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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 May 2001 (31.05.2001)

PCT

(10) International Publication Number
WO 01/39042 A2

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US00/16978

(22) International Filing Date: 20 June 2000 (20.06.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/449,065 24 November 1999 (24.11.1999) US

(71) Applicant: ELEGENT TECHNOLOGIES, INC. [US/US]; 440 Mission Court, Suite 250, Fremont, CA 94539 (US).

(72) Inventors: CHANG, Rong-Wen; 440 Mission Court, Suite 250, Fremont, CA 94539 (US). LEE, John, K.; 440 Mission Court, Suite 250, Fremont, CA 94539 (US). LIN, Ron, K.; 440 Mission Court, Suite 250, Fremont, CA 94539 (US).

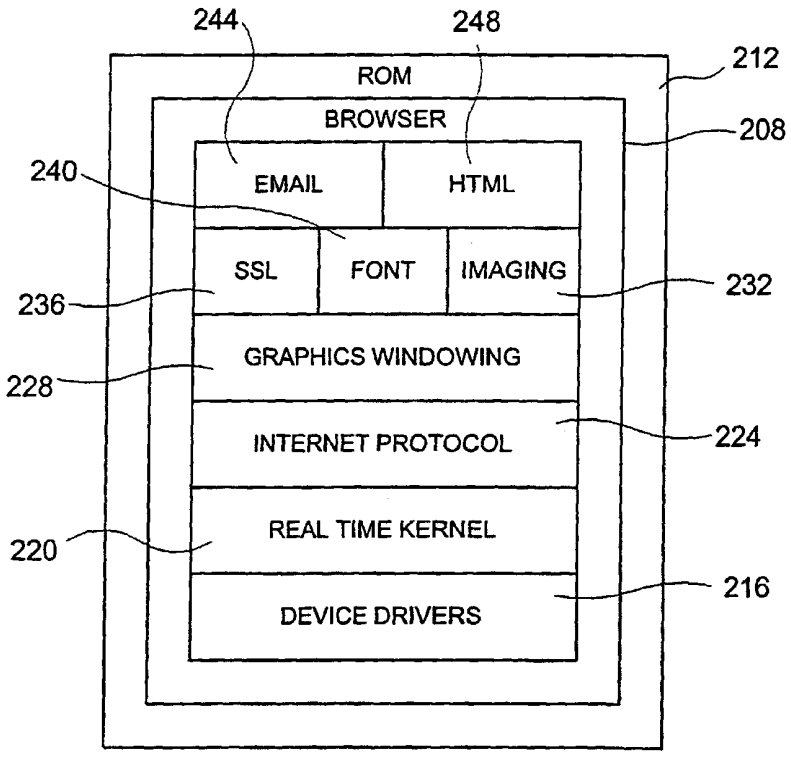
(74) Agent: WOLFF, Jason, W.; Lyon & Lyon LLP, 633 West Fifth Street, Suite 4700, Los Angeles, CA 90071-2066 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, 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, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, 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,

[Continued on next page]

(54) Title: SELF-CONTAINED NETWORK BROWSER WITH DIAGNOSTIC ABILITIES



(57) Abstract: A network browser with diagnostic abilities stored in a persistent memory, wherein the persistent memory is not a hard disk, is provided. The network browser is used to repair failures of peripheral devices in a networked computer, such as a hard drive, so as to avoid forcing a user to manually diagnose or solve the failure. Moreover, the network browser removes strict dependence on a traditional operating system, and thus the hard disk, to make such repairs. According an embodiment, the network browser comprises a plurality of software modules. The modules include: a device driver module (216), which is configured to initialize and test one or more peripheral devices; a real time kernel module (220), which is configured to detect and dispatch data to and from peripheral devices, though said device driver module (216), including processing diagnostic data corresponding to operation of said peripheral devices, and to perform memory management tasks; an internet protocol module (224), which is configured to handle network communications with remote devices; a graphics windowing module (232), which is configured to process visual display data and control; and a hypertext markup language module (248) configured to interpret hypertext markup language documents for display with said graphics windowing module (232).

WO 01/39042 A2

IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *Without international search report and to be republished upon receipt of that 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.

S P E C I F I C A T I O N

TITLE OF INVENTION

SELF-CONTAINED NETWORK BROWSER WITH DIAGNOSTIC ABILITIES

BACKGROUND

1. Field of the Invention.

This invention relates generally to network browser software and more particularly to a self-contained network browser with diagnostic abilities.

2. Background Information.

A web browser (or "browser software") is software that is executed by a personal computer in order to send and receive data from a network. Usually, the network is a wide area network, such as the Internet, or it can be a local area network, such as an intranet. Once the browser software has connected to the network, the browser software can request information, such as hypertext markup language ("HTML") files from another computer or "server" on the network.

An aspect of most browser software, such as the Netscape Navigator (TM) available from Netscape Corporation in Mountain View, California, is that it is an application program. FIG. 1 shows a known software stack 100 for a personal computer. The software that forms the software stack is persistently stored in a read-only memory ("ROM") and a hard drive of the personal computer. In fact, the basic input output system (hereinafter "BIOS") 104, which consists of limited purpose executable firmware code permanently attached to a processor, is stored in the ROM.

The BIOS 104 controls low-level input and output operations when so directed by the operating system.

The operating system 108 is one or more software products (e.g., device drivers) that jointly manage the system resources (e.g., memory management and peripheral devices) of a personal computer, as well as any other programs (e.g., applications) that use the system resources. It is noted that the device drivers can be part of the operating system 108, or they can be added over the top of the operating system 108. For example, some device drivers can be downloaded into memory as needed.

One or more applications 116 are stacked on top of the operating system 108. The applications 116 communicate with the operating system 108 through an application programming interface (hereinafter "API") 112, which contains functions and procedures that are called by the applications 116. The API 112 functions and procedures that are called by the applications 116 are in turn passed to the operating system 108. The operating system 108 then passes any required input/output processes on to the BIOS 104, or processes them directly using a device driver.

The operating system 108, the API 112, and the applications 116, unlike the BIOS 104, are all stored in the hard drive of the personal computer. This is largely due to the size of these software components, but also due to their extensible nature.

The primary reason that most browser software is written as an application is that it will be more "portable", meaning it can be more easily modified to run on different operating systems (e.g., UNIX, Windows 98 (TM), etc.) as well as on different types of computers running the same operating system. This expands a software vendor's market for their product.

In light of recent legal woes, Microsoft Corporation has made much of the fact that their browser (Internet Explorer) is part of their operating system, rather than an application added to the operating system. It is not known whether Microsoft's browser is part of the operating system, or an application added to the operating system. Regardless of whether Microsoft's (or any known) browser is part of the operating system, or that it is an application program, the fact remains that the browser, just like the operating system, is still stored in the hard disk. When the hard disk fails, the browser and the personal computer are usually useless.

SUMMARY OF THE INVENTION

A network browser with diagnostic abilities stored in a persistent memory, wherein the persistent memory is not a hard disk, is provided. The network browser is used to repair failures of peripheral devices in a networked computer, such as a hard drive, so as to avoid forcing a user to manually diagnose or solve the failure. Moreover, the network browser removes strict dependence on a traditional operating system, and thus the hard disk, to make such repairs. According an embodiment, the network browser comprises a plurality of software modules. The modules include: a device driver module (216), which is configured to initialize and test one or more peripheral devices; a real time kernel module (220), which is configured to detect and dispatch data to and from peripheral devices, though said device driver module (216), including processing diagnostic data corresponding to operation of said peripheral devices, and to perform memory management tasks; an internet protocol module (224), which is configured to handle network communications with remote devices; a graphics windowing module (232), which is configured to process visual display data and control; and a hypertext markup language module (248) configured to interpret hypertext markup language documents for display with said graphics windowing module (232). Hardware configurations and software methods for the self-contained browser are disclosed herein.

According to an embodiment of the invention, processing diagnostic data includes initializing peripheral equipment, detecting a failure, contacting a remote server, loading diagnostic interface files, and diagnosing the failure interactively with the remote server using the interface files.

BRIEF DESCRIPTION OF THE DRAWINGS

The figures of the accompanying drawings are shown by way of example and not by way of limitation, in which like reference numerals refer to like components and in which:

FIG. 1 depicts a known software stack.

FIG. 2 is a block diagram of a self-contained ROM-based browser software stack according to the present invention.

FIG. 3 is a high-level block diagram of the present invention.

FIG. 4 is a detailed block diagram of the present invention.

FIG. 5A is a flowchart depicting the invocation of the present invention.

FIGS. 5B and 5C are flowchart depicting useful applications of the present invention.

FIG. 6 is a block diagram of a personal computer.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG. 2 depicts a block diagram of a self-contained, persistently stored browser software stack 208. The browser software stack 208 is persistently stored, but not on a traditional electromagnetic hard drive. As used herein, an "electromagnetic hard drive" is a magnetic storage medium, usually called a platter, that is mechanically rotated. Read and write operations are performed by a read-write head that is part of the hard drive.

Preferably, the browser 208 is stored in PC BIOS in a read-only memory ("ROM"), 212 or a ROM equivalent, such as: electrically programmable read-only memory ("EPROM"), electrically erasable programmable read-only memory ("EEPROM"), electrically alterable programmable read-only memory ("EAPROM"), and flash erasable programmable read-only memory ("FLASH" or "FEPROM"). As used herein, PC BIOS stands for "personal computer basic input output system." The PC BIOS is a firmware code region of memory that is permanently resident in the personal computer. It is primarily responsible for performing low level input output operations, usually on behalf of an operating system. However, according to an embodiment of the present invention, the PC BIOS includes the self-contained browser software stack 208.

The browser 208 is independent of a known operating system and a hard drive. Thus, employing the browser 208 makes a personal computer less susceptible to failures and allows the personal computer to operate in spite of certain types of failures -- whether they are failures that occur at startup, or during run-time operation of the personal computer. Moreover, the browser 208 can be used to diagnose problems associated with hardware components coupled to the personal computer.

The browser 208 depicted in FIG. 2 architecturally defines an embodiment of the invention. According to a presently preferred embodiment, the browser 208 is stored in a ROM 212. Before describing the functional blocks comprising the browser 208 in detail, it is useful to describe the modular elements (e.g., program code and hardware combination) of the browser as they relate to the overall invention. For this we turn to FIG. 3.

OPERATIONAL OVERVIEW

FIG. 3 is a high-level block diagram of the browser 300 as it operates with electronic hardware, such as a personal computer (one embodiment of a personal computer is described below with reference to FIG. 6). According to an embodiment of the present invention, a personal computer (hereinafter "computer 300") having the browser software performs at least four basic operations. One operation includes an input/output function. To this end, the computer 300 includes an input module 304. The input module 304 comprises hardware that provides one or more communication means with an external device, such as a keyboard, a mouse, a stylus, or a touch screen. A serial port, an I/O controller, a USB port, an IEEE 1394 port, and a CEBus are examples of communications means. Software, which is included in the browser, handles interrupts, buffering, and command and control dispatching for inputs received over the communication means. It is noted that output can also be achieved by the communication means of the input module 304, although it is not the primary function of the module 304.

Another operation of the computer 300 is that of lower level network communications. For this, a network communications module 308 is provided that handles communication functions and operations at the physical, data

link, network and transport layers of, for example, the seven layer OSI ("Open Systems Interconnection") Reference Model, which is generally known in the art of networking.

Hardware devices that can implement the network communications module 308 include an Ethernet card, a traditional landline modem, a cable modem, and a wireless modem. Multiple Internet RFCs, which are standards for the Internet, define the specifications of the software that allows these hardware devices to operate. According to embodiments of the invention, the Internet RFC's include 791 (Internet Protocol or "IP"), 792 (Internet Control Message Protocol or "ICMP"), 793 (Transmission Control Protocol or "TCP"), 826 (Address Resolution Protocol or "ARP"), and 1661 (Point-to-Point Protocol or "PPP"), all of which are available on the Internet at the URL http://www.pmg.lcs.mit.edu/rfc.html.

A third operation of the computer 300 is processing data and control to and from the network communications module 308. At the network application module 312, various protocols are employed that interpret messages from the network communications module 308 and either provide a client (or "user interface") specifically for them (for example, terminal monitor software or an electronic mail client), or access another client at a different module (for example in the page rendering module 316, which is described below). If the network application module 312 provides a client for a user, then in addition to handling data to and from the network communications module 308, it can receive data from the input module 304 -- either directly or through another module, such as the page rendering module 316.

Exemplary applications (or "protocols") for the network application module 312 are DNS (Domain Name System), HTTP (Hypertext Transfer Protocol), SSL (Secure Sockets Layer),

HTTPS (HTTP Secure), SMTP (Simple Mail Transfer Protocol), POP3 (Post Office Protocol Version 3), TELNET, and FTP (File Transfer Protocol). These network applications are further described in Internet RFCs: 1034 (DNS), 2068 (HTTP), 821 (SMTP), 1081 (POP3), 854 (TELNET), and 959 (FTP). It is presently preferred that only SMTP, POP3, DNS, HTTP, SSL and HTTPS are simultaneously implemented in the network application module 312. Generally, the less protocols that are included with the browser, the smaller the resulting footprint (i.e., the amount of memory required to store the self-contained network browser is reduced).

At the top of the diagram is the page rendering module 316. The page rendering module 316 handles interpretive aspects of transforming hypertext markup language ("HTML") documents, as well as other document formats. For example, the page rendering module 316 renders graphics files, performs page layout functions, and handles window positioning, sizing, and scrolling in response to data from the input module 304 (for example, data from a mouse). The page rendering module 316 communicates directly with a display module (not shown).

DETAILED MODULE DESCRIPTION

FIG. 4 is a detailed functional block diagram of the software in the self-contained network browser. FIG. 4 is organized in accordance with a typical data flow from a network (for example, over a peripheral device) to its ultimate presentation on a computer display or terminal. Although the functionality of the various detailed modules (FIG. 4) can be moved between the general purpose modules (FIG. 3), they are hereinafter described with reference to a particular general purpose module. For example, the input module 304 (FIG. 3) includes software modules 408 through

412. In an embodiment, the network communications module 308 includes software modules 416 through 432, the network application module 312 includes software modules 436 through 464, and the page rendering module 316 includes software modules 468 through 482.

Data input is received at a peripheral device 404 (i.e., an internal or external device that is added to the computer 300, such as an Ethernet adapter, a keyboard, a mouse, or a wire-line/wireless modem). When the data is received by the input module 304, it is routed from one or more of the computer's input/output ("I/O") ports to its respective module for processing. Generally speaking, each peripheral device is attached to a particular I/O port and data communicated from the peripheral device causes an interrupt in the real-time kernel, which is further described below with reference to FIG. 2. The interrupt identifies an exception vector or memory address of an algorithm that handles the input of data from the peripheral device. For example, a signal from a modem can trigger a first serial port interrupt, which in turn causes the modem point-to-point protocol ("PPP") module 416 to be activated.

Before describing the individual components of the detailed block diagram of FIG. 4, it is noted that the debug console module 412 is not required. The debug console module 412 is used as a debug port for software and hardware testing purposes. Moreover, elements shown with solid lines are part of the browser software, whereas elements shown with dashed lines are part of the overall personal computer while the browser software is executing.

Additionally, the physical page cache 490 and the display 494 are not part of the browser software stack. The page cache 490, preferably formed from a volatile memory, acts as a buffer for data processed by one or more network

applications before the data is passed on to the page rendering module 472. For example, the page cache 490 can pre-cache HTML files (both not-yet-interpreted and interpreted files) before they are passed on to the display 494. The display 494 can be a peripheral device that the page rendering module 472 communicates with or it can be a volatile memory buffer for the actual display device that presents data to a user. The display 494 receives data from the page rendering module 472.

Furthermore, the ROM files 468 are not required (although they are preferred). The ROM files 468 can persistently store one or more user diagnostic interfaces or templates used by a particular remote vendor (e.g., a computer manufacturer or an internet service provider) to diagnose a problem with the personal computer. For example, an HTML form or an executable diagnostic program that is used when the browser connects with a remote vendor can be stored in ROM files 468. Although they are called "ROM files", the files can be stored in an electrically alterable ROM, such as a FLASH memory. This is useful when cookies or other files may be stored in the ROM files 468, since cookies are not necessarily purely static files, but rather, they may change from time to time.

Cursor control module 408 is used to process incoming data from a mouse or stylus. The data is generally two dimensional movement data corresponding to movement of the mouse. The data also includes control signals, such as a mouse selector click. The cursor control module 408 preferably receives the data via a serial port, but it can also receive data from a wireless port, such as an infrared port.

Keyboard module 410 is the keyboard driver. It is used to enter or type text into forms and/or to enter commands

directed to the page rendering module 472, as well as other modules interfacing the page rendering module 472 -- such as modules 452 through 464. An important aspect of the keyboard module 472 is its ability to launch the browser. Typically, a user enters a special key or sequence of keys (e.g., CTRL-ALT-B or CTRL-ALT-HOME) that manually launch the self-contained browser, thereby bypassing the normal operating mode for the personal computer. Typically, the special key sequence is active only when the personal computer is booting up. The keyboard module 410 is also used to control page scrolling and field selection, for example through use of direction and/or tab keys on a keyboard.

The debug console 412 is used for debugging purposes and is included primarily for software tuning. For example, since the device drivers can be different as between chipsets and hardware devices on different computers, the debug console 412 is useful in monitoring the browser performance as it operates with one or more new hardware devices. The debug console 412 is not a necessary component of the browser software.

The modem PPP module 416 is used to process incoming packet data from a public switched telephony network ("PSTN"). The functionality of the module 416 preferably complies with Internet RFC 1616, as it is configured to assist a modem in connecting to an internet service provider ("ISP") or internet access provider ("IAP"), and then making virtual connections with one or more remote servers. The modem PPP module 416 also includes the device drivers for the modem.

The Ethernet ARP module 420 is also used to process incoming packet data from a network. Here, however, the network can be a high-speed local area network ("LAN"), or a

high-speed wide area network ("WAN"). The module 420 includes one or more device drivers for the Ethernet card (a peripheral device). The Ethernet ARP module 420 complies with Internet RFC 826, as the module is configured to translate between IP and Ethernet addresses.

The IP/ICMP module 424 receives packets from modules 416 or 420, depending on the network connection. The IP portion of the IP/ICMP module 424 handles fragmentation, packet routing and re-assembly of IP packets. ICMP is an extension to IP that handles generation of error messages and other information associated with IP. The functionality of the IP/ICMP module 424 is consistent with Internet RFCs 791 and 792.

TCP/UDP module 428 assembles IP packets into messages for the network applications. The TCP/UDP module 428 operates in accordance with Internet RFC 793 (TCP) and Internet RFC 768 (User Datagram Protocol or "UDP"). As the TCP/UDP module 428 is compliant with both TCP and UDP, it can support both connection-oriented communications (TCP) and connectionless communications (UDP).

The socket application programming interface ("API") module 432 provides an interface between the network communications module 308 and the network application module 312 of FIG. 3. As a network application is executed on the computer 300, the network application calls functions and sends and receives data/messages to and from the network communications module 308. To this end, the socket API 432 is used as a dispatcher to create and destroy IP virtual connections, or "sockets", with remote servers. Because both TCP and UDP are supported by the browser software, the sockets can be connection-oriented or connectionless.

As is mentioned above, modules 436 through 464 can be classified as network application modules. The network

application modules are communicatively coupled to both the socket API 432 and the page rendering module 472.

DNS module 460 is used to map (or "resolve") English-type URLs (e.g., "http://www.elegent.com", where "http://" identifies the network application protocol and "www.elegent.com" identifies the URL) to IP addresses (e.g., "206.171.12.20"). (As was mentioned above, Ethernet ARP module 420 maps IP addresses to Ethernet addresses.) DNS is described in Internet RFC 1034. DNS module 460 passes domain name requests ("queries") to a resolver or name server for processing in accordance with RFC 1034.

HTTP module 456 is used for the transfer of HTML ("hypertext markup language") files to or from a remote server. Most of the HTML files are ultimately presented to a user via the page rendering module 472. HTTP module 456 is compliant with Internet RFC 2068.

SMPT module 436, POP3 module 440 and e-mail module 464 are all involved with processing electronic mail messages. SMTP module 436 includes software that handles sending electronic mail messages in accordance with Internet RFC 821, which has been incorporated herein by reference in its entirety. The SMTP module 436 is communicatively coupled to both the socket API 432 and the electronic mail client (or "e-mail module") 464 (which is a user interface for the SMTP and POP3 network applications). POP3 module 440 includes software that handles retrieving electronic mail messages from a remote server. Like the SMTP module 436, the POP3 module 440 is communicatively coupled to both the socket API 432 and the electronic mail client 464. E-mail module 464 is preferably an HTML-based software interface that is interpreted by the page rendering module 472.

Whereas SMTP and POP3 are two presently preferred protocols used in the present invention, other electronic

mail protocols can also be employed. For example, a protocol wherein the electronic mail messages are manipulated on a remote mail server rather than on the computer 300. For instance, modules 436 and 440 could be replaced by Internet Message Access Protocol ("IMAP") compliant software. A recent version of IMAP is described in Internet RFC 2060.

FTP module 444 complies with Internet RFC 959. The FTP module 444 handles file transfers between the computer on which the browser resides and a remote server. The FTP module 444 is not necessary or required for a successful implementation of the browser of the present invention.

SSL module 448 and HTTPS module 452 manage complementary security protocols employed by the browser. These modules are activated when secure exchanges are desired between the self-contained network browser and the remote server. When these modules are activated, dedicated ports (e.g., port 443) between the browser and the remote vendor/server are used to pass IP packets. The browser and the server establish session identifiers and share one or more encryption keys. The session identifiers and encryption keys are used to verify the authenticity of the exchanged information, as well as to protect the information exchanged from snoopers (unauthorized persons who try to eavesdrop on a communication). Both the SSL module 448 and the HTTPS module 452 are not required for successful implementation of the present invention.

Now that the network application modules have been described, the page rendering module 472 is described. The page rendering module 472 engages in two-way communications with the network application modules 312. Exchanges between the page rendering module 472 and the network application modules 312 are generally buffered by the page cache 490.

The page rendering module 472 also communicates screen information to the display 494, which usually includes a graphics accelerator adapter for a bitmap display. The page rendering module 472 also receives data from one or more input devices (such as a mouse or keyboard).

The page rendering module 472 includes five basic elements. One element is as a graphics rendering element 474. The graphics rendering element 474 interprets GIF, JPEG, and MPEG type files for presentation on a bitmap display. The graphics rendering element 474 also handles image scaling. Another page rendering module 472 element is a font engine 476. The font engine 476 supports the various fonts used by HTML files. The multi-lingual element 478 supports the interpretation of characters in HTML files into one of a number of different languages, although only one language needs to be supported by the browser. The page layout element 480 interprets the formatting tags in HTML files, for example the <bold>, , and <color> tags, as well as the <table> and <frame> tags.

The windowing system 482 of the page rendering module 472 provides a window environment for each page or subframe of the HTML files presented to a user. The windowing system 482 handles window sizing for the display 494, as well as the generation of horizontal and vertical scroll bars that a peripheral device can control (via cursor control module 408). Movement of the cursor or stylus detected at the cursor control module 408 is communicated directly to the page rendering module 472, and in particular the windowing system 482, for processing.

BROWSER SOFTWARE ARCHITECTURE

Returning to FIG. 2, it depicts a functional block diagram of the self-contained browser software stack 208 as

depicted and described in detail with reference to FIG. 4. The diagram is useful in understanding the overall architecture of the self-contained network browser, which is preferably contained in a read-only memory 212.

The real-time kernel 220 (which is not shown as a single module in FIG. 4) performs at least two basic tasks. The basic tasks of the real-time kernel 220 include (1) detecting and dispatching data to and from peripheral devices to their appropriate modules and (2) memory management. As for data processing and dispatching, the real-time kernel 220 actively polls peripheral devices 404 in an attempt to detect a change of state in the devices (for example, if peripheral devices share a common bus), or the kernel 220 passively receives direct interrupt requests from the peripheral devices 404. The real-time kernel 220 differs from a traditional operating system in that it handles low level tasks that the traditional operating system may perform, but does not handle higher level tasks. Rather, the higher level tasks are reserved for unique modules in the browser 208.

The memory management aspects of the real-time kernel 220 include management of the page cache 490, as well as other volatile execution memory while processes are running. For example, the real-time kernel 220 performs functions such as memory allocation and garbage collection for global and/or local memory areas utilized by each of the modules depicted in FIG. 4.

According to one embodiment of the invention, the real-time kernel 220 manages the page cache 490 (FIG. 4) as three logical memory regions.

A request queue region 491 stores requests detected by the cursor control 408 and passed to the page cache 490 by the page rendering module 472. The request queue region 491

also stores requests generated by the page rendering module 472.

A pre-processed data region 492 stores data that has not been interpreted by the page rendering module 472. The pre-processed region 492 is particularly useful when one or more network applications pre-fetch batches of HTML files, or request multiple HTML files, prior to actually receiving an explicit request from a user.

A post-processed data region 493 holds data that has been interpreted by the page rendering module 472 but has not yet been sent to the display 494. Data stored in the post-processed data region 493 passes from a network application to the page rendering module 472, and then from the page rendering module 472 to the page cache 490. From the page cache 490 it can again pass through the page rendering module 472 on to the display 494.

The real-time kernel 220 handles input and output between the hardware resources (for example internal and external devices/components) and the software processing modules of the computer 300. Accordingly, the real-time kernel 220 communicates with one or more device drivers 216 associated with individual hardware components to ensure proper communication processing as well as proper hardware component initialization.

According to one embodiment, the real-time kernel 220 is communicatively coupled with an initialization module (not shown). The initialization module detects each peripheral device connected to the computer 300, formats the peripheral device with any initialization parameters needed, and, if the initialization is not successful, then the initialization module triggers a diagnostic processing mode for the browser. Thus, the browser 208 can be invoked by either the initialization module enabling the diagnostic

processing mode, or by a direct call from a traditional operating system or an end-user. In an embodiment, the initialization module is shared with both the browser 208 and a traditional operating system.

The browser 208 includes device driver software 216 that communicates with and/or controls the peripheral devices of the computer. The device driver software 216 can include initialization and testing software that ensures a particular peripheral device is operational (some of the testing software can be separate from the browser 208). The real-time kernel 220 operates over the device driver software 216, and manages the computer system resources and uses the device drivers 216 to communicate with the peripheral devices. Internet protocol software 224 is stacked over the real-time kernel 220, and is used to handle a variety of network communications, such as those described above with reference to FIG. 4.

On top of the internet protocol software 224 is a graphics windowing software 228. The graphics windowing software 228 handles behind-the-scene processing of data that is presented to a user on a display device. For example, window positioning, cursor and keyboard input/control, and graphics processing is handled, in part, by the graphics windowing software 228.

The software components of FIG. 2 that operate above the graphics windowing software 228 are more closely related to end-user applications. For example, SSL software 236 handles security information processing. The font software supports various fonts that are displayed to the end-user, and the imaging software 232 interprets graphics files that are also displayed to the end-user.

The e-mail software 244 is an end-user client that allows the end-user to send and receive electronic messages.

HTML software 248 is an end-user application that interprets HTML files, for example by parsing the files and passing the parsed sections on to the appropriate module for further processing.

BROWSER INVOCATION AND DIAGNOSTIC ABILITIES

FIG. 5A is a flowchart depicting steps for invoking the browser on a personal computer according to an embodiment of the invention. In step 504, the computer system hardware components, internal and external, are initialized. In step 508, a test is performed to determine whether a browser command or trigger has been set. According to one embodiment, the browser command is invoked in response to a key or sequence of keys typed on a keyboard while the personal computer is booting up. In another embodiment, a physical or CMOS setup switch is set that triggers the self-contained network browser.

If no browser trigger was detected in step 508, then the personal computer continues to step 512 where the computer enters a normal operating mode. For example, the traditional operating system boots and general purpose applications are be executed. Similar to step 508, in step 516 a test is performed to determine whether the browser trigger is activated. It is noted that this process can be handled by an exception vector or interrupt routine, or it can be handled by a particular device driver or the traditional operating system. If a browser trigger was detected in step 516, then in step 520 the operating system shuts down and the process continues to step 524.

In step 524, which follows steps 508 or 520, the self-contained browser is invoked. FIGS. 5B and 5C depict particularly useful applications for the self-contained

browser once it has been invoked. The steps shown in FIGS. 5B and 5C replace connector "A" (element 528).

In FIG. 5B, a flowchart is shown depicting use of the browser to diagnose and repair a failed hardware component. In step 532, the browser connects to a network or "remote" server corresponding to a particular vendor. The vendor can be the company that sold the personal computer, or it can be another service provider that handles technical service/repair of the personal computer. In step 536, diagnostic files, for example files stored in ROM files 468, as well as files downloaded from the remote server, are loaded. These diagnostic files allow an end-user to report a problem type, to perform system configuration detection, or to diagnose selected hardware components. The diagnostic files can include interpreted software code, executable software code, or HTML interfaces such as forms. The vendor is thus able to remotely diagnose the problem the personal computer is experiencing.

Before diagnosing the problem, the vendor can send a command that tells the browser launch a particular diagnostic program, or the vendor can send a particular compiled or interpreted diagnostic program to the browser. If a diagnostic program is sent to the browser, then it is sent using a FTP, an HTTP, an HTTPS, or a SMTP protocol. If a public network is used, for example the Internet (versus an intranet), a certificate is used to authenticate the diagnostic program. For example, RSA Data Security, Inc. or X.509 compliant certificates are employed to verify the authenticity of the diagnostic program.

In step 540, the diagnostic program, whether it was stored locally or was sent from a remote vendor, is executed. For example, the diagnostic program can confirm that the hard disk did fail, or it can examine the contents

of an error log corresponding to the operating system and other hardware components. Part of executing the diagnostic program can also include uploading the results to the vendor so that the vendor can take corrective action.

A test is performed in step 544 to determine whether the problem can be repaired. If the problem can be repaired, then it is repaired in step 548, which may also include downloading an additional program from the vendor -- for example, as was described above with reference to step 540. However, if the problem cannot be repaired, then in step 552 a notification is sent to either the user or the vendor (or both) indicating that a repair cannot be made. If the vendor is notified in step 552, then the vendor can follow up with the end-user of the personal computer.

In FIG. 5C, the browser is used to explore or "surf" the Internet as is shown in step 556. For example, a user of the personal computer can use the self-contained network browser to connect to remote servers and send data and retrieve HTML files.

Since aspects of the traditional operating system and the browser overlap, that is each is separately (but not necessarily concurrently) used in one form or another to manage the computer system resources, each can be executed independently of the other. Thus, failures of the traditional operating system or of a peripheral device, in particular the hard disk, will not necessarily affect the browser.

HARDWARE OVERVIEW

FIG. 6 is a block diagram that illustrates an embodiment of a computer system 600 upon which the invention can be implemented.

Computer system 600 includes a bus 602, or other communication mechanism for communicating information, and a processor 604 coupled with bus 602 for processing information. Computer system 600 also includes a main memory 606, such as a random access memory ("RAM"), or other dynamic (or "volatile") storage device, coupled to bus 602. The main memory 606 stores information and instructions executed by processor 604 during execution. Main memory 606 also stores temporary variables or other intermediate information during execution of instructions by processor 606.

Computer system 600 further includes a read only memory ("ROM") 608 or other static (or "persistent") storage device (e.g., FLASH, PROM, EEPROM, etc.) coupled to bus 602. The ROM 608 stores static information and instructions for processor 604, in particular the browser as described herein. It is worth noting that one or more banks of memory can comprise ROM 608. A storage device 610 (or "hard disk", or "hard drive"), such as a magnetic disk or optical disk, is coupled to bus 602. The storage device 610 stores information such as data structures and instructions, for example the operating system or application programs that use the operating system.

Computer system 600 is preferably coupled via bus 602 to a display 612, such as a cathode ray tube ("CRT") or an active or passive-matrix display. The display 612 presents images to an end-user. An input device 614, including alphanumeric and other keys, is coupled to bus 602. The input device 614 communicates information and command selections to processor 604. Another type of user input

device is cursor control 616, such as a mouse, a trackball, or cursor direction keys, for communicating direction information and command selections to processor 604 and for controlling cursor movement on display 612. This input device 614 typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

The invention is a persistently stored, self-contained browser, where the browser is not stored in the hard disk, such as storage device 610. One application for the invention is for diagnosis and repair of the computer system 600. Another is as a diskless internet device. According to an aspect of the invention, the processor 604 in the computer system 600 executes one or more sequences of instructions contained in main memory 606. Such instructions are read into main memory 606 from another computer-readable medium, such as storage device 610 or ROM 608. Execution of the sequences of instructions contained in main memory 606 causes processor 604 to execute the browser and other processes described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor 604 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 610. Volatile media includes dynamic memory, such as main memory 606. Transmission media

includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 602. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infrared data communications.

Common forms of computer-readable media include, a floppy disk, a flexible disk, a hard disk, a magnetic tape, or any other magnetic media, a CD-ROM, any other optical media, punchcards, a paper-tape, any other physical media with patterns of holes, a RAM, a ROM, a FLASH, or any other memory chip or cartridge, a carrier wave as described hereinafter, or any other media from which a computer can read.

Various forms of computer-readable media may be involved in carrying one or more sequences of one or more instructions to processor 604 for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 600 can receive the data on the telephone line and use an infrared transmitter to convert the data to an infrared signal. An infrared detector coupled to bus 602 can receive the data carried in the infrared signal and place the data on bus 602. Bus 602 carries the data to main memory 606, from which processor 604 retrieves and executes the instructions. The instructions received by main memory 606 may optionally be stored on storage device 610 before or after execution by processor 604.

Computer system 600 also includes a communication interface 618 coupled to bus 602. Communication interface 618 provides a two-way data communication coupling to a

network link 620 that is connected to a local network 622. For example, communication interface 618 may be an integrated services digital network ("ISDN") card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 618 may be a local area network ("LAN") card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 618 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 620 preferable provides data communication through one or more networks to other data devices. For example, network link 620 may provide a connection through local network 622 to a host computer 624 or to data equipment operated by an Internet Service Provider ("ISP") 626. ISP 626 in turn provides data communication services through the "Internet" 628 -- for example computer diagnostic services. Local network 622 and Internet 628 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 620 and through communication interface 618, which carry the digital data to and from computer system 600, are exemplary forms of carrier waves transporting the information.

Computer system 600 can send messages and receive data, including program code, through the network(s), network link 620 and communication interface 618. In the Internet example, a server 630 might transmit requested code for an application program through Internet 628, ISP 626, local network 622 and communication interface 618 -- for example using the FTP protocol. In accordance with the invention, one such downloaded application is executable software code

or computer configuration parameters that either further diagnose the computer's problem, or fix the problem outright.

The received code may be executed by processor 604 as it is received, and/or stored in main memory 606, storage device 610, or other non-volatile storage for later execution. In this manner, computer system 600 may obtain application code in the form of a carrier wave.

Referring to FIGS. 3 and 6, it is notable that the input module 304 interacts with input device 614 and cursor control 616. Network communications module 308 and network application module 312 interact with communication interface 618. And page rendering module 316 interacts with display 612.

In one embodiment, all of the self-contained network browser software code is stored in one or more banks of ROM 608. When executed, however, the browser software code is copied to main memory 606. In one embodiment, the page cache 490 is also be a portion of main memory 606.

Advantages of the present invention include a small footprint, self-contained browser architecture that is independent of, yet complimenting, a traditional operation system. A computer incorporating the browser can function without a hard disk. Thus, a user can still operate the computer when the hard disk fails, or the user can operate just the browser software so she has fast and ready access to the Internet without having to wait for the operating system to boot, or other application software to load.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will be evident, however, that various modifications and changes may be made thereto while still remaining consistent with the description above. For example, more or less regions can be specified for the page cache, more or less

functionality can be included in both the network communication and application modules, as well as in the page rendering module. Further still, other embodiments may include a virtual machine for interpreting or executing Java code, or other portable program code. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A self-contained network browser (208) with hardware diagnostic abilities, said network browser (208) comprising a computer readable memory (212) configured to persistently store executable software modules, said software modules including:

a device driver module (216) configured to initialize and test one or more peripheral devices;

a real time kernel module (220) configured to detect and dispatch data to and from peripheral devices, though said device driver module (216), including processing diagnostic data corresponding to operation of said peripheral devices, and to perform memory management tasks;

an internet protocol module (224) configured to handle network communications with remote devices;

a graphics windowing module (232) configured to process visual display data and control; and

a hypertext markup language module (248) configured to interpret hypertext markup language documents for display with said graphics windowing module (232).

2. The self-contained network browser (208) of claim 1, wherein when said diagnostic data corresponding to operation of said peripheral devices indicates a failure has occurred, then said executable software modules are configured to call functions to:

connect to a remote server using said internet protocol module (224);

load one or more diagnostic interface files, including diagnostic interface files stored persistently in said memory (212);

diagnose said failure interactively with said remote server and said loaded one or more interface files; and

repair said failure.

3. The self-contained network browser (208) of claim 2, wherein said executable software modules are further configured to call functions to repair said failure by downloading executable repair code from said remote server using a network protocol with security measures that verify the authenticity of said executable repair code.

4. The self-contained network browser (208) of claim 2, wherein said executable software modules are further configured to call functions that receive a command from said remote server, said command configured to call one or more functions stored in said memory (212) that assist in repairing said failure.

5. The self-contained network browser (208) of claim 1, wherein said memory management in said real time kernel module (220) includes management of a network browser page cache (490) as a request queue region (491), for storing data and commands that have not been serviced, a pre-processed data region (492), for storing data that must be interpreted for output to a display device, including pre-fetched batches of hypertext markup language files, and a post-processed region (493), for storing data that has been interpreted for output to said display device, but has not yet been sent to said display device.

6. A computer for accessing a network, the computer including a processor (604), a hard drive (610), a persistent memory (608), a random access memory (606), and a

communication interface (618), all communicatively coupled to said processor (604) through a bus (602), wherein said persistent memory (608) comprises executable software modules stored in a memory region (212), the executable software modules including:

a device driver module (216) configured to initialize and test said hard drive (610) for a failure;

a real time kernel module (220) configured to detect and dispatch data to and from peripheral devices, though said device driver module (216), including processing diagnostic data corresponding to operation of said hard drive (610), and to perform memory management tasks pertaining to said main memory (606);

an internet protocol module (224) configured to handle network communications with remote devices;

a graphics windowing module (232) configured to process visual display data and control; and

a hypertext markup language module (248) configured to interpret hypertext markup language documents for display with said graphics windowing module (232).

7. The computer of claim 6, wherein said diagnostic data corresponding to operation of said hard drive (610) indicates a failure has occurred and said executable software modules are further configured to cause said processor (604) to:

connect to a remote server (630) using said internet protocol module (224);

load one or more diagnostic interface files, including diagnostic interface files stored persistently in said memory region (212);

diagnose said failure interactively with said remote server (630) using said loaded one or more interface files; and

repair said failure.

8. The computer of claim 7, wherein said executable software modules are further configured to cause said processor (604) to repair said failure by downloading executable repair code from said remote server (630) using a network protocol with security measures that verify the authenticity of said executable repair code.

9. The computer of claim 7, wherein said executable software modules are further configured to cause said processor (604) to receive a command from said remote server (630), said command configured to cause said processor (604) to execute one or more functions stored in said memory region (212) that assist in repairing said failure.

10. The computer of claim 6, wherein said memory management in said real time kernel module (220) includes management of a network browser page cache (490) as a request queue region (491), for storing data and commands that have not been serviced, a pre-processed data region (492), for storing data that must be interpreted for output to a display device (612), including pre-fetched batches of hypertext markup language files, and a post-processed region (493), for storing data that has been interpreted for output to said display device (612), but has not yet been sent to said display device (612).

11. A method for diagnosing and repairing a networked computer with a self-contained network browser (208) stored

in a PC BIOS (212), the method comprising a sequences of steps performed by said networked computer, the steps comprising:

initializing (504) internal and peripheral devices communicatively computed to said networked computer; the invention characterized by:

testing (508) to determine whether a browser trigger has been set, said browser trigger indicating a failure has occurred; and

when said browser trigger has been set then:

invoking (524) said self-contained network browser (208) without loading a full-service operating system; and

connecting (532) to a remote server with said self-contained network browser (208);

loading (536) one or more diagnostic interface files;

diagnosing (540) said failure interactively with said remote server;

determining (544) whether said failure is repairable with said remote server;

repairing (548) said failure when said failure is repairable; and

sending (552) a notification that said failure is not repairable when said failure is not repairable.

12. The method of claim 11, wherein said step of repairing (548) said failure includes downloading executable repair code from said remote server using a network protocol with security measures that verify the authenticity of said executable repair code.

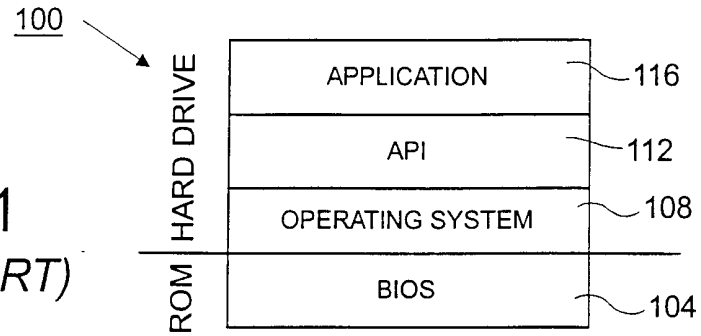


FIG. 1
(PRIOR ART)

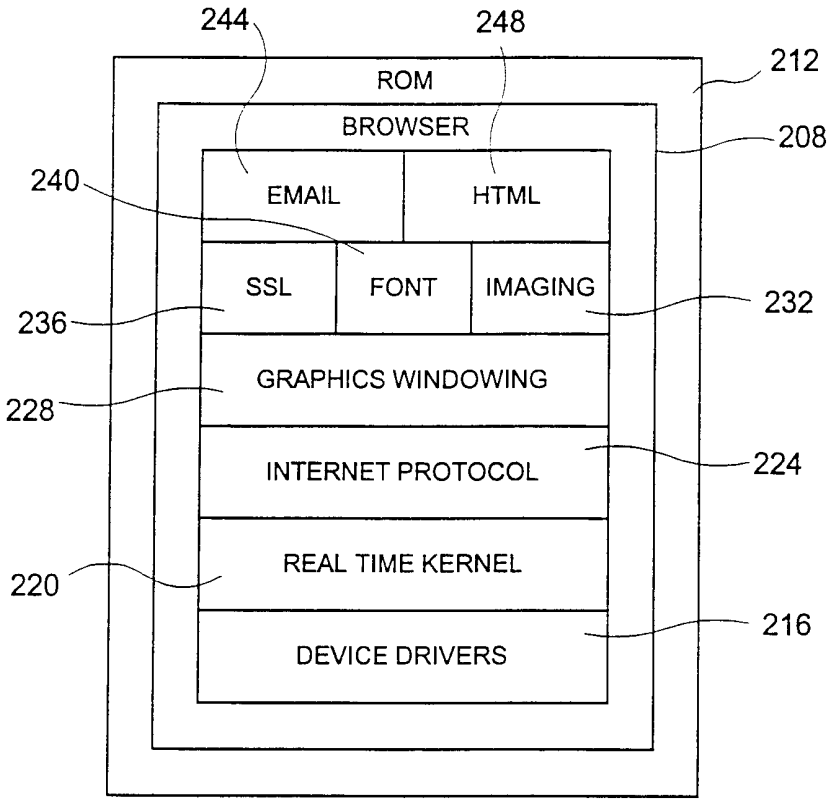


FIG. 2

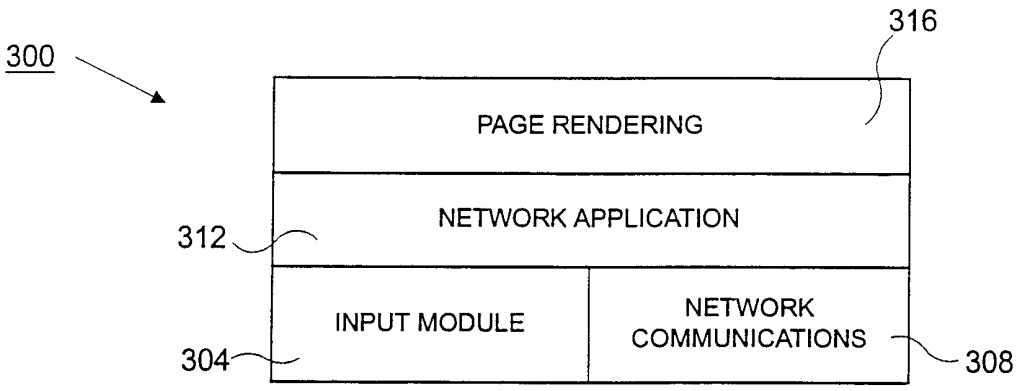


FIG. 3

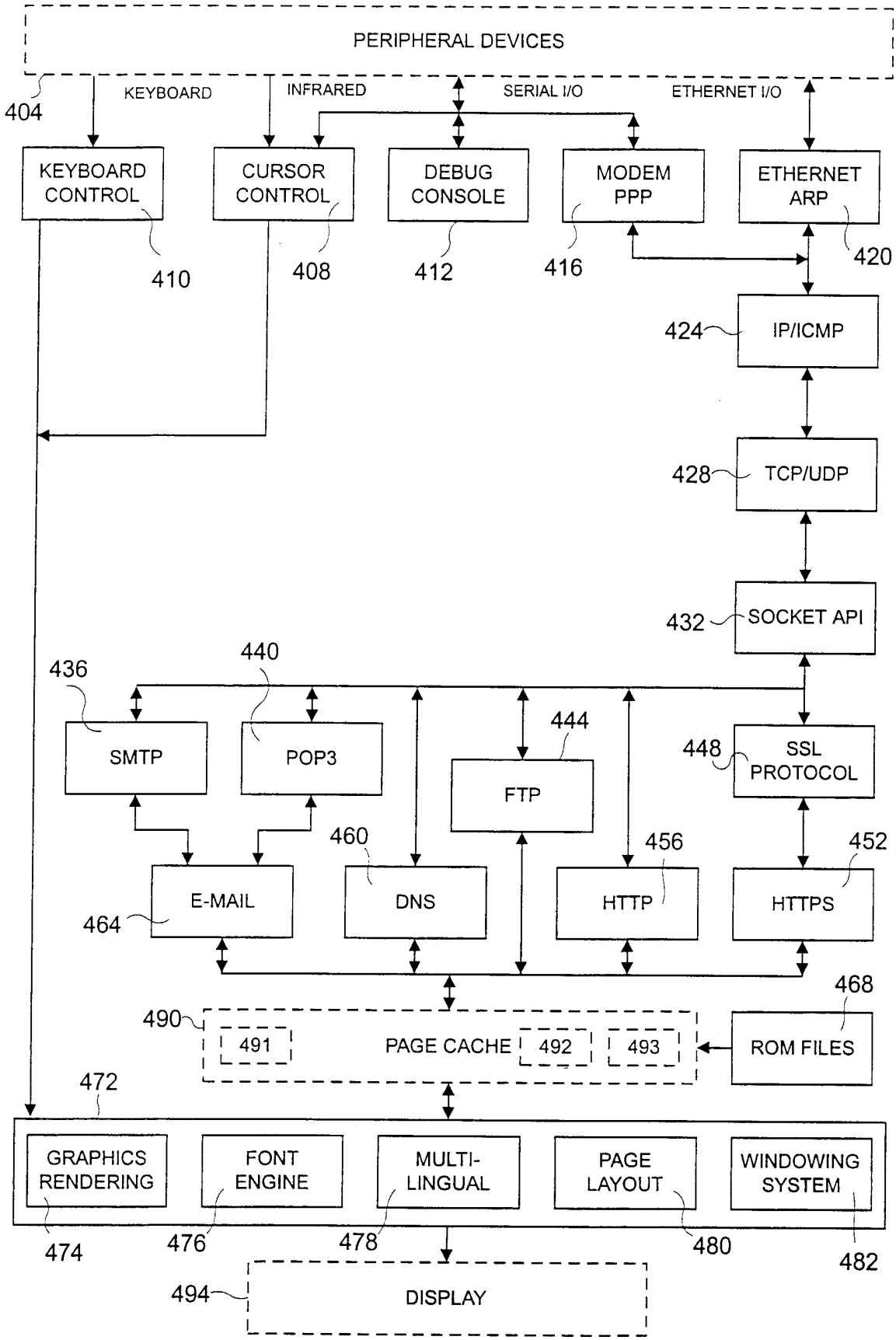

PERIPHERAL DEVICES
404
KEYBOARD
INFRARED
SERIAL I/O
ETHERNET I/O
KEYBOARD CONTROL
CURSOR CONTROL
DEBUG CONSOLE
MODEM PPP
ETHERNET ARP
410
408
412
416
420
424
IP/ICMP
428
TCP/UDP
432
SOCKET API
436
440
444
448
SMTP
POP3
FTP
SSL PROTOCOL
460
456
452
464
E-MAIL
DNS
HTTP
HTTPS
468
490
491
PAGE CACHE
492
493
ROM FILES
472
GRAPHICS RENDERING
FONT ENGINE
MULTI-LINGUAL
PAGE LAYOUT
WINDOWING SYSTEM
474
476
478
480
482
494
DISPLAY

FIG. 4

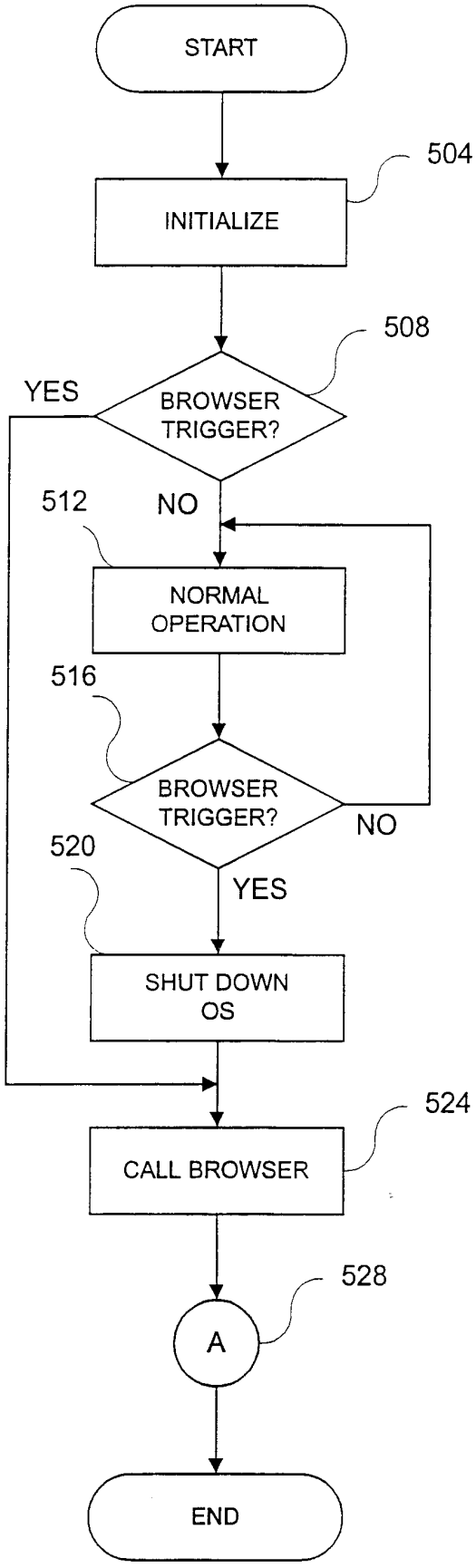

START
504
INITIALIZE
508
YES
BROWSER TRIGGER?
512
NO
NORMAL OPERATION
516
BROWSER TRIGGER?
NO
520
YES
SHUT DOWN OS
524
CALL BROWSER
528
A
END

FIG.5A

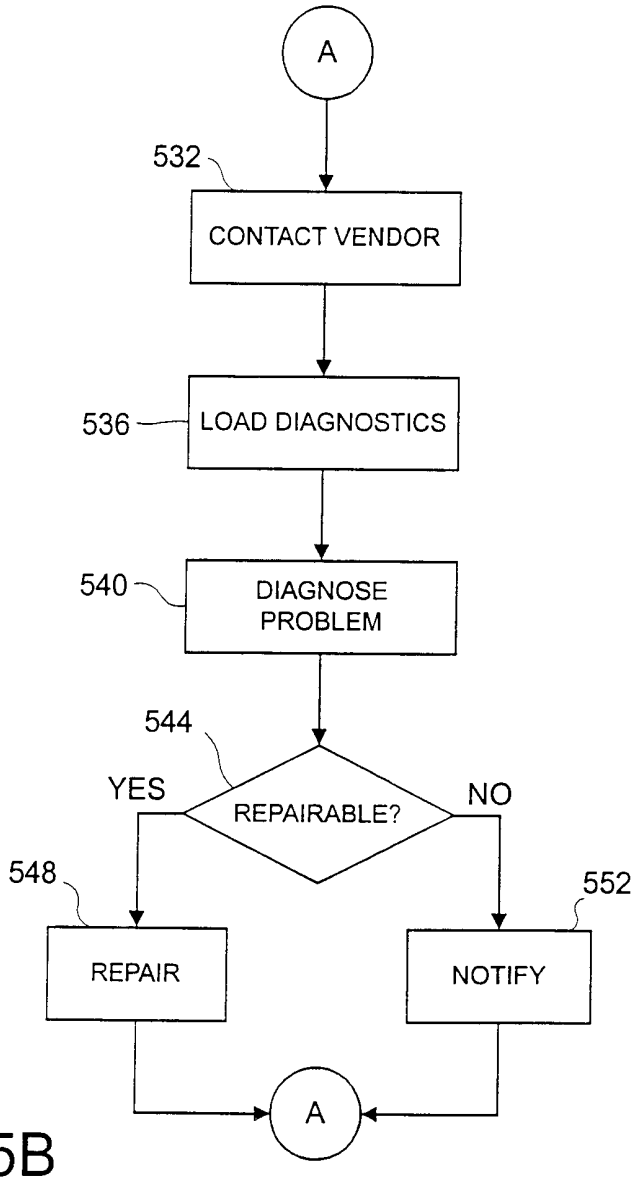

A
532
CONTACT VENDOR
536
LOAD DIAGNOSTICS
540
DIAGNOSE PROBLEM
544
YES
REPAIRABLE?
NO
548
552
REPAIR
NOTIFY
A

FIG. 5B

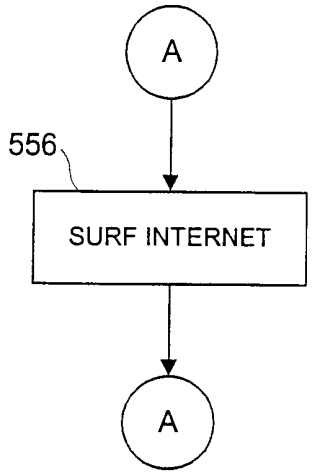

A
556
SURF INTERNET
A

FIG. 5C

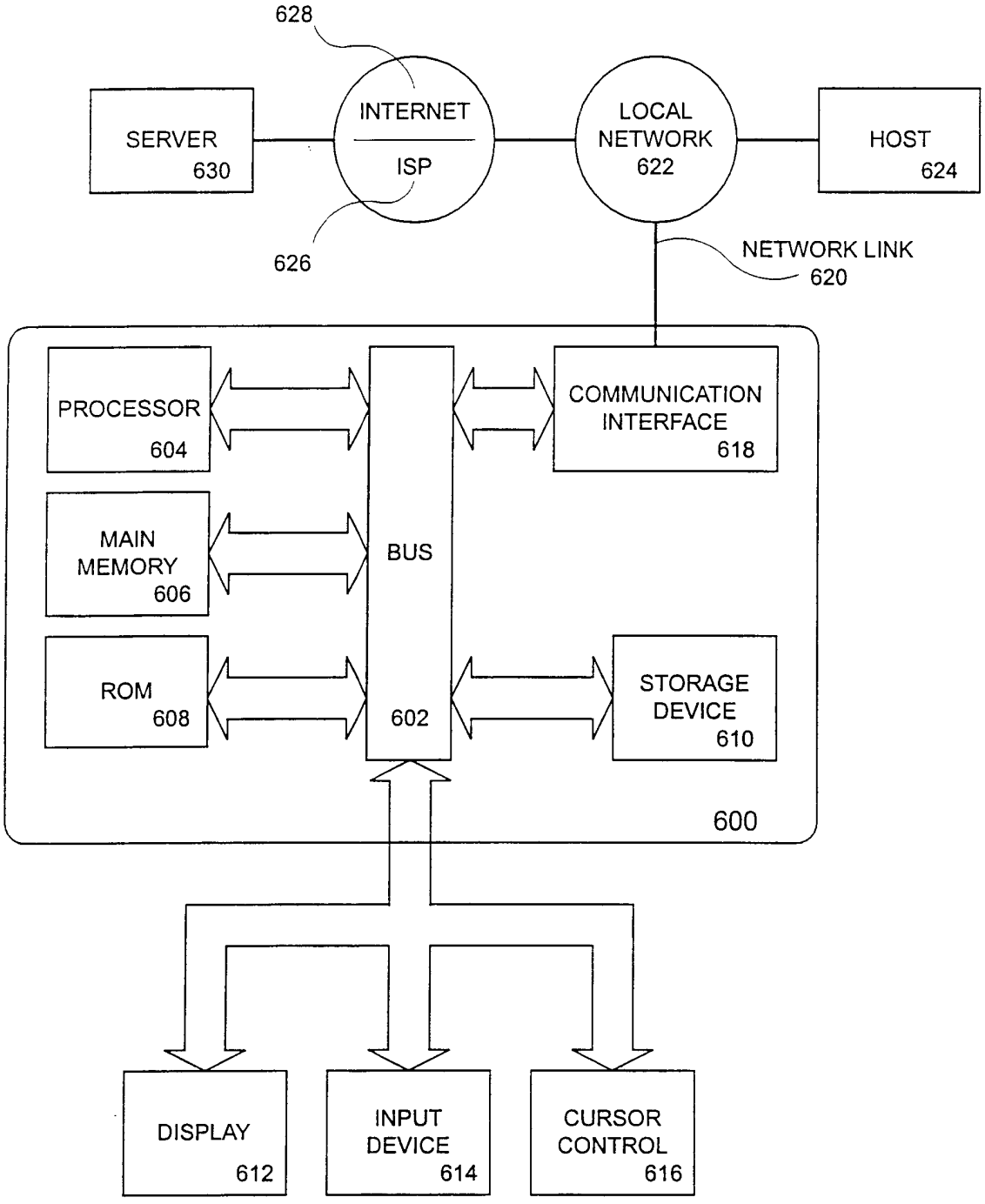

628
INTERNET
ISP
SERVER
630
LOCAL
NETWORK
622
HOST
624
626
NETWORK LINK
620
PROCESSOR
604
COMMUNICATION
INTERFACE
618
MAIN
MEMORY
606
BUS
ROM
608
602
STORAGE
DEVICE
610
600
DISPLAY
612
INPUT
DEVICE
614
CURSOR
CONTROL
616

FIG. 6