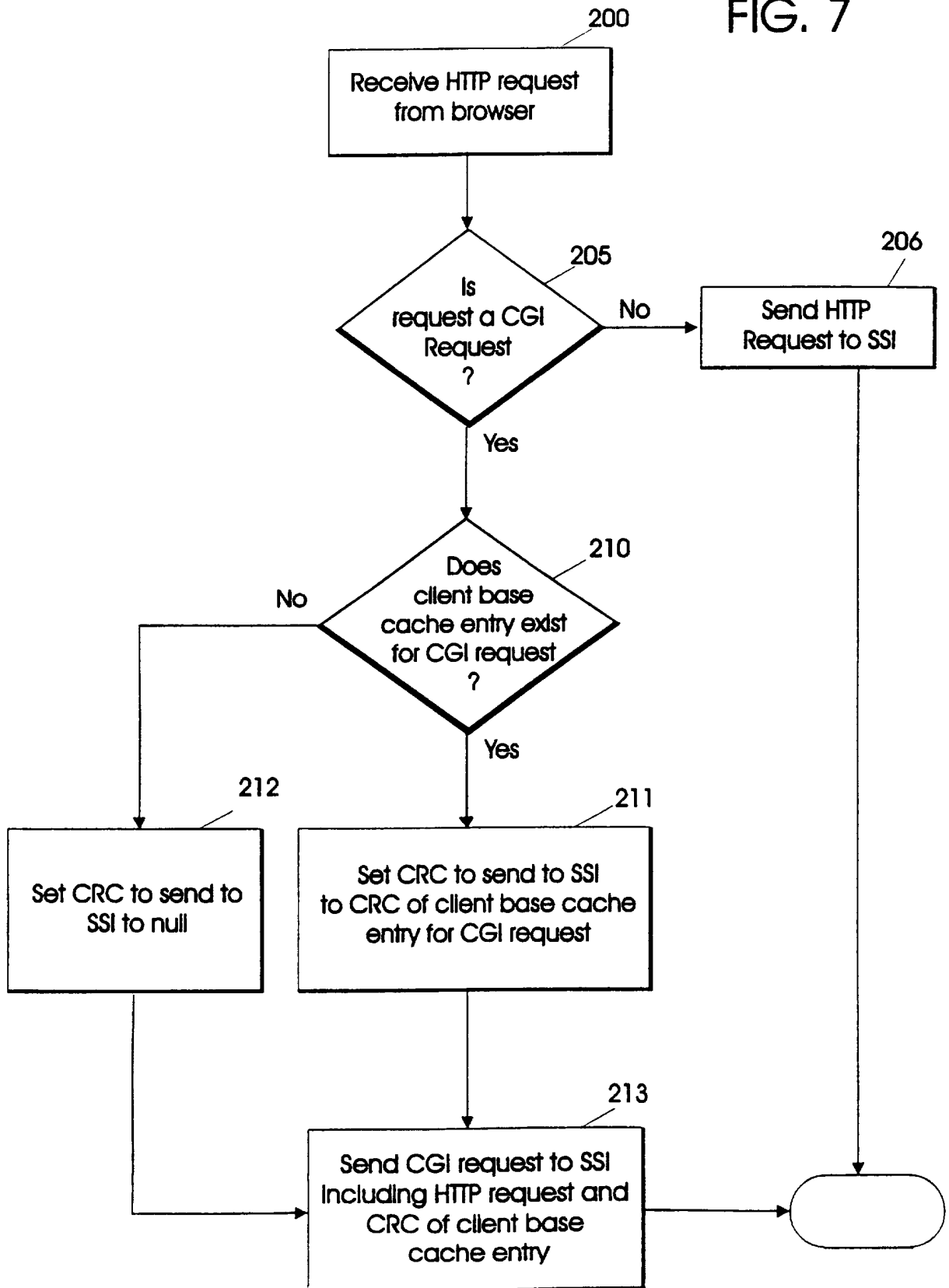


FIG. 8

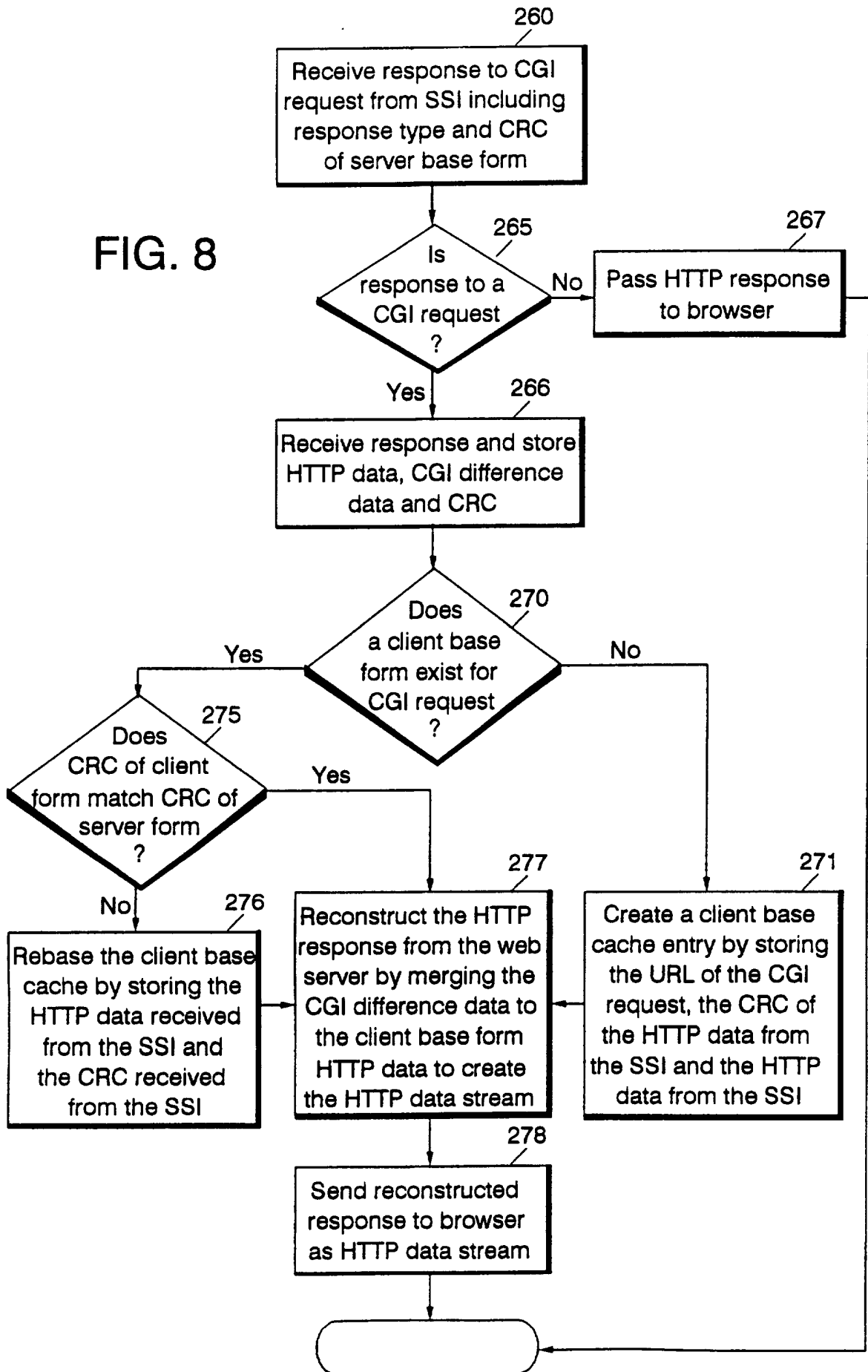


FIG. 9

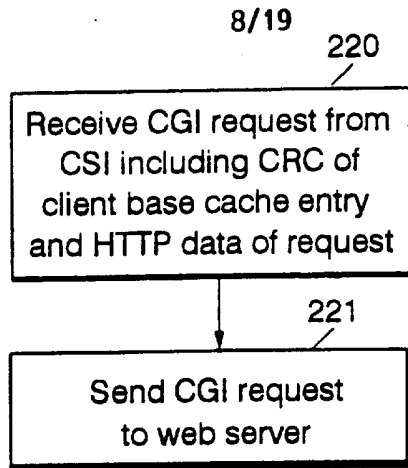
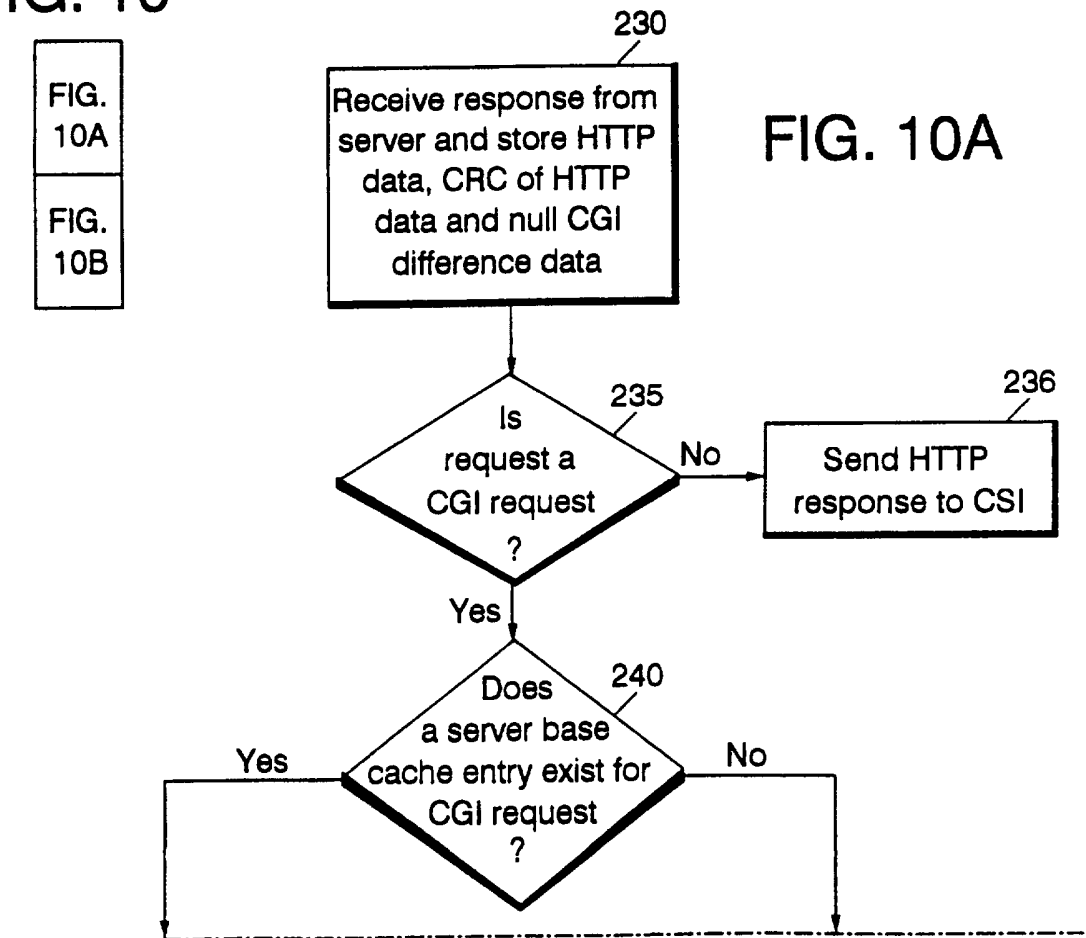


FIG. 10

FIG. 10A

FIG. 10B

FIG. 10A



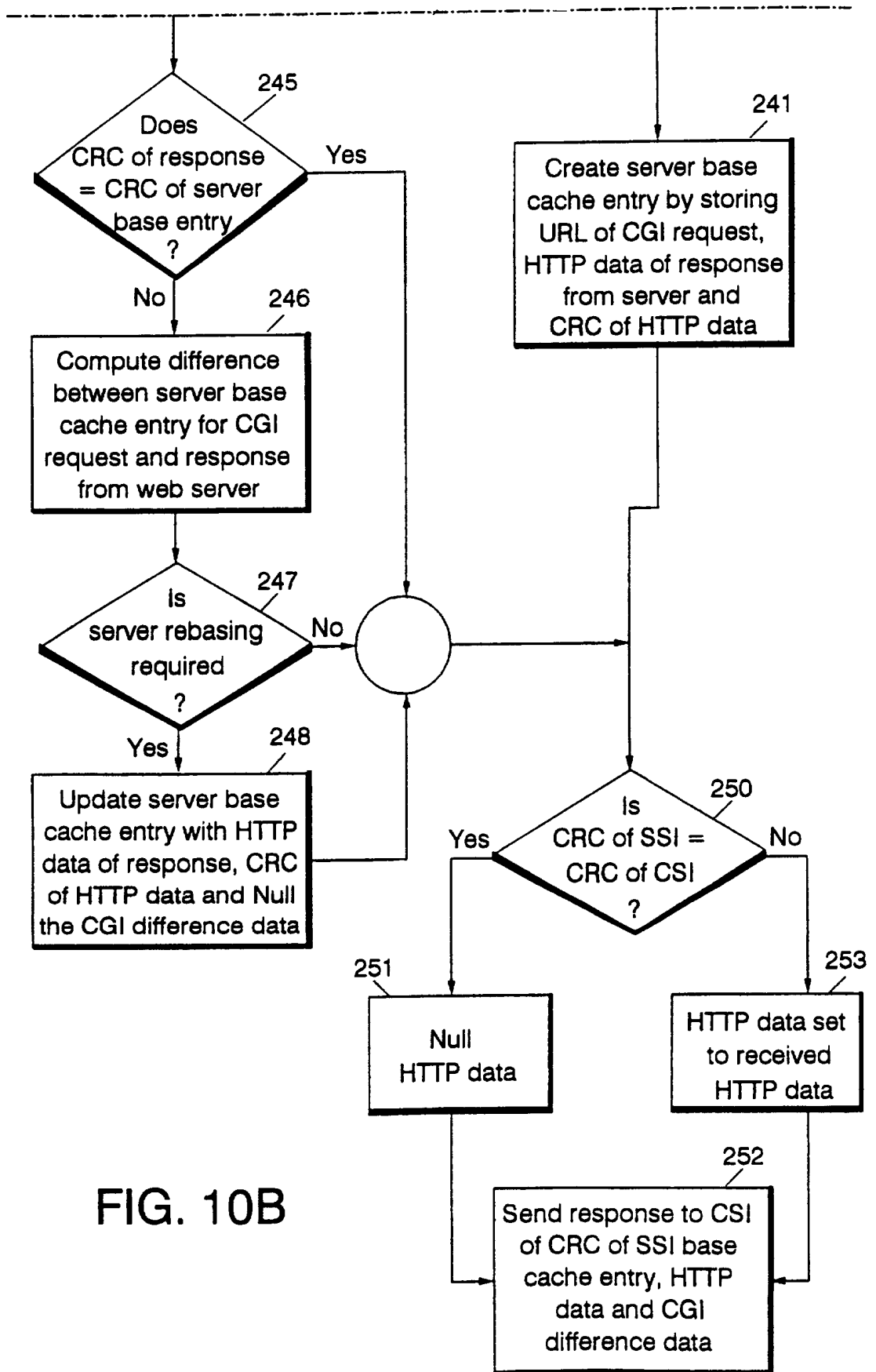


FIG. 10B

FIG. 11

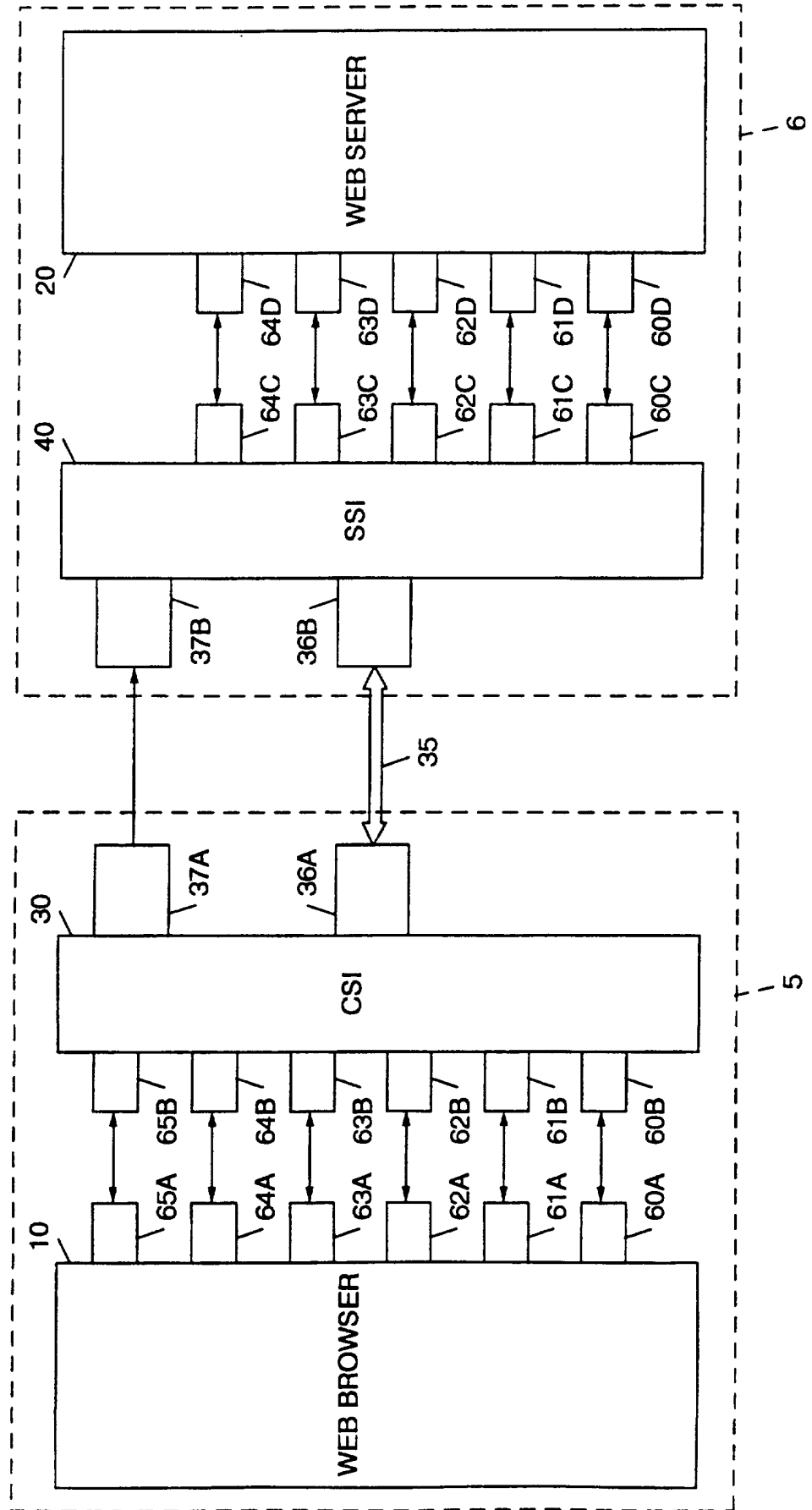


FIG. 12

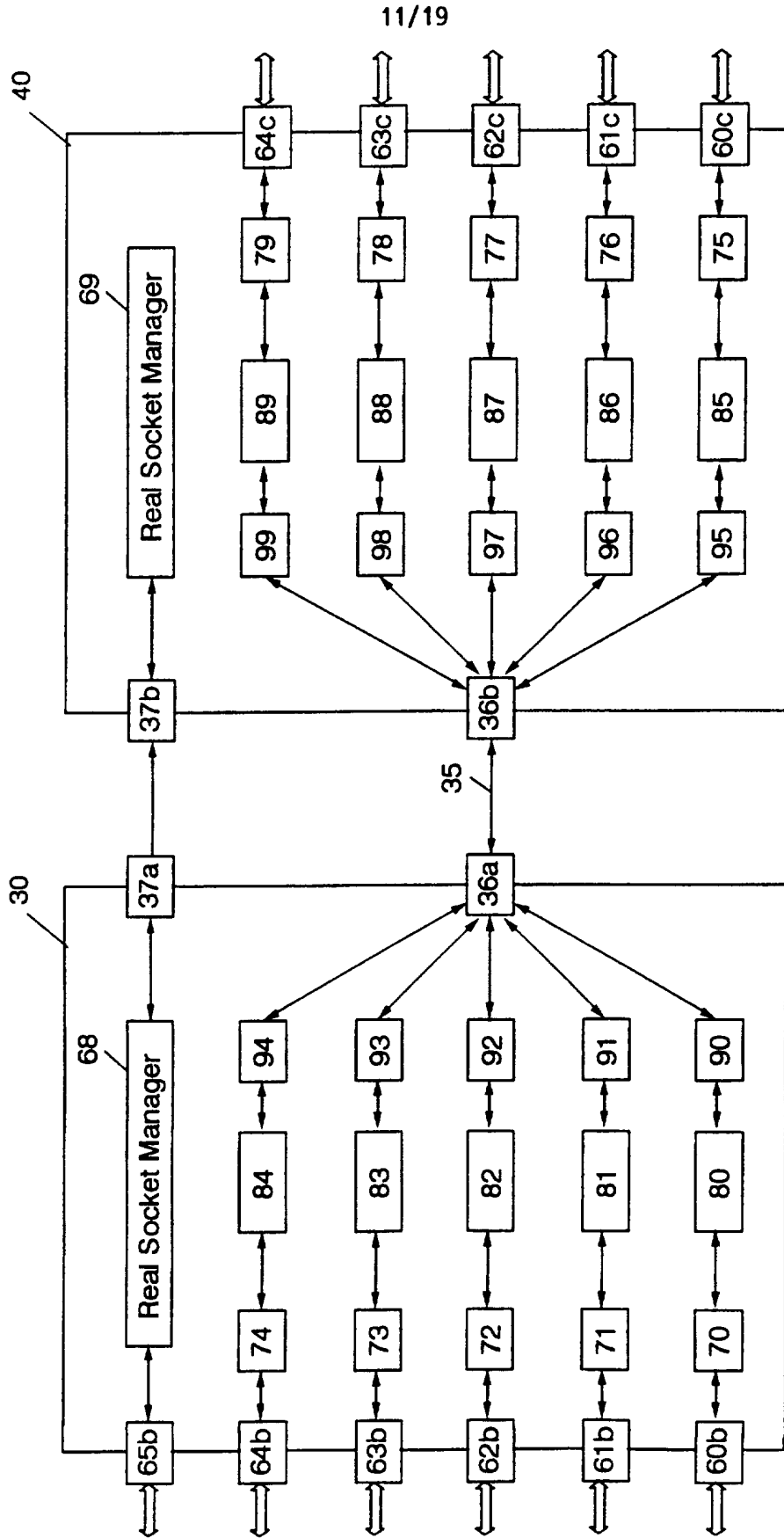


FIG. 13

FIG. 13A
FIG. 13B

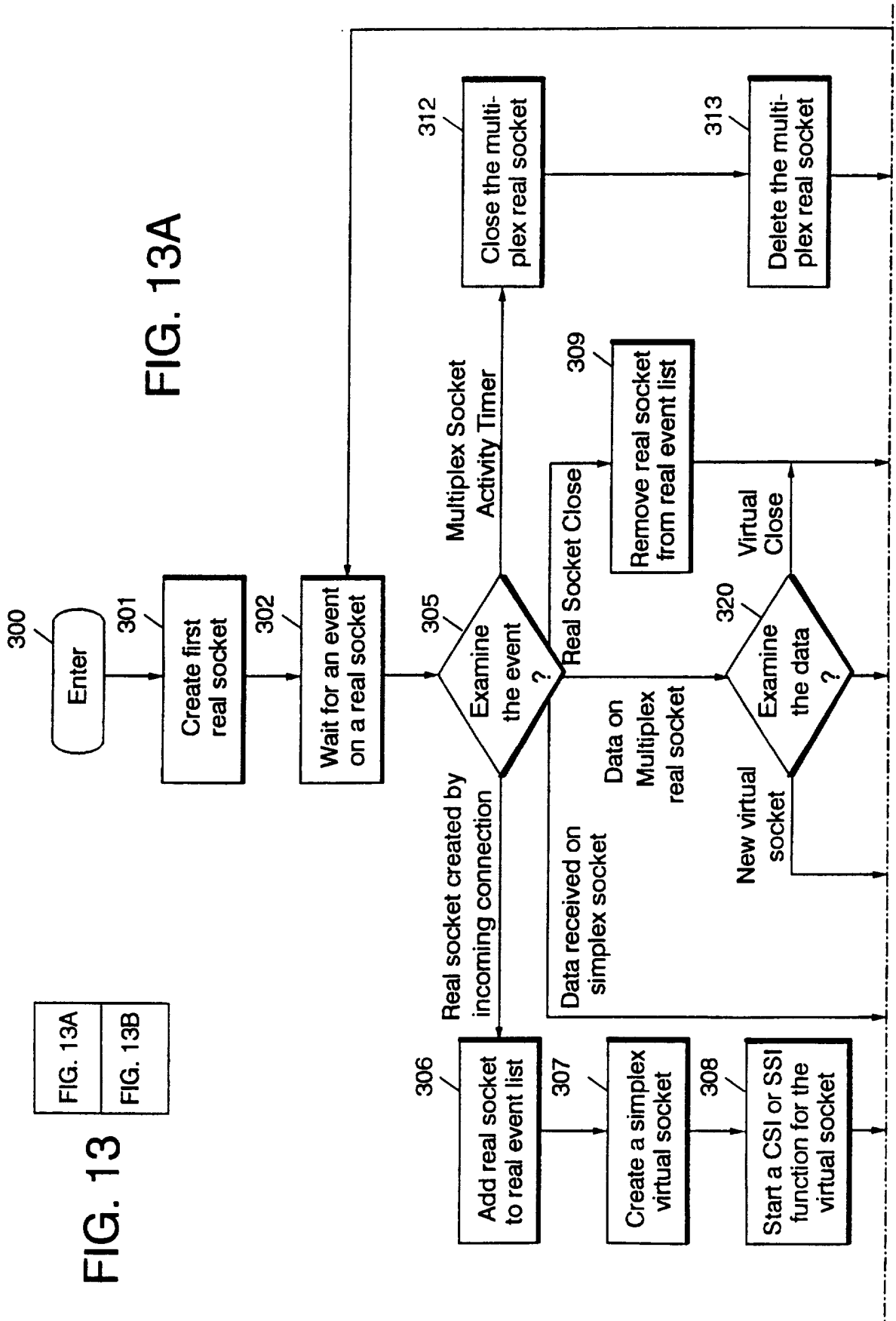


FIG. 13A

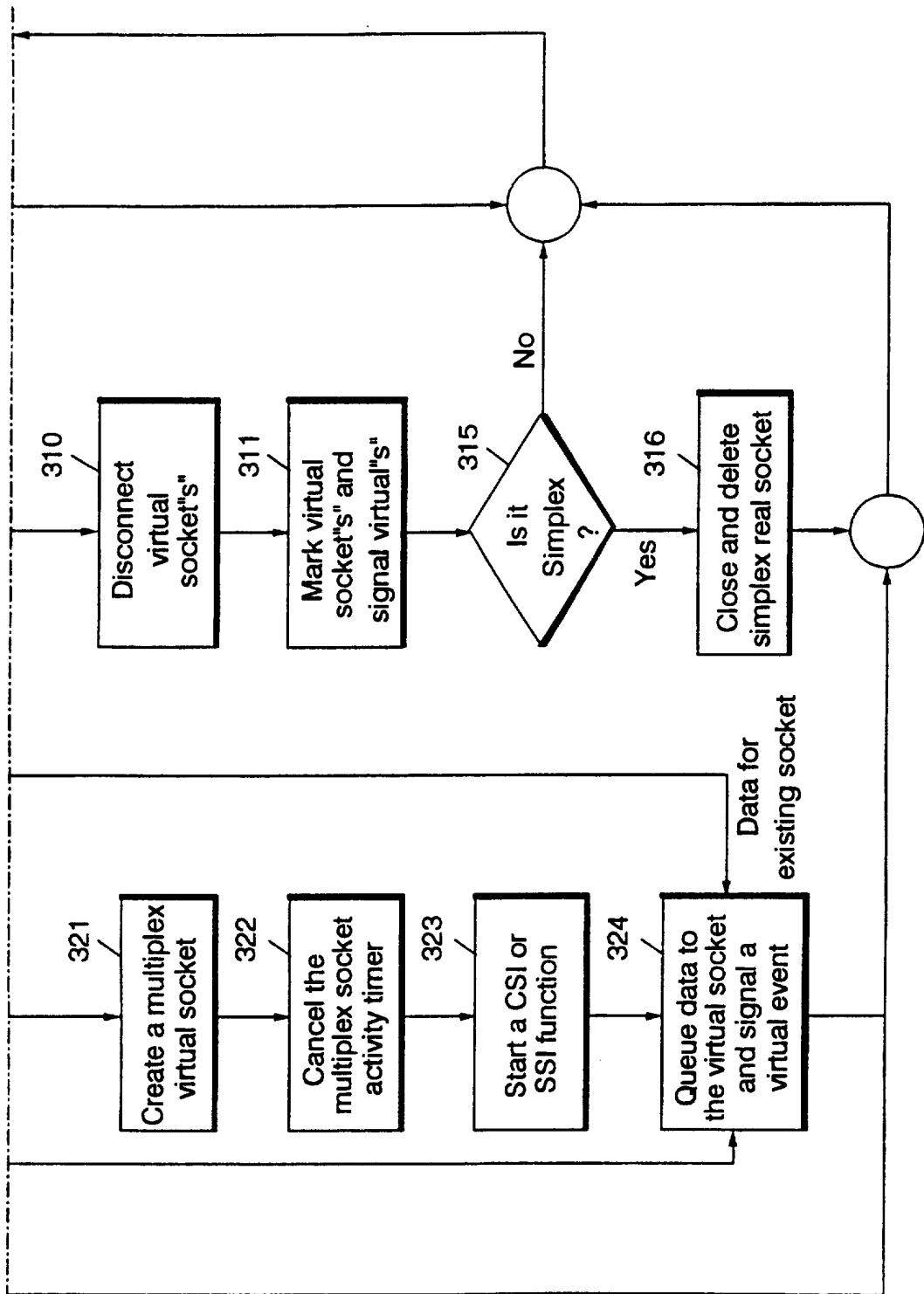


FIG. 13B



FIG. 14

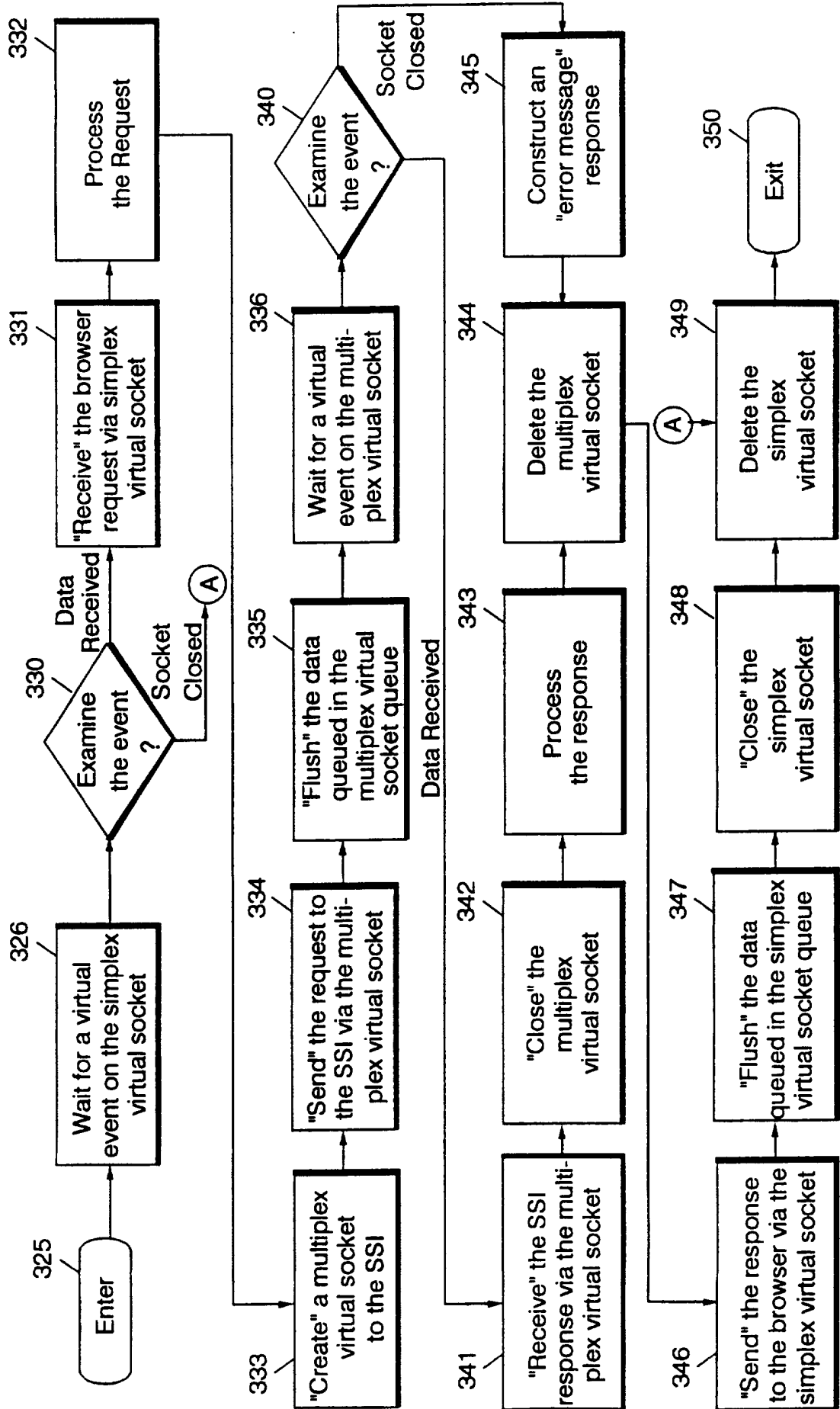


FIG. 15

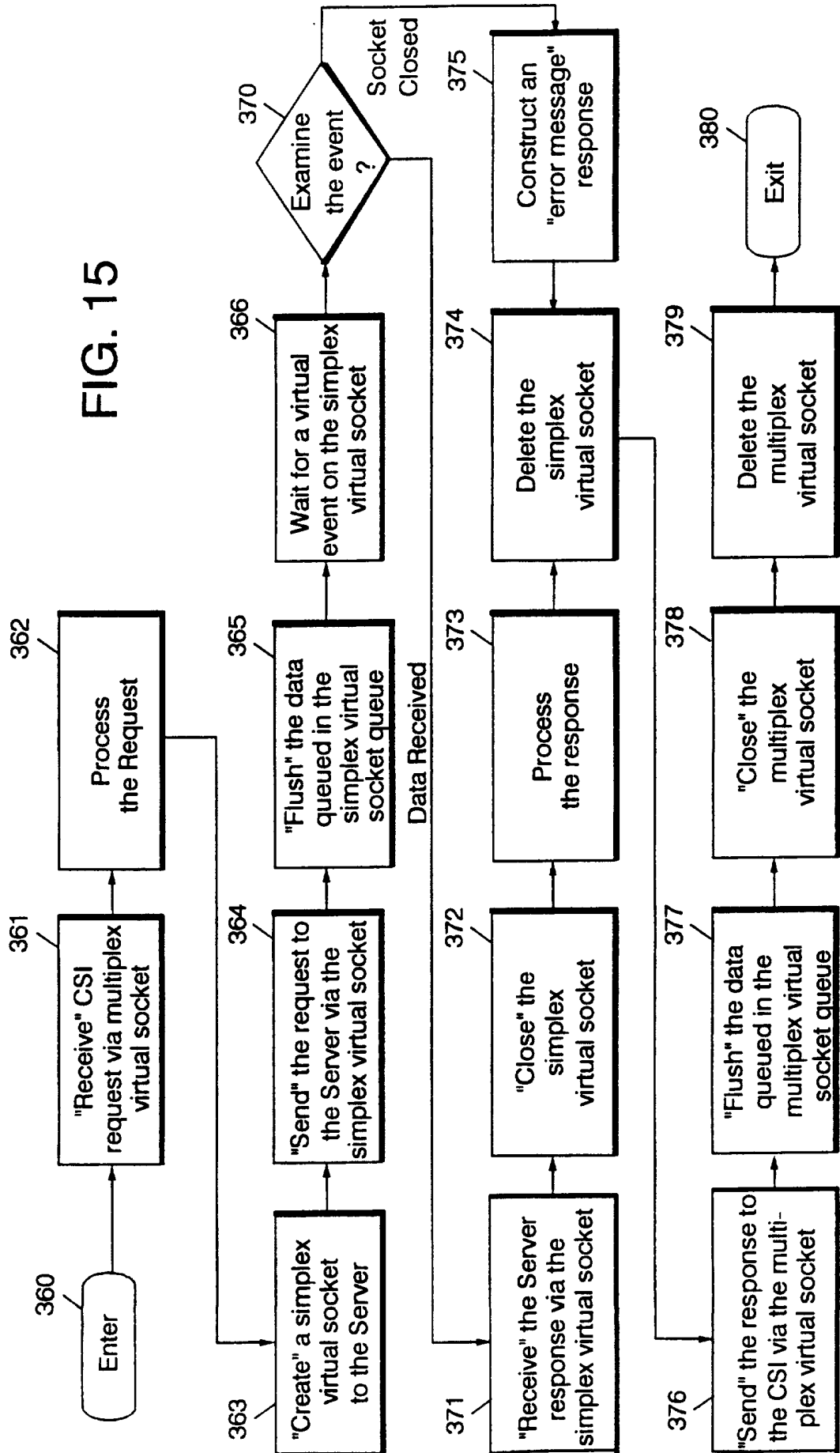


FIG. 16-1

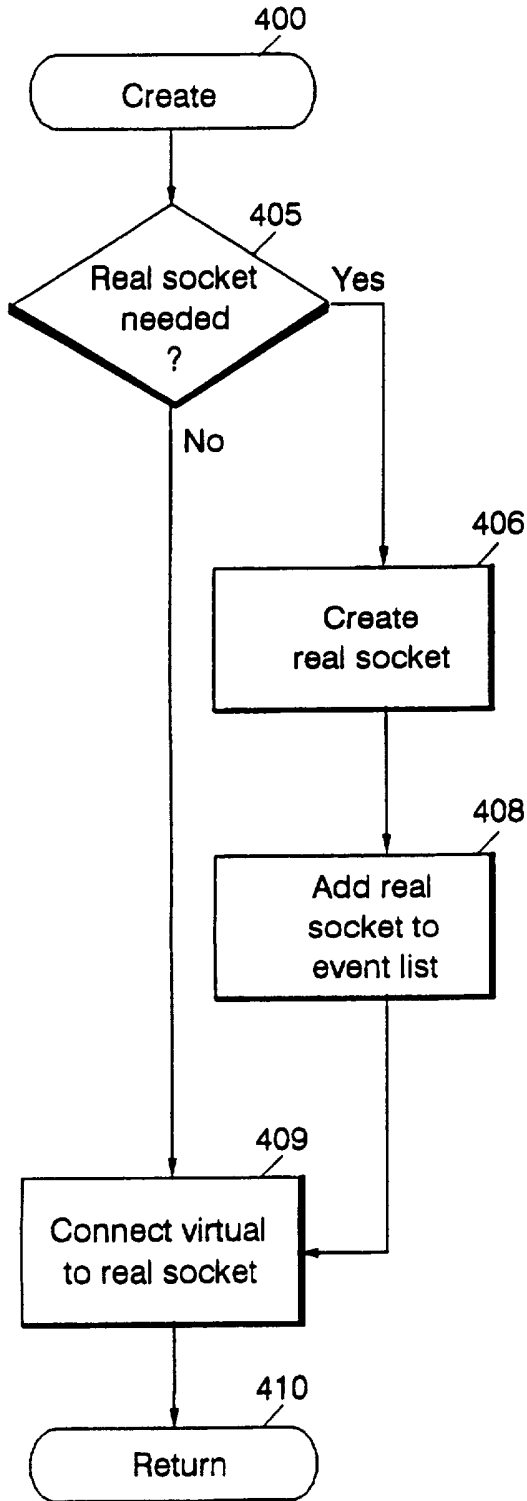


FIG. 16-2

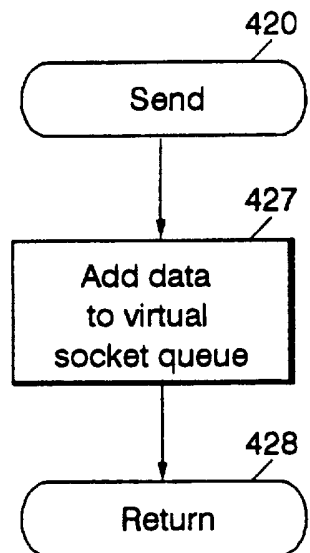


FIG. 16-3

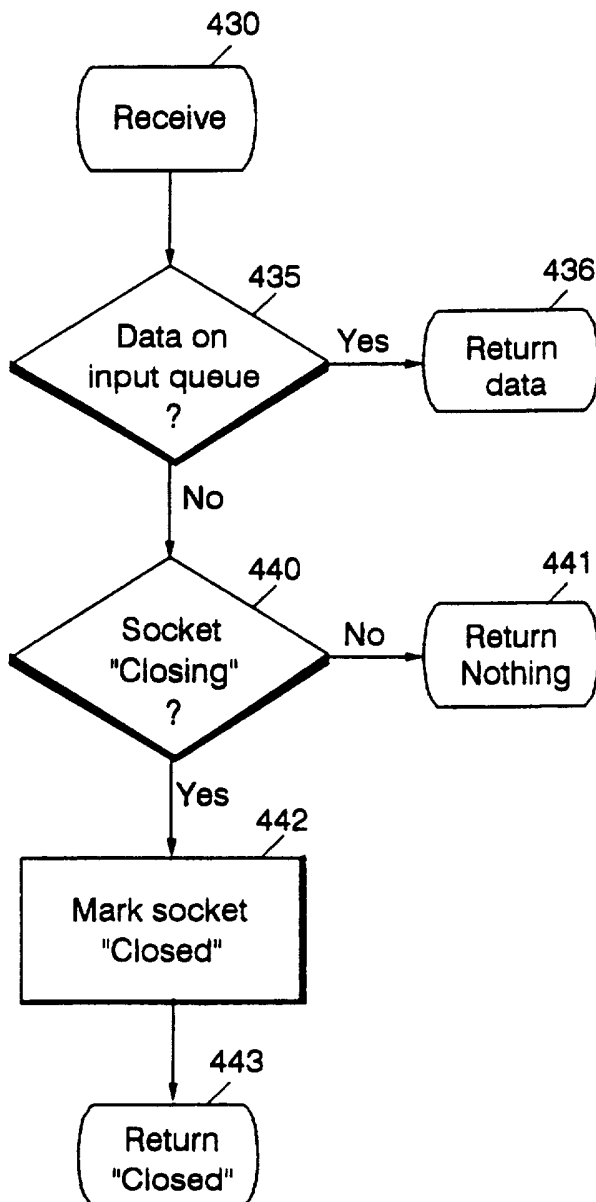


FIG. 16-4

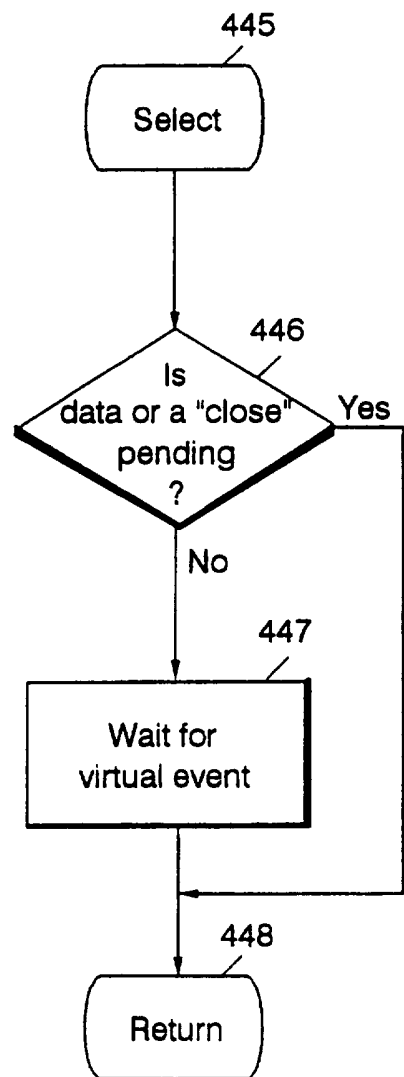


FIG. 17-1

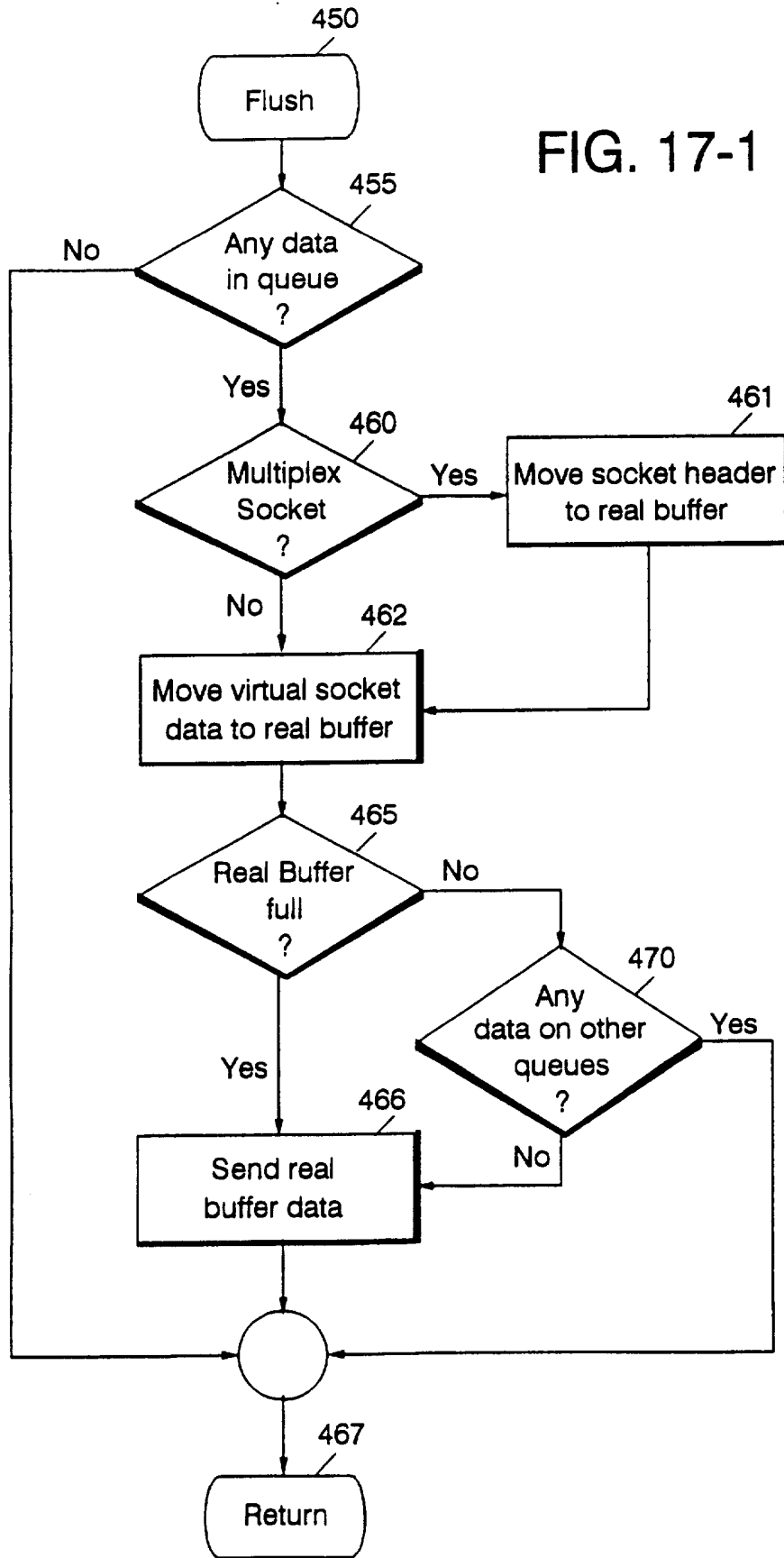


FIG. 17-2

