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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 March 2003 (06.03.2003)

PCT

(10) International Publication Number
WO 03/019901 A1

(51) International Patent Classification[7]: H04L 29/06

(21) International Application Number: PCT/IL01/00793

(22) International Filing Date: 23 August 2001 (23.08.2001)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **PRIVIA INC.** [US/US]; Privia Inc. c/o Guy Blachman, 200 Riverside Blvd., New York, NY 10023 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **ISRAEL, Igal** [IL/IL]; 12/54 Nisenboim Street, 32249 Haifa (IL). **KABISCHER, Boris** [IL/IL]; 17/22 Zahal Street, 36627 Nesher (IL).

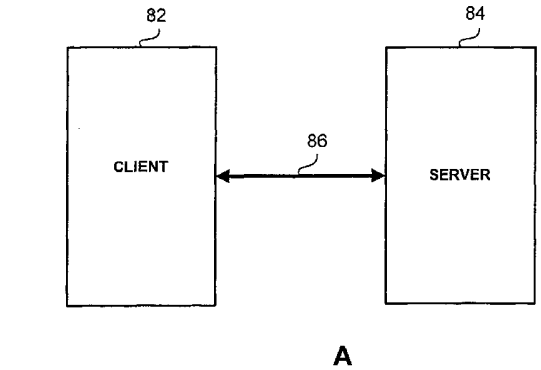
(74) Agents: **AGMON, Jonathan** et al.; Soroker - Agmon, Avocates and Patent Attorneys, 12th Floor, Levinsttein Tower, 23 Petach Tikva Road, 66184 Tel Aviv (IL).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

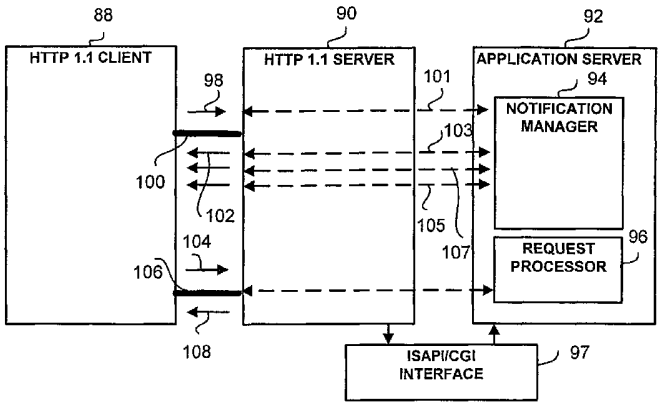
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR ENABLING THE SENDING OF NOTIFICATIONS FROM A SERVER TO A CLIENT WITHOUT POLLING IN A DATA COMMUNICATION NETWORK



A



B

(57) Abstract: In a computing and data communication network a method and system for enabling the sending of notification from a server system to a client system in the framework of a continuously open network connection. Requests for continuous connection are introduced by client systems to server systems. The servers handle the request by updating a continuous connection list, and notify the requesting client and other significant network particles in regard to the opening of the continuous connection. Data, events, and other content received to the server by third-tier database servers are sent to the client in accordance with the suitable control information stored within the backward connection list.

WO 03/019901 A1

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SYSTEM AND METHOD FOR ENABLING THE SENDING OF NOTIFICATIONS FROM A SERVER TO A CLIENT WITHOUT POLLING IN A DATA COMMUNICATION NETWORK

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

The present invention relates in general to the field of client-server computer systems, and more particularly to a system and a non-polling method of operation for enabling the sending of notifications from a server to a client and enabling the client to get data from the server in a data communication network utilizing a stateless session-oriented protocol.

DISCUSSION OF THE RELATED ART

An important field of computing concerns the transfer of information between computing units over data communication networks. These networks are utilized as information retrieval systems having specific architectures and employing specific sets of known protocols. The most common network architecture follows the conventional client-server model. The pair of terms: "client" and "server" refer to the general functionality of a computing unit either as a requester of data i.e., the client, or as a provider of data i.e., the server. Client-server architectures are supported by several client-server application protocols. A client-server application protocol is a set of rules a subset of which regards the manner of the interaction between clients and servers functioning across the data communication network. In a client-server application protocol's request-response process, referred to as a "session", a client introduces a request to a server. The request indicates a specific demand, wrapped within a well-defined formatted message object, for the transmission of specific information content from the server to the client. Subsequent to the request, the client receives a relevant response from server. The response is wrapped within a well-defined formatted message object containing either the desired information content or predefined status information such as an error condition concerning the

inability of the network or the server therein to fulfill the request. In a typical client-server protocol, such as the HyperText Transfer Protocol (HTTP) a session incorporates and identifies a single request-response pair. The session is also associated with the setting up of a network connection between the requesting client and the responding server. A network connection is defined as a transport layer virtual circuit established between two program entities for the purpose of communication. When the request is satisfied through the reception of a paired response, the particular HTTP session associated with the request-response pair terminates. As the client is no longer expects information content from the server, the server can initiate the severance of the connection with the client. Subsequently, the server frees the resources allocated for the client, the session terminates, and the network connection between the client and the server is torn down. In order for the client to submit a new request and for the server to respond to the request a new connection has to be established. According to the interaction rules defined by the protocol, only the client has the capability of setting up new connections. It is important to note that in a session-oriented client-server interaction environment both the clients and the server do not retain information of a session after the session is closed. Therefore, a protocol used in such an environment is also referred to as a stateless protocol.

Fig. 1A illustrates the elements of a highly simplified communicating environment and the steps involved in setting up a connection associated with a session, according to the rules of the HTTP and other stateless protocols. Client 10 sends a request 14 to server 12. The request effectively opens a new network connection to the server 12. Server 12 allocates resources to handle the session, processes the request 14, and communicates an associated response 16 back to the client 10. Client 10 receives the response 16 and server 12 terminates the session by disconnecting from client 10. From this point in time until the setting up of a new session client 10 is disconnected and therefore not functionally receptive to messages sent by the server 12. Thus, if subsequently server 12 attempts to communicate newly generated data 18 to the client 10 the attempt can not be

performed. Only when client 10 initiates a new connection by the submission of a new request to server 12 can the new information 18 be sent.

Fig. 1B illustrates the traditional connectivity method in a simplified networking environment utilizing the HTTP 1.1 application protocol. HTTP 1.1 client 20 introduces requests 28, 34 to HTTP 1.1 server 22. Server 22 can be one of many available HTTP servers such as IIS, Apache, and the like. Server 22 is responsible for low-level transport managed by the HTTP 1.1 protocol, the SSL security protocol, and the like. For each separate requests 28, 34 separate connections 26, 32 are established. Server 22 receives requests 28, 34 separately and forwards the requests to request processor 38 of application server 24. Server 24 can be an ISAPI DLL application, or an Active Server Page (ASP), or a Common Gateway Interface (CGI), or an Enterprise JavaBeans (EJB) application used as an application server "on top" of the HTTP server 22. Server 24 is utilized as a back end providing business rules, logic rules, database access, and the like. Interface 40 provides the ISAPI/CGI interface between server 22 and server 24. Request processor 38 processes requests 28, 34 separately via the appropriate application services and communicates the resulting data or notifications to server 22. Server 22 utilizing the respective communication channels 26, 32 transmits the respective responses 30, 32 to client 20. When client 20 receives response 26 the respective session terminates and connection 26 closes. After client 20 receives response 36 the respective session terminates and channel 32 is closes. Only client 20 is capable of initiating new connections and thereby opening new communication channels to server 22. Servers 22, 24 are unable to communicate with client 20 as long as the connection is closed.

Computing application systems have specific objectives to accomplish. In order to accomplish these objectives, it is preferable for some systems to have a permanently open communication channel between the client and the server. For example, applications involving the sending of real-time updates to user screens on a specific client system, or applications involving the real-time synchronization of a server-side database with local client-side data tables, or application sending

event notifications to client, may substantially benefit from continuous connectivity. "Keeping alive" or sustaining the connection enables the communication of request-independent messages from the server to the client. In order to allow the server to transmit messages independently of a request to the client and to allow a client to be functionally receptive to any potential messages originated by the server independently of any specific request, the configuration of a single request-response pair session should be modified. The session's duration should be expanded to allow the inclusion of more than one server-specific response to at least one client-specific request. The expanded duration of a session can be attained by the provision of an option in respect to the setting up of a continuous connection between the client and the server.

A number of program products such as Instant Messaging systems operating in a client-server environment, utilize non-HTTP communication protocols in order to make available the option of continuous connections between a server and a client. These solutions utilize non-standard communication ports and therefore have problems when operating in a firewall environment. In an environment secured by a firewall, the firewall typically screens incoming messages according to the targeted communication ports. Typically messages sent to non-standard communication ports or by non-standard protocols are blocked. Thus, in such an environment, although the client may be functionally receptive for messages arriving through the right port and according to the well-known standard protocol, and the server may transmit messages to the right port according to the well-known protocol, the messages proper are routinely blocked by the firewall.

Currently the HTTP request-response mechanism has no built-in notification-related features, which are usable in a realistically configured network. Thus, when the server is required to send a notification concerning a specific event or any other type of data to a client then the server must wait for the setting up of a new connection by the specific client. Only after the opening of the connection by the client can the transmission of the pending notification or data

can be completed within the framework of a session as a message responding to a request.

In a session-oriented client-server system no inherent state information exists. Therefore, during periods of disconnection the client has no way of perceiving that the server has pending messages awaiting a connection. A new connection can be established at any point in time when the timing is decided upon only according to the client's local requirements. Thus, the establishment of a new connection does not depend on the presence or the want of pending notifications or data on the server's side. Subsequently, within a highly dynamic application, notifications or new data may arrive to the client too late to be useful.

One conventional method for getting timely notifications or new data from the server to the client is the polling technique. Polling is the periodical initiation of network connections by the submission of repeated requests. The requests are submitted by a client agent in order to open a session through which the reception of the pending notifications or new data from the server is enabled. A client agent is defined as a client program that initiates a request. A client agent can be a browser, an editor, a web-traversal routine, or any other end user tool. During the periods of disconnection the server collects messages or data intended for the client. Typically the client submits a request to the server every few seconds. Thereby, a connection is set up, and a standard request-response session opens. The server responds to the received request by the transmission all or some of the pending notifications or data to the requesting client. After the completion of the transmission the session terminates and the connection is torn down. Until the execution of the next polling action by the client the server again collects the notification or the data intended for transmission to the client.

Currently, polling is the only available HTPP notification method, which is practical in a realistically configured networking environment including firewalls, proxy servers, and gateways. The disadvantage of the method is its inherent inefficiency. Polling creates a plurality of request-response sessions, densely spread over a short period of time. Typically, is it extremely difficult to

synchronize the creation of the pending messages on the server side with the frequency of the polling session initiated by the client agent. Although most polling sessions do not transmit real data, the entire set of polling sessions is necessary in accomplishing timely transmission of up-to-date information.

It will be easily perceived by one with ordinary skill in the art that a need exists for a system and method that will provide continuous connectivity between a client system and a server system. Such a method should substantially eliminate or reduce disadvantages and problems associated with previously developed methods. More particularly such a system and method should allow for continuous bi-directional communication utilizing a session-oriented protocol within the constraints of a realistically configured communication system including proxy servers, gateways, firewalls, and other network particles operative in the active manipulation of the messages.

SUMMARY OF THE PRESENT INVENTION

One aspect of the present invention regards a computing environment accommodating at least one client system connectable to one or more server systems via at least one network control unit in a data communication network. The computing environment includes a non-polling method of transmitting notifications from at least one server system to the at least one client system while utilizing a stateless session-oriented application protocol. The method includes submitting at least one backward request operative in the opening of a continuous connection between a first computer system and a second computer system by the first computer system to be sent to the second computer system. Identifying the backward request submitted by the first computer system by the second computer system. Inserting a backward request entry into a list of the backward connections open between the first computer system and the second computer system by the second computer. Responding to the reception of the backward request by transmitting a response to the first computer system for confirming the opening of the continuous connection between the first computer system and the second computer system from the second computer system. Communicating a notification

message by the second computer system regarding the opening of the continuous connection between the first computer system and the second computer system to at least one network control unit deployed in the communication path between the first computer system and the second computer system. Reacting to at least one application-related event by transmitting at least one notification regarding the at least one event to the first computer system by the second computer system utilizing the communication path associated with the continuous connection.

The second aspect of the present invention regards a computing environment accommodating at least one client system connectable via at least one network control unit to one or more server systems in a data communication network. The environment includes a system for creating, sustaining, and maintaining at least one open continuous connection between the at least one client system and the server system. The system includes a backward connection list implemented on a second computer system to hold at least one record associated with at least one open continuous connection. It also includes an application server implemented on the second computer system, a request processor handler component associated with the application server implemented on the second computer system, and backward connection handler component associated with the application server implemented on the second computer.

The above aspects of the present invention provide for the creation, maintenance, and sustaining of open continuous connections between a first computer system and a second computer system across a data communication network. The above aspects of the present invention provide for the transmission of events, notifications, and data from the second computer to the first computer in the framework of the open continuous connection without the necessity for the first computer to perform polling. The above aspects of the present invention provide for keeping a network connection open and operative in a communication environment implementing a stateless session-oriented protocol.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be understood and appreciated more fully from the following detailed description taken in conjunction with the drawings in which:

Fig. 1A is a block diagram of a simplified communication environment, which is known in the art; and

Fig. 1B is a more detailed block diagram of a simplified communication environment, which is known in the art; and

Fig. 2 is a pictorial representation of the computing and communication environment suitable for the operation of the proposed method, in accordance with a preferred embodiment of the present invention; and

Fig. 3 is a schematic block diagram showing the constituent components of the client and server units, in accordance with a preferred embodiment of the present invention; and

Fig. 4A is a block diagram of a highly simplified communication environment, in accordance with a preferred embodiment of the present invention; and

Fig. 4B is a more detailed block diagram of a simplified communication environment, in accordance with a preferred embodiment of the present invention; and

Fig. 5 is a block diagram of the functional components participating in the method proposed, in accordance with a preferred embodiment of the present invention; and

Fig. 6A depicts the application protocol-specific message fields that are functional to the operation of the proposed method, in accordance with a preferred embodiment of the present invention; and

FIG. 6B depicts the structure of the backward connection list, in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method and system for the creation, maintenance, and the sustaining of a continuous connection between a client and a server within a single communication session is disclosed. Clients communicating with a server introduce two types of requests. The clients initiate a regular communication session via the submission of a request to the server for the transmission of a single associated response from the server. The regular requests are completed by the practically immediate reception of a single response. Subsequent to a received response associated with a regular request the session is terminated and the communication link is disconnected. This type of connection will be referred to as a "forward connection" and the request associated with this type of connection will be referred to as a "forward request". In parallel to the forward requests the client is provided with the option of creating an uncompleted request. The client submits an uncompleted request to the server by introducing a specific predetermined parameter within the message object wrapping the request. The introduced parameter specifies that the network transport connection between the client and the server should remain open independently of the flow of responding messages from the server back to the client. This type of connection will be referred to as a "backward connection" and a request associated with a permanent connection will be referred to as a "backward request". The server is provided with the capability of identifying and handling a backward request. The server intercepts and recognizes a backward request by examining the appropriate

parameter introduced by the client within the body of the message containing the request. When the server identifies a backward request, the accompanying client control data and connection control data is inserted into a backward connection list. The list thus created is a series of records containing control information specifying currently open backward connections. Persons skilled in the art will appreciate that any type of data structure for the connection list is contemplated by the present invention. Additionally, the server responds to the backward request by sending to the requesting client a response message object in order to confirm the opening of a backward connection. The message object is having specific control parameters set such that various operative components in the network disposed along the communication path between the requesting client and the responding server having the autonomous capability of terminating a connection are informed of the setting up of a backward connection. Thus, the confirmation message communicates the event of setting up a backward connection not only to the requesting client, but also to the entire set of relevant network components, such as the web server, and the proxy servers operative in maintaining the connection. Following the confirmation of the backward connection to the client and the notification of all the relevant network particles, the client is connected permanently to the server and considered to be "online".

When the application server originates information, such as a notification of an event, an update sent by an external or internal information source, or the like, which is intended to be sent to a specific client, the server scans the connection records within the backward connection list. If a backward connection record exists in the list for the specific client the server obtains the connection control data belonging to the client and utilizing the specific control information therein transmits a message object containing the new data to the client. Due to the prior backward connection confirmation message, the network particles located along the path of the network connection are aware of the backward connection. Therefore none of the components will terminate the connection. As the client is also aware of the backward connection, after the client

receives the messages, which were sent by the server, the connection remains open.

The client is provided with the capability of sending regular requests or forward requests independently of the open backward connection. The forward requests are handled in the manner known in the art. When the client explicitly terminates the backward connection or the termination of the backward connection is effected as a result of a network failure, the application server is appropriately notified by the web server and the backward connection list is suitably updated by the deletion of the applicable connection record. In order to sustain a continuous connection for a considerably long period various functional connection-related parameters associated with network activity should be reset. Additionally, the auto-reconnect feature of the application protocol can be configured to sustain continuous connections even under extreme circumstances.

The following description is presented in order to enable any person skilled in the art to make and use the invention. For the purposes of the explanation specific terminology is set forth to provide a thorough understanding of the invention. However, it will be apparent to one skilled in the art that the specific details introduced in the description are not required to practice the present invention. Descriptions of a specific application are provided only as example. Various modifications to the preferred embodiment will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. Thus, the present invention is not intended to be limited to the embodiment shown, but it is to be accorded the widest scope consistent with the principles and features disclosed herein.

In accordance with the practices of persons skilled in the art of computer programming, the present invention is described below with reference to acts and symbolic representations of operations that are performed by the processing system. It will be appreciated that the acts and symbolically

represented operations include the manipulation of electric signals by a central processing unit (CPU). The electrical system represent data bits which cause a resulting transformation or reduction of the electrical signal representation, and the maintenance of data bits at memory locations in the memory system to thereby reconfigure or otherwise alter he CPU's operation, as well as other processing of signals. The memory locations are physical locations that have particular electrical, magnetic, optical, or organic properties corresponding to the data bits.

The application server presented in the following description is a set of computer programs implemented on a computing platform. The application server presented in the following description contains computer software instructions specifically developed for the practice of the present invention. The software in the presented application server causes the server to perform the various functions described herein, although it should be noted that it is possible to use dedicated electronic hardware to perform all server functionality described herein. In the second case the application server can be implemented as firmware by the embedding of the predetermined program instructions and/or appropriate control information within suitable electronic hardware devices containing application-specific integrated circuits.

Fig. 2 illustrates a simplified block diagram of a data communication network 41, such as the Internet. Client system 42 implemented on a first computer platform is communicatively coupled to server system 46 implemented on a second computer platform via conventional communication link 48, data communication network 44, and conventional communication link 50. Communication links 42, 46 could be implemented using standard communication hardware components such as modems, network interface cards, standard communication lines, such as twisted pair telephone wires, coaxial cables or fiber optic channels. Links 42, 46 are effectuated by known communication software such as network browsers, communication control routines, and the like. Client 42 intermittently connects to server 46 in order to access and interact with the content

information embedded on server 46. Server 46 responds to the requests of client 42 by sending the requested information back to client 42 utilizing conventional communication link 50, the data communication network 44, and the conventional communication link 48. Although in the drawing only one client, on proxy server, and one server are shown it will be easily perceived that within a realistically configured data communication network a plurality of clients can be communicatively coupled to a plurality of servers via a plurality of proxy servers. In addition communication networks typically contain network components operative in the controlling, processing, filtering, and the transmission of information. Such components are remote access servers, routers, proxy servers, gateways, and the like. A data communication network could also include connection points to other type of communication systems, such as Public Switched Telephone Networks (PSTNs), Cable Television Networks (CATVs), satellite networks, and the like.

Note should be taken that client 42 could be implemented on diverse computing and communicating platforms such as Personal Computers (PCs), Personal Digital Assistants (PDAs), mobile devices, WAP-enabled cellular devices, and the like. Client 42 could be implemented on devices located on diverse communication networks, such as LANs, WANs, cellular networks, satellite networks, and the like. The devices could have provided with advanced access and interaction capabilities to data communication networks such as the Internet when suitable access is made thereto through special gateway devices. Various operational techniques and protocols could be used in the source networks such as HTTP, WAP, and the like. The data communication networks operate according to various communication and application protocols. In the preferred embodiment of the present invention the data communication network is the Internet, and the application protocol utilized therein is the Hypertext Transfer Protocol (HTTP).

With reference to Fig. 3 a client system 52 is implemented on a computing platform, which operates in a computing and communicating environment, such as a data communication network. Client 52 contains central processing unit (CPU) 56, communication device 58, I/O device 60, and memory device 62. Memory device 62 stores operating system 64, and client application 66. CPU 56 is the central unit in the computer containing the logic circuitry that performs the instructions of a computer's programs. Communication device 58 is responsible for communicatively connecting client 52 to a data network. Device 58 is a modem or a network interface card. I/O device 60 transfers data to or from the computer unit. Typical I/O devices are printers, display screens, hard disks, keyboards, and mouses. Memory device 62 is the electronic holding place for instructions and data necessary for the proper operation of the computer platform. Device 62, which is typically a hard disk, contains the operating system 64 and client application programs 66. Operating system 64 is a control program that manages the operation of all the other programs in the computer platform. In addition operating system 64 performs various services for the client applications 66, such as sharing of the memory, management of the input/output, controlling of the multitasking, and the like. Operating system 64 could be any of the known operating systems such as UNIX, Linux, Windows 98, Windows NT, Windows 2000, Windows CE, diverse mobile operating systems (OS), and the like. Client applications 66 are programs that perform specific applications for the users of the client platform. An application, which is central to the preferred embodiment of the present invention is a Web browser that performs content requests submitted by the users of client 52 to information sources within a data network and enables interaction with the content received.

One such information source is located on server system 54. Client 52 is communicatively connected to server 54 via communication link 68 set up by the respective communication devices of client 52, server 54, and other network control components. Server 54 is implemented on a computing platform, which operates in a computing and communicating environment, such as a data

communication network. Server 54 contains central processing unit (CPU) 70, communication device 72, I/O device 74, and memory device 76. Memory device 76 stores operating system 78, and server applications 80. CPU 70 is the central unit in the computer containing the logic circuitry that performs the instructions of a computer's programs. Communication device 72 is responsible for communicatively connecting server 54 to a data network. Device 72 is a modem or a network interface card. I/O device 74 transfers data to or from the computer unit. Memory device 76 is the electronic holding place for instructions and data necessary for the proper operation of the computer platform. Device 76 contains the operating system 78 and server applications 80. Memory device 76 is preferably a hard disk. Operating system 78 is a control program that manages the operation of all the other programs running in the computing platform. In addition operating system 78 performs various services for the server applications 80, such as sharing of the memory, management of the input/output, controlling of the multitasking, and the like. Server applications 80 perform applications for the users of the computing platform. One application relevant to the preferred embodiment of the present invention is a web server, which is a back-end application that handles the requests sent by client 52. As Web server the Internet Information Server (IIS) can be used. IIS contains a set of Internet Server Application Program Interfaces (ISAPIs). ISAPIs are a set of program calls that enable to develop back-end applications. Using ISAPI, a Dynamic Link Library application file can be constructed, which can run as part of the Hypertext Transport Protocol (HTTP) application's process and address space. The DLL files are loaded into the computer when HTTP is started and remain there as long as they are needed. A special kind of ISAPI DLL is called an ISAPI filter, which can be designated to receive control for every HTTP request. An ISAPI filter can be implemented for encryption or decryption, for logging, for request screening, or for other purposes.

Fig. 4A illustrates the elements of a highly simplified communicating environment and the steps involved in setting up a continuous connection, according to the present invention. The application protocol utilized in the environment is the HyperText Transfer Protocol (HTTP) or any other stateless session-oriented protocol. In order to open a continuous connection with server 84 client 82 sends an uncompleted request to server 84. The request is uncompleted such that no single response from the server 84 will effect the termination of the connection. Client 82 allows the connection 86 to remain open. Connection 86 does not close because server 84 does not send a closing instruction for terminating the open connection. The client 82 initiates the uncompleted request by the introduction of a prescribed parameter within the header or the body of the request. The request and the contained prescribed parameter will cause the opening of a continuous network connection 86 between the client 82 and the server 84. The continuous connection 86 will remain open as long as the client 82 desires to "listen" or to be functionally receptive to server-initiated communication from server 84 via transport 86. Therefore, as long as the transport 86 is open whenever server 84 sends information to client 82 the information will be functionally received by client 82. The continuous connection 86 enables client 82 to receive timely information from server 84 without the necessity for polling server 84 i.e., without the need for introducing repetitive requests to server 84. Repetitive requests typically increase the traffic volume within the communication network, and significantly increase workload on server 84 by compelling server 84 to respond to the entire set of the repetitive requests. Thus, the setting up of continuous connections according to the proposed method and system will substantially diminish the volume of the traffic within the network and will substantially reduce the workload on server 84. Following the reception of the uncompleted request server 84 suitably processes the uncompleted request, and communicates a specific response back to client 82 via the transport 86. The communicated response is operative in finalizing the opening of connection 86 as a continuous connection. The communicated

response sent by server 84 is also operative in the transmission of notifications regarding the setting up of the continuous connection 86 to specific network components disposed on the communication path between server 84 and client 82 that have the capability of independently terminating a connection. Connection 86 remains open because server 84 does not send a "close connection" instruction, as a result of the sending of the specific response by the server 84 and because the client has been programmed to keep the connection open. Subsequently connection 86 will not be terminated by the network components having the capability of closing network connections independently.

Fig. 4B illustrates the proposed connectivity technique according to the method and system of the present invention in a simplified networking environment utilizing the communication protocols and the HTTP 1.1 application protocol. HTTP 1.1 client introduces requests 98 and 104 to HTTP 1.1 server 90. Server 90 can be one of many available HTTP servers such as IIS, Apache, and the like. Server 90 is responsible for low-level transport managed by the HTTP 1.1 protocol, the SSL security protocol, and the like. For each separate requests 98, 104 separate connections 100, 106 are established. Request 104 is a conventional "forward" request operative in the opening of conventional "forward" connection 106. Server 90 receives forward request 104 and forwards the request 104 to request processor 96 of application server 92. Server 92 can be an ISAPI DLL application, a CGI application used as an application server "on top" of the HTTP server 90 and the like. Server 92 is utilized as a back-end providing business rules, logic rules, database access, and the like. Interface 97 provides the ISAPI/CGI interface between server 90 and server 92. Request processor 96 processes request 104 via the appropriate application services and after recognizing the request as a conventional forward request communicates the resulting data or notifications to server 90. Server 90 utilizing the communication channel 106 transmits a conventional practically immediate response 108 to client 88. When client 88 receives the immediate response 108 the session associated

with the request-response pair 104-108 terminates and the connection 106 closes. Client 88 is capable of initiating new connections and thereby opening new communication channels to server 90. Request 98 is a "backward" request operative in the opening of a "backward" connection 100. Server 90 receives backward request 98 and sends the backward request 89 to request processor 96 of application server 92. Server 92 can be an ISAPI DLL application, a CGI application used as an application server "on top" of the HTTP server 90 and the like. Server 92 is utilized as a back-end providing business rules, logic rules, database access, and the like. Interface 97 provides the ISAPI/CGI interface between server 90 and server 92. Request processor 96 processes request 98 via the appropriate application services and after recognizing the request as a backward request assembles a specific confirmation and notification response concerning the finalization of connection 100 as a continuous connection. The response 101 wrapped within a message is communicated to server 90. Server 90 utilizing the communication channel 100 transmits the message 101 to client 88. Client 88 initially creates the connection 100 to remain constantly open. As results of the message 101 client 88 receives notification that its request was fulfilled. Connection 100 remain open because server: a) does not close it; b) server put to response header (or in the body) corresponding values for not closing the connection; c) client 88 is set to a constant connection state and does not close the connection by itself. As a result of the message client 88 allows the connection 100 to remain open. Connection 100 is finalized as a "backward" connection or a continuous connection. The notification message 101 is also operative in notifying other network components (not shown) disposed along the communication path between client 88 and server 90 regarding the finalization of connection 100 as a continuous connection. As a result of the notification the network components allow the connection 100 to remain permanently open. When server 92 is updated with new data originating from diverse external or internal information sources or routines (not shown) and the updates are operative in the activation of processes for the notification of client 98. The notification manager 94 of server 92 may

assemble suitable data objects 103, 105, 107 containing the new data. The data objects 103,105,107 are transmitted to server 90. Server 90 sends the data objects within the well-defined formatted HTTP 1.1 messages 102 and sends the messages 102 to client 88 via transport 100. As a result of the continuous status of the transport 100 the client 88 is functionally receptive to transmission of messages from server 90. Therefore, as long as connection 100 is open the messages 102 sent by server 90 are functionally received and appropriately processed by client 88. Note should be taken that in parallel with a number of open backward connections client 88 is allowed to submit forward requests effecting a number of conventional forward connections. Client 88 can also terminate the continuous connection 100 through the activation of predetermined procedures.

Referring now to FIGs. 5 and 6 respectively, FIG.5 illustrates the connectivity flow between the various components of a simplified data communication network in accordance with the proposed method and system of the present invention. Client 110 is implemented on a computing platform communicatively linked to other computing platforms across a data communication network. Client 110 includes a client agent 120 and can also include a local database 122. Client agent 120 is preferably a Web browser. Agent 120 is an application program that provides a way to look at and interact with information sources on a data network such as the World Wide Web (the Web). Agent 120 is a client program that uses the Hypertext Transfer Protocol (HTTP) to make requests of Web servers throughout the Internet on behalf of the browser user. Agent 120 can be any network browsers implementing stateless session-oriented protocols such as HTTP or the like. Agent 120 can also be a specifically written agent for accomplishing the tasks described herein. Known network browsers that can be utilized are the Netscape Navigator or the Microsoft Internet Explorer (MSIE). Optional local database 122 is a formatted data

structure operative in storing client-side application data. Other methods for storing the client-side application data can be equally used.

Client 110 is communicatively linked to server 111 via a data communications network. A proxy server 112 is deployed in the path of communication between the client 110 and the server 111. Server 112 is used as an intermediary between a client and the data network in order to ensure security, administrative control, and caching service. Typically, a proxy server is associated with or is a part of a gateway server that separates a local data network from the external data network and a firewall server that protects the local network from outside intrusion.

Web server 114 is contained within the server system 111 implemented on a computing platform communicatively connected to the data network. Server 114 can be any one of the HTTP server extendible applications such as the Internet Information Server (IIS) developed and distributed by Microsoft Corporation and the like. Other extendible HTTP servers can be used, such as UNIX-specific HTTP servers using CGI technology and the like. Any other HTTP server can be employed in the context of the present invention. In the preferred embodiment of the present invention server 112 is an IIS server. IIS can be easily extended by using the built-in Internet Services API (API) routines. ISAPI is a set of program calls that provide the capability of writing back-end applications. An ISAPI extension is a Dynamic Link Library (DLL) routine. The DLL is called by submitting to the IIS a Uniform Resource Locator (URL) with a virtual path to the DLL. A special kind of ISAPI DLL is called an ISAPI filter, which can be designated to receive control for every HTTP request. An ISAPI filter can be created for various purposes such as for request screening, or the like. Other elements having like operation and functionality can be similarly employed and are equally contemplated by the present invention. Front-end is a term used to characterize program interfaces and services relative to the initial user of these interfaces and services. The user may be a human being or a program. A front-end application is one that application users interact with directly. In contrast, a

back-end application program serves indirectly in support of the front-end services, usually by being closer to the required resource or having the capability to communicate with the required resource. The back-end application may interact directly with the front-end. The back-end application is typically a program called from an intermediate program that mediates front-end and back-end activities. In the preferred embodiment of the present invention, server 116 is developed using ISAPI extensions. Application server 116 is a server-side back-end application implemented as part of the server system 111 embedded on a computing platform within a data communication network. Server 116 includes request processor 126, backward connection handler 130, and backward connection list 132. Request processor 126 is operative in servicing requests submitted by client 110. Processor 126 accepts requests from client 110, and forwards the requests to a database server 118 for processing. Processor 126 is also operative in receiving incoming content, services, notifications, or updates from database server 118 and appropriately transmitting the received content back to client 118. In the preferred embodiment of the present invention, processor 126 controls backward connection handler 130. Handler 130 is activated by processor 126 in order to identify backward requests, to effect the insertion of backward connection records into backward connection list 132, to search for and obtain backward connection records from backward connection list 132 in order to identify operative continuous connections and for effecting the transmission of new data via the open connections. List 132 is a formatted data structure which stores data associated with continuous connections. Database server 118 is a back-end application implemented within a server system on a computing platform communicatively linked to a data network. Database server 118 is operative in supplying content information such as Web pages, data, or files within the framework of responses to the requests of client 110. Server 118 is linked to content database 132. Server 118 is operative in processing requests from request processor 116. Server 118 accesses content database 134, obtains the content information requested by client 110 and transmits the requested content to request

processor 126. For the clarity of description FIG. 5 includes a single database server associated with a single content database. In the preferred embodiment of the present invention, a plurality of database servers associated with a plurality of content databases could be operative. However, persons skilled in the art will appreciate that the presence or absence of databases serves only to better describe the present invention. The present invention may function equally in the absence or with the presence of databases. In order to initiate a backward connection to the server 111, client 110 submits an HTTP request via the services of client agent 120. Agent 120 can be a Web browser and operates in the known manner of network browsers such as the Netscape Navigator or the Microsoft Internet Explorer. Agent 120 can be a specially written computer program for connecting to a network and forwarding an receiving network-related instructions. An HTTP request is submitted by the utilization of a Uniform Resource Locator (URL). The URL will typically contain the address of the server 110, the relevant port number, the identification of the ISAPI extension routine to be activated, and the value of a predetermined parameter, such as BACKWARD=YES in the body or header of the message. In addition extendable header fields can be employed. The structure of an exemplary URL to be submitted is shown as the following statement:

"http://server-ip/server-port/server.dll?Backward"

In addition to the URL, the client 110 supplies a regular block of HTTP headers, containing identification and control information to allow the server 111 to identify the backward connection and associate the information thereof with the particular client 110. The identification and control information enables the server 111 to keep a list of open backward connections 132 with associated control information identifying the backward connection-initiating client systems. Thereby, notifications received from diverse external or internal information sources, such as a database server and are intended to be sent to the client system 110 are transmitted by the server 111 through utilizing the backward connection control information stored within the backward connection list 132. The request introduced by client 110 is formatted as an HTTP request and

transmitted to the server 111. The request forwarded to the web server 114 and to the application server 116. The request processor 126 forwards the request to the backward connection handler 130. Handler 130 examines the request contents for the presence of the particular BACKWARD parameter. Requests without the specific parameter are handled like conventional forward requests in the manner known in the art. Identified backward messages are appropriately processed by the extraction of the relevant control information from the header or the body of the request. The processing of the backward request also includes the building and the insertion of a backward connection record 159, 161 of FIG. 6B into backward connection list 132. The backward connection record 159 of FIG. 6A includes connection identification 156, and client identification 158. Record 159 is utilized as a pointer to backward connection record 161 of FIG. 6B, Backward connection record 161 of FIG. 6B contains target Internet Protocol (IP) address 160, a port number 162 and a source IP address 164. Server 111 will create a suitable response 140 of FIG. 6A intended as confirmation and/or notification to client 110 and to other network components. The response 140 of FIG. 6A is effective in finalizing the continuous state of the connection. The response message header 142 of FIG. 6A includes a Connection field 146 with the value of the field set to "KEEP-ALIVE", a Connection-Length field 148 with the value of the field set to a maximum possible value, such as "ULONG_MAX", and a Content-Length 150 field with the field set to a maximum possible value, such as "ULONG_MAX". The client 110 is notified regarding the setting up of a continuous connection by the setting of the variable Connection-Length 148 to the maximum possible value "ULONG_MAX". The maximum possible value of the variable Connection-Length 148 will require client 110 to wait for an indefinite period before effecting a disconnection due to a client-side timeout signal. The maximum possible value of the variable Connection-Length 148 of FIG. 6A enables the client agent 120 to receive blocks of content information transmitted from the server 111 independently of specific requests. In addition to notifying client 110 other specific network components should be notified in regard to the setting up

of a continuous connection. The value of the variable Content-Length 150 of response header 142 is also set to the maximum possible value such as ULONG_MAX. Thereby, information of the IIS detection threads, of the proxy server 112, which is disposed on the communication path between the client 110 and the server 111 and of client 110, is effected. The Content-Length 150 value indicates that the current request is not yet completed and will be completed only after a maximum possible value of bytes will be transmitted from server 111 to client 110. Proxy server 112 disposed along the communication path between client 110 and server 111 may close the continuous connection independently of the actions performed by server 111 or by client 110. Typically, for reasons of performance optimization proxy server 112 has been provided with sophisticated built-in routines in order to perform autonomously supplementary connectivity optimizations within the communication network. By analyzing sets of HTTP instructions, and by utilizing a set of configurable timeout values, proxy server 112 may terminate a network connection independently of connection-specific operative components such as the client 110 or the server 111. Depending on the precise network installation and configuration actions such as these, which follow, should be performed in order to prevent the proxy server 112 from interfering with the sustaining of a continuous connection.

a) The field Connection 146 within HTTP header 142 of HTTP response 140 on FIG. 6B should contain the value "KEEP-ALIVE" to inform all the relevant network components that the relevant TCP socket should not be closed even if a proxy server considers the current request as already completed. The KEEP-ALIVE messages can be sent on the application protocol level from the server to the client. Such signals can be used by the client to restore or reestablish broken connection.

b) The field Connection-Length 148 within HTTP header 142 of HTTP response 140 on FIG. 6B should contain the maximum possible value such as "ULONG_MAX" in order to disable the proxy server timeouts. The maximum possible value such as "ULONG_MAX" will make the proxy

server to consider the request uncompleted even after a considerably long period.

c) IIS connection timeout values should be set to a maximum possible value such as "ULONG_MAX". IIS connection timeout values can be configured via the Internet Service Manager application.

d) The HTTP client 110 connection timeout value should be set to a maximum possible value. HTTP specifications require HTTP clients not to close a connection since typically the HTTP servers are responsible for the closing a connection. However, many HTTP clients perform their own timeout checking and occasionally do close a connection when such a timeout occurs. For example, Microsoft Internet Explorer (MSIE) utilizes a default timeout value set to twenty minutes. The connection timeout option of an HTTP client is should be disabled by appropriately modifying the suitable registry settings.

e) As a "natural" timeout can still occur under certain circumstances, like for example after transmitting a considerably large number of notification data over a considerably long period of time, an auto-reconnect feature restoring backward connection, should be added to the client/server implementation scheme.

When the application server 116 wants to notify a client about some event, the backward connection handler 130 is activated in order to look up the client's connection identification 158 of FIG. 6B in the list of the backward connections 132. Subsequently the connection identification 158 of FIG. 6B and associated connection data 160,162,164 of FIG.6B is utilized to transmit data via the backward connection to the client 110. Client agent 120 is operative in the processing of the received information. Optionally by the utilization of client-side routines the received information is inserted into local database 122. The IIS WriteClient function can be used to send data to the client 110 identified by the given connection identification fields 160,162,164 of FIG. 6B. When client 110

disconnects or the continuous connection is terminated as a result of diverse network-related events, such as a network failure, the IIS 114 informs backward connection handler 130 in regard to the connection loss. The disconnection-related information is accompanied by an appropriate reason code. The receiving of the disconnection-related information effects the removal of the connection identification 158 of FIG. 6B from the backward connection list 132.

The proposed method and system is operative in the creation, maintenance, and the sustaining of continuous network connections within the framework of a single communication session. The method and system according to the teachings of the present invention operate within a data network environment that utilizes a stateless application protocol such as the HTTP. Messages transported via the HTTP protocol are sent to and received from a well-known port, such as the standard HTTP port 80. As the present invention accomplishes a continuous connection transport within the framework of the HTTP, the requests and responses transmitted between the operative components of the proposed system are communicated via port number 80. As port number 80 is a well-known standard port the messages will not be blocked by firewall routines configured to filter messages transmitted through non-standard ports. Thus a method and system for the creating, maintaining, and sustaining a continuous connectivity transport between operative components of data communication network incorporating the teachings of the present invention has been described. The present invention is not limited to what has been particularly shown and described hereinabove. Rather the scope of the present invention is defined only by the claims which follow.

I/WE CLAIM:

1. In a computing environment accommodating at least one client system connectable to at least one server system via at least one network control unit in a data communication network, a non-polling method of transmitting notifications from the one or more server systems to the at least one client system utilizing a stateless session-oriented application protocol, the method comprising the steps of:

submitting at least one backward request operative in opening of a continuous connection between a first computer system and a second computer system by the first computer to be sent to the second computer;

identifying the backward request transmitted by the first computer system by the second computer system;

inserting a backward request entry associated with the transmitted backward request into a list of the backward connections open between the first computer system and the second computer system by the second computer system;

responding the the reception of the backward request by transmitting a response to the first computer system for confirming the opening of the continuous connection between the first computer system and the second computer system from the second computer system;

communicating a notification message by the second computer system regarding the opening of the continuous connection between the first computer system and the second computer system to at least one network control unit deployed in the communications path between the first computer system and the second computer system;

reacting to at least one application-related event by transmitting at least one notification regarding the at least one event to the first computer system by the second computer system utilizing the communication path associated with the continuous connection.

2. The method of claim 1 further comprising establishing a list of continuous connections between the first computer system and the second computer system.

3. The method of claim 1 further comprising transmitting regular forward requests independently of the continuous connection open between the first computer system and the second computer system by the first computer system to the second computer system.

4. The method of claim 1 further comprising terminating the continuous connection between the first computer system and the second computer system by the first computer system.

5. The method of claim 4 wherein the step of terminating further comprising deleting the entry associated with the continuous connection from the backward connections list on the second computer system.

6. The method of claim 1 wherein the step of identifying comprises the step of examining the backward request transmitted by the first computer system to the second computer system for the presence of a predetermined parameter in the body of the backward request.

7. The method of claim 1 wherein the step of communicating comprises the step of introducing into the response message predefined parameters operative in informing the at least one network control unit regarding the opening of the continuous connection between the the first computer system and the second computer system

8. The method of claim 1 wherein the step of submitting comprises the step of introducing a predetermined parameter within a message object warping the backward request.

9. The method of claim 1 wherein the step of reacting comprises the sub-steps of:

identifying the first computer as the target of the received content information;

scanning the backward connections list in order to obtain the control information needed to forward the content to the first computer;

utilizing the control information to transmit the content message to the first computer.

10. The method of claim 8 wherein the predetermined parameter specifies that the network transport connection should remain open between the first computer system and the second computer system independently of the flow of messages from the second computer system to the first computer system.

11. The method of claim 1 further comprises an auto-connect feature to sustain continuous connections under exceptional circumstances.

12. In a computing environment accommodating at least one client system connectable via at least one network control unit to one or more server systems in a data communication network, a system for creating, sustaining and maintaining at least one open continuous connection between the at least one client system and the server system, the system comprising:

a backward connection list implemented on the second computer system to hold at least one record associated with at least one continuous connections;

an application server implemented on the second computer system;

a backward connection handler component associated with an application server implemented on the second computer system;

13. The system of claim 12 wherein the backward connection handler identifies backward requests submitted by the web browser from the first computer, inserts backward connection records into backward connection table, an obtains continous connection records from the backward connection table to identify operative continuous connections between the second computer system and the first computer system.

14. The system of claim 12 wherein the backward connection list is a data structure holding control information associated with the operative continuous connection records.

15. The system of claim 14 wherein the backward connection list comprises:
a user identification to identify the use associated with the continuous connection;
a connection pointer to identify a continuous connection record stored in the backward connections list;
a target IP address to identify the first computer Internet Protocol address;
a port number to identify the port associated with the session;
a source IP address to identify the source of the content, data, or event to be transmitted to the first computer.

16. The system of claim 12 wherein the application server comprises a request processor.

17. The system of claim 14 wherein the request processor is operative in servicing requests submitted from the first computer system.

18. The system of claim 14 wherein the request processor further operative in receiving incoming content, services, notifications, and updates from the database server.

19. The system of claim 14 wherein the request processor further operative in transmitting the received content to the web browser on the first computer system.

20. The system of claim 12 further comprising:

a web browser implemented on the first computer system designed to submit requests to web servers across a data communication network;

a web server implemented on the second computer system to handle requests submitted by web browser from the first computer system;

a database server implemented on the second computer system to handle request for content submitted by the web server;

a content database implemented on the second computer system desgined to hold content information.

21. The system of claim 20 wherein the database baser is implemented on a third computer system across the data communication network.

22. The system of claim 12 wherein the web server is a Hypertext Transfer Protocol (HTTP) server.

23. The system of claim 12 wherein the system implements a stateless session-oriented application protocol.

24. The system of claim 23 wherein the stateless session-oriented protocol is the HyperText Transfer Protocol (HTTP).

25. The system of claim 11 further comprising:

a network control server deployed on the communication path between the first computer and the second computer utilized as a firewall, a gateway, a router, or as a proxy server.

26. The system of claim 11 wherein the data communications network is the Internet.

27. The system of claim 11 wherein the client system is implemented on a Personal computer (PC) platform, a Personal Digital Assistant (PDA) device, a mobile device, or a WAP-enabled mobile cellular device,

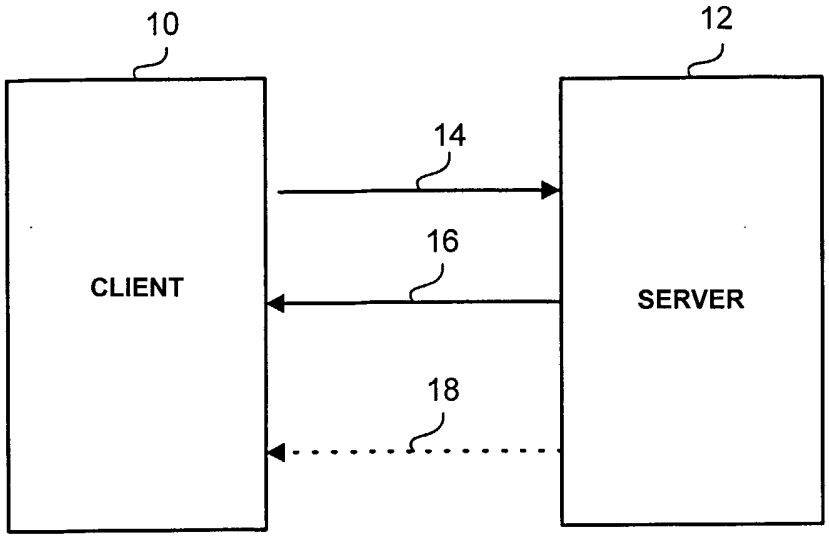


FIG. 1A
PRIOR ART

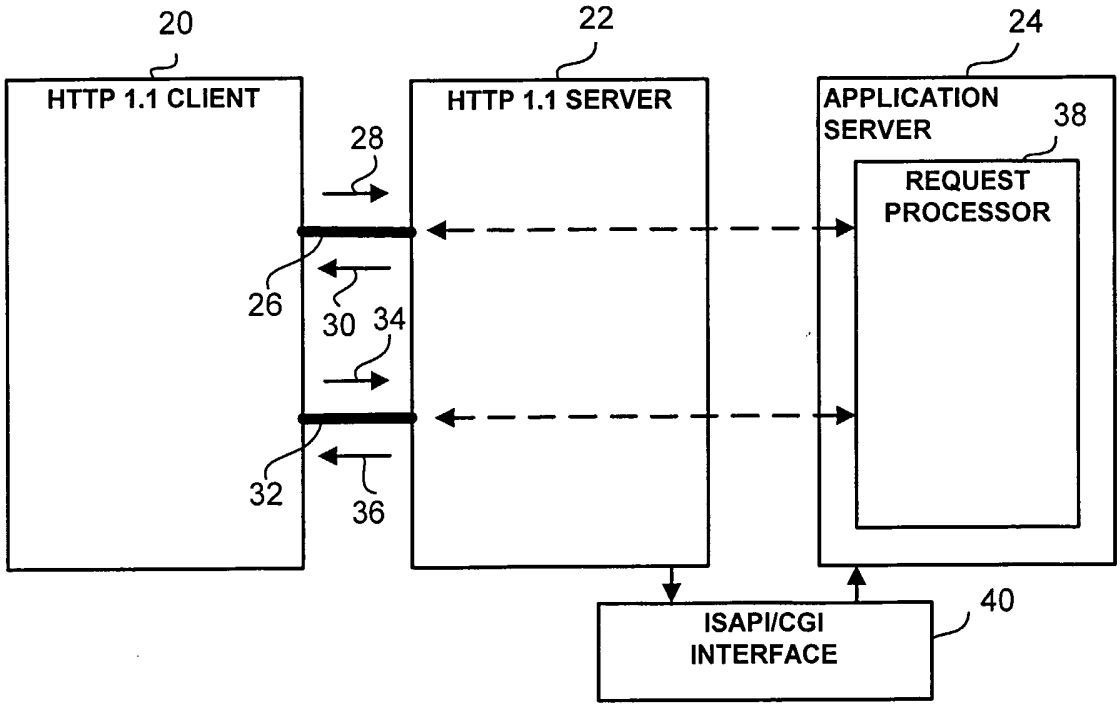


FIG.1B
PRIOR ART

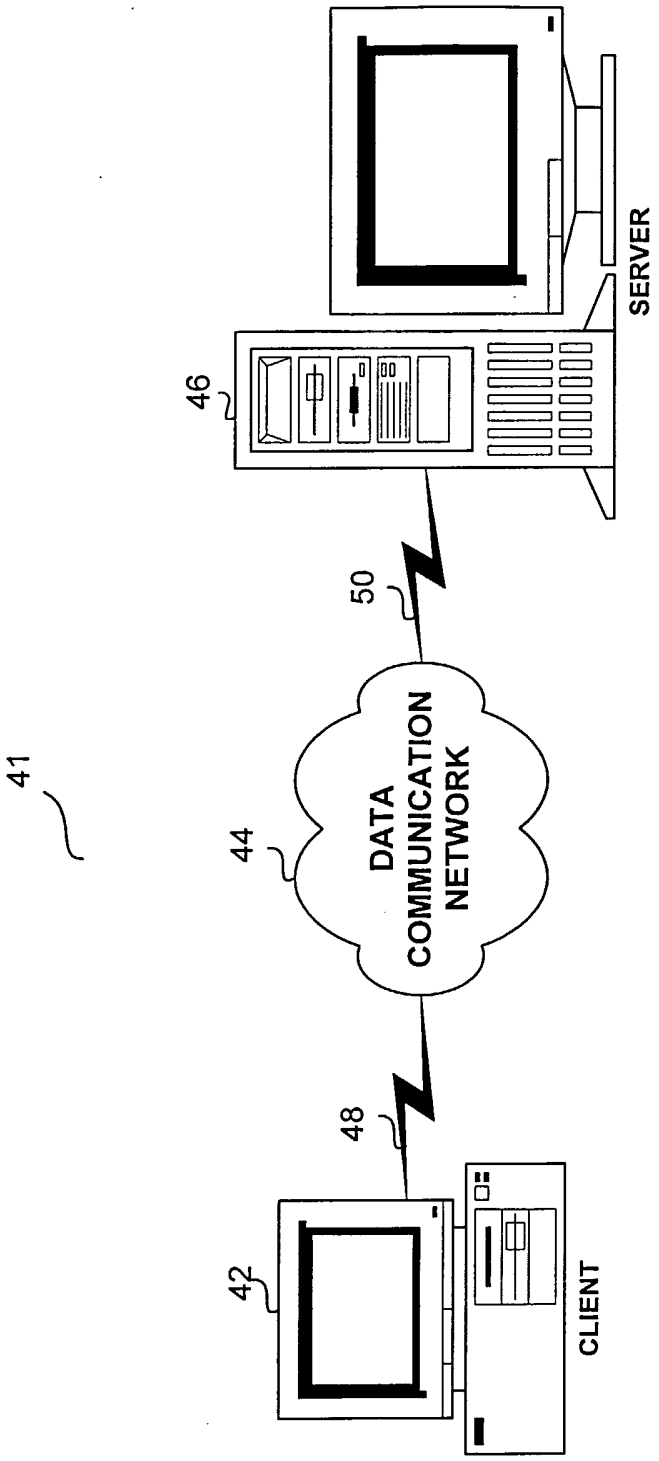


FIG. 2

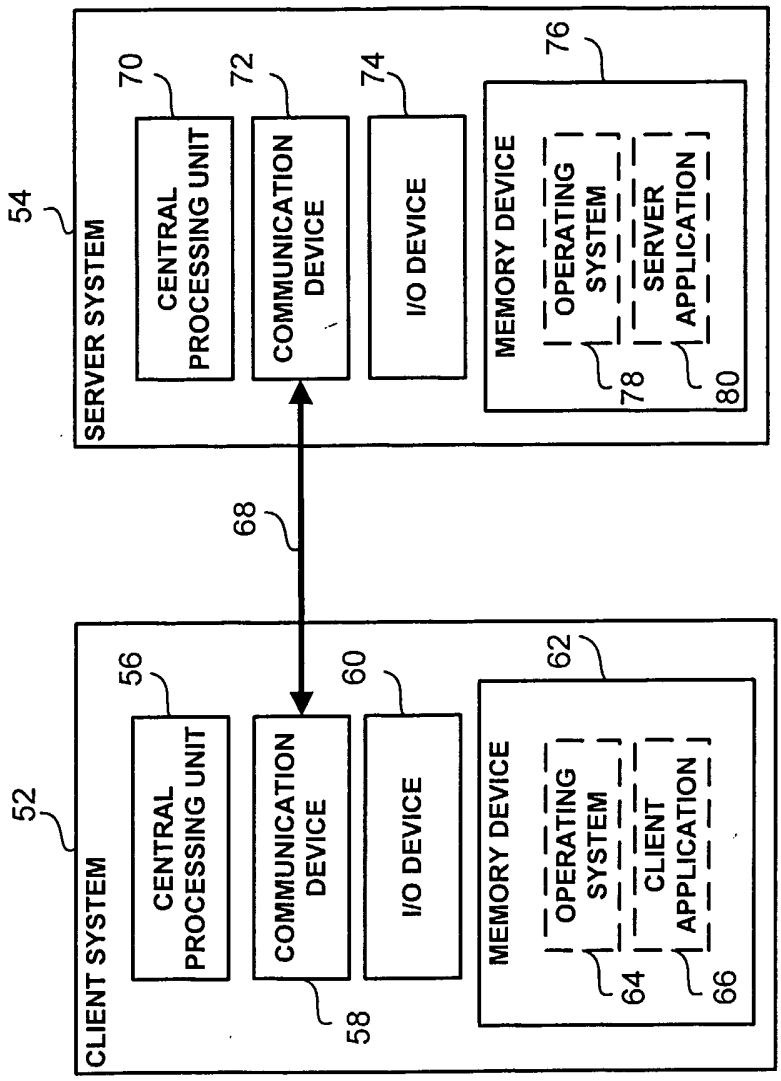

SERVER SYSTEM
54
CENTRAL PROCESSING UNIT
70
COMMUNICATION DEVICE
72
I/O DEVICE
74
MEMORY DEVICE
76
OPERATING SYSTEM
78
SERVER APPLICATION
80
68
CLIENT SYSTEM
52
CENTRAL PROCESSING UNIT
56
COMMUNICATION DEVICE
58
I/O DEVICE
60
MEMORY DEVICE
62
OPERATING SYSTEM
64
CLIENT APPLICATION
66

FIG. 3

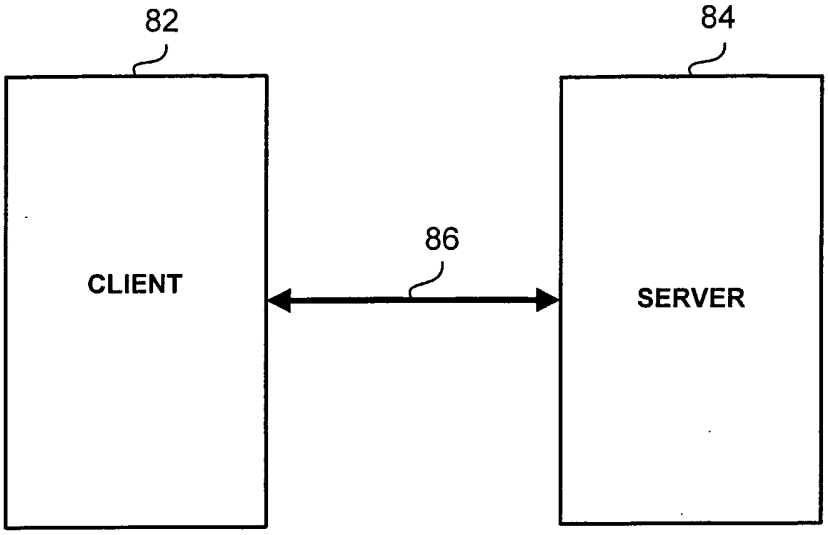

82
84
CLIENT
86
SERVER

FIG. 4A

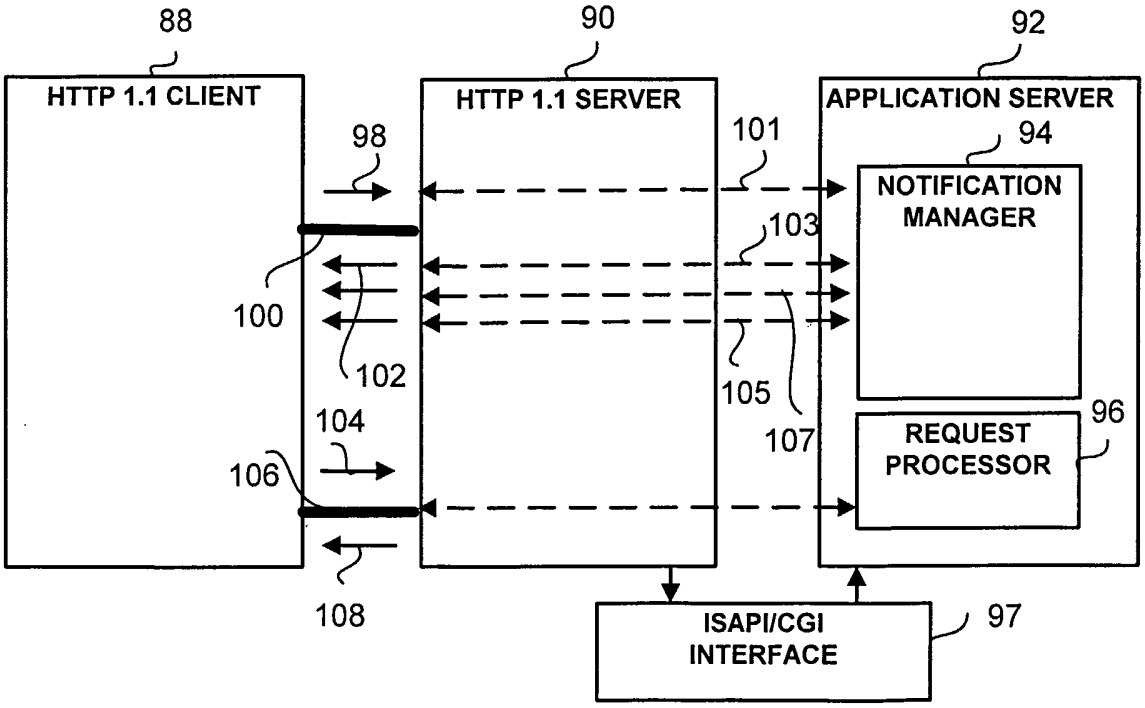

88
90
92
HTTP 1.1 CLIENT
HTTP 1.1 SERVER
APPLICATION SERVER
98
101
94
NOTIFICATION MANAGER
103
100
102
105
107
104
106
96
REQUEST PROCESSOR
108
ISAPI/CGI INTERFACE
97

FIG. 4B

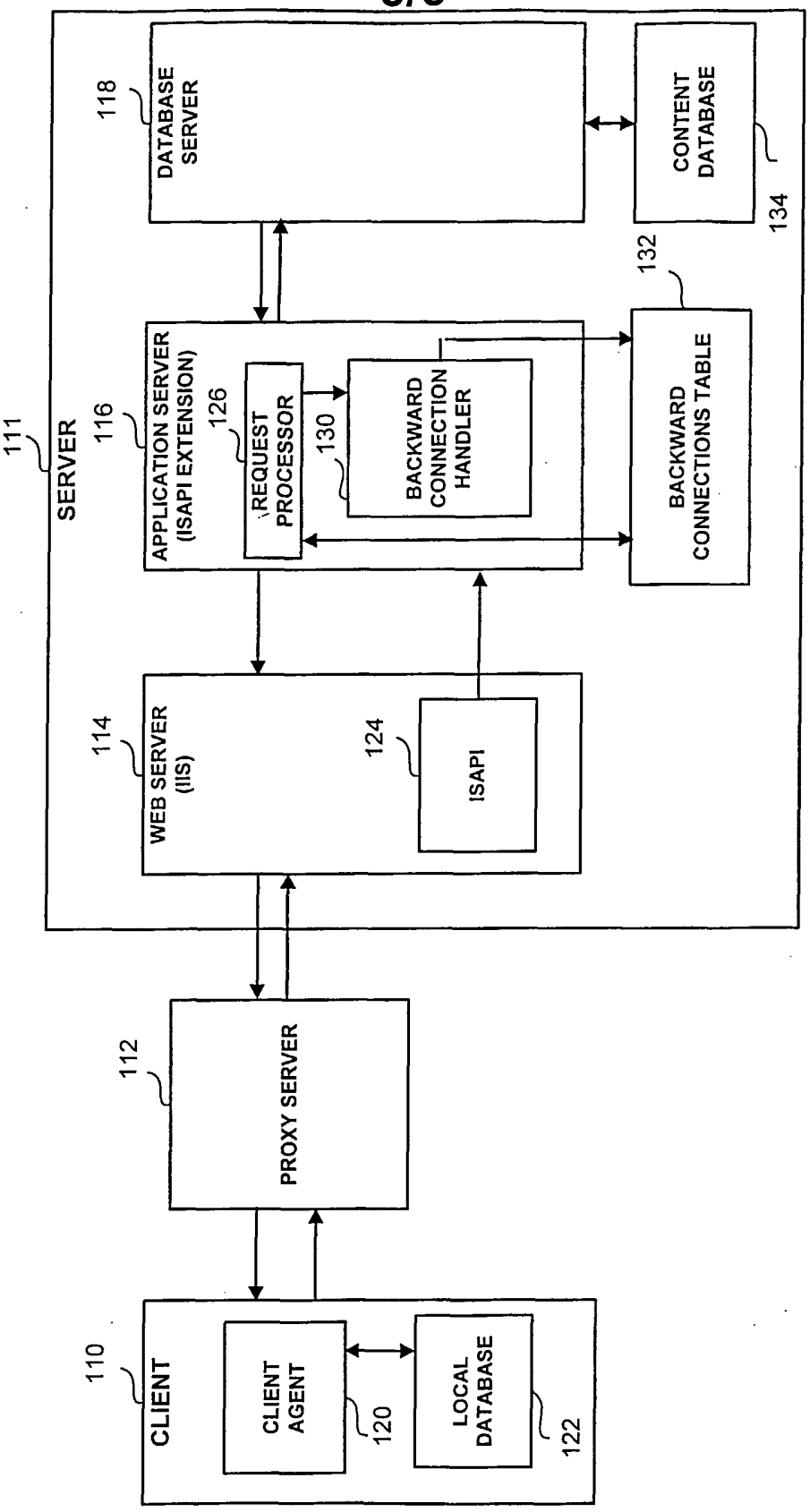

CLIENT
110
CLIENT AGENT
120
LOCAL DATABASE
122
PROXY SERVER
112
SERVER
111
WEB SERVER (IIS)
114
ISAPI
124
APPLICATION SERVER (ISAPI EXTENSION)
116
REQUEST PROCESSOR
126
BACKWARD CONNECTION HANDLER
130
BACKWARD CONNECTIONS TABLE
132
DATABASE SERVER
118
CONTENT DATABASE
134

FIG. 5

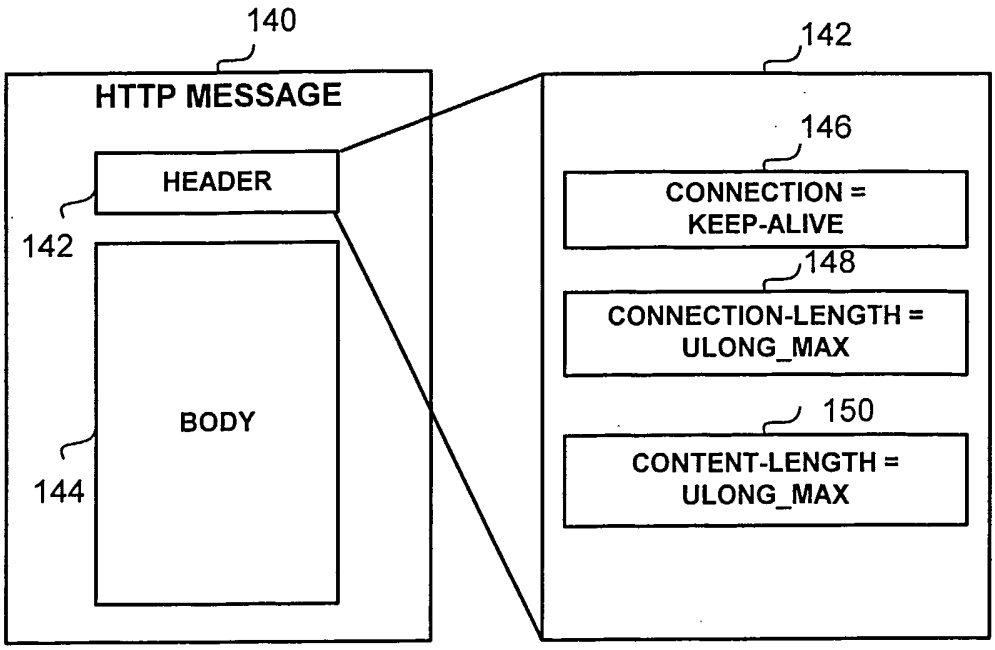

140
HTTP MESSAGE
HEADER
142
BODY
144
142
146
CONNECTION = KEEP-ALIVE
148
CONNECTION-LENGTH = ULONG_MAX
150
CONTENT-LENGTH = ULONG_MAX

FIG. 6A

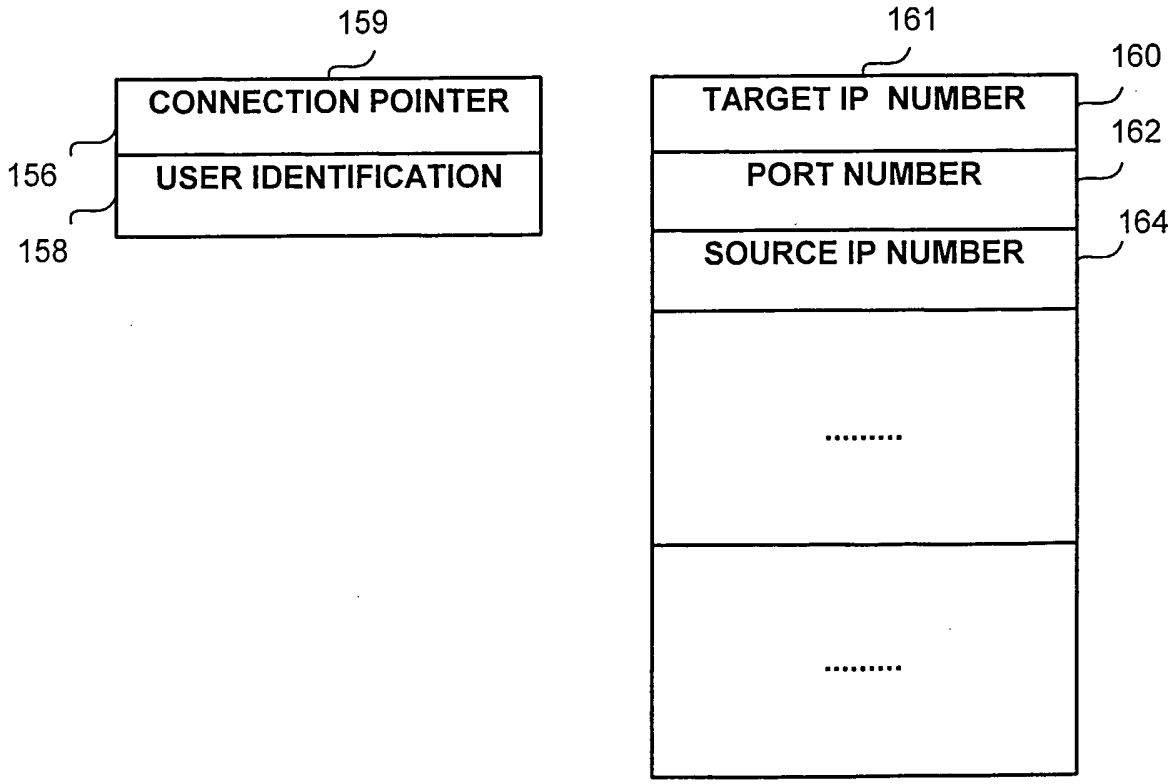

159
CONNECTION POINTER
156
USER IDENTIFICATION
158
161
160
TARGET IP NUMBER
162
PORT NUMBER
164
SOURCE IP NUMBER
.....
.....

FIG. 6B

INTERNATIONAL SEARCH REPORT

International Application No
PCT/IL 01/00793

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, PAJ, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	MOGUL J C: "THE CASE FOR PERSISTENT-CONNECTION HTTP" COMPUTER COMMUNICATIONS REVIEW, ASSOCIATION FOR COMPUTING MACHINERY. NEW YORK, US, vol. 25, no. 4, 1 October 1995 (1995-10-01), pages 299-313, XP000541665 ISSN: 0146-4833	1,3,4,10
Y	column 1, line 1 -column 12, line 46	2,5,9, 12-15,25
Y	WO 00 72537 A (EEROLA SEVERI ;NOKIA CORP (FI)) 30 November 2000 (2000-11-30) abstract; figure 6 page 8, line 10 - line 30	2,5,9, 12-15,25

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 March 2002	08/04/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Pereira, M

3

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	PADMANABHAN V N ET AL: "Improving HTTP latency" COMPUTER NETWORKS AND ISDN SYSTEMS, NORTH HOLLAND PUBLISHING. AMSTERDAM, NL, vol. 28, no. 1, 1 December 1995 (1995-12-01), pages 25-35, XP004001208 ISSN: 0169-7552 the whole document	1-27
A	PADMANABHAN V N: "IMPROVING WORLD WIDE WEB LATENCY" UNIVERSITY OF CALIFORNIA REPORT, XX, XX, 1 May 1995 (1995-05-01), pages 1-24, XP002041557 the whole document	1-27
A	US 6 038 601 A (VAN DER RIJN DANIEL J G ET AL) 14 March 2000 (2000-03-14) the whole document	1-27

INTERNATIONAL SEARCH REPORT

ormation on patent family members

Int[redacted]tional Application No
PCT/IL 01/00793

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 0072537 A	30-11-2000	FI 991180 A	25-11-2000
		AU 4759500 A	12-12-2000
		EP 1183837 A1	06-03-2002
		WO 0072537 A1	30-11-2000
US 6038601 A	14-03-2000	AU 8578898 A	10-02-1999
		EP 0996893 A1	03-05-2000
		WO 9904345 A1	28-01-1999