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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 December 2004 (02.12.2004)

PCT

(10) International Publication Number
WO 2004/104825 A1

(51) International Patent Classification[7]: G06F 9/44

(21) International Application Number:
PCT/US2004/015077

(22) International Filing Date: 13 May 2004 (13.05.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/471,523 15 May 2003 (15.05.2003) US

(71) Applicant *(for all designated States except US)*: **APPLIANZ TECHNOLOGIES, INC.** [US/US]; 7 Lydia Court, Hawthorn Woods, IL 60047 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **JANZEN, Mark** [US/US]; 7676 East Polo Dr., #39, Wichita, KS 67206 (US).

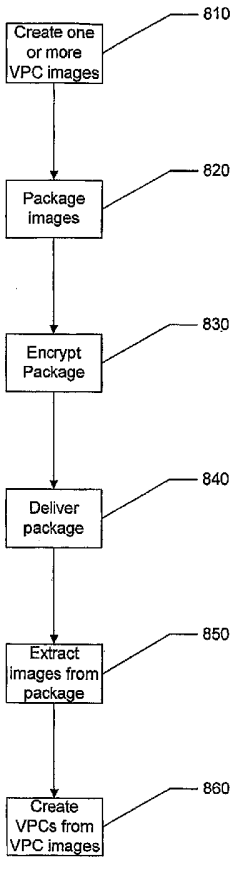
(74) Agent: **THRONSON, Mark, J.**; Dickstein Shapiro Morin & Oshinsky LLP, 2101 L Street, N.W., Washington, DC 20037-1526 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, 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, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM,

[Continued on next page]

(54) Title: SYSTEMS AND METHODS OF CREATING AND ACCESSING SOFTWARE SIMULATED COMPUTERS



(57) Abstract: The system and methods of the present application comprise one or more computers that generate and maintain a plurality of software-simulated computers (860). Each software-simulated computer is adapted to efficiently run an installed application program. Additional security layers provide access to the installed application through a remote user interface installed on a user's coomputing device. The system generates a new copy of the software-simulated computer for each user session (810), that prevents configuration problems from interfering with the proper operation of the application program, thereby consistently running the application in an optimized fashion, regardless of changes made to the software-simulated computer by the user or a virus. These software-simulated computer are unaffected by changes a user makes on their own client device. To this end, the system provides robust, web accessible capabilities to application software that may not have been adapted for user on the internet.

WO 2004/104825 A1

ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

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.

SYSTEMS AND METHODS OF CREATING AND ACCESSING SOFTWARE SIMULATED COMPUTERS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims benefit of U.S. Provisional Patent Application Serial No. 60/471,523, entitled "Systems and Methods of Creating and Accessing Software Simulated Computers," filed May 15, 2003, which is hereby incorporated by reference in its entirety for each of its teachings and embodiments.

FIELD OF THE INVENTION

[0002] This invention relates to the field of virtual computer servers. In particular, it relates to a computer that serves virtual computers on demand.

BACKGROUND OF THE INVENTION

[0003] With the advent of the personal computer and networking technologies, client/server application programs were written that helped to improve worker productivity in small and medium sized companies. Since most such companies lacked the resources to staff an Information Technology Department, and could not generate their own customized applications, most companies implemented standardized application program packages at a fraction of their development cost.

[0004] However, the computer technology industry has been rapidly evolving over the course of the last decade. Networking technologies that once dominated the industry have been supplanted by TCP/IP, the communication protocol that underlies the ubiquitous World-Wide-Web and the Internet. However, networks often suffer from a wide variety of problems that can directly impact application software performance. Consequently, users depend upon technical support personnel to troubleshoot and repair countless system problems that may arise from these network related problems that prevent users from accessing application programs, such as viruses or denial of service attacks.

[0005] In addition, new computer languages, such as Java, have developed to implement new technologies in the present Internet computing paradigm. But application

programs that were written before such languages were even conceived cannot take advantage of the functionality provided by such new computer languages.

[0006] In addition, many standardized application programs were designed to run on communication protocols that are incompatible with TCP/IP. Unfortunately, these application program packages may also have been designed on an older client/server model and not an Internet-based model. Consequently, a complete source code rewrite for these application programs would be necessary to implement them in a modern Internet-based computing paradigm. Such a rewrite would not only be cost prohibitive, but might also exceed a mid-sized business's resources. Further, the application program may no longer be supported by the original software developers and vendors. A mid-sized company that wishes to improve their access to key computer applications would be faced with a dilemma of either purchasing a new application program and incurring the additional cost of converting data accumulated over many years into a new format used by such a new application program, or incurring the maintenance expense for a legacy application program and forsaking the freedom of accessing the application through the ubiquitous Internet.

[0007] Therefore, there exists a need for a computing platform that can transform older, legacy applications into a modern-day, Internet-based application without bearing the expense and effort of rewriting source code. In addition, there exists a need for a robust platform that provides a consistent application program performance without being affected by changes made by a user, a virus, or other malicious software such as Trojan horses, spyware, or adware.

BRIEF SUMMARY OF THE INVENTION

[0008] The system and methods of the present application comprise one or more computers that generate and maintain a plurality of software-simulated computers. Each software-simulated computer is adapted to efficiently run an installed application program. Additional security layers provide access to the installed application through a remote user interface installed on a user's computing device. The system generates a new copy of the software-simulated computer for each user session, which prevents configuration problems from interfering with the proper operation of the application program, thereby consistently running the application in an optimized fashion, regardless of changes made to the software-simulated computer by the user or a virus. These software-simulated computers

are unaffected by changes a user makes on their own client device. To this end, the system provides robust, web accessible capabilities to application software that may not have been adapted for use on the Internet.

[0009] In one aspect, the present invention is directed to a method of deploying and remotely accessing a plurality of software-simulated computers, comprising:

creating a software-simulated computer image, said image comprising:

simulated hardware device specifications;

a bootable application;

a guest process manager, and;

one or more application programs;

cloning said image to create said plurality of software-simulated computers;

branding each software simulated computer in said plurality with unique, machine-differentiation information;

selecting a software-simulated computer in said plurality; and

establishing communications for remote access across a network to said selected software-simulated computer.

[0010] In another aspect of the present invention, said step of branding prevents communication conflicts between machines on said network.

[0011] In another aspect of the present invention, said bootable application is a Windows variant, and said machine differentiation information includes a system identifier.

[0012] In another aspect of the present invention, the method further comprises: loading user specific information into said image before performing said cloning step.

[0013] In another aspect of the present invention, said user specific information comprises software license numbers.

[0014] In another aspect of the present invention, said user specific information includes one or more of company names and individual names.

[0015] In another aspect of the present invention, said user specific information includes user identifiers and associated passwords.

[0016] In another aspect of the present invention, said branding prevents conflicts between machines on said network.

[0017] In another aspect of the present invention, the method further comprises booting each software-simulated computer.

[0018] In another aspect of the present invention, the method further comprises evaluating quality of said communications and selecting a remote control communications protocol based on said quality.

[0019] In another aspect of the present invention, the method further comprises accessing said selected software-simulated computer through a remote user interface.

[0020] In another aspect of the present invention, the method further comprises configuring a firewall to permit communications with said selected software-simulated computer.

[0021] In another aspect, the present invention is directed to a method of creating one or more software-simulated computers on a remote computer, comprising:

creating one or more software-simulated computer images;

generating a package comprising said images; and

delivering said package to said remote computer, wherein said remote computer extracts said images and automatically creates said software-simulated computers.

[0022] In another aspect of the present invention, said package includes an xml document comprising installation instructions and said remote computer generates said software-simulated computers in accordance with said instructions.

[0023] In another aspect of the present invention, said package is encrypted.

[0024] In another aspect of the present invention, each of said images comprises:

simulated hardware device specifications;

a bootable application;

a guest process manager, and;

one or more application programs.

[0025] In another aspect of the present invention, one of said images represents a server computer.

[0026] In another aspect of the present invention, one of said images represents a firewall computer.

[0027] In another aspect, the present invention is directed to a software-simulated computer server for providing a client device access to an application program on a software-simulated computer through a network, comprising:

one or more hardware computers;

an image that defines a software-simulated computer having a copy of said application program stored thereon; and

a host control program that causes said one or more hardware computers to create a plurality of software-simulated computers from said image and to generate unique, machine-differentiation information for each software-simulated computer in said plurality;

wherein said host control program further causes said hardware computers to select a software-simulated computer from said plurality, to negotiate a communications connection between said selected software-simulated computer and said client device, and to enable said client device to access said application program running on said selected software-simulated computer through said network.

[0028] In another aspect of the present invention, said host control program causes said one or more hardware computers to shutdown, recreate, and restart said plurality of software-simulated computers.

[0029] In another aspect of the present invention, said host control program further causes said one or more hardware computers to copy user generated data to backup storage.

[0030] In another aspect of the present invention, said host control program recreates and restarts said plurality of software-simulated computers after said user generated data has been copied.

[0031] In another aspect of the present invention, said host control program responds to client device requests using one or more communication protocols from a list comprising FTP, HTTP, HTTPS, MPLS, SFTP, SMTP, and SSH.

[0032] In another aspect of the present invention, said application program is designed to be used on a single personal computer.

[0033] In another aspect of the present invention, said application program is a client/server application.

[0034] In another aspect of the present invention, said application program uses one or more communication protocols from a list consisting of IPX/SPX, netbios, raw IP sockets, UDP/IP, TCP/IP, IPv6, IPSEC, HTTP, and netbeui.

[0035] In another aspect of the present invention, said host control program balances load on said hardware computers when making said selection of said software-simulated computer.

[0036] In another aspect of the present invention, said load is determined by one or more of available memory, processor utilization, and a number of unused software-simulated computers.

[0037] In another aspect of the present invention, said software-simulated computer is adapted to accept and communicate with and to provide concurrent interaction of said application with more than one client device.

[0038] In another aspect of the present invention, said software-simulated computer is adapted to record user input from said client device.

[0039] In another aspect of the present invention, the software-simulated computer server further comprises a control center computer in communication with a host control virtual computer, wherein said control center computer transmits said image for said software simulated computer to said host control virtual computer.

[0040] In another aspect of the present invention, said control center computer receives status information about a software-simulated computer.

[0041] In another aspect of the present invention, said control center computer is adapted to issue a reboot command that causes a particular software-simulated computer to be shutdown, recreated, and restarted.

[0042] In another aspect of the present invention, said control center computer transmits an updated image comprising an updated copy of said application program and said host control program causes said one or more hardware computers to shutdown, recreate using said updated image, and restart said plurality of software-simulated computers.

[0043] In another aspect, the present invention is directed to a media storing a computer program that causes a processor that executes said program to perform a method of deploying and remotely accessing a plurality of software-simulated computers, the steps comprising:

cloning an image that defines a software simulated computer to create said plurality of software-simulated computers;

branding each software simulated computer in said plurality with unique, machine-differentiation information;

selecting a software-simulated computer in said plurality; and

establishing communications for remote access across a network to said selected software-simulated computer.

[0044] In another aspect of the present invention, said computer program causes said processor to perform steps comprising: loading user specific information into said image before performing said cloning step.

[0045] In another aspect of the present invention, said computer program causes said processor to perform steps comprising: evaluating quality of said communications and selecting a remote control client based on said quality.

[0046] In another aspect of the present invention, said computer program causes said processor to perform steps comprising: configuring a firewall to permit communications with said selected software-simulated computer.

BRIEF DESCRIPTION OF THE DRAWINGS

[0047] FIG. 1 is a block diagram that illustrates a system that serves virtual computers and provides a client device access to an application program on such virtual computers through a network;

[0048] FIG. 2 is a block diagram that illustrates an image for a virtual computer;

[0049] FIG. 3A is a block diagram that illustrates a virtual computer;

[0050] FIG. 3B is a block diagram that illustrates a host controller virtual computer;

[0051] FIG. 4 is a block diagram that illustrates remote user interface components;

[0052] FIG. 5 is a state diagram that illustrates a virtual computer's life cycle;

[0053] FIG. 6 is a flowchart that depicts a method of deploying and remotely accessing a virtual computer;

[0054] FIG. 7A is a block diagram that illustrates a collection of servers and data storage structures known as Mission Control;

[0055] FIG. 7B is a block diagram that illustrates a package and associated components;

[0056] FIG. 8 is a flowchart that depicts a preferred embodiment of remotely creating one or more software-simulated computers;

[0057] FIGS. 9A and B are an XML listing illustrating instructions for a watchdog process; and

[0058] FIG. 10 is a block diagram that illustrates various types of media.

DETAILED DESCRIPTION OF THE INVENTION

[0059] The present invention comprises a system and methods for serving virtual personal computers (VPCs). The system provides a means for a computer user to access an application without installing the application on her client computing device. The system also provides a means for the user to access the application from any location where a communications connection can be established with the system. One with skill in the art will understand that this system provides a centralized means for administering the

distribution and operation of computer applications, which improves application reliability and increases employee productivity.

[0060] Although the invention has been described herein as a system and method for serving VPCs, one of ordinary skill in the art will appreciate that the invention is not so limited (e.g., may be used as system for maintaining a local area network (LAN)) and may include any modification that permits interoperability of a legacy personal computer application with other modern computer networks and interfaces. For example, a prior art client/server application utilizing Novell Netware's™ IPX/SPX communication protocol can be installed in this system and run over the Internet (which uses a completely different communication protocol, namely TCP/IP) without any modification to the application and despite whether the application was designed for Internet accessibility.

[0061] Furthermore, it should be understood that the detailed description and specific examples, while indicating exemplary embodiments of the present invention, are given for purposes of illustration only and not for limitation. Although the present invention described herein principally details exemplary traditional client/server applications, it should be appreciated that this system is not so limited and would accommodate single-user or standalone applications as well.

[0062] Additionally, the present invention may be described herein in terms of functional block components, code listings, optional selections and various processing steps. It should be appreciated that such functional blocks may be realized by any number of hardware and/or software components configured to perform the specified functions. For example, the present invention may employ various integrated circuit components, e.g., memory elements, processing elements, logic elements, look-up tables, and the like, which may carry out a variety of functions under the control of one or more microprocessors or other control devices.

[0063] Similarly, the software elements of the present invention may be implemented with any programming or scripting language such as C, C++, C#, Java, COBOL, assembler, PERL, or the like, with the various algorithms being implemented with any combination of data structures, objects, processes, routines or other programming elements. Preferably, the computer code used to provide the described functionality is developed with Microsoft Visual Studio. The computer code is preferably programmed in Visual Basic 6, C, C++, C#, Visual Basic .NET, and Transact SQL. The object code

created can be executed by any computer having a Windows™ 2000 or higher operating system and the Microsoft .NET Framework™ version 1.1, and VMWare Workstation™ version 4.05 or higher.

[0064] Further, it should be noted that the present invention may employ any number of conventional techniques for data transmission, signaling, data processing, network control, and the like. For a basic introduction of cryptography, please review a text written by Bruce Schneider which is entitled "Applied Cryptography: Protocols, Algorithms, And Source Code In C," published by John Wiley & Sons (second edition, 1996), which is hereby incorporated by reference.

[0065] It should be appreciated that the particular implementations shown and described herein are illustrative of the invention and its best mode and are not intended to otherwise limit the scope of the present invention in any way. Indeed, for the sake of brevity, conventional data networking, application development and other functional aspects of the systems (and components of the individual operating components of the systems) may not be described in detail herein. Furthermore, the connecting lines shown in the various figures contained herein are intended to represent exemplary functional relationships and/or physical or virtual couplings between the various elements. It should be noted that many alternative or additional functional relationships or physical or virtual connections may be present in a practical electronic data communications system.

[0066] As will be appreciated by one of ordinary skill in the art, the present invention may be embodied as a method, a data processing system, a device for data processing, and/or a computer program product. Accordingly, the present invention may take the form of an entirely software embodiment, an entirely hardware embodiment, or an embodiment combining aspects of both software and hardware. Furthermore, the present invention may take the form of a computer program product on a computer-readable storage medium having computer-readable program code means embodied in the storage medium. Any suitable computer-readable storage medium may be utilized, including hard disks, CD-ROM, optical storage devices, magnetic storage devices, and/or the like.

[0067] The present invention is described below with reference to block diagrams and flowchart illustrations of methods, apparatus (e.g., systems), and computer program products according to various aspects of the invention. It will be understood that each functional block of the block diagrams and the flowchart illustrations, and combinations of

functional blocks in the block diagrams and flowchart illustrations, respectively, can be implemented by computer program instructions. These computer program instructions may be loaded onto a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions that execute on the computer or other programmable data processing apparatus create means for implementing the functions specified in the flowchart block or blocks.

[0068] These computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instruction means that implement the function specified in the flowchart block or blocks. The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer-implemented process such that the instructions that execute on the computer or other programmable apparatus provide steps for implementing the functions specified in the flowchart block or blocks.

[0069] These computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instruction means that implement the function specified in the flowchart block or blocks. The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer-implemented process such that the instructions that execute on the computer or other programmable apparatus provide steps for implementing the functions specified in the flowchart block or blocks.

[0070] Accordingly, functional blocks of the block diagrams and flowchart illustrations support combinations of means for performing the specified functions, combinations of steps for performing the specified functions, and program instruction means for performing the specified functions. It will also be understood that each functional block of the block diagrams and flowchart illustrations, and combinations of functional blocks in the block diagrams and flowchart illustrations, can be implemented by either special purpose

hardware-based computer systems that perform the specified functions or steps, or suitable combinations of special purpose hardware and computer instructions.

[0071] The scope of the invention should be determined by the appended claims and their legal equivalents, rather than by the examples given herein. For example, the steps recited in any method claims may be executed in any order and are not limited to the order presented in the claims. Moreover, no element is essential to the practice of the invention unless specifically described herein as "critical" or "essential."

System Architecture

[0072] Fig. 1 is a block diagram illustrating a preferred embodiment for serving virtual personal computers (VPCs), also known as software-simulated computers. As shown in Fig. 1, system architecture 100 preferably comprises a master virtual computer server 110, zero or more slave virtual computer servers 150, a network 140, a client device 170 and physical communication connections 145.

[0073] In addition, system architecture 100 may also include a connection to Internet 148 and a remote client device 180, attached to Internet 148.

[0074] A collection of fileservers and databases, collectively known as Mission Control 190, described in further detail in connection with Fig. 7 below, may also be part of system architecture 100, and may be used to monitor operation of system 100.

[0075] Master virtual computer server 110 is preferably coupled to slave virtual computer server 150 through one or more network communications lines 145. Although Fig. 1 illustrates master virtual computer server 110 and slave virtual computer server 150 as distinct computers, one skilled in the art will recognize that master virtual computer server 110 and slave virtual computer server 150 may, for example, be implemented in a single hardware computer having one or more processors, and may be implemented as concurrently processed applications running in said master virtual computer server 110 and slave virtual computer server 150.

[0076] Master virtual computer server 110 and slave virtual computer server 150 are preferably dedicated, high-performance computers adapted to serve VPCs to a user. These servers 110/150 comprise a processor, storage, and communications interfaces sufficient to network these computers with client devices 170 and/or 180. Preferably, virtual computer

servers 110/150 comprise AMD Opteron 144 processors, 4GB of ECC RAM, and mirrored 10K SATA hard drives.

[0077] In a preferred embodiment, master virtual computer server 110 comprises an image 115, a host controller VPC 120, a host updater program 122, a remote process manager (RPM) 125, and one or more VPCs 130.

[0078] In addition, master virtual computer server 110 comprises many virtual communication connections 135, some of which are illustrated in Fig. 1. In a preferred embodiment, virtual communication connections 135/165 are virtual hubs, switches, and connections that are created with VMWare's VMNet bridge protocol program, or similar virtual network creation software.

[0079] Image 115 is a data file that contains information that describes a VPC. Further description of image 115 is presented below in connection with Fig. 2. Images are preferably created at Mission Control 190, as described below. Alternatively, an image author may also generate images for use in system 100 or server 110.

[0080] Host controller VPC 120 is a virtual computer that contains a program known as a host controller program. Host controller VPC 120, through the host controller program, is responsible for managing virtual computers in master virtual computer server 110 and slave virtual computer server 150 as described below.

[0081] Host updater 122 is a short program that fetches a copy of RPM 125 from host controller VPC 120, loads it into master virtual computer server 110's memory, and starts RPM 125.

[0082] RPM 125 is a program that manages processes running on master virtual computer server 110. RPM 125 starts, stops, suspends, and monitors these processes, and follows instructions received from host controller VPC 120.

[0083] In a preferred embodiment, slave virtual computer server 150 comprises a host updater 152, a remote process manager 155, and one or more VPCs 160. In addition, slave virtual computer server 150 further comprises virtual communication connections 165, some of which are illustrated in Fig. 1.

[0084] Like master virtual computer server 110, slave virtual computer server 150's RPM 155 is a program responsible for managing processes on slave virtual computer server

150. Similarly, host updater 152 is a short program that fetches a copy of RPM 155 from host controller VPC 120, loads it into slave virtual computer server 150's memory, and starts RPM 155.

[0085] Fig. 1 illustrates a client device 170 that contains a remote user interface program 175. Remote user interface program 175 enables client device 170 to interact with a VPC on master virtual computer server 110 or slave virtual computer server 150. This interaction takes place through network 140.

[0086] Fig. 1 illustrates a remote client device 180, which also contains a remote user interface program 185 that enables device 180 to interact with VPCs on servers 110 and/or 150. Remote client device 180 is connected to servers 110/150 through Internet 148 merely for illustrative purposes. One skilled in the art will appreciate that network 140 and such connection between remote client device 180 may include any system for exchanging data, such as an Intranet, an Extranet, WAN, LAN, satellite communications, and/or the like.

[0087] Client devices 170/180 include any computing device such as a keyboard, mouse, kiosk, personal digital assistant, hand held computer (e.g., Palm Pilot™), cellular phone and/or the like. Similarly, the invention could be used in conjunction with any type of personal computer, network computer, workstation, mini-computer, mainframe, video game system or the like running any operating system such as any versions of Windows, Windows NT, Windows 2000, Windows 2003, Windows 98, Windows 95, Windows XP, Windows XP embedded, MAC OS, OS/2, BEOS, Linux, UNIX, or the like.

[0088] Moreover, although the invention is frequently described herein as being implemented with TCP/IP communications protocols, it will be readily understood that the invention could also be implemented using IPX/SPX, Appletalk, Netbios, raw IP sockets, UDP/IP, IP v6, IP sec, Netbeui, FTP, HTTP, HTTPS, SFTP, SMTP, and SSH, or any number of existing or future communication protocols.

[0089] Figure 2 illustrates an image for generating a VPC. Image 115 comprises a bootable application image 205, simulated hardware device specifications 220, an application image 230, a guest process manager image 240, and a guest updater image 250.

[0090] Bootable application image 205 is a copy of an operating system or some other application that can be run by a personal computer at boot time. Such a bootable application can be, for example, Windows, Linux, or one of the aforementioned operating systems.

[0091] Simulated hardware device specifications 220 comprise specifications that define simulated hardware on a VPC. Such specifications include specifications for a hard disk storage device, random access memory, a processor, and interfaces such as parallel or serial ports, Ethernet network interface cards, video cards, keyboards, or mouse interfaces.

[0092] Guest process manager image 230 is a copy of a program that manages other processes on the VPC. Guest process manager represents an application program that runs under the auspices of a boot application.

[0093] Application image 215 is a copy of any computer program designed to run on a personal computer. Application image 215 is therefore designed to run under the auspices of a boot application. Application image 215 also is invoked by a guest process manager.

[0094] Figs. 3A and 3B illustrate a generic VPC 130/160 and a host controller VPC 120, respectively. VPCs 130/160 are generated from images loaded into master virtual computer server 110 or slave virtual computer server 150, as described in more detail below. Host controller VPC 120 is generated from an image loaded into master virtual computer server 110.

[0095] Fig. 3A illustrates a VPC 130/160. As shown in Fig. 3A, VPC 130/160 comprises a bootable application 305, simulated hardware devices 320, a guest process manager 330, a guest updater 335 and one or more application programs 340.

[0096] Bootable application 305 is generated from boot application image 205, and comprises an operating system used to run VPC 130/160. Simulated hardware devices 320 are generated from simulated hardware device specifications 220, and represent virtual hardware devices in VPC 130/160.

[0097] Guest process manager 330 is a special application that runs under the auspices of boot application 305. Guest process manager 330 monitors the processors running on VPC 130/160, starts said processes, stops said processes, and generally maintains said

processes. Guest process manager 330 accepts commands and executes orders from host controller program 390.

[0098] Guest updater 335 is a short program fetches a copy of guest process manager 330 from host controller VPC 120 at boot time, loads it into VPC 130/160's virtual memory, and executes it.

[0099] Application 340 is a computer program that runs under the auspices of boot application 305. Application 340 can be, but is not limited to a program, an active X component that runs on a web browser, or a java applet that runs on a web browser. Typically, application 340 may authenticate a user by query for a product key code or by other means. Such authentication is described in a system operation section below.

[0100] Turning to Fig. 3B, host controller VPC 120 comprises a bootable application 355, simulated hardware devices 370, and a host controller program 390.

[0101] Bootable application 355 is generated from boot application image 205, and comprises an operating system used to run host controller VPC 120. Simulated hardware devices 370 are generated from simulated hardware device specifications 220, and represent virtual hardware devices in host controller VPC 120.

[0102] Host controller program 390 is an application program that runs under the auspices of boot application 355. Host controller program 390 performs functions such as creating VPCs, deleting VPCs, cloning VPCs, and managing VPCs in master virtual computer server 110 and slave virtual computer server 150 as well as reporting status and control information. Host controller program 390's functions are described in more detail in connection with system 100's operation below.

[0103] Fig. 4 illustrates a remote user interface 175/185. Remote user interface 175/185 comprises a communication interface 410, a display renderer 420, and a user input interface 430.

[0104] Communication interface 410 is a program module that communicates with user input interface 430 and display renderer 420. Communication interface also communicates with other devices via network link 145. Communication interface 410 receives user input from user input interface 430 and either recasts it as a request to a VPC or passes it on to display renderer 420.

[0105] Display renderer 420 is a program module that places information on a display of client device 170/180. Display renderer 420 provides a view of data received and renders a facsimile of a screen that application 340 would present to a user.

[0106] User input interface 430 is a program module that receives user input. Such input would normally be provided by the user interacting with application 340, such as keystrokes, mouse commands, etc.

[0107] In summary, remote user interface 175/185 creates a convincing illusion that the user is interacting with an application program installed on client device 170/180. One with skill in the art would know that remote user interface 175/185 can be implemented with generic, off-the-shelf software, or by a custom application. In a preferred embodiment, remote user interface is implemented through a combination of code to communicate with host controller program 390 and either a web browser, Microsoft Remote Desktop, VNC, or similar desktop remote user interface technology.

[0108] Communication between remote user interface 175/185 and servers 110/150 is accomplished through any suitable communication means, such as, for example, a telephone network, Intranet, Internet 148, point of interaction device (point of sale device, personal digital assistant, cellular phone, kiosk, etc.), online communications, off-line communications, wireless communications, and/or the like.

System Operation

[0109] The following discussion describes system functions performed by host controller program 390. Preferably, host controller program 390 is running on host controller VPC 120, but alternatively may be run as a stand-alone process in servers 110/150. In such event, it should be understood that references to host controller VPC 120 also comprise the activities of host controller program 390, and such terms are to be considered interchangeable.

[0110] Fig. 5 is a state diagram that illustrates a life cycle of a VPC. As shown in Fig. 5, initially a VPC starts out in a created state 510. Next, the VPC transitions to a booted state 520. Once the VPC has booted, it establishes communications with host controller VPC 120, and then transitions into a ready state 530 once such communications have been established.

[0111] While in ready state 530, the VPC is available for users to connect to it. It also listens to host controller VPC 120 for any commands and updates host controller VPC with its status. In order for a user to connect to VPC 130/160 and use application 340, application 340 typically authenticates a user by querying a product key code or it may display a list of products that have a registered license and that provide access to the user. The user may connect to as many products as there are licenses, however, application 340 may have other constraints, e.g., it may only allow only a single login per user.

[0112] Once a user has selected an application 340, remote user interface 175/185 attempts to connect to master virtual computer server 110 in general, and host controller VPC 120 in particular. If host controller VPC 120 is not located (at the last known IP address), remote user interface 175/185 will query Mission Control 190 for network connection information. In response, Mission Control 190 returns one or more IP addresses where host controller VPC 120 may be found. Remote user interface 175/185 then tries the returned information until it establishes communications with host controller VPC 120.

[0113] Remote user interface 175/185 queries host controller VPC 120 for the network connection information of a VPC from the plurality of VPCs 130/160 that has a required application 340. Host controller VPC 120 selects the VPC based on load balancing considerations. Such considerations include the amount of available memory the processor utilization, and/or the number of ready VPCs in server 110/150. If no VPCs are ready, host controller VPC 120 will create another VPC. If the maximum number of VPCs for application 340 are already running, host controller VPC 120 returns a message to remote user interface 175/185 that no more sessions are available. If application 340 is not present on servers 110/150, then host controller VPC 120 returns a message that such application was not found.

[0114] Then, host controller VPC 120 returns the necessary information, such as the IP address, port number, and communication protocols for the selected VPC 130/160. Preferably, host controller VPC 120 ensures that remote user interface 175/185 connects to the selected VPC 130/160 by creating internal communication paths 135/165 to the selected VPC and by reconfiguring other VPCs, such as a firewall VPC, as described below.

[0115] Once a user connects to the selected VPC, the VPC transitions to connected state 540. While in connected state 540, the VPC informs host controller VPC 120 that a user has connected to it and then it transitions to running state 550.

[0116] While in running state 550, several scripts and programs are executed in response to commands issued by host controller VPC. For example, such scripts may map network drives or change environment settings for the application's use when establishing a connection with a fileserver. But, primarily the connected user interacts with application program 340 while the VPC is in running state 550.

[0117] In a preferred embodiment, more than one client device 170/180 can connect with a VPC, for collaboration projects, video conferencing, etc. In yet another preferred embodiment, user input is re-ordered so that it can be used later, e.g., for demonstrative purposes.

[0118] While in running state 550, the user may download files from the selected VPC to her computing device 170/180. During such file transfers, if the communications protocol does not provide a direct way of effecting the transfer, host controller VPC 120 may broker the transfer in a two-step process. In the first step, host controller VPC 120 uses the communications protocol to receive the file and to temporarily store the file. Then, in the second step, host controller VPC 120 transfers the file to the destination. In a preferred embodiment, these transfers are effected in a secure, encrypted manner and authenticated by host controller VPC 120.

[0119] In addition, if a user requests printing, the print job is transmitted to computing device 170/180, where the user chooses which printer to use and any other commonly toggled printing options necessary to direct the printed document's output.

[0120] The user may complete his task and disconnect from the VPC in running state 550. When this occurs, the VPC transitions to a disconnected state 560. While in disconnected state 560, the VPC notifies host controller VPC 120 that the user has disconnected from it and awaits further instructions from host controller VPC 120.

[0121] If the user temporarily loses the communication connection between remote user interface 175/185 and the selected VPC, the VPC transitions to disconnected state 560 from running state 550. Remote user interface 175/185 requests reconnection to the very same selected VPC from host controller VPC 120. If the connection cannot be

reestablished, host controller VPC 120 informs remote user interface 175/185, shuts down the selected VPC as described below, and negotiates a new connection with another VPC selected from VPCs 130/160.

[0122] In response to a user's request, or if a user logs out, the VPC may also transition from running state 550 to a shutting down state 570. In either case, the VPC informs host controller VPC 120 of the change in state. The VPC may also transition to shutting down state 570 from disconnected state 560 or connected state 540 in response to commands from host controller VPC 120.

[0123] While in shutting down state 570, the VPC proceeds to perform an orderly shutdown. The VPC warns any connected users of the shutdown. Then the VPC transitions to shutdown state 580. In addition, if host controller VPC 120 notes that the VPC is taking too long to shutdown, host controller VPC 120 cleans up the faulty shutdown and ensures that the VPC properly transitions to shutdown state 580.

[0124] In shutdown state 580, the VPC is unable to communicate with host controller VPC 120 for any further commands. When host controller VPC 120 deletes the VPC, the VPC transitions to a destroyed state 590.

[0125] Master virtual computer server 110's operation is now described in terms of its components, but this description also applies to equivalent components found in slave computer server 150 unless otherwise indicated. When a virtual computer server is turned on, the machine boots up in a native operating system installed in the server's boot device. Next, host updater 122 determines whether the server is a master or slave, based on a configuration file stored in the server (not shown). If the virtual computer server is a master, host updater 122 generates and launches host controller VPC 120. Once host control VPC 120 has booted up, host updater 122 establishes communications with host controller VPC 120.

[0126] If the virtual computer server is a slave, then host updater 152 waits for a host controller VPC 120 to boot up on a master virtual computer server 110. Once host controller VPC 120 is running, host updaters 122 and 152 download software required to run RPMs 125 and 155, respectively. Then, host updaters 122 and 152 execute RPM 125 and 155 in servers 110 and 150 respectively.

[0127] As stated above, RPMs 125 and 155 are programs that control the operation of processes in master virtual computer server 110 and slave virtual computer server 150, respectively. RPMs 125 and 155 are in communication with host control VPC 120. RPMs 125 and 155 are responsible for starting processes, stopping processes and monitoring processes. RPMs 125 and 155 also check the health of virtual computer servers 110/150, check for remote login into any computer server via any kind of remote shell or control program, and performing other general security functions. Communications between RPMs 125 and 155 and host control VPC 120 are accomplished various different communication transport protocols. Preferably HTTP is used, but .NET remote computing, JAVA RMI, virtualization software (VM ware, virtual PC, etc.) hidden communication paths, or the like may be used.

[0128] In a preferred embodiment, host updater 122/152 and RPM 125/155 monitors the status of host controller VPC 120 and a special VPC known as a firewall VPC, which is described in more detail below. If a significant period of time has passed since the start up of either host controller VPC 120 or firewall VPC, and communications have not been properly established, host updater 122/152 and/or RPM 125/155 configures the native OS with communication parameters or make a DHCP request, and then reports the trouble back to Mission Control 190, so that a technician can take corrective action.

[0129] Once communications have been established between host controller VPC 120 and RPMs 125 and 155, host controller VPC 120 issues commands and receives events from these RPMs. A typical command issued by host controller VPC 120 is, for example, to launch a VPC that is configured as a firewall. Such a firewall boots up and report back to host controller VPC 120 that it is up and running and is configured.

[0130] Host controller VPC 120 creates one or more VPCs (illustrated as 1 through N in Fig. 1) on master virtual computer server 110 and one or more VPCs (illustrated as 1 through M) on slave virtual computer server 150 by issuing commands to RPMs 125 and 155. Host control VPC 120 uses image 115's and other instructions described below in connection with a package contents to determine the components contained in VPCs 130/160.

[0131] In a preferred embodiment, the first VPC generated after host controller VPC 120 in master virtual computer server 110 is a firewall, indicated as VPC1 in VPC plurality

130, shown in Fig. 1. After the firewall VPC is up and running, all communications between client devices and selected VPCs must pass through this firewall VPC, as illustrated in Fig. 1. In this way, firewall VPC maintains the security of the access to VPCs in system 100. Firewall VPC can be reconfigured at any time by host controller VPC 120. Typical configuration settings include required parameters such as an IP address, subnet mask, gateway address, etc. In addition, host controller VPC 120 may cause firewall VPC to perform tasks such as establishing trusted subnets for remote connections, opening ports, closing ports, and setting up network address and port translation of received data packets. This design provides additional security functionality heretofore unavailable with an unmonitored hardware firewall solution.

[0132] In a preferred embodiment, one of the VPCs created by host controller VPC 120 is a fileserver. Fileserver VPC preferably contains databases used by a client/server application, and may employ communication protocols incompatible with TCP/IP. Access to the fileserver VPC is achieved by client device 170/180 through a selected VPC. In this manner, the client/server paradigm is virtualized in servers 110/150, and the client side user interface is presented to the user through remote user interface 175/185. Even a TCP/IP incompatible client/server application becomes an Internet-ready application without rewriting a single line of application source code!

[0133] In a preferred embodiment, fileserver VPC has a Linux OS with an installed Samba server. Alternatively, fileserver VPC may have a Windows XP/XP embedded/2000/2003 operating system, depending upon the requirements of the different applications that run on the other VPCs. In another variation, fileserver VPC may be, e.g., a database server, or a hardware-device server, for example, a fax server, modem server, or an IP telephony server. In addition, fileserver VPC and the other application VPCs may comprise different versions of guest process manager 330.

[0134] In a preferred embodiment, image 115 is stored at Mission Control 190. Typically, image 115 is delivered to master virtual computer server 110 through Internet 148 via communication lines 145, but image 115 may also be installed from media (depicted in Fig. 9 below) in servers 110/150.

[0135] Fig. 6 is a flowchart that depicts a preferred method of deploying and remotely accessing a virtual computer. VPC 130/160's generation is orchestrated by commands

generated from host control VPC 120 issued to remote process managers 125 and 155 in master virtual computer server 110 or slave virtual computer server 150 respectively.

[0136] As shown in Fig. 6, in step 610, a VPC image is created. In a preferred environment, a technician at Mission Control 190 generates VPC images from specifications required to run application program 340. Alternatively, an image author may create a VPC image.

[0137] In step 620, master virtual computer server 110 clones said image to create a VPC. The VPC is modified by instructions found in a package, described below, and by runtime parameters. In a preferred embodiment, host controller VPC 120 checks that each VPC shares the same base virtual hard drive, and configures the VPC to write changes to a new virtual hard drive. Preferably, during the boot process, key differentiation information passed directly into the VPC from cloning step 620 causes the VPC to brand itself in accordance with configuration parameters such as a unique MAC address, a unique computer name, and a unique IP address. Preferably, the computer name is randomly generated by the VPC during boot, but the name can also be assigned by host controller program 390. Preferably, the IP address is assigned by a firewall VPC through DHCP.

[0138] In step 630, the VPC is booted.

[0139] In step 640, the VPC is branded with unique identifying information. As explained above, this branding preferably takes place during the boot sequence.

[0140] In step 650, host controller VPC 120 checks that a sufficient number of VPCs have been generated for the plurality of VPCs 130/160. If not, steps 620 through 640 are repeated as necessary.

[0141] In step 660, in response to a request from client device 170/180, host control VPC 120 selects a VPC from the plurality of VPCs 130/160, and start up the selected VPC. Host control VPC 120 accomplishes this task by sending a command to the appropriate RPM 125/155 on master virtual computer server 110 or slave virtual computer server 150, respectively. In turn, RPMs 125/155 start the selected VPC.

[0142] In step 670, host control VPC 120 reports the IP address of the selected VPC to client device 170/180. Remote user interface 175/185 attempts to establish communications with the selected VPC. Client device 170/180 evaluates the quality of

the communication connection to the selected VPC. Host control VPC 120 presents remote control communications protocols to remote user interface 175/185, and remote user interface will choose a protocol based on the quality of the communications connection in order to create the best possible experience for the user. Preferably, the user may choose to override the automatic selection, based on user preferences such as responsiveness, picture quality, or bandwidth.

Mission Control Architecture and Operation

[0143] Fig. 7A is a block diagram that illustrates a collection of servers and data storage structures known as Mission Control 190. Mission Control 190 comprises a collection of fileservers and databases used in virtual computer server system 100. Fig. 7A illustrates an exemplary embodiment of Mission Control 190, but one with skill in the art would understand that Mission Control may comprise many different combinations of fileservers 710, encrypted fileservers 720 and databases 730, that provide secure web services.

[0144] In a preferred embodiment, as shown in Fig. 7A, fileserver 710 is connected by communication link 145 to Internet 148. Fileserver 710 also comprises a database 730 containing a package 750. Preferably, fileserver 710 is connected to an encrypted fileserver 720.

[0145] In a preferred embodiment, fileserver 710 provides copies of remote user interfaces 175/185, which are downloaded and installed by client devices 170/180.

[0146] In a preferred embodiment, Mission Control 190 comprises an encrypted fileserver 720 that is used to store files and other information received from master virtual computer server 110 and slave virtual computer server 150 through said communication lines 145, network 140, and Internet 148.

[0147] In a preferred embodiment, database 730 contains one or more packages 750 and other information used to configure and maintain master virtual computer server 110 and slave virtual computer server 150. One skilled in the art will also appreciate that, for security reasons, any databases, systems, or components of the present invention may consist of any combination of databases or components at a single location or at multiple locations, wherein each database or system includes any of various suitable security features, such as firewalls, access codes, encryption, de-encryption, compression, decompression, and/or the like.

[0148] One primary purpose of Mission Control 190 is to generate one or more packages 750 containing images and instructions. As shown in Fig. 7B, package 750 comprises one or more images 751, 752 and instructions 755, 756. Images 751, 752 define VPCs described above. Instructions 755, 756 provide the necessary steps to install package 750 on servers 110/150. Preferably, instructions 755, 756 are written in a structured language such as XML.

[0149] Fig. 8 is a flowchart that depicts a preferred embodiment of remotely creating one or more software simulated computers. As shown in Fig. 8, in step 810, one or more VPC images are created at Mission Control 190.

[0150] Next, in step 820, images are packaged together along with instructions for installing the images. These instructions include dependencies between various VPCs that are created on master virtual computer server 110 and slave virtual computer server 150.

[0151] In step 830, package 750 is encrypted. Encryption is not a requirement, but in a preferred embodiment, encryption offers an additional level of security when transmitting package 750 across an insecure data network such as Internet 148.

[0152] In step 840, package 750 is delivered to master virtual computer server 110 or slave virtual computer server 150. The delivery method may take the form of an electronic transmission, or package 750 may be recorded on media 1001 et seq., described below, and installed locally from media onto master virtual computer server 110 or slave virtual computer server 150.

[0153] In step 850, master virtual computer server 110 or slave virtual computer server 150 extracts images 751, 752 from package 750.

[0154] In step 860, new VPCs are created from VPC images 751, 752 contained in package 750 in accordance with instructions 755, 756.

[0155] Booting up multiple VPCs is a complex process that must be carefully orchestrated by host controller VPC 120. Consequently, certain events must be successfully achieved before subsequent events are embarked. These dependencies are defined in instructions 755, 756 contained in package 750. XML instructions 755, 756 are scripts that orchestrate the complex dependencies involve with generating and maintaining VPCs 130/160.

[0156] As an example, consider the exemplary XML instruction listing for a watchdog process illustrated in Figs. 9A and B. Figs. 9A and B show instructions that cause RPMs 125/155 to perform the following tasks: (1) start a VMManager process that runs until 3 a.m.; (2) start a background downloader process that runs until 3 a.m.; (3) start an LCD Manager process that runs permanently; (4) restart any of the aforementioned processes if any one of them fails; (5) shutdown every process at 3:01 a.m.; (6) run an internal backup; and (7) restart a server 110/150.

[0157] In addition, Mission Control 190 is used to monitor connections of remote client devices 170/180 with master virtual computer server VPC 130 and slave virtual computer server VPC 160. Mission Control 190 maintains status of every connection for every VPC 130/160 and client device 170/180.

[0158] Finally, Mission Control 190 serves as a repository for storing a back up of master virtual computer server 110 and slave virtual computer server 150. Alternatively, the backup is stored on master virtual computer server 110 and slave virtual computer server 150.

[0159] In either backup storage case, a backup utility program is invoked by remote process manager 125, typically, once every evening. Preferably, the backup utility is executed after host controller VPC 120 has shut itself down. The utility checks for available storage space, and removes old backup copies as necessary. The backup utility generates a backup copy of the simulated storage devices for each VPC 130/160. In a preferred embodiment, the utility keeps one copy per day for a week, one copy per week for a month, one copy per month for a year, and an annual copy for as many years as storage space permits. After the backup utility has completed, remote process manager 125 starts a full warm reboot of master virtual computer server 110 and slave virtual computer server 150.

Delivery of Packages and Software on Media

[0160] In the specification, the term "media" means any medium that can record data therein. Examples of a recording medium are illustrated in Fig. 10.

[0161] The term "media" includes, for instance, a disk shaped media for 1001 such as CD-ROM (compact disc-read only memory), magneto optical disc or MO, digital video disc-read only memory or DVD-ROM, digital video disc-random access memory or DVD-

RAM, a floppy disc 1002, a memory chip 1004 such as random access memory or RAM, read only memory or ROM, erasable programmable read only memory or E-PROM, electrical erasable programmable read only memory or EE-PROM, a rewriteable card-type read only memory 1005 such as a smart card, a magnetic tape, a hard disc 1003, and any other suitable means for storing a program therein.

[0162] A recording media storing a program for accomplishing the above mentioned apparatus maybe accomplished by programming functions of the above mentioned apparatuses with a programming language readable by a computer 1000 or processor, and recording the program on a media such as mentioned above.

[0163] A server equipped with a hard disk drive may be employed as a recording media. It is also possible to accomplish the present invention by storing the above mentioned computer program on such a hard disk in a server and reading the computer program by other computers through a network.

[0164] As a computer processing device 1000, any suitable device for performing computations in accordance with a computer program may be used. Examples of such devices include a personal computer, a laptop computer, a microprocessor, a programmable logic device, or an application specific integrated circuit.

[0165] Having thus described at least illustrative embodiments of the invention, various modifications and improvements will readily occur to those skilled in the art and are intended to be within the scope of the invention. Accordingly, the foregoing description is by way of example only and is not intended as limiting. The invention is limited only as defined in the following claims and the equivalents thereto.

CLAIMS

What is claimed is:

1. A method of deploying and remotely accessing a plurality of software-simulated computers, comprising:

creating a software-simulated computer image, said image comprising:

simulated hardware device specifications;

a bootable application;

a guest process manager, and;

one or more application programs;

cloning said image to create said plurality of software-simulated computers;

branding each software simulated computer in said plurality with unique, machine-differentiation information;

selecting a software-simulated computer in said plurality; and

establishing communications for remote access across a network to said selected software-simulated computer.

2. The method of claim 1 wherein said step of branding prevents communication conflicts between machines on said network.

3. The method of claim 2 wherein said bootable application is a Windows variant, and said machine differentiation information includes a system identifier.

4. The method of claim 1, further comprising:

loading user specific information into said image before performing said cloning step.

5. The method of claim 4 wherein said user specific information comprises software license numbers.

6. The method of claim 4 wherein said user specific information includes one or more of company names and individual names.

7. The method of claim 4 wherein said user specific information includes user identifiers and associated passwords.

8. The method of claim 1 wherein said branding prevents conflicts between machines on said network.

9. The method of claim 1 further comprising booting each software-simulated computer.

10. The method of claim 1 further comprising evaluating quality of said communications and selecting a remote control communications protocol based on said quality.

11. The method of claim 10, further comprising accessing said selected software-simulated computer through a remote user interface.

12. The method of claim 10, further comprising configuring a firewall to permit communications with said selected software-simulated computer.

13. A method of creating one or more software-simulated computers on a remote computer, comprising:

creating one or more software-simulated computer images;

generating a package comprising said images; and

delivering said package to said remote computer, wherein said remote computer extracts said images and automatically creates said software-simulated computers.

14. The method of claim 13, wherein said package includes an xml document comprising installation instructions and said remote computer generates said software-simulated computers in accordance with said instructions.

15. The method of claim 13, wherein said package is encrypted.

16. The method of claim 13, wherein each of said images comprises:

simulated hardware device specifications;

a bootable application;

a guest process manager, and;

one or more application programs.

17. The method of claim 13, wherein one of said images represents a server computer.

18. The method of claim 13, wherein one of said images represents a firewall computer.

19. A software-simulated computer server for providing a client device access to an application program on a software-simulated computer through a network, comprising:

one or more hardware computers;

an image that defines a software-simulated computer having a copy of said application program stored thereon; and

a host control program that causes said one or more hardware computers to create a plurality of software-simulated computers from said image and to generate unique, machine-differentiation information for each software-simulated computer in said plurality;

wherein said host control program further causes said hardware computers to select a software-simulated computer from said plurality, to negotiate a

communications connection between said selected software-simulated computer and said client device, and to enable said client device to access said application program running on said selected software-simulated computer through said network.

20. The software-simulated computer server of claim 19, wherein said host control program causes said one or more hardware computers to shutdown, recreate, and restart said plurality of software-simulated computers.

21. The software-simulated computer server of claim 20, wherein said host control program further causes said one or more hardware computers to copy user generated data to backup storage.

22. The software-simulated computer server of claim 21, wherein said host control program recreates and restarts said plurality of software-simulated computers after said user generated data has been copied.

23. The software-simulated computer server of claim 19 wherein said host control program responds to client device requests using one or more communication protocols from a list comprising FTP, HTTP, HTTPS, MPLS, SFTP, SMTP, and SSH.

24. The software-simulated computer server of claim 23 wherein said application program is designed to be used on a single personal computer.

25. The software-simulated computer server of claim 23 wherein said application program is a client/server application.

26. The software-simulated computer server of claim 25 wherein said application program uses one or more communication protocols from a list consisting of IPX/SPX, netbios, raw IP sockets, UDP/IP, TCP/IP, IPv6, IPSEC, HTTP, and netbeui.

27. The software-simulated computer server of claim 19, wherein said host control program balances load on said hardware computers when making said selection of said software-simulated computer.

28. The software-simulated computer server of claim 27, wherein said load is determined by one or more of available memory, processor utilization, and a number of unused software-simulated computers.

29. The software-simulated computer server of claim 19, wherein said software-simulated computer is adapted to accept and communicate with and to provide concurrent interaction of said application with more than one client device.

30. The software-simulated computer server of claim 19, wherein said software-simulated computer is adapted to record user input from said client device.

31. The software-simulated computer server of claim 19, further comprising a control center computer in communication with a host control virtual computer, wherein said control center computer transmits said image for said software simulated computer to said host control virtual computer.

32. The software-simulated computer server of claim 31, wherein said control center computer receives status information about a software-simulated computer.

33. The server appliance system of claim 31, wherein said control center computer is adapted to issue a reboot command that causes a particular software-simulated computer to be shutdown, recreated, and restarted.

34. The server appliance system of claim 31, wherein said control center computer transmits an updated image comprising an updated copy of said application program and said host control program causes said one or more hardware computers to shutdown, recreate using said updated image, and restart said plurality of software-simulated computers.

35. A media storing a computer program that causes a processor that executes said program to perform a method of deploying and remotely accessing a plurality of software-simulated computers, the steps comprising:

cloning an image that defines a software simulated computer to create said plurality of software-simulated computers;

branding each software simulated computer in said plurality with unique, machine-differentiation information;

selecting a software-simulated computer in said plurality; and

establishing communications for remote access across a network to said selected software-simulated computer.

36. The media of claim 35, wherein said computer program causes said processor to perform steps comprising:

loading user specific information into said image before performing said cloning step.

37. The media of claim 35, wherein said computer program causes said processor to perform steps comprising:

evaluating quality of said communications and selecting a remote control client based on said quality.

38. The media of claim 35, wherein said computer program causes said processor to perform steps comprising:

configuring a firewall to permit communications with said selected software-simulated computer.

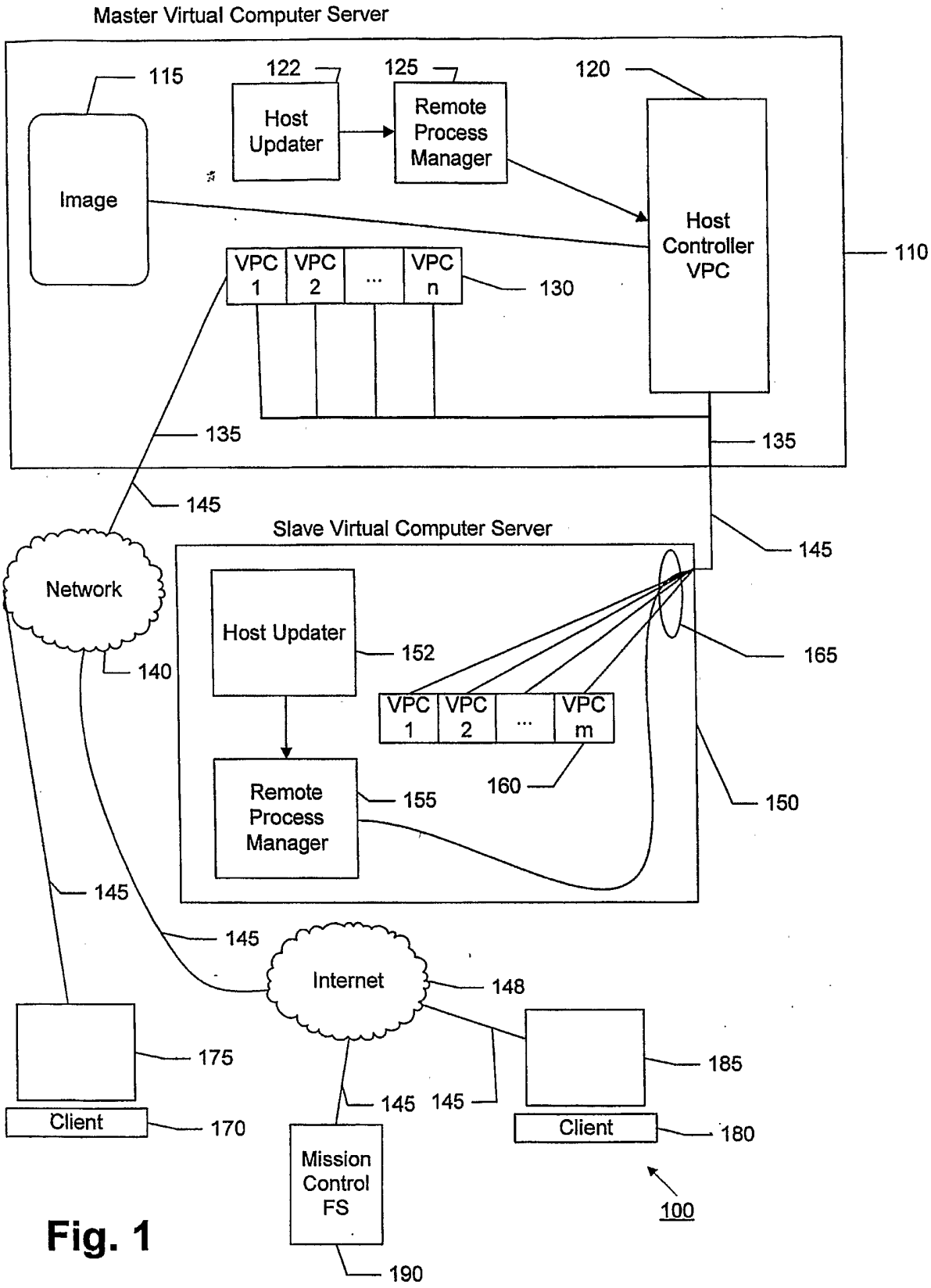

Master Virtual Computer Server
115
122
125
120
Host Updater
Remote Process Manager
Image
Host Controller VPC
110
VPC 1
VPC 2
...
VPC n
130
135
135
145
Slave Virtual Computer Server
145
Network
Host Updater
152
165
140
VPC 1
VPC 2
...
VPC m
160
Remote Process Manager
155
150
145
145
Internet
148
175
185
145
145
Client
170
Client
180
Mission Control FS
100
190

Fig. 1

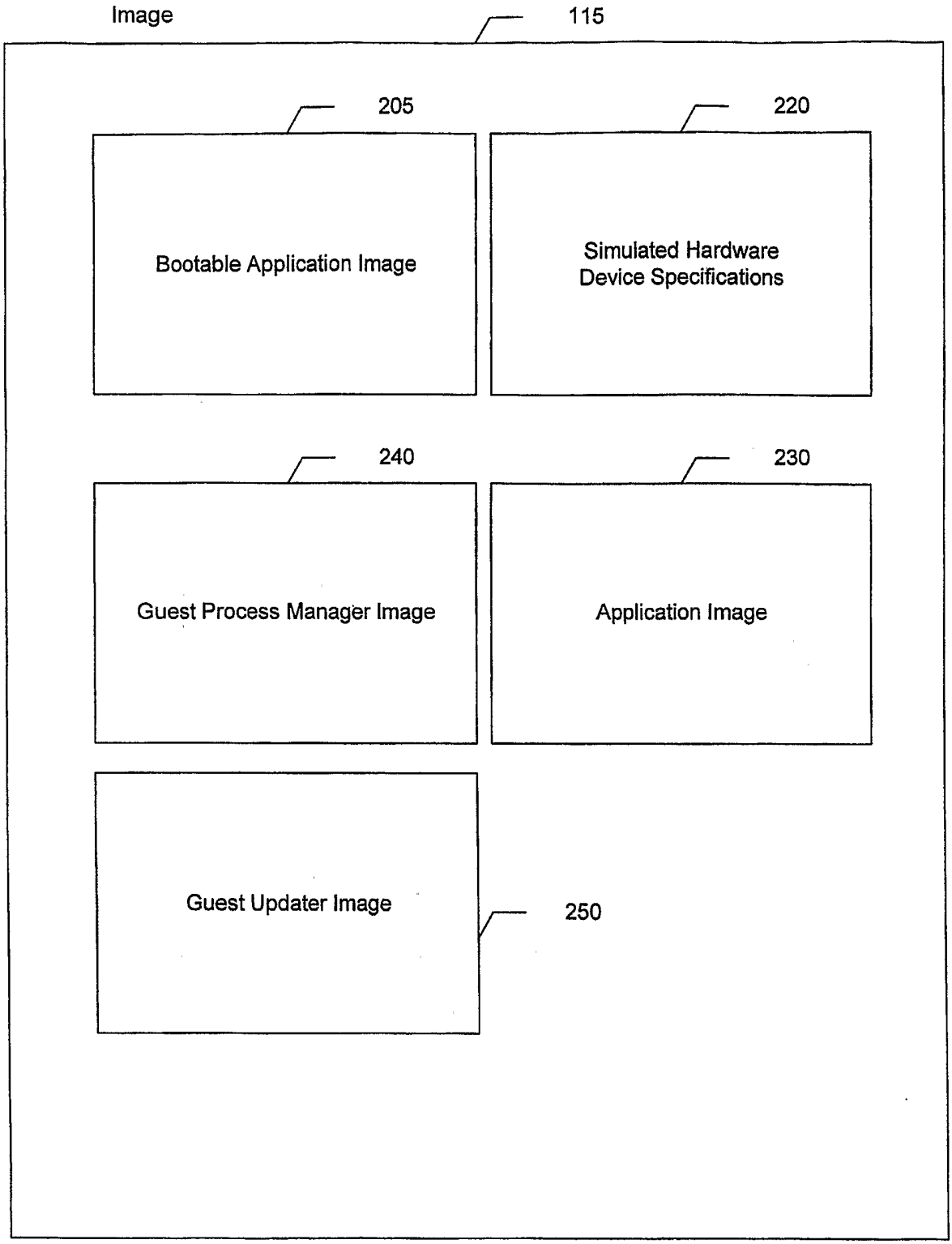


Image
115
205
Bootable Application Image
220
Simulated Hardware Device Specifications
240
Guest Process Manager Image
230
Application Image
Guest Updater Image
250

Fig. 2

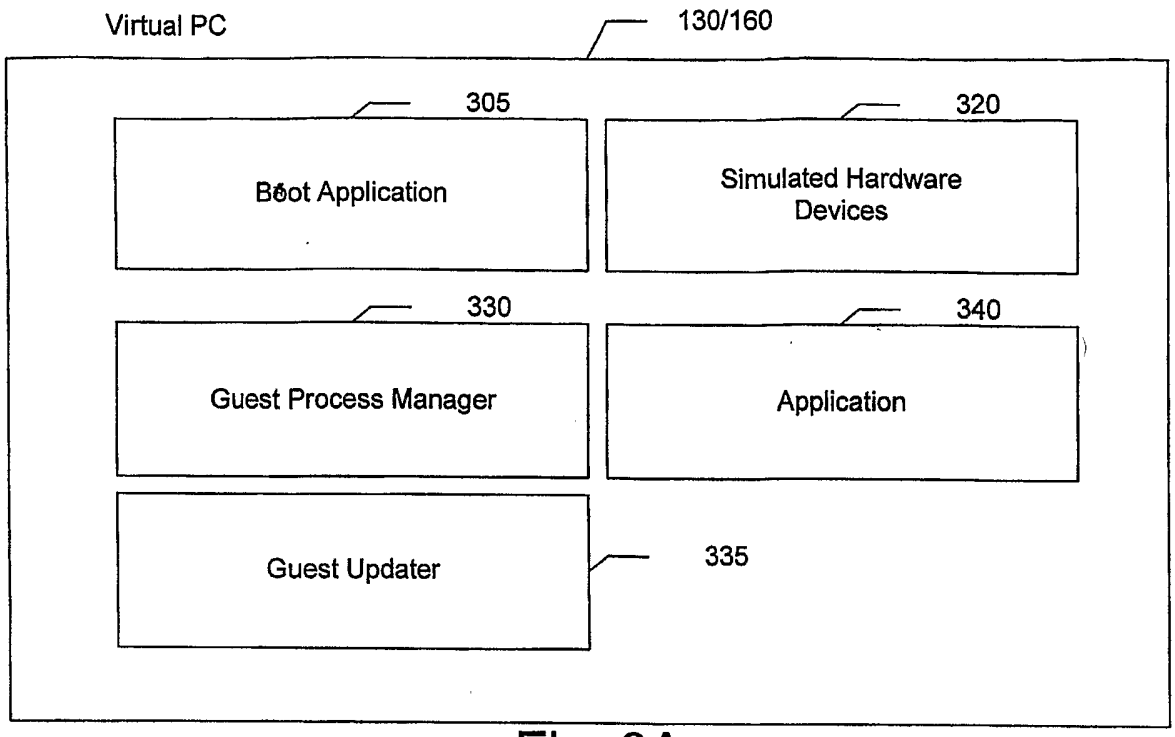

Virtual PC
130/160
305
Boot Application
320
Simulated Hardware Devices
330
Guest Process Manager
340
Application
Guest Updater
335

Fig. 3A

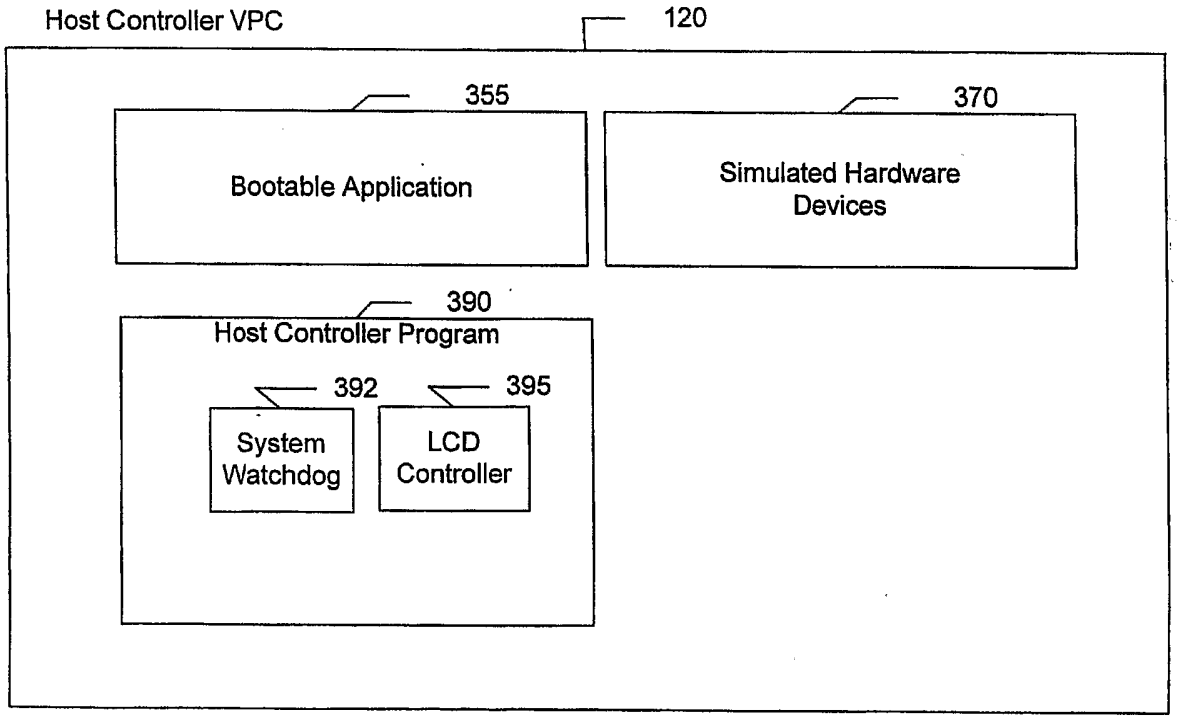

Host Controller VPC
120
355
Bootable Application
370
Simulated Hardware Devices
390
Host Controller Program
392
System Watchdog
395
LCD Controller

Fig. 3B

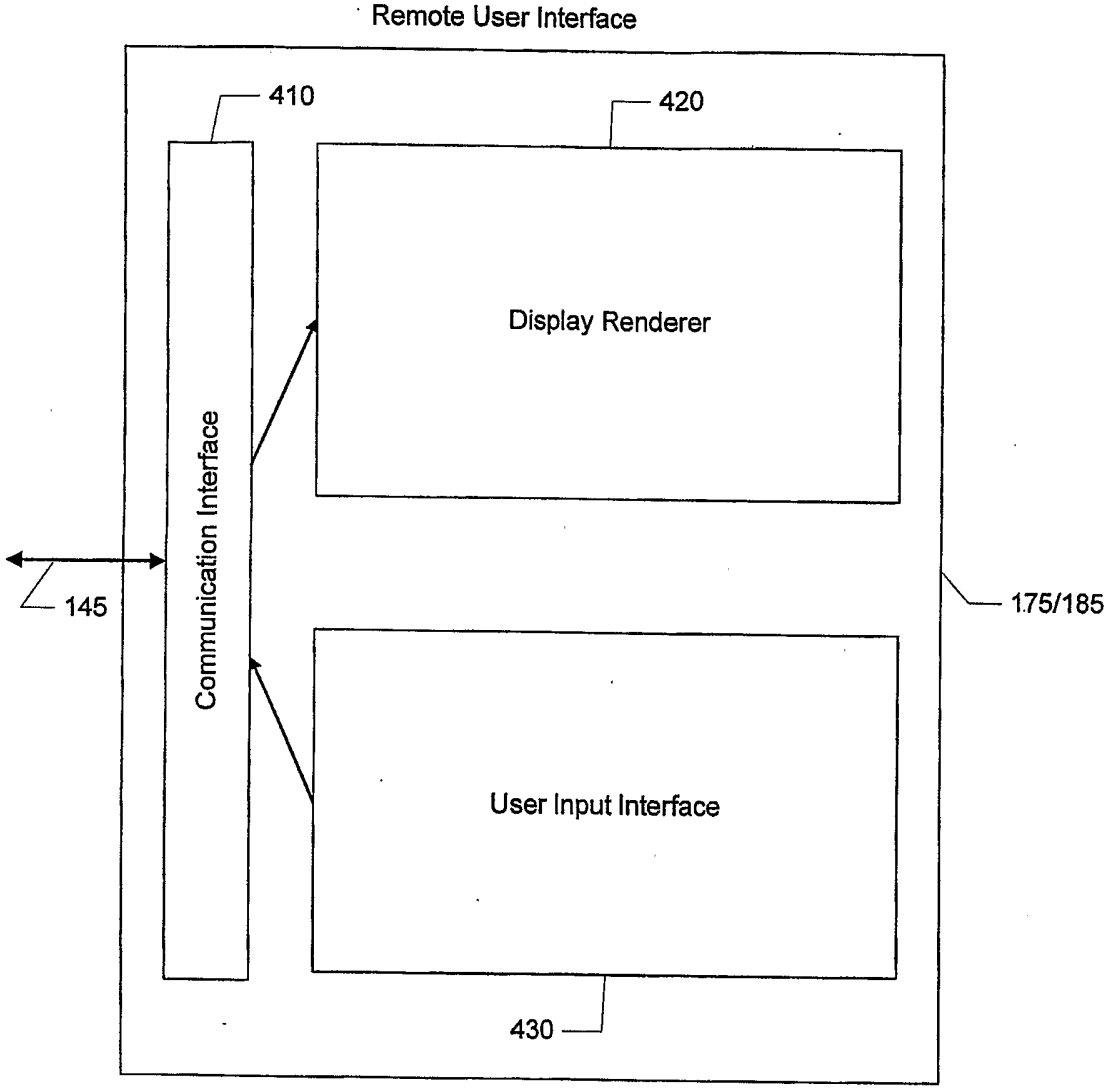

Remote User Interface
410
420
Display Renderer
Communication Interface
145
175/185
User Input Interface
430

Fig. 4

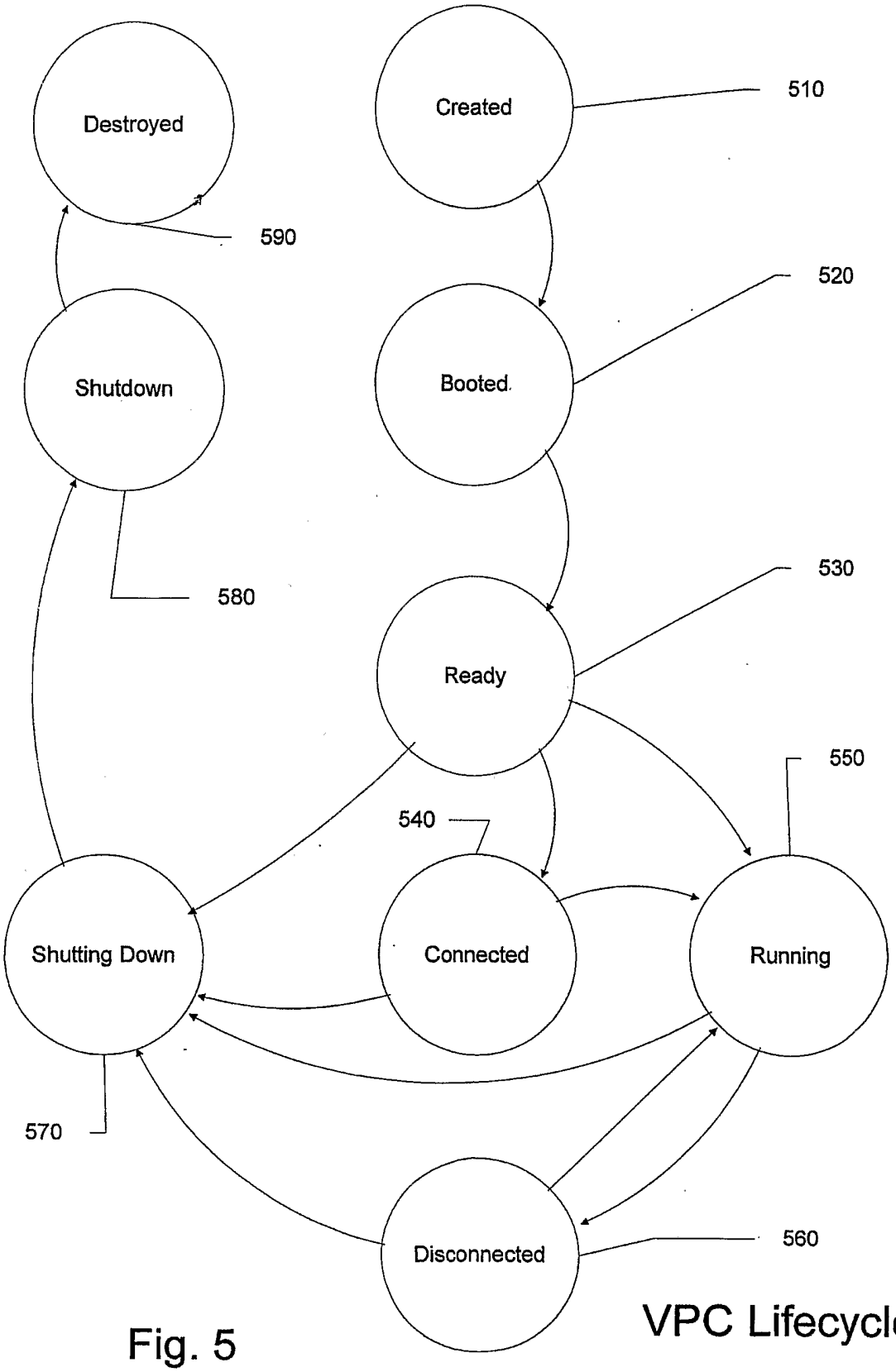

Destroyed
Created
510
590
520
Shutdown
Booted
530
580
Ready
550
540
Shutting Down
Connected
Running
570
Disconnected
560
VPC Lifecycle

Fig. 5

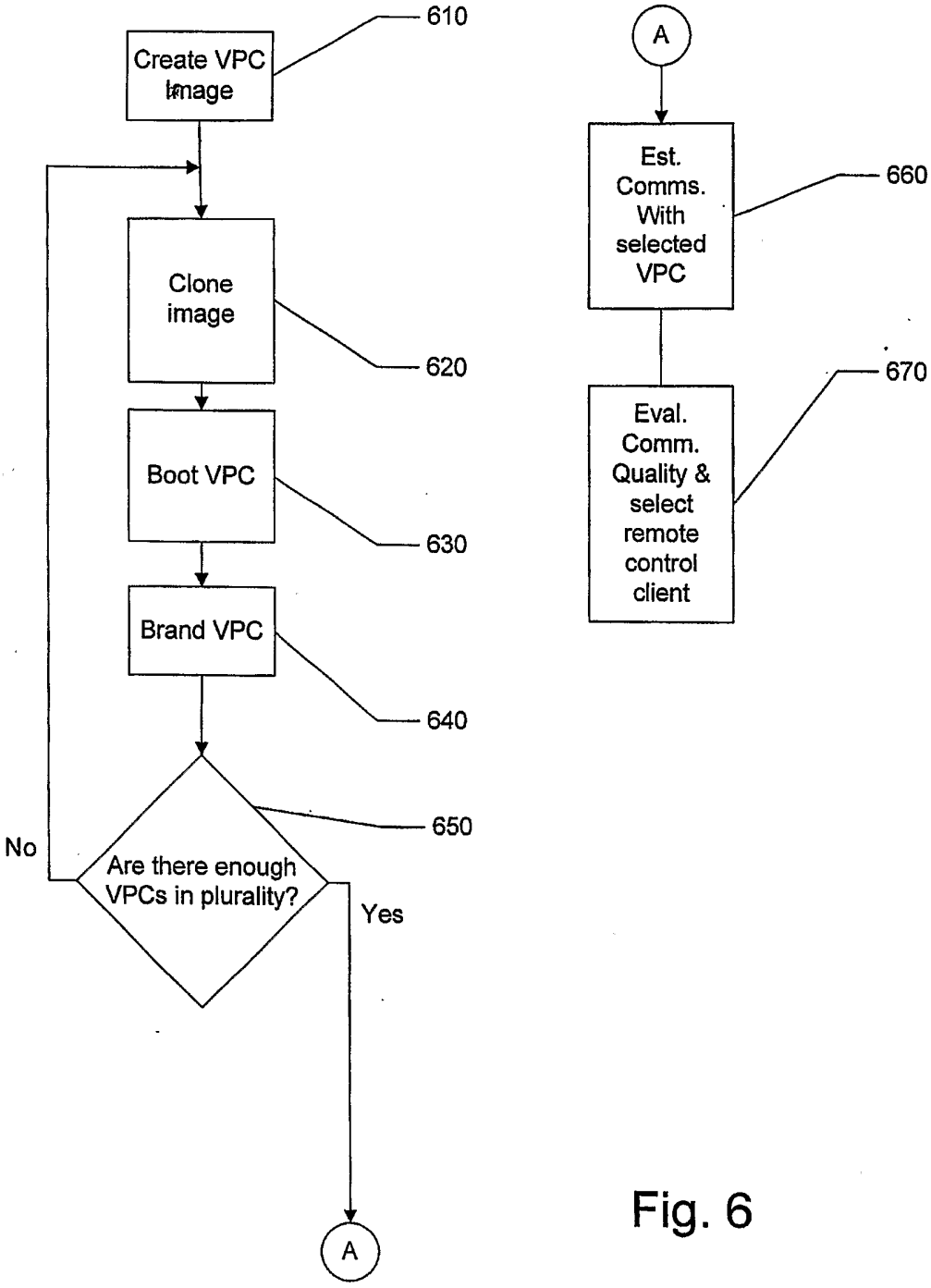

Create VPC Image
610
Clone image
620
Boot VPC
630
Brand VPC
640
Are there enough VPCs in plurality?
650
No
Yes
A
A
Est. Comms. With selected VPC
660
Eval. Comm. Quality & select remote control client
670

Fig. 6

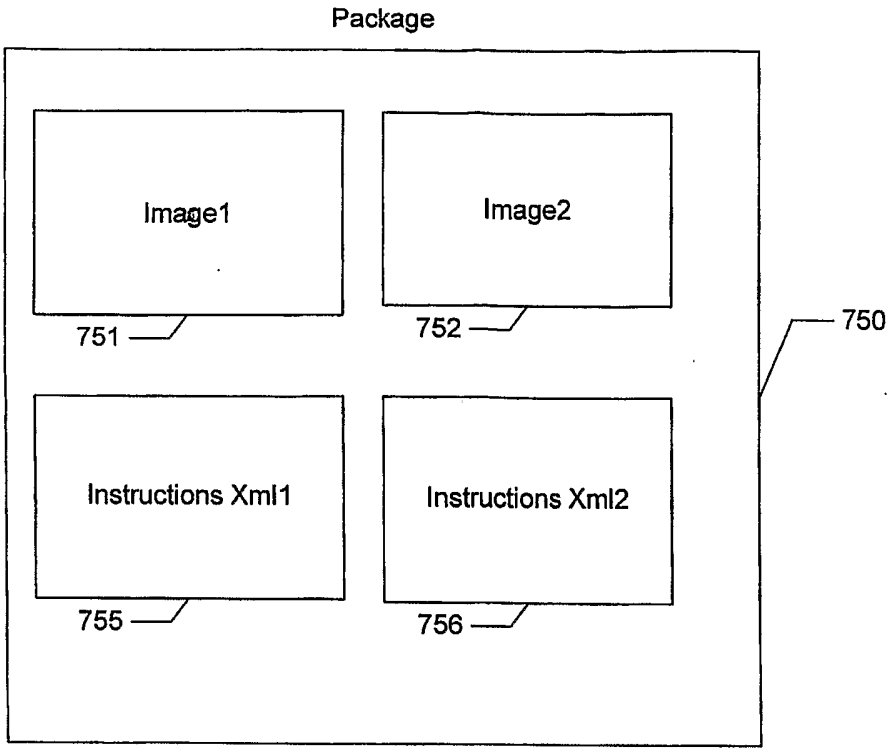

Package
Image1
Image2
751
752
750
Instructions Xml1
Instructions Xml2
755
756

Fig. 7B

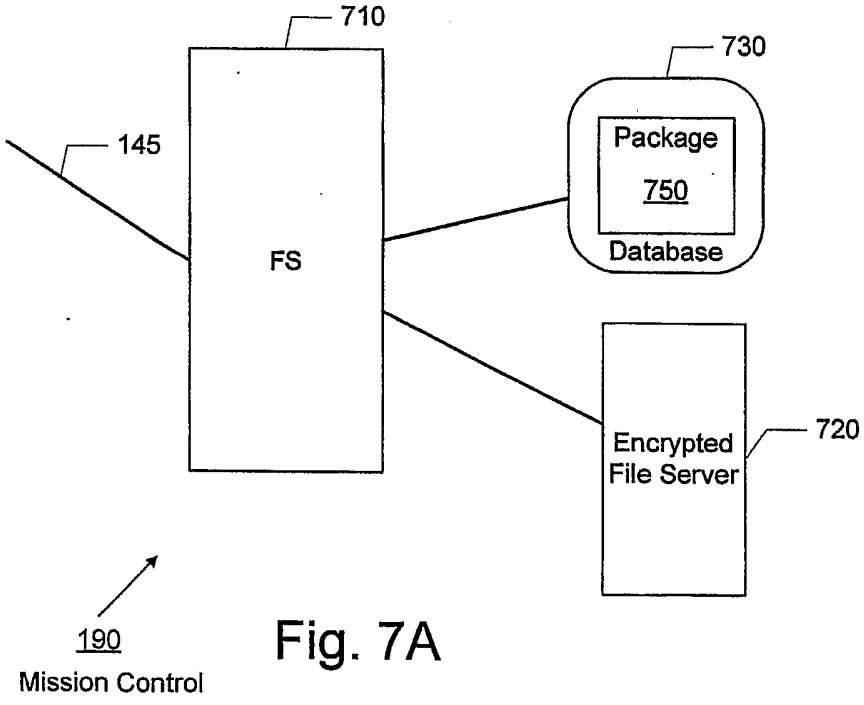

710
730
145
Package
750
Database
FS
Encrypted
File Server
720
190
Mission Control

Fig. 7A

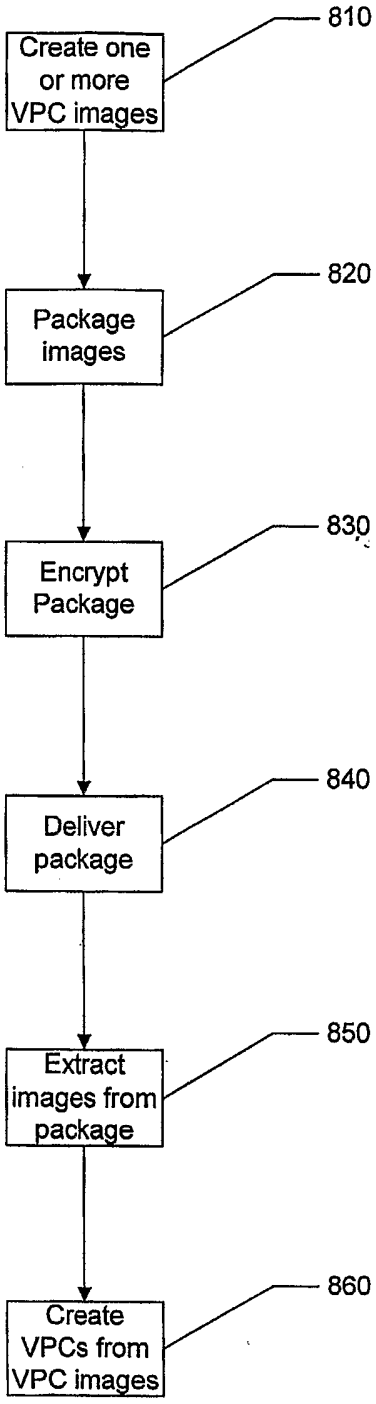

Create one or more VPC images
810
Package images
820
Encrypt Package
830
Deliver package
840
Extract images from package
850
Create VPCs from VPC images
860

Fig. 8

```
<?xml version="1.0" encoding="utf-8" ?>
- <configuration guid="6529C140-F8ED-4F39-964C-07027381A7B0">
  - <watchdog>
      <pingEvery>1</pingEvery>
    </watchdog>
  - <processes>
    - <process id="Ping" applianzProcess="false">
      - <execution autostart="true">
          <path>cmd</path>
          <shutdown type="FORCE" />
        - <parameters>
            <param>/c ping 127.0.0.1 -n 60</param>
          </parameters>
        </execution>
      </process>
    - <process id="ApplianzVMManager" applianzProcess="true">
      - <execution autostart="true">
          <path>C:\program
            files\applianz\Host\ApplianzVMManager.exe</path>
          <shutdown type="API" />
        - <dependencies>
            <dependOn type="CLOSED">Ping</dependOn>
          </dependencies>
        - <parameters>
            <param>2:00AM</param>
          </parameters>
        </execution>
      </process>
    - <process id="Backup" applianzProcess="false">
      - <execution autostart="true">
          <path>C:\program files\applianz\Host\backup.exe</path>
          <shutdown type="API" />
        - <dependencies>
            <dependOn type="CLOSED">ApplianzVMManager</dependOn>
          </dependencies>
        - <parameters>
            <param>10000 "C:\Program Files\Applianz\vms\applianzlinux"
              "C:\Program Files\Applianz\vms\QB"</param>
            <!-- first param is the max megs and the rest are just files
            to backup  -->
          </parameters>
        </execution>
      </process>
    - <process id="AdminCopy" applianzProcess="false">
      - <execution autostart="true">
          <path>xcopy</path>
          <shutdown type="API" />
        - <dependencies>
            <dependOn type="CLOSED">Backup</dependOn>
          </dependencies>
        - <parameters>
            <param>"C:\Program Files\Applianz\vms\QBAdmin\*.*"
```

FIG. 9A

```
      "C:\Program Files\Applianz\vms\QB" /Y</param>
      </parameters>
    </execution>
  </process>
- <process id="Shutdown" applianzProcess="false">
  - <execution autostart="true">
      <path>shutdown</path>
      <shutdown type="API" />
    - <dependencies>
        <dependOn type="CLOSED">AdminCopy</dependOn>
      </dependencies>
    - <parameters>
        <param>-r -f -t 0</param>
      </parameters>
    </execution>
  </process>
- <process id="LCDController" applianzProcess="true">
  - <execution autostart="true">
      <path>C:\program files\applianz\Host\lcdcontroller.exe</path>
      <shutdown type="API" />
    - <dependencies>
        <dependOn type="CLOSED">Ping</dependOn>
      </dependencies>
    </execution>
  </process>
- <process id="CommandPrompt" applianzProcess="false">
  - <execution autostart="true">
      <path>cmd</path>
      <shutdown type="FORCE" />
    - <dependencies>
        <dependOn type="CLOSED">Ping</dependOn>
      </dependencies>
    </execution>
  </process>
 </processes>
</configuration>
```

FIG. 9B

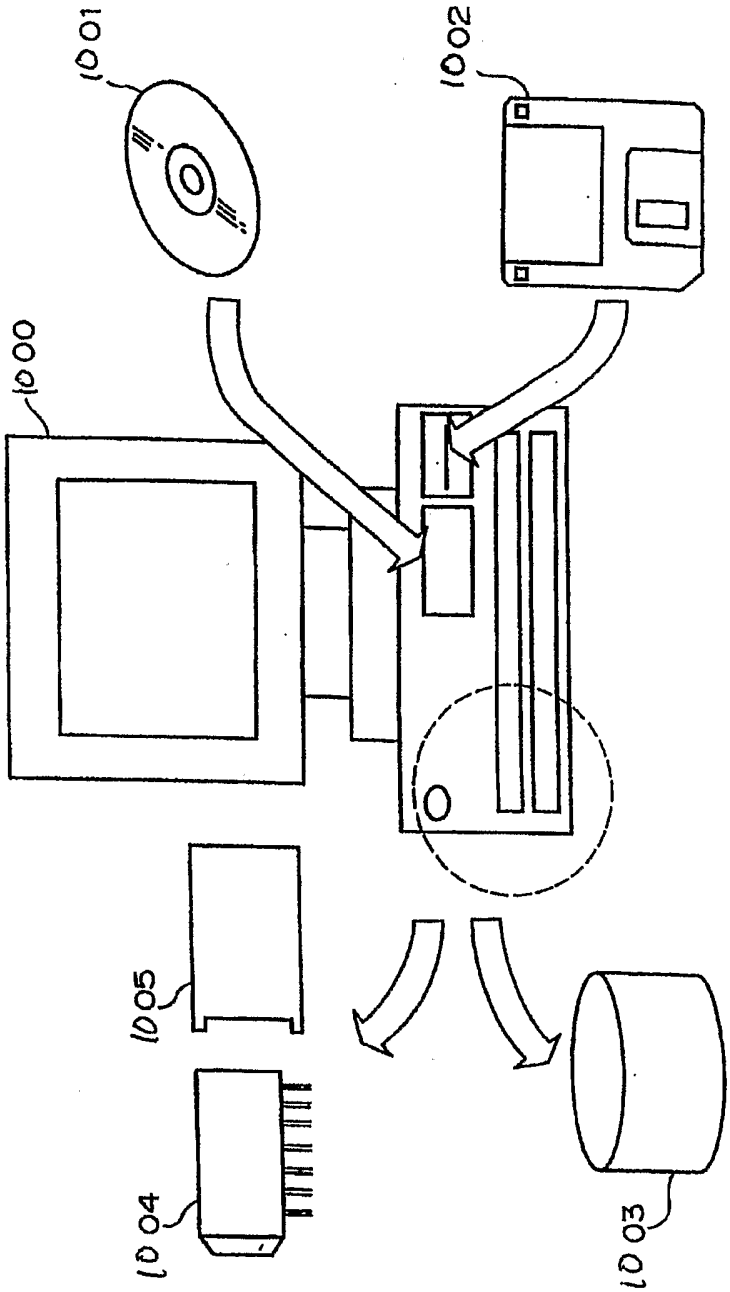


FIG. 10 MEDIA

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US04/15077

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 9/44

US CL : 717/134, 135, 138, 703/13, 22

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)

U.S. : 717/134, 135, 138, 703/13, 22

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 practicable, search terms used)

WEST, IEEE online, ACM online

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,313,615 A (NEWMAN et al) 17 May 1994 (17.05.1994), col. 4 line 55 to col. 7 line 30.	13-18
A	US 6,760,630 B2 (TURNAUS) 06 July 2004 (06.07.2004).	1-38
A	US 6,708,329 B1 (WHITEHILL et al) 16 March 2004 (16.03.2004).	1-38

☐ Further documents are listed in the continuation of Box C. ☐ See patent family 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 application or patent 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

19 August 2004 (19.08.2004)

Date of mailing of the international search report

10 SEP 2004

Name and mailing address of the ISA/US

Mail Stop PCT, Attn: ISA/US
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

Facsimile No. (703)305-3230

Authorized officer

Tuan Dam

Telephone No. (703) 305-9600

Form PCT/ISA/210 (second sheet) (January 2004)