

Under the Paperwork Reduction Act of 1995 no persons are required to respond to a collection of information unless it displays a valid OMB control number

UTILITY PATENT APPLICATION TRANSMITTAL <i>(Only for new nonprovisional applications under 37 CFR 1.53(b))</i>	Attorney Docket No.	FIN0001-CON1-CIP1-CON7-DIV1
	First Named Inventor	Yigal Mordechai EDERY, et al.
	Title	Malicious Mobile Code Runtime Monitoring System and Methods
	Express Mail Label No.	

APPLICATION ELEMENTS <i>See MPEP chapter 600 concerning utility patent application contents.</i>	Commissioner for Patents P.O. Box 1450 Alexandria, VA 22313-1450
1. ☐ Fee Transmittal Form (PTO/SB/17 or equivalent) 2. ☐ Applicant asserts small entity status. See 37 CFR 1.27 3. ☐ Applicant certifies micro entity status. See 37 CFR 1.29. Applicant must attach form PTO/SB/15A or B or equivalent. 4. ☒ Specification [Total Pages <u>35</u>] Both the claims and abstract must start on a new page. (See MPEP § 608.01(a) for information on the preferred arrangement) 5. ☒ Drawing(s) (35 U.S.C. 113) [Total Sheets <u>10</u>] 6. Inventor's Oath or Declaration [Total Pages <u> </u>] (including substitute statements under 37 CFR 1.64 and assignments serving as an oath or declaration under 37 CFR 1.63(e)) a. ☐ Newly executed (original or copy) b. ☐ A copy from a prior application (37 CFR 1.63(d)) 7. ☒ Application Data Sheet * See note below. See 37 CFR 1.76 (PTO/AIA/14 or equivalent) 8. CD-ROM or CD-R in duplicate, large table, or Computer Program (Appendix) ☐ Landscape Table on CD 9. Nucleotide and/or Amino Acid Sequence Submission (if applicable, items a. – c. are required) a. ☐ Computer Readable Form (CRF) b. ☐ Specification Sequence Listing on: i. ☐ CD-ROM or CD-R (2 copies); or ii. ☐ Paper c. ☐ Statements verifying identity of above copies	ACCOMPANYING APPLICATION PAPERS 10. ☐ Assignment Papers (cover sheet & document(s)) Name of Assignee _____ 11. ☐ 37 CFR 3.73(c) Statement ☐ Power of Attorney (when there is an assignee) 12. ☐ English Translation Document (if applicable) 13. ☐ Information Disclosure Statement (PTO/SB/08 or PTO-1449) ☐ Copies of citations attached 14. ☐ Preliminary Amendment 15. ☐ Return Receipt Postcard (MPEP § 503) (Should be specifically itemized) 16. ☐ Certified Copy of Priority Document(s) (if foreign priority is claimed) 17. ☐ Nonpublication Request Under 35 U.S.C. 122(b)(2)(B)(i). Applicant must attach form PTO/SB/35 or equivalent. 18. ☒ Other: Filed Electronically. _____ _____ _____ _____

***Note:** (1) Benefit claims under 37 CFR 1.78 and foreign priority claims under 1.55 **must** be included in an Application Data Sheet (ADS).
(2) For applications filed under 35 U.S.C. 111, the application must contain an ADS specifying the applicant if the applicant is an assignee, person to whom the inventor is under an obligation to assign, or person who otherwise shows sufficient proprietary interest in the matter. See 37 CFR 1.46(b).

19. CORRESPONDENCE ADDRESS					
☒ The address associated with Customer Number: <u>115222</u> OR ☐ Correspondence address below					
Name					
Address					
City		State		Zip Code	
Country		Telephone		Email	
Signature	/Dawn-Marie Bey/			Date	June 5, 2015
Name (Print/Type)	Dawn-Marie Bey			Registration No. (Attorney/Agent)	44,442

This collection of information is required by 37 CFR 1.53(b). The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

**MALICIOUS MOBILE CODE RUNTIME MONITORING
SYSTEM AND METHODS**

PRIORITY REFERENCE TO RELATED APPLICATIONS

[0001] This application is a division of U.S. patent application serial no. 14/619,363, filed February 11, 2015 by inventors Yigal Mordechai Edery, et al., entitled "Malicious Mobile Code Runtime Monitoring System and Methods," which is a continuation of U.S. patent application serial no. 14/155,835, filed January 15, 2014 by inventors Yigal Mordechai Edery, et al., entitled "Malicious Mobile Code Runtime Monitoring System and Methods," which is a continuation of U.S. patent application serial no. 13/290,708, filed November 7, 2011 by inventors Yigal Mordechai Edery, et al., entitled "Malicious Mobile Code Runtime Monitoring System and Methods," which is a continuation of U.S. patent application serial no. 12/471,942, filed May 26, 2009 by inventors Yigal Mordechai Edery, et al., now U.S. Patent No. 8,079,086, entitled "Malicious Mobile Code Runtime Monitoring System and Methods," which is a continuation of assignee's U.S. patent application serial no. 11/370,114, filed March 7, 2006 by inventors Yigal Mordechai Edery, et al., now U.S. Patent No. 7,613,926, entitled "Method and System for Protecting a Computer and a Network from Hostile Downloadables," which is a continuation of assignee's U.S. patent application serial no. 09/861,229, filed on May 17, 2001 by inventors Yigal Mordechai Edery, et al., now U.S. Patent No. 7,058,822, entitled "Malicious Mobile Code Runtime Monitoring System And Methods," all of which are hereby incorporated by reference. U.S. patent application serial no. 09/861,229, now U.S. Patent No. 7,058,822, claims benefit of provisional U.S. patent application serial no. 60/205,591, entitled "Computer Network Malicious Code Run-Time Monitoring," filed on May 17, 2000 by inventors Nimrod Itzhak Vered, et al., which is hereby incorporated by reference. U.S. patent application serial no. 09/861,229, now U.S. Patent No. 7,058,822, is also a Continuation-In-Part of assignee's U.S. patent application serial no. 09/539,667, entitled "System and Method for Protecting a Computer and a Network From Hostile Downloadables," filed on March 30, 2000 by inventor Shlomo Touboul, now U.S. Patent No. 6,804,780, and hereby incorporated by reference, which is a continuation of assignee's U.S. patent application serial no. 08/964,388, filed on November 6, 1997 by inventor Shlomo Touboul, now U.S. Patent No. 6,092,194, also entitled "System and Method for Protecting a Computer and a

Network from Hostile Downloadables" and hereby incorporated by reference, which application claims the benefit of provisional U.S. application serial no. 60/030,639, filed November 8, 1996 by inventors Shlomo Touboul, entitled "System and Method For Protecting a Computer From Hostile Downloadables." U.S. Serial No. 09/861,229, now U.S. Patent No. 7,058,822, is also a Continuation-In-Part of assignee's U.S. patent application serial no. 09/551,302, entitled "System and Method for Protecting a Client During Runtime From Hostile Downloadables," filed on April 18, 2000 by inventor Shlomo Touboul, now U.S. Patent No. 6,480,962, which is hereby incorporated by reference, which is a continuation of U.S. application serial no. 08/790,097, filed January 29, 1997 by inventor Shlomo Touboul, now U.S. Patent No. 6,167,520, entitled "System and Method For Protecting a Client From Hostile Downloadables" which claims the benefit of U.S. provisional application no. 60/030,639, filed on November 8, 1996 by inventor Shlomo Touboul, entitled "System and Method For Protecting a Computer From Hostile Downloadables."

BACKGROUND OF THE INVENTION

Field of the Invention

[0002] This invention relates generally to computer networks, and more particularly provides a system and methods for protecting network-connectable devices from undesirable downloadable operation.

Description of the Background Art

[0003] Advances in networking technology continue to impact an increasing number and diversity of users. The Internet, for example, already provides to expert, intermediate and even novice users the informational, product and service resources of over 100,000 interconnected networks owned by governments, universities, nonprofit groups, companies, etc. Unfortunately, particularly the Internet and other public networks have also become a major source of potentially system-fatal or otherwise damaging computer code commonly referred to as "viruses."

[0004] Efforts to forestall viruses from attacking networked computers have thus far met with only limited success at best. Typically, a virus protection program designed to identify and remove or protect against the initiating of known viruses is installed on a network firewall

or individually networked computer. The program is then inevitably surmounted by some new virus that often causes damage to one or more computers. The damage is then assessed and, if isolated, the new virus is analyzed. A corresponding new virus protection program (or update thereof) is then developed and installed to combat the new virus, and the new program operates successfully until yet another new virus appears - and so on. Of course, damage has already typically been incurred.

[0005] To make matters worse, certain classes of viruses are not well recognized or understood, let alone protected against. It is observed by this inventor, for example, that Downloadable information comprising program code can include distributable components (e.g. Java™ applets and JavaScript scripts, ActiveX™ controls, Visual Basic, add-ins and/or others). It can also include, for example, application programs, Trojan horses, multiple compressed programs such as zip or meta files, among others. U.S. Patent 5,983,348 to Shuang, however, teaches a protection system for protecting against only distributable components including “Java applets or ActiveX controls”, and further does so using resource intensive and high bandwidth static Downloadable content and operational analysis, and modification of the Downloadable component; Shuang further fails to detect or protect against additional program code included within a tested Downloadable. U.S. Patent 5,974,549 to Golan teaches a protection system that further focuses only on protecting against ActiveX controls and not other distributable components, let alone other Downloadable types. U.S. patent 6,167,520 to Touboul enables more accurate protection than Shuang or Golan, but lacks the greater flexibility and efficiency taught herein, as do Shuang and Golan.

[0006] Accordingly, there remains a need for efficient, accurate and flexible protection of computers and other network connectable devices from malicious Downloadables.

SUMMARY OF THE INVENTION

[0007] The present invention provides protection systems and methods capable of protecting a personal computer (“PC”) or other persistently or even intermittently network accessible devices or processes from harmful, undesirable, suspicious or other “malicious” operations that might otherwise be effectuated by remotely operable code. While enabling the capabilities of prior systems, the present invention is not nearly so limited, resource intensive or inflexible, and yet enables more reliable protection. For example, remotely operable code

that is protectable against can include downloadable application programs, Trojan horses and program code groupings, as well as software “components”, such as Java™ applets, ActiveX™ controls, JavaScript™/Visual Basic scripts, add-ins, etc., among others. Protection can also be provided in a distributed interactively, automatically or mixed configurable manner using protected client, server or other parameters, redirection, local/remote logging, etc., and other server/client based protection measures can also be separately and/or interoperably utilized, among other examples.

[0008] In one aspect, embodiments of the invention provide for determining, within one or more network “servers” (e.g. firewalls, resources, gateways, email relays or other devices/processes that are capable of receiving-and-transferring a Downloadable) whether received information includes executable code (and is a “Downloadable”). Embodiments also provide for delivering static, configurable and/or extensible remotely operable protection policies to a Downloadable-destination, more typically as a sandboxed package including the mobile protection code, downloadable policies and one or more received Downloadables. Further client-based or remote protection code/policies can also be utilized in a distributed manner. Embodiments also provide for causing the mobile protection code to be executed within a Downloadable-destination in a manner that enables various Downloadable operations to be detected, intercepted or further responded to via protection operations. Additional server/information-destination device security or other protection is also enabled, among still further aspects.

[0009] A protection engine according to an embodiment of the invention is operable within one or more network servers, firewalls or other network connectable information re-communicating devices (as are referred to herein summarily one or more “servers” or “re-communicators”). The protection engine includes an information monitor for monitoring information received by the server, and a code detection engine for determining whether the received information includes executable code. The protection engine also includes a packaging engine for causing a sandboxed package, typically including mobile protection code and downloadable protection policies to be sent to a Downloadable-destination in conjunction with the received information, if the received information is determined to be a Downloadable.

[0010] A sandboxed package according to an embodiment of the invention is receivable by and operable with a remote Downloadable-destination. The sandboxed package includes mobile protection code (“MPC”) for causing one or more predetermined malicious operations or operation combinations of a Downloadable to be monitored or otherwise intercepted. The sandboxed package also includes protection policies (operable alone or in conjunction with further Downloadable-destination stored or received policies/MPCs) for causing one or more predetermined operations to be performed if one or more undesirable operations of the Downloadable is/are intercepted. The sandboxed package can also include a corresponding Downloadable and can provide for initiating the Downloadable in a protective “sandbox”. The MPC/policies can further include a communicator for enabling further MPC/policy information or “modules” to be utilized and/or for event logging or other purposes.

[0011] A sandbox protection system according to an embodiment of the invention comprises an installer for enabling a received MPC to be executed within a Downloadable-destination (device/process) and further causing a Downloadable application program, distributable component or other received downloadable code to be received and installed within the Downloadable-destination. The protection system also includes a diverter for monitoring one or more operation attempts of the Downloadable, an operation analyzer for determining one or more responses to the attempts, and a security enforcer for effectuating responses to the monitored operations. The protection system can further include one or more security policies according to which one or more protection system elements are operable automatically (e.g. programmatically) or in conjunction with user intervention (e.g. as enabled by the security enforcer). The security policies can also be configurable/extensible in accordance with further downloadable and/or Downloadable-destination information.

[0012] A method according to an embodiment of the invention includes receiving downloadable information, determining whether the downloadable information includes executable code, and causing a mobile protection code and security policies to be communicated to a network client in conjunction with security policies and the downloadable information if the downloadable information is determined to include executable code. The determining can further provide multiple tests for detecting, alone or together, whether the downloadable information includes executable code.

[0013] A further method according to an embodiment of the invention includes forming a sandboxed package that includes mobile protection code (“MPC”), protection policies, and a received, detected-Downloadable, and causing the sandboxed package to be communicated to and installed by a receiving device or process (“user device”) for responding to one or more malicious operation attempts by the detected-Downloadable from within the user device. The MPC/policies can further include a base “module” and a “communicator” for enabling further up/downloading of one or more further “modules” or other information (e.g. events, user/user device information, etc.).

[0014] Another method according to an embodiment of the invention includes installing, within a user device, received mobile protection code (“MPC”) and protection policies in conjunction with the user device receiving a downloadable application program, component or other Downloadable(s). The method also includes determining, by the MPC, a resource access attempt by the Downloadable, and initiating, by the MPC, one or more predetermined operations corresponding to the attempt. (Predetermined operations can, for example, comprise initiating user, administrator, client, network or protection system determinable operations, including but not limited to modifying the Downloadable operation, extricating the Downloadable, notifying a user/another, maintaining a local/remote log, causing one or more MPCs/policies to be downloaded, etc.)

[0015] Advantageously, systems and methods according to embodiments of the invention enable potentially damaging, undesirable or otherwise malicious operations by even unknown mobile code to be detected, prevented, modified and/or otherwise protected against without modifying the mobile code. Such protection is further enabled in a manner that is capable of minimizing server and client resource requirements, does not require pre-installation of security code within a Downloadable-destination, and provides for client specific or generic and readily updateable security measures to be flexibly and efficiently implemented. Embodiments further provide for thwarting efforts to bypass security measures (e.g. by “hiding” undesirable operation causing information within apparently inert or otherwise “friendly” downloadable information) and/or dividing or combining security measures for even greater flexibility and/or efficiency.

[0016] Embodiments also provide for determining protection policies that can be downloaded and/or ascertained from other security information (e.g. browser settings, administrative policies, user input, uploaded information, etc.). Different actions in response to different Downloadable operations, clients, users and/or other criteria are also enabled, and embodiments provide for implementing other security measures, such as verifying a downloadable source, certification, authentication, etc. Appropriate action can also be accomplished automatically (e.g. programmatically) and/or in conjunction with alerting one or more users/administrators, utilizing user input, etc. Embodiments further enable desirable Downloadable operations to remain substantially unaffected, among other aspects.

BRIEF DESCRIPTION OF THE DRAWINGS

[0017] FIG. 1a is a block diagram illustrating a network system in accordance with an embodiment of the present invention;

[0018] FIG. 1b is a block diagram illustrating a network subsystem example in accordance with an embodiment of the invention;

[0019] FIG. 1c is a block diagram illustrating a further network subsystem example in accordance with an embodiment of the invention;

[0020] FIG. 2 is a block diagram illustrating a computer system in accordance with an embodiment of the invention;

[0021] FIG. 3 is a flow diagram broadly illustrating a protection system host according to an embodiment of the invention;

[0022] FIG. 4 is a block diagram illustrating a protection engine according to an embodiment of the invention;

[0023] FIG. 5 is a block diagram illustrating a content inspection engine according to an embodiment of the invention;

[0024] FIG. 6a is a block diagram illustrating protection engine parameters according to an embodiment of the invention;

[0025] FIG. 6b is a flow diagram illustrating a linking engine use in conjunction with ordinary, compressed and distributable sandbox package utilization, according to an embodiment of the invention;

[0026] FIG. 7a is a flow diagram illustrating a sandbox protection system operating within a destination system, according to an embodiment of the invention;

[0027] FIG. 7b is a block diagram illustrating memory allocation usable in conjunction with the protection system of FIG. 7a, according to an embodiment of the invention;

[0028] FIG. 8 is a block diagram illustrating a mobile protection code according to an embodiment of the invention;

[0029] FIG. 9 is a flowchart illustrating a protection method according to an embodiment of the invention;

[0030] FIG. 10a is a flowchart illustrating method for determining if a potential-Downloadable includes or is likely to include executable code, according to an embodiment of the invention;

[0031] FIG. 10b is a flowchart illustrating a method for forming a protection agent, according to an embodiment of the invention;

[0032] FIG. 11 is a flowchart illustrating a method for protecting a Downloadable destination according to an embodiment of the invention;

[0033] FIG. 12a is a flowchart illustrating a method for forming a Downloadable access interceptor according to an embodiment of the invention; and

[0034] FIG. 12b is a flowchart illustrating a method for implementing mobile protection policies according to an embodiment of the invention.

DETAILED DESCRIPTION

[0035] In providing malicious mobile code runtime monitoring systems and methods, embodiments of the invention enable actually or potentially undesirable operations of even unknown malicious code to be efficiently and flexibly avoided. Embodiments provide, within one or more “servers” (e.g. firewalls, resources, gateways, email relays or other information re-communicating devices), for receiving downloadable-information and detecting whether

the downloadable-information includes one or more instances of executable code (e.g. as with a Trojan horse, zip/meta file etc.). Embodiments also provide for separately or interoperably conducting additional security measures within the server, within a Downloadable-destination of a detected-Downloadable, or both.

[0036] Embodiments further provide for causing mobile protection code (“MPC”) and downloadable protection policies to be communicated to, installed and executed within one or more received information destinations in conjunction with a detected-Downloadable. Embodiments also provide, within an information-destination, for detecting malicious operations of the detected-Downloadable and causing responses thereto in accordance with the protection policies (which can correspond to one or more user, Downloadable, source, destination, or other parameters), or further downloaded or downloadable-destination based policies (which can also be configurable or extensible). (Note that the term “or”, as used herein, is generally intended to mean “and/or” unless otherwise indicated.)

[0037] FIGS. 1a through 1c illustrate a computer network system **100** according to an embodiment of the invention. FIG. 1a broadly illustrates system **100**, while FIGS. 1b and 12 of 1c illustrate exemplary protectable subsystem implementations corresponding with system **104** or **106** of FIG. 1a.

[0038] Beginning with FIG. 1a, computer network system **100** includes an external computer network **101**, such as a Wide Area Network or “WAN” (e.g. the Internet), which is coupled to one or more network resource servers (summarily depicted as resource server-1 **102** and resource server-N **103**). Where external network **101** includes the Internet, resource servers 1-N (**102**, **103**) might provide one or more resources including web pages, streaming media, transaction-facilitating information, program updates or other downloadable information, summarily depicted as resources **121**, **131** and **132**. Such information can also include more traditionally viewed “Downloadables” or “mobile code” (i.e. distributable components), as well as downloadable application programs or other further Downloadables, such as those that are discussed herein. (It will be appreciated that interconnected networks can also provide various other resources as well.)

[0039] Also coupled via external network **101** are subsystems **104-106**. Subsystems **104-106** can, for example, include one or more servers, personal computers (“PCs”), smart

appliances, personal information managers or other devices/processes that are at least temporarily or otherwise intermittently directly or indirectly connectable in a wired or wireless manner to external network **101** (e.g. using a dialup, DSL, cable modem, cellular connection, IR/RF, or various other suitable current or future connection alternatives). One or more of subsystems **104-106** might further operate as user devices that are connectable to external network **101** via an internet service provider (“ISP”) or local area network (“LAN”), such as a corporate intranet, or home, portable device or smart appliance network, among other examples.

[0040] FIG. 1a also broadly illustrates how embodiments of the invention are capable of selectively, modifiably or extensibly providing protection to one or more determinable ones of networked subsystems **104-106** or elements thereof (not shown) against potentially harmful or other undesirable (“malicious”) effects in conjunction with receiving downloadable information. “Protected” subsystem **104**, for example, utilizes a protection in accordance with the teachings herein, while “unprotected” subsystem-N **105** employs no protection, and protected subsystem-M **106** might employ one or more protections including those according to the teachings herein, other protection, or some combination.

[0041] System **100** implementations are also capable of providing protection to redundant elements **107** of one or more of subsystems **104-106** that might be utilized, such as backups, failsafe elements, redundant networks, etc. Where included, such redundant elements are also similarly protectable in a separate, combined or coordinated manner using embodiments of the present invention either alone or in conjunction with other protection mechanisms. In such cases, protection can be similarly provided singly, as a composite of component operations or in a backup fashion. Care should, however, be exercised to avoid potential repeated protection engine execution corresponding to a single Downloadable; such “chaining” can cause a Downloadable to operate incorrectly or not at all, unless a subsequent detection engine is configured to recognize a prior packaging of the Downloadable.

[0042] FIGS. 1b and 1c further illustrate, by way of example, how protection systems according to embodiments of the invention can be utilized in conjunction with a wide variety of different system implementations. In the illustrated examples, system elements are generally configurable in a manner commonly referred to as a “client-server” configuration, as

is typically utilized for accessing Internet and many other network resources. For clarity sake, a simple client-server configuration will be presumed unless otherwise indicated. It will be appreciated, however, that other configurations of interconnected elements might also be utilized (e.g. peer-peer, routers, proxy servers, networks, converters, gateways, services, network reconfiguring elements, etc.) in accordance with a particular application.

[0043] The FIG. 1b example shows how a suitable protected system **104a** (which can correspond to subsystem-l **104** or subsystem-M **106** of FIG. 1) can include a protection-initiating host “server” or “re-communicator” (e.g. ISP server **140a**), one or more user devices or “Downloadable-destinations” **145**, and zero or more redundant elements (which elements are summarily depicted as redundant client device/process **145a**). In this example, ISP server **140a** includes one or more email, Internet or other servers **141a**, or other devices or processes capable of transferring or otherwise “re-communicating” downloadable information to user devices **145**. Server **141a** further includes protection engine or “PE” **142a**, which is capable of supplying mobile protection code (“MPC”) and protection policies for execution by client devices **145**. One or more of user devices **145** can further include a respective one or more clients **146** for utilizing information received via server **140a**, in accordance with which MPC and protection policies are operable to protect user devices **145** from detrimental, undesirable or otherwise “malicious” operations of downloadable information also received by user device **145**.

[0044] The FIG. 1c example shows how a further suitable protected system **104b** can include, in addition to a “re-communicator”, such as server **142b**, a firewall **143c** (e.g. as is typically the case with a corporate intranet and many existing or proposed home/smart networks.) In such cases, a server **141b** or firewall **143** can operate as a suitable protection engine host. A protection engine can also be implemented in a more distributed manner among two or more protection engine host systems or host system elements, such as both of server **141** and firewall **143**, or in a more integrated manner, for example, as a standalone device. Redundant system or system protection elements **11**) can also be similarly provided in a more distributed or integrated manner (see above).

[0045] System **104b** also includes internal network **144** and user devices **145**. User devices **145** further include a respective one or more clients **146** for utilizing information

received via server **140a**, in accordance with which the MPCs or protection policies are operable. (As in the previous example, one or more of user devices **145** can also include or correspond with similarly protectable redundant system elements, which are not shown.)

[0046] It will be appreciated that the configurations of FIGS 1a-1c are merely exemplary. Alternative embodiments might, for example, utilize other suitable connections, devices or processes. One or more devices can also be configurable to operate as a network server, firewall, smart router, a resource server servicing deliverable third-party/manufacture postings, a user device operating as a firewall/server, or other information-suppliers or intermediaries (i.e. as a “re-communicator” or “server”) for servicing one or more further interconnected devices or processes or interconnected levels of devices or processes. Thus, for example, a suitable protection engine host can include one or more devices or processes capable of providing or supporting the providing of mobile protection code or other protection consistent with the teachings herein. A suitable information-destination or “user device” can further include one or more devices or processes (such as email, browser or other clients) that are capable of receiving and initiating or otherwise hosting a mobile code execution.

[0047] FIG. 2 illustrates an exemplary computing system **200**, that can comprise one or more of the elements of FIGS. 1a through 1c. While other application-specific alternatives might be utilized, it will be presumed for clarity sake that system **100** elements (FIGS. 1a-c) are implemented in hardware, software or some combination by one or more processing systems consistent therewith, unless otherwise indicated.

[0048] Computer system **200** comprises elements coupled via communication channels (e.g. bus **201**) including one or more general or special purpose processors **202**, such as a Pentium® or Power PC®, digital signal processor (“DSP”), etc. System **200** elements also include one or more input devices **203** (such as ~ mouse, keyboard, microphone, pen, etc.), and one or more output devices **204**, such as a suitable display, speakers, actuators, etc., in accordance with a particular application.

[0049] System **200** also includes a computer readable storage media reader **205** coupled to a computer readable storage medium **206**, such as a storage/memory device or hard or removable storage/memory media; such devices or media are further indicated separately as storage device **208** and memory **209**, which can include hard disk variants, floppy/compact

disk variants, digital versatile disk (“DVD”) variants, smart cards, read only memory, random access memory, cache memory, etc., in accordance with a particular application. One or more suitable communication devices **207** can also be included, such as a modem, OSL, infrared or other suitable transceiver, etc. for providing inter-device communication directly or via one or more suitable private or public networks that can include but are not limited to those already discussed.

[0050] Working memory further includes operating system (“OS”) elements and other programs, such as application programs, mobile code, data, etc. for implementing system **100** elements that might be stored or loaded therein during use. The particular OS can vary in accordance with a particular device, features or other aspects in accordance with a **110** particular application (e.g. Windows, Mac, Linux, Unix or Palm OS variants, a proprietary OS, etc.). Various programming languages or other tools can also be utilized, such as C++, Java, Visual Basic, etc. As will be discussed, embodiments can also include a network client such as a browser or email client, e.g. as produced by Netscape, Microsoft or others, a mobile code executor such as an OS task manager, Java Virtual Machine (“JVM”), etc., and an application program interface (“API”), such as a Microsoft Windows or other suitable element in accordance with the teachings herein. (It will also become apparent that embodiments might also be implemented in conjunction with a resident application or combination of mobile code and resident application components.)

[0051] One or more system **200** elements can also be implemented in hardware, software or a suitable combination. When implemented in software (e.g. as an application program, object, downloadable, servlet, etc. in whole or part), a system **200** element can be communicated transitionally or more persistently from local or remote storage to memory (or cache memory, etc.) for execution, or another suitable mechanism can be utilized, and elements can be implemented in compiled or interpretive form. Input, intermediate or resulting data or functional elements can further reside more transitionally or more persistently in a storage media, cache or more persistent volatile or non-volatile memory, (e.g. storage device **207** or memory **208**) in accordance with a particular application.

[0052] FIG. 3 illustrates an interconnected re-communicator **300** generally consistent with system **140b** of FIG. 1, according to an embodiment of the invention. As with system **140b**,

system **300** includes a server **301**, and can also include a firewall **302**. In this implementation, however, either server **301** or firewall **302** (if a firewall is used) can further include a protection engine (**310** or **320** respectively). Thus, for example, an included firewall can process received information in a conventional manner, the results of which can be further processed by protection engine **310** of server **301**, or information processed by protection engine **320** of an included firewall **302** can be processed in a conventional manner by server **301**. (For clarity sake, a server including a singular protection engine will be presumed, with or without a firewall, for the remainder of the discussion unless otherwise indicated. Note, however, that other embodiments consistent with the teachings herein might also be utilized.)

[0053] FIG. 3 also shows how information received by server **301** (or firewall **302**) can include non-executable information, executable information or a combination of non-executable and one or more executable code portions (e.g. so-called Trojan horses that include a hostile Downloadable within a friendly one, combined, compressed or otherwise encoded files, etc.). Particularly such combinations will likely remain undetected by a firewall or other more conventional protection systems. Thus, for convenience, received information will also be referred to as a “potential-Downloadable”, and received information found to include executable code will be referred to as a “Downloadable” or equivalently as a “detected-Downloadable” (regardless of whether the executable code includes one or more application programs, distributable “components” such as Java, ActiveX, add-in, etc.).

[0054] Protection engine **310** provides for detecting whether received potential-Downloadables include executable code, and upon such detection, for causing mobile protection code (“MPC”) to be transferred to a device that is a destination of the potential-Downloadable (or “Downloadable-destination”). Protection engine **310** can also provide protection policies in conjunction with the MPC (or thereafter as well), which MPC/policies can be automatically (e.g. programmatically) or interactively configurable in accordance user, administrator, downloadable source, destination, operation, type or various other parameters alone or in combination (see below). Protection engine **310** can also provide or operate separately or interoperably in conjunction with one or more of certification, authentication, downloadable tagging, source checking, verification, logging, diverting or other protection services via the MPC, policies, other local/remote server or destination processing, etc. (e.g.

which can also include protection mechanisms taught by the above-noted prior applications; see FIG. 4).

[0055] Operationally, protection engine **310** of server **301** monitors information received by server **301** and determines whether the received information is deliverable to a protected destination, e.g. using a suitable monitor/data transfer mechanism and comparing a destination-address of the received information to a protected destination set, such as a protected destinations list, array, database, etc. (All deliverable information or one or more subsets thereof might also be monitored.) Protection engine **310** further analyzes the potential-Downloadable and determines whether the potential-Downloadable includes executable code. If not, protection engine **310** enables the not executable potential-Downloadable **331** to be delivered to its destination in an unaffected manner.

[0056] In conjunction with determining that the potential-Downloadable is a detected-Downloadable, protection engine **310** also causes mobile protection code or “MPC” **341** to be communicated to the Downloadable-destination of the Downloadable, more suitably in conjunction with the detected-Downloadable **343** (see below). Protection engine **310** further causes downloadable protection policies **342** to be delivered to the Downloadable-destination, again more suitably in conjunction with the detected-Downloadable. Protection policies **342** provide parameters (or can additionally or alternatively provide additional mobile code) according to which the MPC is capable of determining or providing applicable protection to a Downloadable-destination against malicious Downloadable operations.

[0057] (One or more “checked”, tag, source, destination, type, detection or other security result indicators, which are not shown, can also be provided as corresponding to determined non-Downloadables or Downloadables, e.g. for testing, logging, further processing, further identification tagging or other purposes in accordance with a particular application.)

[0058] Further MPCs, protection policies or other information are also deliverable to a the same or another destination, for example, in accordance with communication by an MPC/protection policies already delivered to a downloadable-destination. Initial or subsequent MPCs/policies can further be selected or configured in accordance with a Downloadable-destination indicated by the detected-Downloadable, destination-user or administrative information, or other information providable to protection engine **310** by a user,

administrator, user system, user system examination by a communicated MPC, etc. (Thus, for example, an initial MPC/policies can also be initially provided that are operable with or optimized for more efficient operation with different Downloadable-destinations or destination capabilities.)

[0059] While integrated protection constraints within the MPC might also be utilized, providing separate protection policies has been found to be more efficient, for example, by enabling more specific protection constraints to be more easily updated in conjunction with detected-Downloadable specifics, post-download improvements, testing, etc. Separate policies can further be more efficiently provided (e.g. selected, modified, instantiated, etc.) with or separately from an MPC, or in accordance with the requirements of a particular user, device, system, administration, later improvement, etc., as might also be provided to protection engine **310** (e.g. via user/MPC uploading, querying, parsing a Downloadable, or other suitable mechanism implemented by one or more servers or Downloadable-destinations).

[0060] (It will also become apparent that performing executable code detection and communicating to a downloadable-Destination an MPC and any applicable policies as separate from a detected-Downloadable is more accurate and far less resource intensive than, for example, performing content and operation scanning, modifying a Downloadable, or providing completely Downloadable-destination based security.)

[0061] System **300** enables a single or extensible base-MPC to be provided, in anticipation or upon receipt of a first Downloadable, that is utilized thereafter to provide protection of one or more Downloadable-destinations. It is found, however, that providing an MPC upon each detection of a Downloadable (which is also enabled) can provide a desirable combination of configurability of the MPC/policies and lessened need for management (e.g. given potentially changing user/destination needs, enabling testing, etc.).

[0062] Providing an MPC upon each detection of a Downloadable also facilitates a lessened demand on destination resources, e.g. since information-destination resources used in executing the MPC/policies can be re-allocated following such use. Such alternatives can also be selectively, modifiably or extensibly provided (or further in accordance with other application-specific factors that might also apply.) Thus, for example, a base-MPC or base-policies might be provided to a user device that is/are extensible via additionally

downloadable “modules” upon server **301** detection of a Downloadable deliverable to the same user device, among other alternatives.

[0063] In accordance with a further aspect of the invention, it is found that improved efficiency can also be achieved by causing the MPC to be executed within a Downloadable-destination in conjunction with, and further, prior to initiation of the detected Downloadable. One mechanism that provides for greater compatibility and efficiency in conjunction with conventional client-based Downloadable execution is for a protection engine to form a sandboxed package **340** including MPC **341**, the detected-Downloadable **343** and any policies **342**. For example, where the Downloadable is a binary executable to be executed by an operating system, protection engine **310** forms a protected package by concatenating, within sandboxed package **340**, MPC **341** for delivery to a Downloadable-destination first, followed by protection policies **342** and Downloadable **343**. (Concatenation or techniques consistent therewith can also be utilized for providing a protecting package corresponding to a Java applet for execution by a NM of a Downloadable-destination, or with regard to ActiveX controls, add-ins or other distributable components, etc.)

[0064] The above concatenation or other suitable processing will result in the following. Upon receipt of sandboxed package **340** by a compatible browser, email or other destination-client and activating of the package by a user or the destination-client, the operating system (or a suitable responsively initiated distributed component host) will attempt to initiate sandboxed package **340** as a single Downloadable. Such processing will, however, result in initiating the MPC **341** and -in accordance with further aspects of the invention- the MPC will initiate the Downloadable in a protected manner, further in accordance with any applicable included or further downloaded protection policies **342**. (While system **300** is also capable of ascertaining protection policies stored at a Downloadable-destination, e.g. by poll, query, etc. of available destination information, including at least initial policies within a suitable protecting package is found to avoid associated security concerns or inefficiencies.)

[0065] Turning to FIG. 4, a protection engine **400** generally consistent with protection engine **310** (or **320**) of FIG. 3 is illustrated in accordance with an embodiment of the invention. Protection engine **400** comprises information monitor **401**, detection engine **402**, and protected packaging engine **403**, which further includes agent generator **431**, storage **404**,

linking engine **405**, and transfer engine **406**. Protection engine **400** can also include a buffer **407**, for temporarily storing a received potential-Downloadable, or one or more systems for conducting additional authentication, certification, verification or other security processing (e.g. summarily depicted as security system **408**.) Protection engine **400** can further provide for selectively re-directing, further directing, logging, etc. of a potential/detected Downloadable or information corresponding thereto in conjunction with detection, other security, etc., in accordance with a particular application.

[0066] (Note that FIG. 4, as with other figures included herein, also depicts exemplary signal flow arrows; such arrows are provided to facilitate discussion, and should not be construed as exclusive or otherwise limiting.)

[0067] Information monitor **401** monitors potential-Downloadables received by a host server and provides the information via buffer **407** to detection engine **402** or to other system **400** elements. Information monitor **401** can be configured to monitor host server download operations in conjunction with a user or a user-device that has logged-on to the server, or to receive information via a server operation hook, servlet, communication channel or other suitable mechanism.

[0068] Information monitor **401** can also provide for transferring, to storage **404** or other protection engine elements, configuration information including, for example, user, MPC, protection policy, interfacing or other configuration information (e.g. see FIG. 6). Such configuration information monitoring can be conducted in accordance with a user/device logging onto or otherwise accessing a host server, via one or more of configuration operations, using an applet to acquire such information from or for a particular user, device or devices, via MPC/policy polling of a user device, or via other suitable mechanisms.

[0069] Detection engine **402** includes code detector **421**, which receives a potential-Downloadable and determines, more suitably in conjunction with inspection parameters **422**, whether the potential-Downloadable includes executable code and is thus a “detected-Downloadable”. (Code detector **421** can also include detection processors for performing me decompression or other “decoding”, or such detection-facilitating processing as decryption, utilization/support of security system **408**, etc. in accordance with a particular application.)

[0070] Detection engine **402** further transfers a detected-downloadable (“XEQ”) to protected packaging engine **403** along with indicators of such detection., or a determined non-executable (“NXEQ”) to transfer engine **406**. (Inspection parameters **422** enable analysis criteria to be readily updated or varied, for example, in accordance with particular source, destination or other potential Downloadable impacting parameters, and are discussed in greater detail with reference to FIG. 5). Detection engine **402** can also provide indicators for delivery of initial and further MPCs/policies, for example, prior to or in conjunction with detecting a Downloadable and further upon receipt of an indicator from an already downloaded MPC/policy. A downloaded MPC/policy can further remain resident at a user device with further modules downloaded upon or even after delivery of a sandboxed package. Such distribution can also be provided in a configurable manner, such that delivery of a complete package or partial packages are automatically or interactively determinable in accordance with user/administrative preferences/policies, among other examples.

[0071] Packaging engine **403** provides for generating mobile protection code and protection policies, and for causing delivery thereof (typically with a detected-Downloadable) to a Downloadable-destination for protecting the Downloadable-destination against malicious operation attempts by the detected Downloadable. In this example, packaging engine **403** includes agent generator **431**, storage **404** and linking engine **405**.

[0072] Agent generator **431** includes an MPC generator **432** and a protection policy generator **433** for “generating” an MPC and a protection policy (or set of policies) respectively upon receiving one or more “generate MPC/policy” indicators from detection engine **402**, indicating that a potential-Downloadable is a detected-Downloadable. MPC generator **432** and protection policy generator **433** provide for generating MPCs and protection policies respectively in accordance with parameters retrieved from storage **404**. Agent generator **431** is further capable of providing multiple MPCs/policies, for example, the same or different MPCs/policies in accordance with protecting ones of multiple executables within a zip file, or for providing initial MPCs/policies and then further MPCs/policies or MPC/policy “modules” as initiated by further indicators such as given above, via an indicator of an already downloaded MPC/policy or via other suitable mechanisms. (It will be appreciated that pre-constructed MPCs/policies or other processing can also be utilized, e.g. via retrieval from storage **404**, but with a potential decrease in flexibility.)

[0073] MPC generator **432** and protection policy generator **433** are further configurable. Thus, for example, more generic MPCs/policies can be provided to all or a grouping of serviced destination-devices (e.g. in accordance with a similarly configured/administered intranet), or different MPCs/policies that can be configured in accordance with one or more of user, network administration, Downloadable-destination or other parameters (e.g. see FIG. 6). As will become apparent, a resulting MPC provides an operational interface to a destination device/process. Thus, a high degree of flexibility and efficiency is enabled in providing such an operational interface within different or differently configurable user devices/processes or other constraints.

[0074] Such configurability further enables particular policies to be utilized in accordance with a particular application (e.g. particular system uses, access limitations, user interaction, treating application programs or Java components from a particular known source one way and unknown source ActiveX components, or other considerations). Agent generator **431** further transfers a resulting MPC and protection policy pair to linking engine **405**.

[0075] Linking engine **405** provides for forming from received component elements (see above) a sandboxed package that can include one or more initial or complete MPCs and applicable protection policies, and a Downloadable, such that the sandboxed package will protect a receiving Downloadable-destination from malicious operation by the Downloadable. Linking engine **405** is implementable in a static or configurable manner in accordance, for example, with characteristics of a particular user device/process stored intermittently or more persistently in storage **404**. Linking engine **405** can also provide for restoring a Downloadable, such as a compressed, encrypted or otherwise encoded file that has been decompressed, decrypted or otherwise decoded via detection processing **20** (e.g. see FIG. 6b). It is discovered, for example, that the manner in which the Windows OS initiates a binary executable or an ActiveX control can be utilized to enable protected initiation of a detected-Downloadable. Linking engine **405** is, for example, configurable to form, for an ordinary single-executable Downloadable (e.g. an application program, applet, etc.) a sandboxed package **340** as a concatenation of ordered elements including an MPC **341**, applicable policies **342** and the Downloadable or “XEQ” **343** (e.g. see FIG. 4).

[0076] Linking engine **405** is also configurable to form, for a Downloadable received by a server as a compressed single or multiple-executable Downloadable such as a zipped or meta file, a protecting package **340** including one or more MPCs, applicable policies and the one or more included executables of the Downloadable. For example, a sandboxed package can be formed in which a single MPC and policies precede and thus will affect all such executables as a result of inflating and installation. An MPC and applicable policies can also, for example, precede each executable, such that each executable will be separately sandboxed in the same or a different manner according to MPC/policy configuration (see above) upon inflation and installation. (See also FIGS. 5 and 6.) Linking engine is also configurable to form an initial MPC, MPC-policy or sandboxed package (e.g. prior to upon receipt of a downloadable) or an additional MPC, MPC-policy or sandboxed package (e.g. upon or following receipt of a downloadable), such that suitable MPCs/policies can be provided to a Downloadable-destination or other destination in a more distributed manner. In this way, requisite bandwidth or destination resources can be minimized (via two or more smaller packages) in compromise with latency or other considerations raised by the additional required communication.

[0077] A configurable linking engine can also be utilized in accordance with other requirements of particular devices/processes, further or different elements or other permutations in accordance with the teachings herein. (It might, for example be desirable to modify the ordering of elements, to provide one or more elements separately, to provide additional information, such as a header, etc., or perform other processing in accordance with a particular device, protocol or other application considerations.)

[0078] Policy/authentication reader-analyzer **481** summarily depicts other protection mechanisms that might be utilized in conjunction with Downloadable detection, such as already discussed, and that can further be configurable to operate in accordance with policies or parameters (summarily depicted by security/authentication policies **482**). Integration of such further protection in the depicted configuration, for example, enables a potential-Downloadable from a known unfriendly source, a source failing authentication or a provided-source that is confirmed to be fictitious to be summarily discarded, otherwise blocked, flagged, etc. (with or without further processing). Conversely, a potential-Downloadable from a known friendly source (or one confirmed as such) can be transferred with or without further

processing in accordance with particular application considerations. (Other configurations including pre or post Downloadable detection mechanisms might also be utilized.)

[0079] Finally, transfer engine **406** of protection agent engine **303** provides for receiving and causing linking engine **405** (or other protection) results to be transferred to a destination user device/process. As depicted, transfer engine **406** is configured to receive and transfer a Downloadable, a determined non-executable or a sandboxed package. However, transfer engine **406** can also be provided in a more configurable manner, such as was already discussed for other system **400** elements. (Anyone or more of system **400** elements might be configurably implemented in accordance with a particular application.) Transfer engine **406** can perform such transfer, for example, by adding the information to a server transfer queue (not shown) or utilizing another suitable method.

[0080] Turning to FIG. 5 with reference to FIG. 4, a code detector **421** example is illustrated in accordance with an embodiment of the invention. As shown, code detector **421** includes data fetcher **501**, parser **502**, file-type detector **503**, inflater **504** and control **506**; other depicted elements. While implementable and potentially useful in certain instances, are found to require substantial overhead, to be less accurate in certain instances (see above) and are not utilized in a present implementation; these will be discussed separately below. Code detector elements are further configurable in accordance with stored parameters retrievable by data fetcher **501**. (A coupling between data fetcher **501** and control **506** has been removed for clarity sake.)

[0081] Data fetcher **501** provides for retrieving a potential-Downloadable or portions thereof stored in buffer **407** or parameters from storage **404**, and communicates such information or parameters to parser **502**. Parser **502** receives a potential-Downloadable or portions thereof from data fetcher **501** and isolates potential-Downloadable elements, such as file headers, source, destination, certificates, etc. for use by further processing elements.

[0082] File type detector **502** receives and determines whether the potential-Downloadable (likely) is or includes an executable file type. File-reader **502** can, for example, be configured to analyze a received potential-Downloadable for a file header, which is typically included in accordance with conventional data transfer protocols, such as a portable executable or standard “.exe” file format for Windows OS application programs, a Java class header for Java

applets, and so on for other applications, distributed components, etc. “Zipped”, meta or other compressed files, which might include one or more executables, also typically provide standard single or multi-level headers that can be read and used to identify included executable code (or other included information types). File type detector **502** is also configurable for analyzing potential-Downloadables for all potential file type delimiters or a more limited subset of potential file type delimiters (e.g. “.exe” or “.com” in conjunction with a DOS or Microsoft Windows as Downloadable-destination).

[0083] Known file type delimiters can, for example, be stored in a more temporary or more persistent storage (e.g. storage **404** of FIG. 4) which file type detector **502** can compare to a received potential-Downloadable. (Such delimiters can thus also be updated in storage **404** as a new file type delimiter is provided, or a more limited subset of delimiters can also be utilized in accordance with a particular Downloadable-destination or other considerations of a particular application.) File type detector **502** further transfers to controller **506** a detected file type indicator indicating that the potential-Downloadable includes or does not include (i.e. or likely include) an executable file type.

[0084] In this example, the aforementioned detection processor is also included as pre-detection processor or, more particularly, a configurable file inflator **504**. File inflator **504** provides for opening or “inflating” compressed files in accordance with a compressed file type received from file type detector **503** and corresponding file opening parameters received from data fetcher **501**. Where a compressed file (e.g. a meta file) includes nested file type information not otherwise reliably provided in an overall file header or other information, inflator **504** returns such information to parser **502**. File inflator **504** also provides any now-accessible included executables to control **506** where one or more included files are to be separately packaged with an MPC or policies.

[0085] Control **506**, in this example, operates in accordance with stored parameters and provides for routing detected non-Downloadables or Downloadables and control information, and for conducting the aforementioned distributed downloading of packages to Downloadable-destinations. In the case of a non-Downloadable, for example, control **506** sends the non-Downloadable to transfer engine **406** (FIG. 4) along with any indicators that might apply. For an ordinary single-executable Downloadable, control **506** sends control information to agent

generator **431** and the Downloadable to linking engine **405** along with any other applicable indicators (see **641** of FIG. 6b). Control **506** similarly handles a compressed single-executable Downloadable or a multiple downloadable to be protected using a single sandboxed package. For a multiple-executable Downloadable, control **506** sends control information for each corresponding executable to agent generator agent generator **431**, and sends the executable to linking engine **405** along with controls and any applicable indicators, as in **643b** of FIG. 6b. (The above assumes, however, that distributed downloading is not utilized; when used - according to applicable parameters- control **506** also operates in accordance with the following.)

[0086] Control **506** conducts distributed protection (e.g. distributed packaging) by providing control signals to agent generator **431**, linking engine **405** and transfer engine **406**. In the present example, control **506** initially sends controls to agent generator **431** and linking engine **405** (FIG. 4) causing agent generator to generate an initial MPC and initial policies, and sends control and a detected-Downloadable to linking engine **405**. Linking engine **405** forms an initial sandboxed package, which transfer engine causes (in conjunction with further controls) to be downloaded to the Downloadable destination (**643a** of FIG. 6b). An initial MPC within the sandboxed package includes an installer and a communicator and performs installation as indicated below. The initial MPC also communicates via the communicator controls to control **506** (FIG. 5) in response to which control **506** similarly causes generation of MPC-M and policy-M modules **643c**, which linking engine **405** links and transfer engine **406** causes to be sent to the Downloadable destination, and so on for any further such modules.

[0087] (It will be appreciated, however, that an initial package might be otherwise configured or sent prior to receipt of a Downloadable in accordance with configuration parameters or user interaction. Information can also be sent to other user devices, such as that of an administrator. Further MPCs/policies might also be coordinated by control **506** or other elements, or other suitable mechanisms might be utilized in accordance with the teachings herein.)

[0088] Regarding the remaining detection engine elements illustrated in FIG. 5, where content analysis is utilized, parser **502** can also provide a Downloadable or portions thereof to

content detector **505**. Content detector **505** can then provide one or more content analyses. Binary detector **551**, for example, performs detection of binary information; pattern detector **552** further analyzes the Downloadable for patterns indicating executable code, or other detectors can also be utilized. Analysis results therefrom can be used in an absolute manner, where a first testing result indicating executable code confirms Downloadable detection, which result is then sent to control **506**. Alternatively, however, composite results from such analyses can also be sent to control **506** for evaluation. Control **506** can further conduct such evaluation in a summary manner (determining whether a Downloadable is detected according to a majority or minimum number of indicators), or based on a weighting of different analysis results. Operation then continues as indicated above. (Such analysis can also be conducted in accordance with aspects of a destination user device or other parameters.)

[0089] FIG. 6a illustrates more specific examples of indicators/parameters and known (or “knowledge base”) elements that can be utilized to facilitate the above-discussed system **400** configurability and detection. For clarity sake, indicators, parameters and knowledge base elements are combined as indicated “parameters.” It will be appreciated, however, that the particular parameters utilized can differ in accordance with a particular application, and indicators, parameters or known elements, where utilized, can vary and need not correspond exactly with one another. Any suitable explicit or referencing list, database or other storage structure(s) or storage structure configuration(s) can also be utilized to implement a suitable user/device based protection scheme, such as in the above examples, or other desired protection schema.

[0090] Executable parameters **601** comprise, in accordance with the above examples, executable file type parameters **611**, executable code parameters **612** and code pattern parameters **613** (including known executable file type indicators, header/code indicators and patterns respectively, where code patterns are utilized). Use parameters **602** further comprise user parameters **621**, system parameters **622** and general parameters **623** corresponding to one or more users, user classifications, user-system correspondences or destination system, device or processes, etc. (e.g. for generating corresponding MPCs/policies, providing other protection, etc.). The remaining parameters include interface parameters **631** for providing MPC/policy (or further) configurability in accordance with a particular device or for enabling communication with a device user (see below), and other parameters **632**.

[0091] FIG. 6b illustrates a linking engine **405** according to an embodiment of the invention. As already discussed, linking engine **405** includes a linker for combining MPCs, policies or agents via concatenation or other suitable processing in accordance with an OS, JVM or other host executor or other applicable factors that might apply. Linking engine **405** also includes the aforementioned post-detection processor which, in this example, comprises a compressor **508**. As noted, compressor **508** receives linked elements from linker **507** and, where a potential-Downloadable corresponds to a compressed file that was inflated during detection, re-forms the compressed file. (Known file information can be provided via configuration parameters, substantially reversal of inflating or another suitable method.) Encryption or other post-detection processing can also be conducted by linking engine **508**.

[0092] FIGS. 7a, 7b and 8 illustrate a “sandbox protection” system, as operable within a receiving destination-device, according to an embodiment of the invention.

[0093] Beginning with FIG. 7a, a client **146** receiving sandbox package **340** will “recognize” sandbox package **340** as a (mobile) executable and cause a mobile code installer **711** (e.g. an OS loader, JVM, etc.) to be initiated. Mobile code installer **711** will also recognize sandbox package **340** as an executable and will attempt to initiate sandbox package **340** at its “beginning.” Protection engine **400** processing corresponding to destination **700** use of a such a loader, however, will have resulted in the “beginning” of sandbox package **340** as corresponding to the beginning of MPC **341**, as noted with regard to the above FIG. 4 example.

[0094] Such protection engine processing will therefore cause a mobile code installer (e.g. OS loader **711**, for clarity sake) to initiate MPC **341**. In other cases, other processing might also be utilized for causing such initiation or further protection system operation. Protection engine processing also enables MPC **341** to effectively form a protection “sandbox” around Downloadable (e.g. detected-Downloadable or “XEQ”) **343**, to monitor Downloadable **343**, intercept determinable Downloadable **343** operation (such as attempted accesses of Downloadable **343** to destination resources) and, if “malicious”, to cause one or more other operations to occur (e.g. providing an alert, offloading the Downloadable, offloading the MPC, providing only limited resource access, possibly in a particular address space or with regard to a particularly “safe” resource or resource operation, etc.).

[0095] MPC **341**, in the present OS example, executes MPC element installation and installs any policies, causing MPC **341** and protection policies **342** to be loaded into a first memory space, PI. MPC **341** then initiates loading of Downloadable **343**. Such Downloadable initiation causes OS loader **711** to load Downloadable **343** into a further working memory space-P2 **703** along with an API import table ("IAT") **731** for providing Downloadable **631** with destination resource access capabilities. It is discovered, however that the IA T can be modified so that any call to an API can be redirected to a function within the MPC. The technique for modifying the IA T is documented within the MSDN (Microsoft Developers Network) Library CD in several articles. The technique is also different for each operating system (e.g. between Windows **9x** and Windows NT), which can be accommodated by agent generator configurability, such as that given above. MPC **341** therefore has at least initial access to API IAT **731** of Downloadable **632**, and provides for diverting, evaluating and responding to attempts by Downloadable **632** to utilize system APIs **731**, or further in accordance with protection policies **342**. In addition to API diverting, MPC **341** can also install filter drivers, which can be used for controlling access to resources such as a Downloadable-destination file system or registry. Filter driver installation can be conducted as documented in the MSDN or using other suitable methods.

[0096] Turning to FIG. 8 with reference to FIG. 7b, an MPC **341** according to an embodiment of the invention includes a package extractor **801**, executable installer **802**, sandbox engine installer **803**, resource access diverter **804**, resource access (attempt) analyzer **805**, policy enforcer **806** and MPC de-installer **807**. Package extractor **801** is initiated upon initiation of MPC **341**, and extracts MPC **341** elements and protection policies **342**. Executable installer **802** further initiates installation of a Downloadable by extracting the downloadable from the protected package, and loading the process into memory in suspended mode (so it only loads into memory, but does not start to run). Such installation further causes the operating system to initialize the Downloadable's IAT **731** in the memory space of the downloadable process, P2, as already noted.

[0097] Sandbox engine installer **803** (running in process space PI) then installs the sandbox engine (**803-805**) and policies **342** into the downloadable process space P2. This is done in different way in each operating system (e.g. see above). Resource access diverter **804** further modifies those Downloadable-API IAT entries that correspond with protection policies

342, thereby causing corresponding Downloadable accesses via Downloadable-API IAT **731** to be diverted resource access analyzer **805**.

[0098] During Downloadable operation, resource access analyzer or “RAA” **805** receives and determines a response to diverted Downloadable (i.e. “malicious”) operations in accordance with corresponding protection policies of policies **342**. (RAA **805** or further elements, which are not shown, can further similarly provide for other security mechanisms that might also be implemented.) Malicious operations can for example include, in a Windows environment: file operations (e.g. reading, writing, deleting or renaming a file), network operations (e.g. listen on or connect to a socket, send/receive data or view intranet), OS registry or similar operations (read/write a registry item), OS operations (exit as/client, kill or change the priority of a process/thread, dynamically load a class library), resource usage thresholds (e.g. memory, CPU, graphics), etc.

[0099] Policy enforcer **806** receives RAA **805** results and causes a corresponding response to be implemented, again according to the corresponding policies. Policy enforcer **806** can, for example, interact with a user (e.g. provide an alert, receive instructions, etc.), create a log file, respond, cause a response to be transferred to the Downloadable using “dummy” or limited data, communicate with a server or other networked device (e.g. corresponding to a local or remote administrator), respond more specifically with a better known Downloadable, verify accessibility or user/system information (e.g. via local or remote information), even enable the attempted Downloadable access, among a wide variety of responses that will become apparent in **20** view of the teachings herein.

[0100] The FIG. 9 flowchart illustrates a protection method according to an embodiment of the invention. In step **901**, a protection engine monitors the receipt, by a server or other re-communicator of information, and receives such information intended for a protected information-destination (i.e. a potential-Downloadable) in step **903**. Steps **905-911** depict an adjunct trustworthiness protection that can also be provided, wherein the protection engine determines whether the source of the received information is known to be “unfriendly” and, if so, prevents current (at least unaltered) delivery of the potential-Downloadable and provides any suitable alerts. (The protection engine might also continue to perform Downloadable detection and nevertheless enable delivery or protected delivery of a non-Downloadable, or

avoid detection if the source is found to be "trusted", among other alternatives enabled by the teachings herein.)

[0101] If, in step **913**, the potential-Downloadable source is found to be of an unknown or otherwise suitably authenticated/certified source, then the protection engine determines whether the potential-Downloadable includes executable code in step **915**. If the potential-Downloadable does not include executable code, then the protection engine causes the potential-Downloadable to be delivered to the information-destination in its original form in step **917**, and the method ends. If instead the potential-Downloadable is found to include executable code in step **915** (and is thus a “detected-Downloadable”), then the protection engine forms a sandboxed package in step **919** and causes the protection agent to be delivered to the information-Destination in step **921**, and the method ends. As was discussed earlier, a suitable protection agent can include mobile protection code, policies and the detected-Downloadable (or information corresponding thereto).

[0102] The FIG. 10a flowchart illustrates a method for analyzing a potential-Downloadable, according to an embodiment of the invention. As shown, one or more aspects can provide useful indicators of the inclusion of executable code within the potential-Downloadable. In step **1001**, the protection engine determines whether the potential-Downloadable indicates an executable file type, for example, by comparing one or more included file headers for file type indicators (e.g. extensions or other descriptors). The indicators can be compared against all known file types executable by all protected Downloadable destinations, a subset, in accordance with file types executable or desirably executable by the Downloadable-destination, in conjunction with a particular user, in conjunction with available information or operability at the destination, various combinations, etc.

[0103] Where content analysis is conducted, in step **1003** of FIG. 10a, the protection engine analyzes the potential-Downloadable and determines in accordance therewith whether the potential-Downloadable does or is likely to include binary information, which typically indicates executable code. The protection engine further analyzes the potential-Downloadable for patterns indicative of included executable code in step **1003**. Finally, in step **1005**, the protection engine determines whether the results of steps **1001** and **1003** indicate that the

potential-Downloadable more likely includes executable code (e.g. via weighted comparison of the results with a suitable level indicating the inclusion or exclusion of executable code). The protection engine, given a suitably high confidence indicator of the inclusion of executable code, treats the potential-Downloadable as a detected-Downloadable.

[0104] The FIG. 10b flowchart illustrates a method for forming a sandboxed package according to an embodiment of the invention. As shown, in step **1011**, a protection engine retrieves protection parameters and forms mobile protection code according to the parameters. The protection engine further, in step **1013**, retrieves protection parameters and forms protection policies according to the parameters. Finally, in step **1015**, the protection engine couples the mobile protection code, protection policies and received-information to form a sandboxed package. For example, where a Downloadable-destination utilizes a standard windows executable, coupling can further be accomplished via concatenating the MPC for delivery of MPC first, policies second, and received information third. (The protection parameters can, for example, include parameters relating to one or more of the Downloadable destination device/process, user, supervisory constraints or other parameters.)

[0105] The FIG. 11 flowchart illustrates how a protection method performed by mobile protection code (“MPC”) according to an embodiment of the invention includes the MPC installing MPC elements and policies within a destination device in step **1101**. In step **1102**, the MPC loads the Downloadable without actually initiating it (i.e. for executables, it will start a process in suspended mode). The MPC further forms an access monitor or “interceptor” for monitoring or “intercepting” downloadable destination device access attempts within the destination device (according to the protection policies in step **1103**, and initiates a corresponding Downloadable within the destination device in step **1105**.

[0106] If, in step **1107**, the MPC determines, from monitored/intercepted information, that the Downloadable is attempting or has attempted a destination device access considered undesirable or otherwise malicious, then the MPC performs steps **1109** and **1111**; otherwise the MPC returns to step **1107**. In step **1109**, the MPC determines protection policies in accordance with the access attempt by the Downloadable, and in step **1111**, the MPC executes the protection policies. (protection policies can, for example, be retrieved from a temporary, e.g. memory/cache, or more persistent storage.)

[0107] As shown in the FIG. 12a example, the MPC can provide for intercepting Downloadable access attempts by a Downloadable by installing the Downloadable (but not executing it) in step **1201**. Such installation will cause a Downloadable executor, such as a the Windows operating system, to provide all required interfaces and parameters (such as the IAT, process ill, etc.) for use by the Downloadable to access device resources of the host device. The MPC can thus cause Downloadable access attempts to be diverted to the MPC by modifying the Downloadable IAT, replacing device resource location indicators with those of the MPC (step **1203**).

[0108] The FIG. **12b** example further illustrates an example of how the MPC can apply suitable policies in accordance with an access attempt by a Downloadable. As shown, the MPC receives the Downloadable access request via the modified IAT in step **1211**. The MPC further queries stored policies to determine a policy corresponding to the Downloadable access request in step **1213**.

[0109] The foregoing description of preferred embodiments of the invention is provided by way of example to enable a person skilled in the art to make and use the invention, and in the context of particular applications and requirements thereof. Various modifications to the embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles, features and teachings disclosed herein. The embodiments described herein are not intended to be exhaustive or limiting. The present invention is limited only by the following claims.

CLAIMS

What is claimed is:

1. A system for reviewing an operating system call issued by a downloadable, comprising:

an operating system probe associated with an operating system function for intercepting an operating system call being issued by a downloadable to an operating system and associated with the operating system function;

a runtime environment monitor for comparing the operating system call against a predetermined security policy including multiple security rules to determine if execution of the operating system call violates one or more of the multiple security rules before allowing the operating system to process the operating system call and for forwarding a message to a response engine when the comparison by the runtime environment monitor indicates a violation of one or more of the multiple security rules;

a response engine for compiling each rule violation indicated in the messages forwarded by the runtime environment monitor, for blocking execution of operating system calls that are forbidden according to the security policy when execution of the operating system calls would result in a violation of a predetermined combination of multiple security rules of the predetermined security policy and for allowing execution of operating system calls that are permitted according to the security policy.

2. The system of claim 1, wherein the downloadable is one of a Java component, an ActiveX control, executable code, or interpretable code.

3. The system of claim 1, further comprising:

a request broker for receiving a notification message from the downloadable engine regarding the extension call;

a file system probe and a network system probe each being associated with an operating system function for receiving the request message from the downloadable engine and

intercepting an operating system call being issued by the downloadable to an operating system and associated with the operating system function;

an event router for receiving the notification message from the request broker regarding the extension call and an event message from one of the file system probe and the network system probe regarding the operating system call;

the runtime environment monitor for receiving the notification message and the event message from the event router and comparing the extension call and the operating system call against a predetermined security policy before allowing the operating system to process the extension call and the operating system call; and

the response engine for receiving a violation message from the runtime environment monitor when one of the extension call and the operating system call violate one or more rules of the predetermined security policy and blocking extension calls and operating system calls that are forbidden according to the predetermined security policy, and for allowing extension calls and operating system calls that are permitted according to the predetermined security policy.

4. The system of claim 3, wherein the downloadable is one of a Java component, an ActiveX control, executable code, or interpretable code.

5. A system for reviewing an operating system call issued by a downloadable, comprising: a downloadable engine for intercepting a request message being issued by a downloadable to an operating system, wherein the request message includes an extension call;

a request broker for receiving a notification message from the downloadable engine regarding the extension call;

a file system probe and a network system probe each being associated with an operating system function for receiving the request message from the downloadable engine and intercepting an operating system call being issued by the downloadable to an operating system and associated with the operating system function;

a runtime environment monitor for receiving the notification message and the event message from the event router and comparing the extension call and the operating system call

against a predetermined security policy before allowing the operating system to process the extension call and the operating system call; and

a response engine for receiving a violation message from the runtime environment monitor when one of the extension call and the operating system call violate one or more rules of the predetermined security policy and blocking extension calls and operating system calls that are forbidden according to the predetermined security policy, and for allowing extension calls and operating system calls that are permitted according to the predetermined security policy.

6. The system of claim 5, wherein the Downloadable is one of a Java component, an ActiveX control, executable code, or interpretable code.
7. The system of claim 5, wherein the response engine stores results of the comparison in an event log.
8. The system of claim 5, wherein the response engine informs the user when the security policy has been violated.
9. The system of claim 5, wherein the response engine stores information on the Downloadable in a suspicious Downloadable database when it includes extension calls or operating system calls that are forbidden according to the predetermined security policy.
10. The system of claim 5, wherein the response engine discards the Downloadable when it includes extension calls or operating system calls that are forbidden according to the predetermined security policy.

ABSTRACT

Protection systems and methods provide for protecting one or more personal computers (“PCs”) and/or other intermittently or persistently network accessible devices or processes from undesirable or otherwise malicious operations of Java TN applets, ActiveX™ controls, JavaScript™ scripts, Visual Basic scripts, add-ins, downloaded/uploaded programs or other “Downloadables” or “mobile code” in whole or part. A protection engine embodiment provides for monitoring information received, determining whether received information does or is likely to include executable code, and if so, causes mobile protection code (MPC) to be transferred to and rendered operable within a destination device of the received information. An MPC embodiment further provides, within a Downloadable-destination, for initiating the Downloadable, enabling malicious Downloadable operation attempts to be received by the MPC, and causing (predetermined) corresponding operations to be executed in response to the attempts.

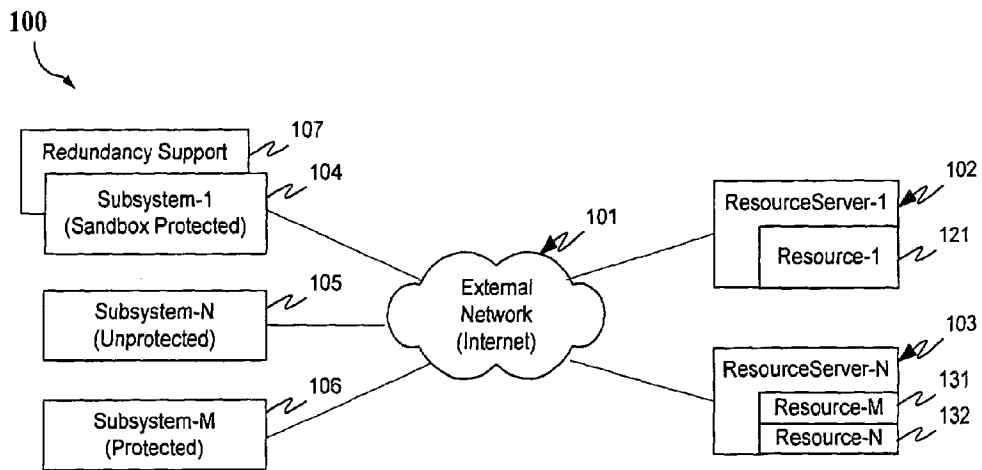


FIG. 1a

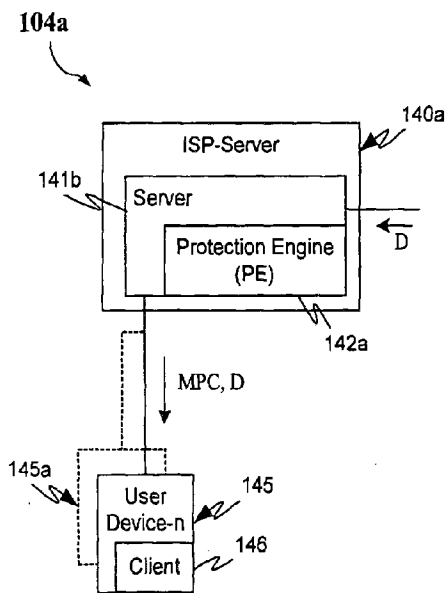


FIG. 1b

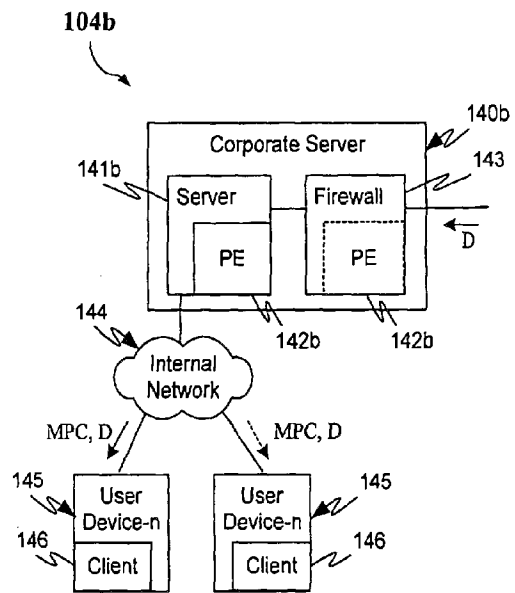


FIG. 1c

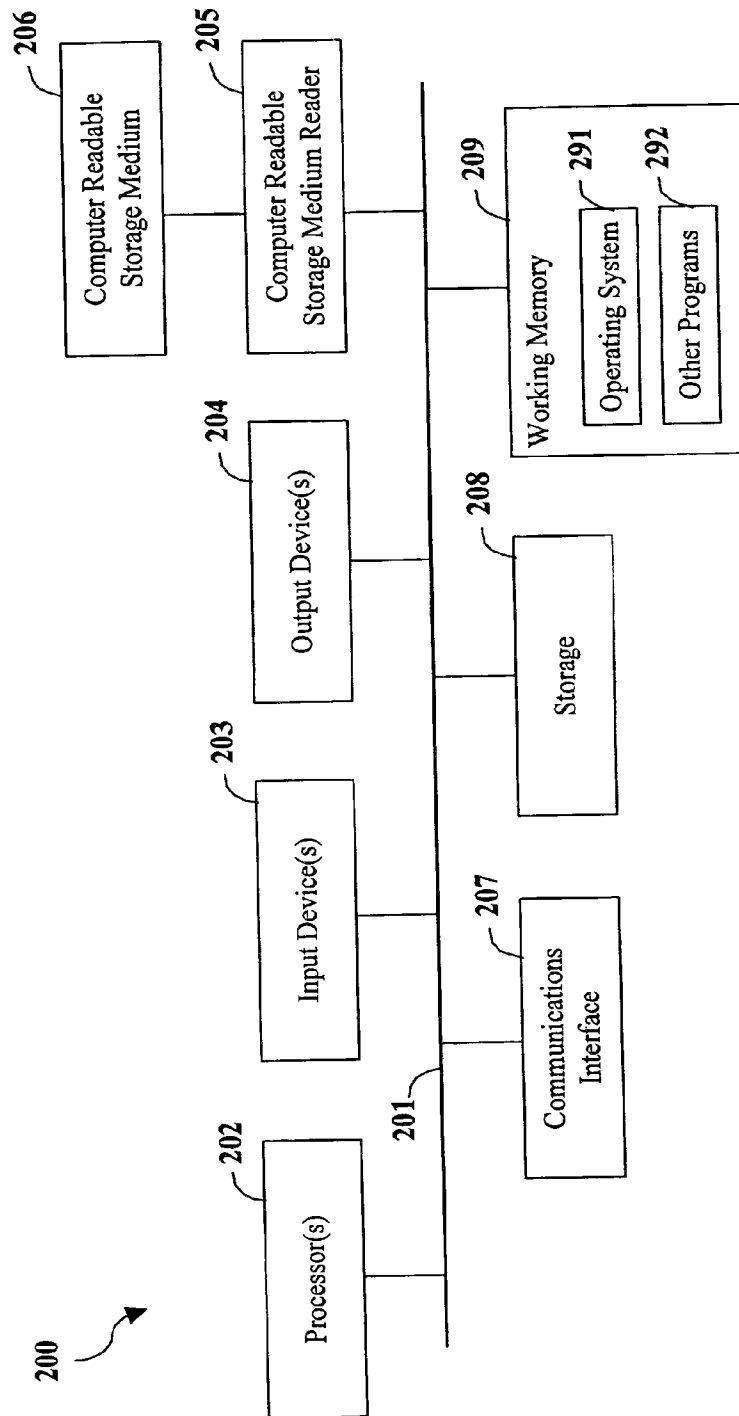


FIG. 2

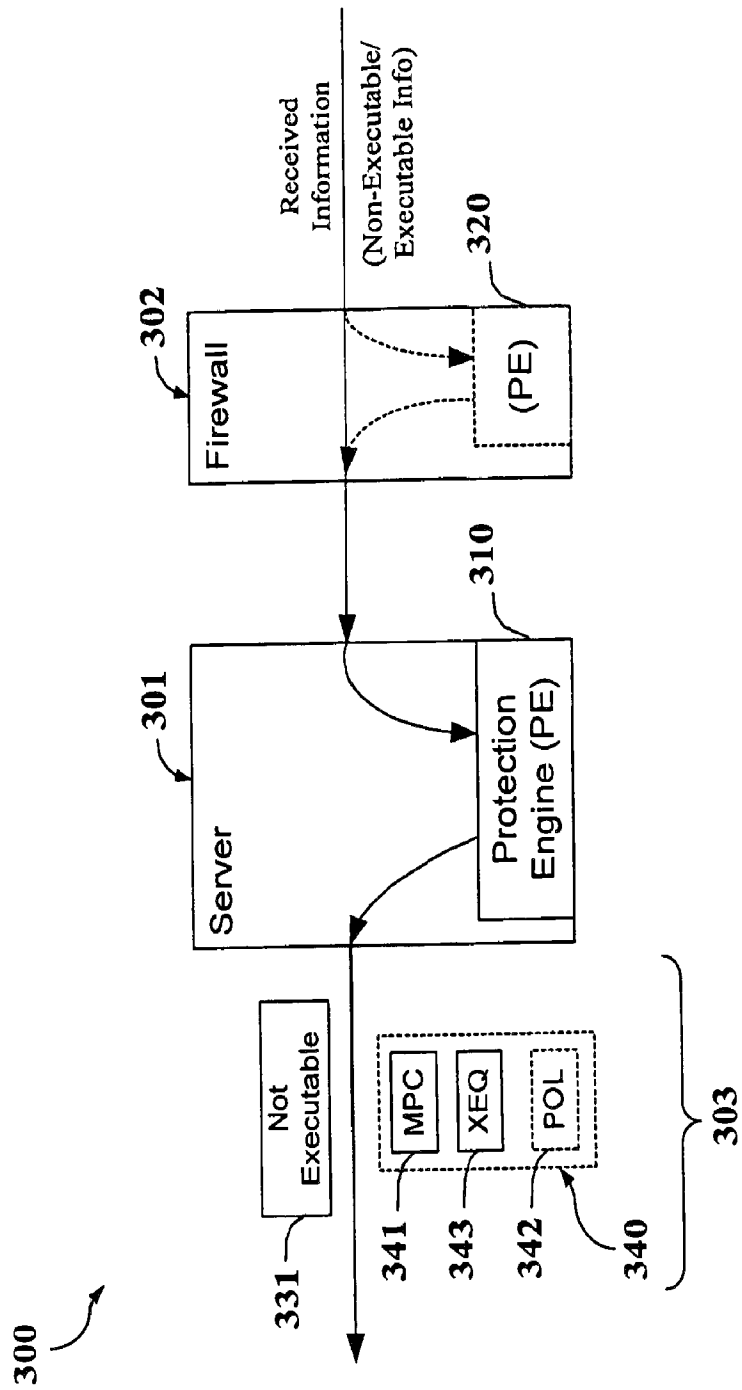


FIG. 3

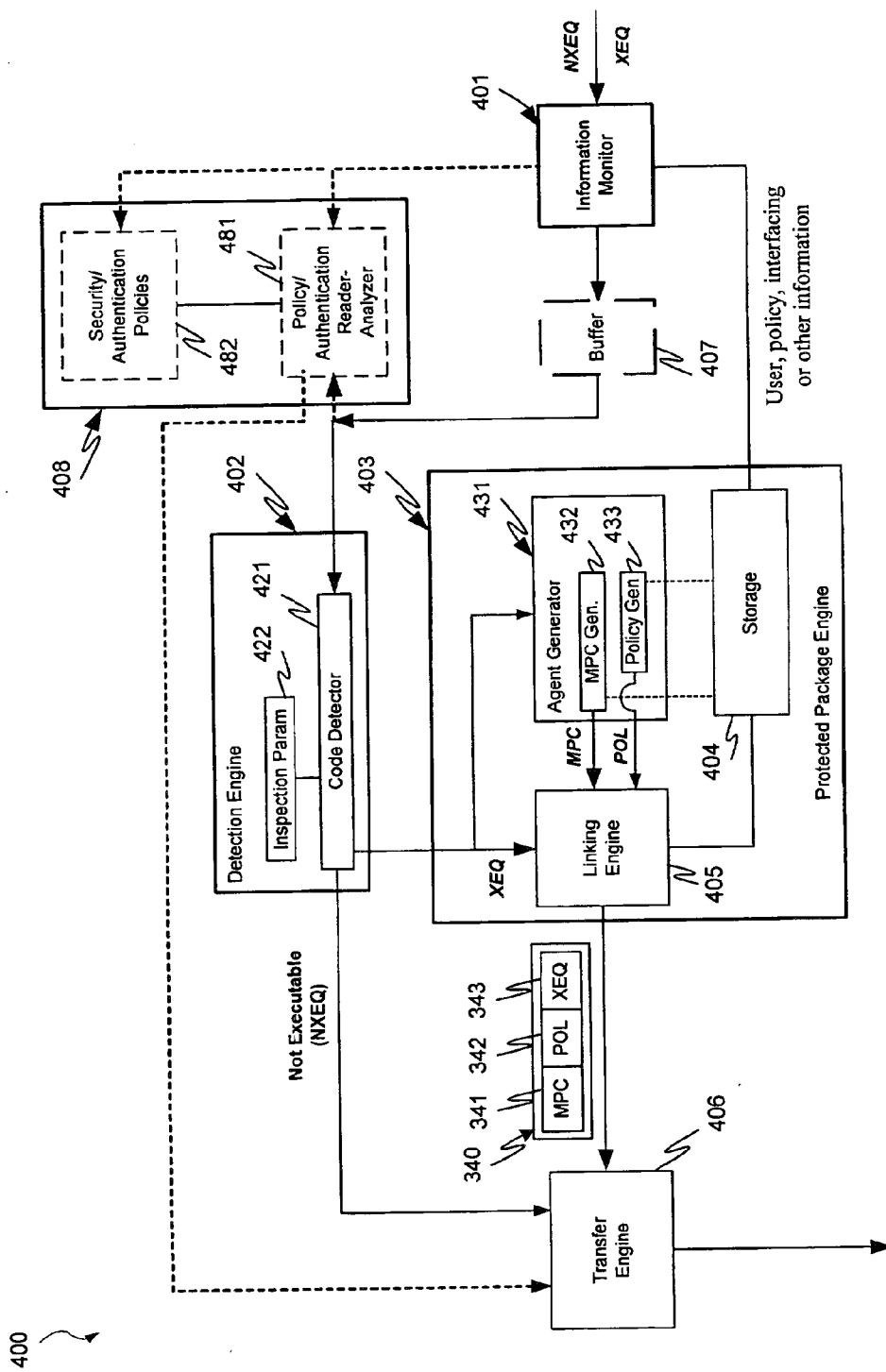


FIG. 4

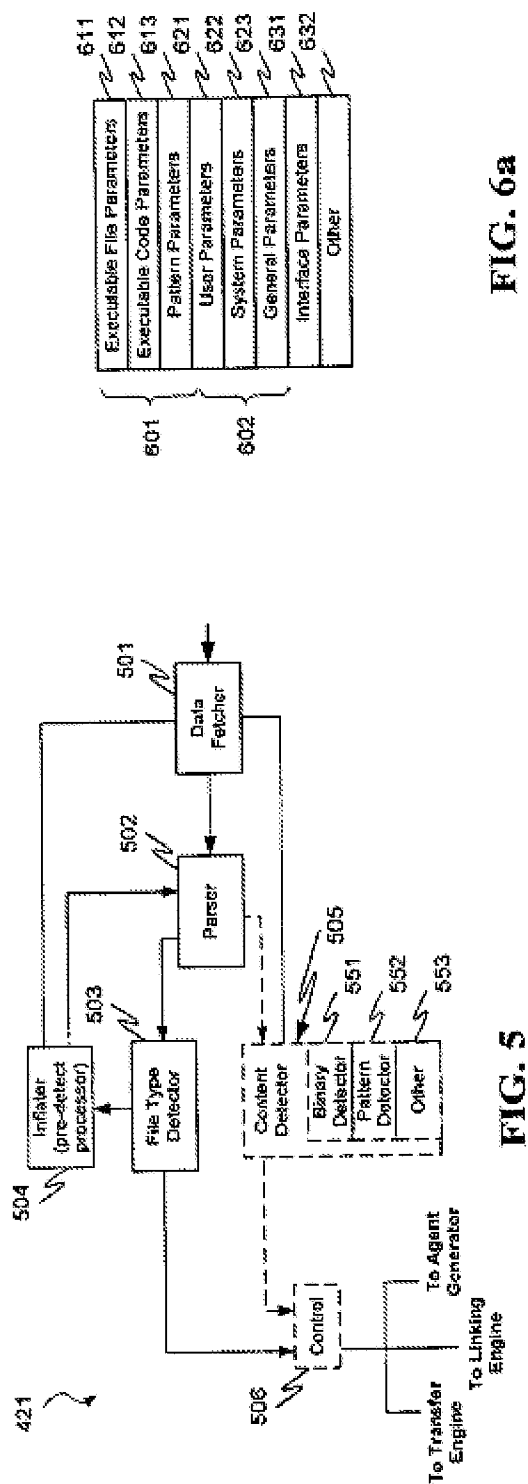
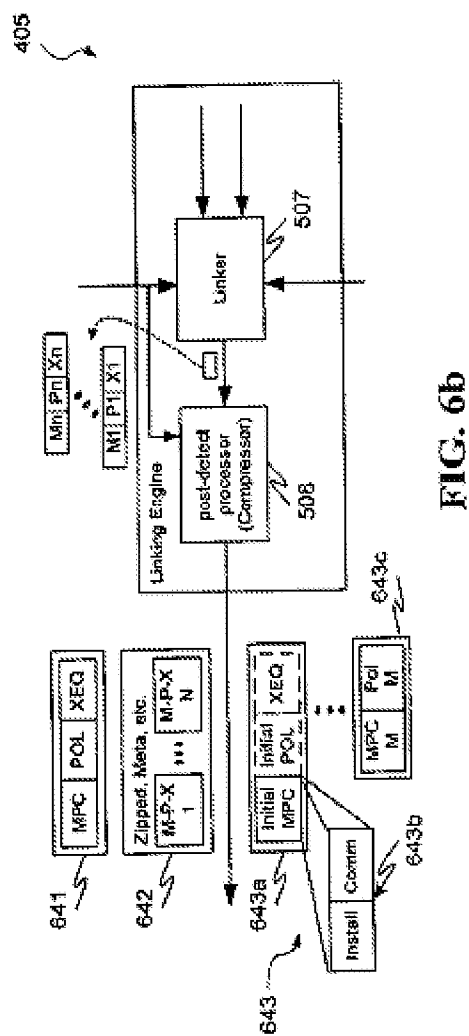


FIG. 6a



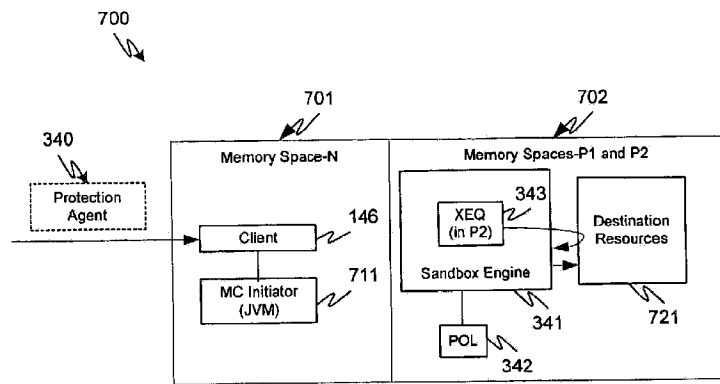


FIG. 7a

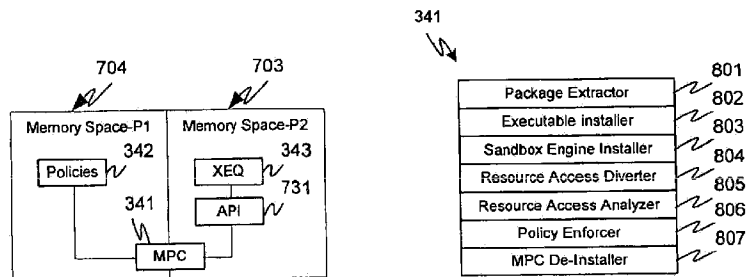


FIG. 7b

FIG. 8

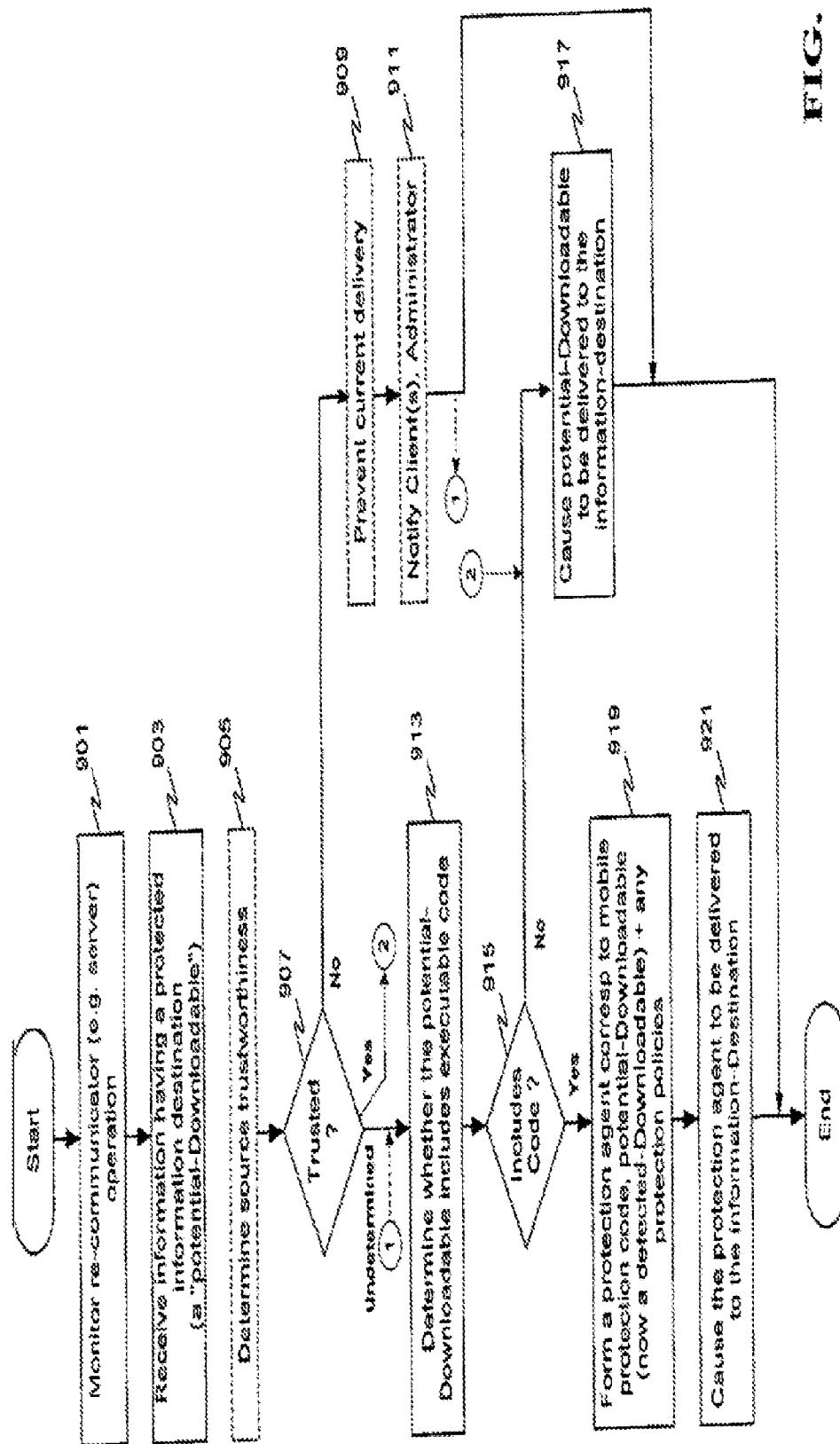


FIG. 9

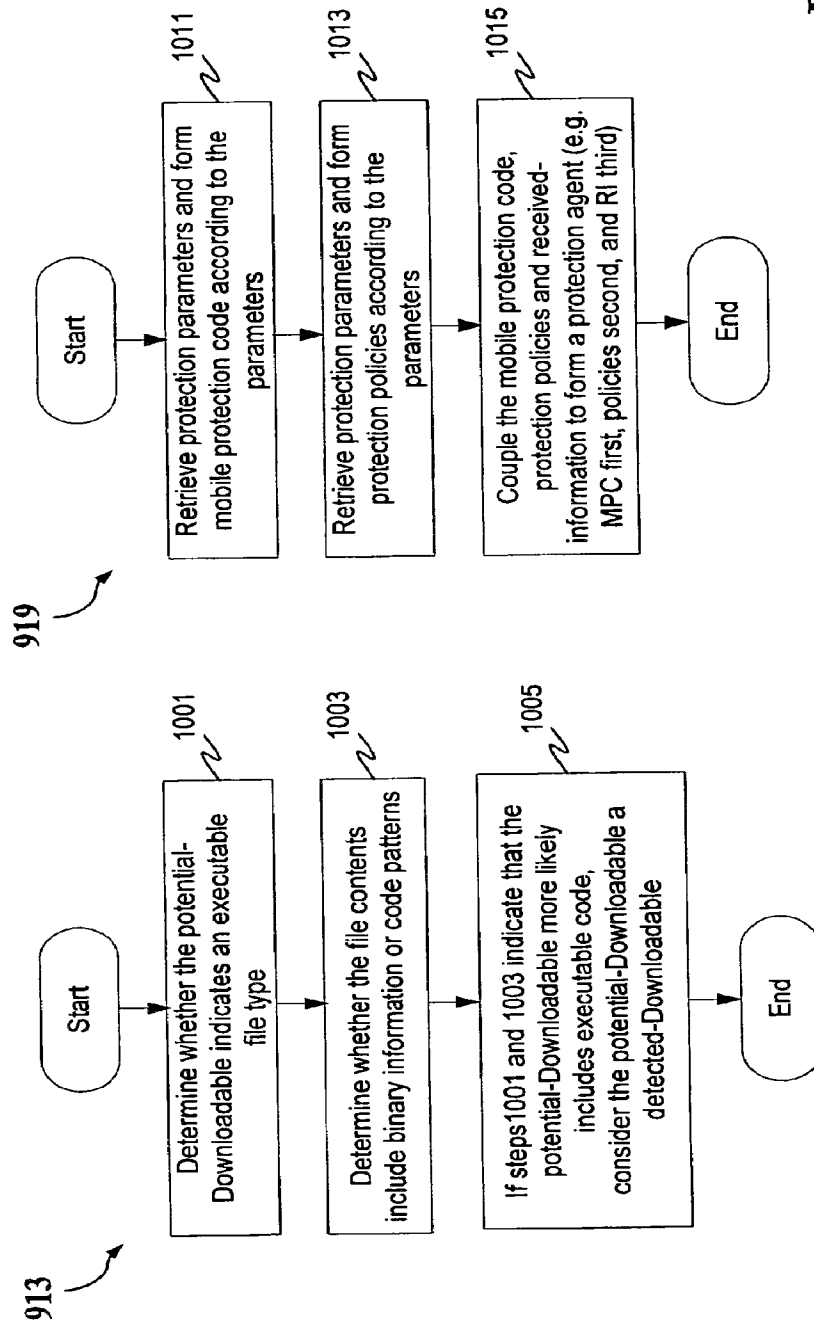


FIG. 10A

FIG. 10B

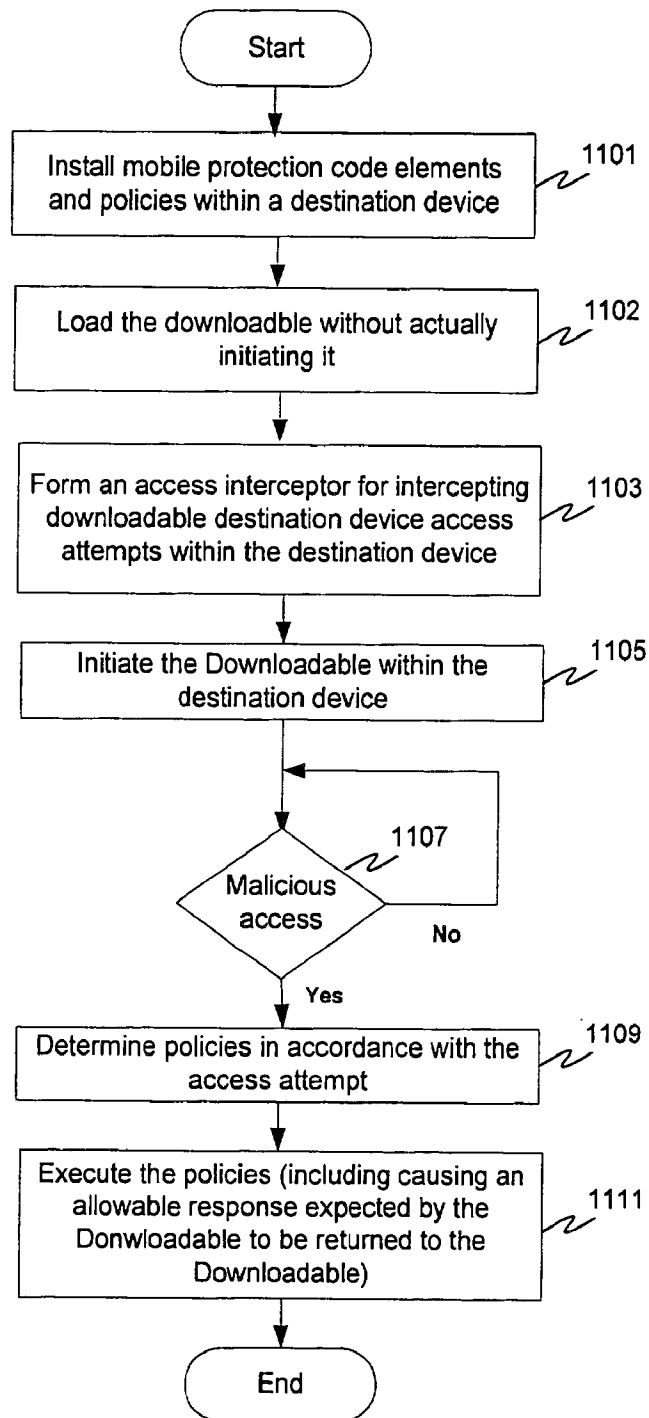


FIG. 11

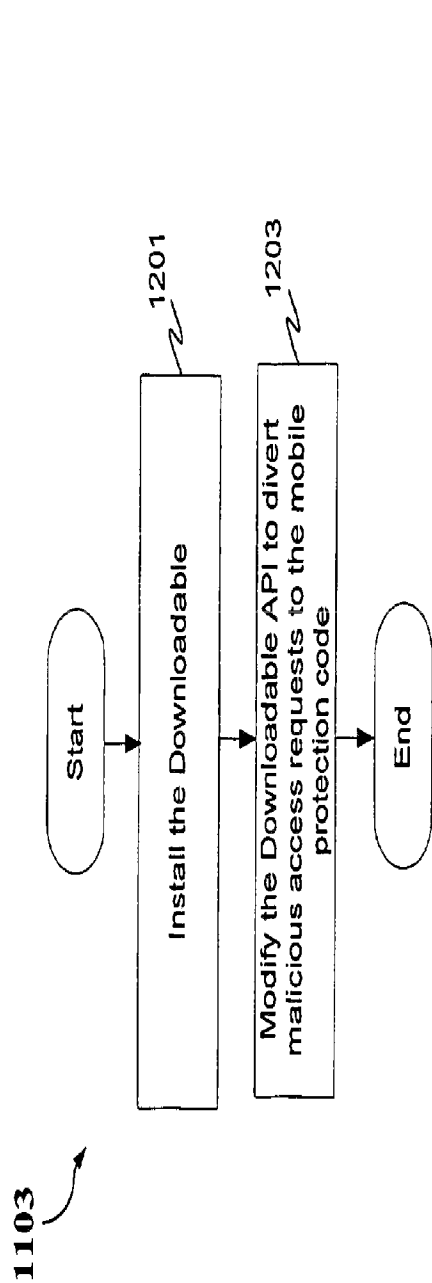


FIG. 12a

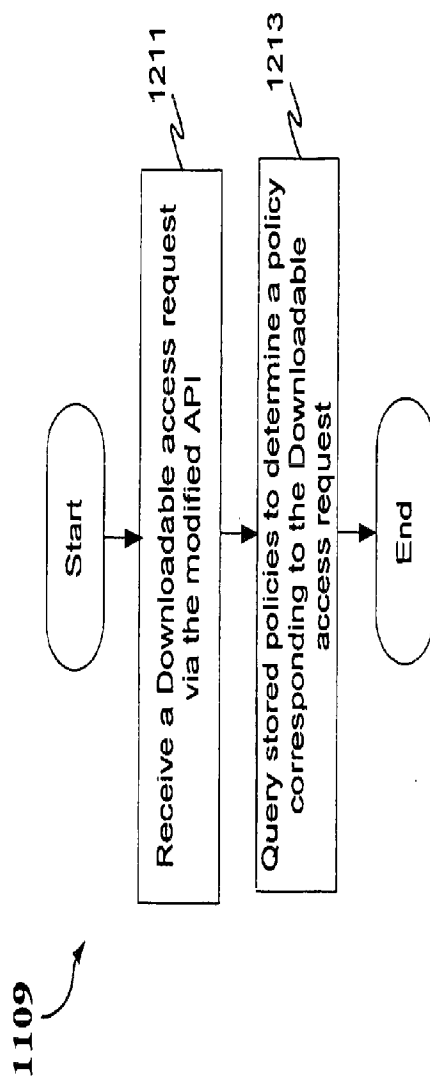


FIG. 12b

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	FIN0001-CON1-CIP1-C7-DIV1
		Application Number	
Title of Invention	Malicious Mobile Code Runtime Monitoring System and Methods		
The application data sheet is part of the provisional or nonprovisional application for which it is being submitted. The following form contains the bibliographic data arranged in a format specified by the United States Patent and Trademark Office as outlined in 37 CFR 1.76. This document may be completed electronically and submitted to the Office in electronic format using the Electronic Filing System (EFS) or the document may be printed and included in a paper filed application.			

Secrecy Order 37 CFR 5.2

☐	Portions or all of the application associated with this Application Data Sheet may fall under a Secrecy Order pursuant to 37 CFR 5.2 (Paper filers only. Applications that fall under Secrecy Order may not be filed electronically.)

Inventor Information:

Inventor 1					Remove
Legal Name					
Prefix	Given Name	Middle Name	Family Name	Suffix	
	Yigal	Mordechai	Edery		
Residence Information (Select One) ☐ US Residency ☒ Non US Residency ☐ Active US Military Service					
City	Pardesia	Country of Residence i	IL		

Mailing Address of Inventor:

Address 1	Hashikma 11				
Address 2	POS 1115				
City	Pardesia	State/Province	IL		
Postal Code	42815	Country i	IL		

Inventor 2					Remove
Legal Name					
Prefix	Given Name	Middle Name	Family Name	Suffix	
	Nimrod	Itzhak	Vered		
Residence Information (Select One) ☐ US Residency ☒ Non US Residency ☐ Active US Military Service					
City	Goosh Tel-Mond	Country of Residence i	IL		

Mailing Address of Inventor:

Address 1	Moshav Miameret #81				
Address 2					
City	Goosh Tel-Mond	State/Province	IL		
Postal Code	40695	Country i	IL		

Inventor 3					Remove
Legal Name					

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	FIN0001-CON1-CIP1-C7-DIV1
		Application Number	
Title of Invention	Malicious Mobile Code Runtime Monitoring System and Methods		

Prefix	Given Name	Middle Name	Family Name	Suffix
	David	R.	Kroll	
Residence Information (Select One) ☒ US Residency ☐ Non US Residency ☐ Active US Military Service				
City	San Jose	State/Province	CA	Country of Residence i US

Mailing Address of Inventor:

Address 1	4656 Kingbrook Drive			
Address 2				
City	San Jose	State/Province	CA	
Postal Code	95124	Country i	US	

Inventor 4

Remove

Legal Name

Prefix	Given Name	Middle Name	Family Name	Suffix
	Shlomo		Touboul	
Residence Information (Select One) ☐ US Residency ☒ Non US Residency ☐ Active US Military Service				
City	Kefar-Haim	Country of Residence i	IL	

Mailing Address of Inventor:

Address 1	9 Beher Hamaim Street			
Address 2				
City	Kefar-Haim	State/Province	IL	
Postal Code	4294500	Country i	IL	

All Inventors Must Be Listed - Additional Inventor Information blocks may be generated within this form by selecting the **Add** button.

Add

Correspondence Information:

Enter either Customer Number or complete the Correspondence Information section below.
For further information see 37 CFR 1.33(a).

☐ An Address is being provided for the correspondence Information of this application.

Customer Number	115222		
Email Address		Add Email	Remove Email

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	FIN0001-CON1-CIP1-C7-DIV1
		Application Number	
Title of Invention	Malicious Mobile Code Runtime Monitoring System and Methods		

Application Information:

Title of the Invention	Malicious Mobile Code Runtime Monitoring System and Methods		
Attorney Docket Number	FIN0001-CON1-CIP1-C7-DIV1	Small Entity Status Claimed	☐
Application Type	Nonprovisional		
Subject Matter	Utility		
Total Number of Drawing Sheets (if any)	10	Suggested Figure for Publication (if any)	

Filing By Reference :

Only complete this section when filing an application by reference under 35 U.S.C. 111(c) and 37 CFR 1.57(a). Do not complete this section if application papers including a specification and any drawings are being filed. Any domestic benefit or foreign priority information must be provided in the appropriate section(s) below (i.e., "Domestic Benefit/National Stage Information" and "Foreign Priority Information").

For the purposes of a filing date under 37 CFR 1.53(b), the description and any drawings of the present application are replaced by this reference to the previously filed application, subject to conditions and requirements of 37 CFR 1.57(a).

Application number of the previously filed application	Filing date (YYYY-MM-DD)	Intellectual Property Authority or Country

Publication Information:

☐ Request Early Publication (Fee required at time of Request 37 CFR 1.219)
☐ Request Not to Publish. I hereby request that the attached application not be published under 35 U.S.C. 122(b) and certify that the invention disclosed in the attached application has not and will not be the subject of an application filed in another country, or under a multilateral international agreement, that requires publication at eighteen months after filing.

Representative Information:

Representative information should be provided for all practitioners having a power of attorney in the application. Providing this information in the Application Data Sheet does not constitute a power of attorney in the application (see 37 CFR 1.32). Either enter Customer Number or complete the Representative Name section below. If both sections are completed the customer Number will be used for the Representative Information during processing.			
Please Select One:	☒ Customer Number	☐ US Patent Practitioner	☐ Limited Recognition (37 CFR 11.9)
Customer Number	115222		

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	FIN0001-CON1-CIP1-C7-DIV1
		Application Number	
Title of Invention	Malicious Mobile Code Runtime Monitoring System and Methods		

Domestic Benefit/National Stage Information:

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121, or 365(c) or indicate National Stage entry from a PCT application. Providing this information in the application data sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78.

When referring to the current application, please leave the application number blank.

Prior Application Status		Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)
	Division of	14/619363	2015-02-11
Prior Application Status		Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)
14/619363	Continuation of	14/155835	2014-01-15
Prior Application Status		Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)
14/155835	Continuation of	13/290708	2011-11-07
Prior Application Status		Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)
13/290708	Continuation of	12/471942	2009-05-26
Prior Application Status		Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)
12/471942	Continuation of	11/370114	2006-03-07
Prior Application Status		Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)
11/370114	Continuation of	09/861229	2001-05-17
Prior Application Status		Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)
09/861229	Continuation in part of	09/551302	2000-04-18
Prior Application Status		Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)
09/551302	Continuation of	08/790097	1997-01-29
Prior Application Status		Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)
08/790097	Claims benefit of provisional	60/030639	1996-11-08
Prior Application Status		Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)
09/861229	Claims benefit of provisional	60/205591	2000-05-17
Prior Application Status		Remove	

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	FIN0001-CON1-CIP1-C7-DIV1
		Application Number	
Title of Invention	Malicious Mobile Code Runtime Monitoring System and Methods		

Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)
09/861229	Continuation in part of	09/539667	2000-03-30
Prior Application Status		Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)
09539667	Continuation of	08/964388	1997-11-06
Prior Application Status		Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)
08/964388	Claims benefit of provisional	60/030639	1996-11-08
Additional Domestic Benefit/National Stage Data may be generated within this form by selecting the Add button. Add			

Foreign Priority Information:

This section allows for the applicant to claim priority to a foreign application. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55(d). When priority is claimed to a foreign application that is eligible for retrieval under the priority document exchange program (PDX) the information will be used by the Office to automatically attempt retrieval pursuant to 37 CFR 1.55(h)(1) and (2). Under the PDX program, applicant bears the ultimate responsibility for ensuring that a copy of the foreign application is received by the Office from the participating foreign intellectual property office, or a certified copy of the foreign priority application is filed, within the time period specified in 37 CFR 1.55(g)(1).

[Remove](#)

Application Number	Country ⁱ	Filing Date (YYYY-MM-DD)	Access Code ^j (if applicable)
Additional Foreign Priority Data may be generated within this form by selecting the Add button. Add			

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications

This application (1) claims priority to or the benefit of an application filed before March 16, 2013 and (2) also contains, or contained at any time, a claim to a claimed invention that has an effective filing date on or after March 16, 2013.

☐ NOTE: By providing this statement under 37 CFR 1.55 or 1.78, this application, with a filing date on or after March 16, 2013, will be examined under the first inventor to file provisions of the AIA.

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	FIN0001-CON1-CIP1-C7-DIV1
		Application Number	
Title of Invention	Malicious Mobile Code Runtime Monitoring System and Methods		

Authorization to Permit Access:

☐ Authorization to Permit Access to the Instant Application by the Participating Offices

If checked, the undersigned hereby grants the USPTO authority to provide the European Patent Office (EPO), the Japan Patent Office (JPO), the Korean Intellectual Property Office (KIPO), the World Intellectual Property Office (WIPO), and any other intellectual property offices in which a foreign application claiming priority to the instant patent application is filed access to the instant patent application. See 37 CFR 1.14(c) and (h). This box should not be checked if the applicant does not wish the EPO, JPO, KIPO, WIPO, or other intellectual property office in which a foreign application claiming priority to the instant patent application is filed to have access to the instant patent application.

In accordance with 37 CFR 1.14(h)(3), access will be provided to a copy of the instant patent application with respect to: 1) the instant patent application-as-filed; 2) any foreign application to which the instant patent application claims priority under 35 U.S.C. 119(a)-(d) if a copy of the foreign application that satisfies the certified copy requirement of 37 CFR 1.55 has been filed in the instant patent application; and 3) any U.S. application-as-filed from which benefit is sought in the instant patent application.

In accordance with 37 CFR 1.14(c), access may be provided to information concerning the date of filing this Authorization.

Applicant Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

Applicant 1

[Remove](#)

If the applicant is the inventor (or the remaining joint inventor or inventors under 37 CFR 1.45), this section should not be completed. The information to be provided in this section is the name and address of the legal representative who is the applicant under 37 CFR 1.43; or the name and address of the assignee, person to whom the inventor is under an obligation to assign the invention, or person who otherwise shows sufficient proprietary interest in the matter who is the applicant under 37 CFR 1.46. If the applicant is an applicant under 37 CFR 1.46 (assignee, person to whom the inventor is obligated to assign, or person who otherwise shows sufficient proprietary interest) together with one or more joint inventors, then the joint inventor or inventors who are also the applicant should be identified in this section.

[Clear](#)

☒ Assignee
 ☐ Legal Representative under 35 U.S.C. 117
 ☐ Joint Inventor

☐ Person to whom the inventor is obligated to assign.
 ☐ Person who shows sufficient proprietary interest

If applicant is the legal representative, indicate the authority to file the patent application, the inventor is:

Name of the Deceased or Legally Incapacitated Inventor :

If the Applicant is an Organization check here. ☒

Organization Name

Finjan, Inc.

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	FIN0001-CON1-CIP1-C7-DIV1
		Application Number	
Title of Invention	Malicious Mobile Code Runtime Monitoring System and Methods		

Mailing Address Information:			
Address 1	2000 University Avenue		
Address 2	Suite 600		
City	East Palo Alto	State/Province	CA
Country i	US	Postal Code	94303
Phone Number		Fax Number	
Email Address			
Additional Applicant Data may be generated within this form by selecting the Add button. Add			

Assignee Information including Non-Applicant Assignee Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

Assignee 1			
Complete this section if assignee information, including non-applicant assignee information, is desired to be included on the patent application publication. An assignee-applicant identified in the "Applicant Information" section will appear on the patent application publication as an applicant. For an assignee-applicant, complete this section only if identification as an assignee is also desired on the patent application publication.			
Remove			
If the Assignee or Non-Applicant Assignee is an Organization check here. ☒			
Organization Name	Finjan, Inc.		
Mailing Address Information For Assignee including Non-Applicant Assignee:			
Address 1	2000 University Avenue		
Address 2	Suite 600		
City	East Palo Alto	State/Province	CA
Country i	US	Postal Code	94303
Phone Number		Fax Number	
Email Address			
Additional Assignee or Non-Applicant Assignee Data may be generated within this form by selecting the Add button. Add			

Signature:

Remove

NOTE: This form must be signed in accordance with 37 CFR 1.33. See 37 CFR 1.4 for signature requirements and certifications			
Signature	/Dawn-Marie Bey/	Date (YYYY-MM-DD)	2015-06-05

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	FIN0001-CON1-CIP1-C7-DIV1		
		Application Number			
Title of Invention	Malicious Mobile Code Runtime Monitoring System and Methods				
First Name	Dawn-Marie	Last Name	Bey	Registration Number	44442
Additional Signature may be generated within this form by selecting the Add button.					Add

This collection of information is required by 37 CFR 1.76. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 23 minutes to complete, including gathering, preparing, and submitting the completed application data sheet form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these records.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

**CERTIFICATION AND REQUEST FOR PRIORITIZED EXAMINATION
UNDER 37 CFR 1.102(e) (Page 1 of 1)**

First Named Inventor:	Yigal Mordechai Edery, et al.	Nonprovisional Application Number (if known):	To Be Assigned
Title of Invention:	Malicious Mobile Code Runtime Monitoring System and Methods		

APPLICANT HEREBY CERTIFIES THE FOLLOWING AND REQUESTS PRIORITIZED EXAMINATION FOR THE ABOVE-IDENTIFIED APPLICATION.

1. The processing fee set forth in 37 CFR 1.17(i)(1) and the prioritized examination fee set forth in 37 CFR 1.17(c) have been filed with the request. The publication fee requirement is met because that fee, set forth in 37 CFR 1.18(d), is currently \$0. The basic filing fee, search fee, and examination fee are filed with the request or have been already been paid. I understand that any required excess claims fees or application size fee must be paid for the application.
2. I understand that the application may not contain, or be amended to contain, more than four independent claims, more than thirty total claims, or any multiple dependent claims, and that any request for an extension of time will cause an outstanding Track I request to be dismissed.
3. The applicable box is checked below:

I. ☒ Original Application (Track One) - Prioritized Examination under § 1.102(e)(1)

- i. (a) The application is an original nonprovisional utility application filed under 35 U.S.C. 111(a). This certification and request is being filed with the utility application via EFS-Web.
---OR---
(b) The application is an original nonprovisional plant application filed under 35 U.S.C. 111(a). This certification and request is being filed with the plant application in paper.
- ii. An executed inventor's oath or declaration under 37 CFR 1.63 or 37 CFR 1.64 for each inventor, or the application data sheet meeting the conditions specified in 37 CFR 1.53(f)(3)(i) is filed with the application.

II. ☐ Request for Continued Examination - Prioritized Examination under § 1.102(e)(2)

- i. A request for continued examination has been filed with, or prior to, this form.
- ii. If the application is a utility application, this certification and request is being filed via EFS-Web.
- iii. The application is an original nonprovisional utility application filed under 35 U.S.C. 111(a), or is a national stage entry under 35 U.S.C. 371.
- iv. This certification and request is being filed prior to the mailing of a first Office action responsive to the request for continued examination.
- v. No prior request for continued examination has been granted prioritized examination status under 37 CFR 1.102(e)(2).

Signature /Dawn-Marie Bey/	Date June 5, 2015
Name (Print/Typed) Dawn-Marie Bey	Practitioner Registration Number 44,442

Note: This form must be signed in accordance with 37 CFR 1.33. See 37 CFR 1.4(d) for signature requirements and certifications. Submit multiple forms if more than one signature is required.*

☒ *Total of 1 forms are submitted.

Electronic Patent Application Fee Transmittal				
Application Number:				
Filing Date:				
Title of Invention:		Malicious Mobile Code Runtime Monitoring System and Methods		
First Named Inventor/Applicant Name:		Yigal Mordechai Edery		
Filer:		Dawn-Marie Bey./Jeanne Paoella-Bald		
Attorney Docket Number:		FIN0001-CN1_CIP1-CN7-DIV1		
Filed as Large Entity				
Filing Fees for Track I Prioritized Examination - Nonprovisional Application under 35 USC 111(a)				
Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
Basic Filing:				
Utility application filing	1011	1	280	280
Utility Search Fee	1111	1	600	600
Utility Examination Fee	1311	1	720	720
Request for Prioritized Examination	1817	1	4000	4000
Pages:				
Claims:				
Miscellaneous-Filing:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
Publ. Fee- Early, Voluntary, or Normal	1504	1	0	0
PROCESSING FEE, EXCEPT PROV. APPLS.	1830	1	140	140
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Miscellaneous:				
Total in USD (\$)				5740

Electronic Acknowledgement Receipt

EFS ID:	22553513
Application Number:	14732188
International Application Number:	
Confirmation Number:	1047
Title of Invention:	Malicious Mobile Code Runtime Monitoring System and Methods
First Named Inventor/Applicant Name:	Yigal Mordechai Edery
Customer Number:	115222
Filer:	Dawn-Marie Bey./Jeanne Paoletta-Bald
Filer Authorized By:	Dawn-Marie Bey.
Attorney Docket Number:	FIN0001-CN1_CIP1-CN7-DIV1
Receipt Date:	05-JUN-2015
Filing Date:	
Time Stamp:	16:33:13
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	\$5740
RAM confirmation Number	3262
Deposit Account	
Authorized User	

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

File Listing:					
Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Transmittal of New Application	fin0001con1cip1con7div1_executed_utilityapplicationtransmittal.pdf	275953	no	2
			3c40a752ecfa490735b98a0d9b8c48be2ad9bde7		
Warnings:					
Information:					
2	Specification	fin0001con1cip1con7div1_specification.pdf	212903	no	35
			d37a28af752d71102e8914cd3381a6cece6b6c44		
Warnings:					
Information:					
3	Drawings-only black and white line drawings	fin0001con1cip1con7div1_figures.pdf	4007288	no	10
			e77a317bf448680e57b06cbfd07bb4be643f542c		
Warnings:					
Information:					
4	Application Data Sheet	fin0001con1cip1con7div1_executed_applicationdatasheet.pdf	1561994	no	9
			bb4766634ad966049a0248be2e72e7f179821deb		
Warnings:					
Information:					
5	TrackOne Request	fin0001con1cip1con7div1_executed_track1requestforprioritizedexamination.pdf	148761	no	1
			0f523ed704da5bbf1589c622ff3ffe03a3285ba1		
Warnings:					
Information:					
6	Fee Worksheet (SB06)	fee-info.pdf	40538	no	2
			202058987c6948c8bef303be311ab478fa540fc2		
Warnings:					
Information:					
Total Files Size (in bytes):			6247437		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : H04L 29/06	A1	(11) International Publication Number: WO 98/40993 (43) International Publication Date: 17 September 1998 (17.09.98)
(21) International Application Number: PCT/IL98/00083 (22) International Filing Date: 23 February 1998 (23.02.98) (30) Priority Data: 120420 10 March 1997 (10.03.97) IL (71) Applicant (for all designated States except US): SECURITY-7 (SOFTWARE) LTD. (IL/IL); P.O. Box 107, 20692 Yoqneam (IL). (72) Inventors; and (75) Inventors/Applicants (for US only): ELGRESSY, Doron (IL/IL); 31 Kish Street, 33531 Haifa (IL). JOSPE, Asher (IL/IL); Nurit Street 39, 42670 Natanya (IL). (74) Agents: LUZZATTO, Kfir et al.; Luzzatto & Luzzatto, P.O. Box 5352, 84152 Beer-Sheva (IL).		(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG). Published With international search report. Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.
(54) Title: METHOD AND SYSTEM FOR PREVENTING THE DOWNLOADING AND EXECUTION OF EXECUTABLE OBJECTS		
<pre> graph TD SP --> CC1 CC1 --> GW1 CC1 --> GW2 CC1 --> GW3 CC1 --> GW4 CC1 --> GW5 CC1 --> CC2 CC2 --> GW4 CC2 --> GW5 GW3 --> PC1((PC)) GW3 --> PC2((PC)) </pre>		
(57) Abstract A method for selectively preventing the downloading and execution of undesired Executable Objects in a computer, comprising the steps of: (a) providing one or more Control Centers (CC1, CC2), each connected to one or more gateways (GW1... GW5) located between a LAN (PC) and an external computer communication network; (b) providing means coupled to each of said gateways (GW1... GW5), to detect Executable Objects reaching said gateway (GW1... GW5), to analyze the header of each of said Executable Objects, and to determine the resources of the computer (PC) that the Executable Object needs to utilize; (c) providing means coupled to each of said gateways (GW1... GW5), to store a user's Security Policy representing the resources, or combination of resources, that the user allows or does not allow an Executable Object to utilize within its LAN, wherein the Security Policy is received from and/or stored in each of said one or more Control Centers (CC1, CC2); (d) when an Executable Object is detected at the gateway; (GW1... GW5) comparing the resources of the computer (PC) that the Executable Object needs to utilize with the Security Policy.		

FOR THE PURPOSES OF INFORMATION ONLY

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

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

-1-

METHOD AND SYSTEM FOR PREVENTING THE DOWNLOADING
AND EXECUTION OF EXECUTABLE OBJECTS

Field of the Invention

The present invention relates to the security management of computer networks. More particularly, the invention relates to methods and systems for preventing the downloading and execution of undesirable Executable Objects in a workstation of a computer network.

Background of the Invention

The Internet has developed very much both in respect of its contents and of the technology employed, since it began a few years ago. In the early days of the Internet, web sites included text only, and after a while graphics was introduced. As the Internet developed, many compressed standards, such as pictures, voice and video files, were developed and with them programs used to play them (called "players"). Initially, such files were downloaded to the user's workstation only upon his request, and extracted only by the appropriate player, and after a specific order from the user.

When, in the natural course of the development of the World Wide Web the search for a way to show nicer, interactive and animated Web Pages began, Sun Microsystems Inc. developed Java - a language that allows the webmaster to write a program, a list of commands - Network Executables -

-2-

that will be downloaded to the user workstation without his knowledge, and executed by his browser at his workstation. The executables are used, e.g., to provide photographic animation and other graphics on the screen of the web surfer. Such executables have some ways approaching the user workstation's resources, which lead to a great security problem. Although some levels of security were defined in the Java language, it was very soon that a huge security hole was found in the language.

Since Java was developed, Microsoft developed ActiveX, which is another Network Executable format, also downloaded into the workstation. ActiveX has also security problems of the same kind.

The Internet has been flooded with "Network Executables" which may be downloaded -- deliberately or without the knowledge of the users -- into workstations within organizations. These codes generally contain harmless functions. Although usually safe, they may not meet the required security policy of the organization.

Once executed, codes may jam the network, cause considerable irreversible damage to the local database, workstations and servers, or result in unauthorized retrieval of information from the servers/workstations. Such elements may appear on Java applets, ActiveX components, DLLs and other object codes, and their use is increasing at an unparalleled pace. The majority of these small programs are downloaded

-3-

into the organization unsolicited and uncontrolled. The enterprise has no way of knowing about their existence or execution and there is no system in place for early detection and prevention of the codes from being executed.

The security problem was solved partially by the browser manufactures which allow the user to disable the use of executables. Of course this is not a reasonable solution, since all the electronic commerce and advertising are based on the use of executables. The security problem is much more serious once such an executable can approach the enterprise servers, databases and other workstations.

It is therefore clear that it is highly needed to be able to prevent undesirable Executable Objects from infiltrating the LAN/WAN in which we work and, ultimately, our workstation and server. However, so far the art has failed to provide comprehensive solutions which are safe and quick enough to be practically useful. Systems such as "Firewall" or "Finjan", distributed for use by Internet users, provide only partial solutions and, furthermore, are difficult to install and to update.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide a comprehensive method for selectively preventing the downloading and execution of undesired

-4-

Executable Objects in a computer, which overcomes the aforesaid drawbacks of prior art systems.

It is another object of the invention to provide such a system which is easy to install and which can be quickly and easily updated.

It is a further object of the invention to provide such a method which can be used with a large number of gateways, LAN's and workstations.

It is yet another object of the invention to provide such a security management system which is independent of the physical infrastructure and network layout.

It is a further object of the invention to provide a system which analyzes the executables "on the fly", and does not hinder the downloading and the operation of harmless executables.

It is yet a further object of the invention to provide a system of the kind described above, which operates as a central security system to which peripheral gateways may be added as needed, to provide a simple, dynamically growing security system.

It is furthermore an object of the invention to provide a central system which permits to define sub-groups of users, each group being subject to a different security policy.

Also encompassed by the invention is a computer system which utilizes the method of the invention.

Other advantages and objects of the invention will become apparent as the description proceeds.

The method for selectively preventing the downloading and execution of undesired Executable Objects in a computer, according to the invention, comprises the steps of:

- (a) providing one or more Control Centers, each connected to one or more gateways located between a LAN and an external computer communication network;

- (b) providing means coupled to each of said gateways, to detect Executable Objects reaching said gateway, to analyze the header of each of said Executable Objects, and to determine the resources of the computer that the Executable Object needs to utilize;

- (c) providing means coupled to each of said gateways, to store a user's Security Policy representing the resources, or combination of resources, that the user allows or does not allow an Executable Object to

-6-

utilize within its LAN, wherein the Security Policy is received from and/or stored in each of said one or more Control Centers;

(d) when an Executable Object is detected at the gateway;

1. analyzing the header of said Executable Object;
2. determining the resources of the computer that the Executable Object needs to utilize;
3. comparing the resources of the computer that the Executable Object needs to utilize with the Security Policy and;

(i) if the resources of the computer that the Executable Object needs to utilize are included in the list of the resources allowed for use by the Security Policy, allowing the Executable Object to pass through the gateway and to reach the computer which has initiated its downloading; and

(ii) if the resources of the computer that the Executable Object needs to utilize are included in the list of the resources prohibited for use by the Security Policy, preventing the Executable Object from passing through the gateway, thereby preventing it from reaching the computer which has initiated its downloading.

A Control Center (CC) may be a central control unit, e.g., a PC or other computer, which is connected to a plurality of gateways, and which updates the memory means containing relevant data, e.g., the Security Policy. As will be understood from the description to follow, once the CC is updated, e.g., by the addition of an additional limitation to the Security Policy, all gateways are updated at once. The use of the CC to control the operation of the security elements of the gateways obviates the need (which exists in prior art systems) to update each gateway every time that a change in policy is made.

A LAN (Local Area Network) may be (but is not limited to), e.g., a network of computers located in an office or building. The LAN is typically connected to outside communications networks, such as the World Wide Web, or to more limited LANs, e.g., of a client or supplier, through one or more gateways. The larger the organization, the larger the number of gateways employed, in order to keep communications at a reasonable speed.

Generally speaking, a LAN can also be made of a plurality of smaller LANs, located geographically nearby or far apart, but even if small LANs are found within the same organization, the security requirements may vary from one department to the other, and it may be necessary to keep high security levels, including preventing Executables from migrating from one department to the other, even within the same organization.

The means coupled to each of said gateways, to detect Executable Objects reaching said gateway, to analyze the header of each of said Executable Objects, and to determine the resources of the computer that the Executable Object needs to utilize may be of many different types. Typically, the executable object is "trapped" and analyzed at the gateway by listening on the communication line to the TCP/IP protocol, as well as to the object transfer protocols, such as SMTP, HTTP, FTP, etc. Hooking into the communication line and extracting the contents of the header of the executable object are steps which are understood by the skilled person, and which can be effected by means of conventional programming, and they are therefore not described herein in detail, for the sake of brevity.

Once the header of the Executable Object (EO) has been analyzed, comparing the resources of the computer that the EO needs to utilize with the Security Policy can be easily done, e.g., by comparing them with a look-up table provided to the gateway by the CC, which represents the Security Policy. Comparison can also be carried out against the data stored in the CC, and in such a case specific memory means and comparing means may not be necessary in the gateway. However, speed and performance considerations will often dictate that such operations be carried out at the gateway itself.

The gateway must be installed in each Internet server within the organization. It comprises a small real time database which contains all the relevant operational information for the gateway. The gateway "listens" to the data being transferred between the enterprise and the Internet. It knows when an object is coming into the LAN, it analyzes it and compares it with the Security Policy to decide what action is to be taken.

According to a preferred embodiment of the invention, as stated, if the resources of the computer that the Executable Object needs to utilize are included in the list of the resources allowed for use by the Security Policy, no steps are taken by the system to prevent the Executable Object from passing through the gateway and reaching the computer which has initiated its downloading. However, if the resources of the computer that the Executable Object needs to utilize are included in the list of the resources prohibited for use by the Security Policy, steps will be taken to prevent the Executable Object from passing through the gateway. Such steps may include, e.g., re-routing the executable to a destination outside the gateway, canceling or garbling part of it, so as to make it inoperative, etc.

The invention is not limited to any specific EO. However, according to a preferred embodiment of the invention, the system analyzes EO's including, *inter alia*, Java Applets, Active-X, OCX, Win32 Executables,

-10-

DLLs, or the like executable objects. However, as will be apparent to the skilled person, EO's are constantly developed, and the invention is by no means intended to be limited to the use with specific EOs, and the actual nature of the EO is not of critical importance.

According to another preferred embodiment of the invention, the method further comprises the steps of:

- (1) when an undesirable Executable Object is detected at a gateway, providing an identifying value therefrom, and notifying all gateways thereof; and
- (2) providing memory means and suitable identity verification means, coupled to each gateway, to identify undesirable Executable Objects already analyzed by another gateway, and from preventing it from passing the gateway.

Notifying the other gateways of the existence of undesirable EO's is important inasmuch as this procedure may save considerable time if the EO reaches another gateway, which then does not need to analyze it in detail, to determine the contents of its header, but may decide not to allow its passage by a simpler and quicker procedure, such as checksum.

When more than one Control Centers are provided, Security Policies can be disseminated from one main Control Center to the remaining Control

Centers, and each Control Center, in turn, controls the operation of the gateways connected to it.

According to a preferred embodiment of the invention, each Control Center and each group of gateways is provided with its own individual Security Policy, which may be the same or different from the Security Policy received from the main Control Center. Of course, each subordinate Control Center may add additional limitations to the Security Policy received from the main Control Center, by the addition of resources to the list of those the use of which is not allowed, but according to a preferred embodiment of the invention, it may not remove limitations from the list of limited resources contained in the Security Policy distributed by the main Control Center.

According to a preferred embodiment of the invention, when the system is first installed on the network, the person in charge of security (called hereinafter "CSO" - Chief Security Officer) defines the identity of other Security Officers (SO) who can log-in to the Control Center and make changes in Security Policies. The CSO can define different levels of authority at which the various SOs can operate and make changes to security policies. Other SOs can make changes in the Control Center only if allowed to do so by the CSO, and those changes can affect only clients hierarchically found under their own responsibility. Such changes can only

-12-

tighten their client's security policy, with respect to the basic Security Policy, but not loosen it.

Also encompassed by the invention is a computer system comprising one or more LANs, each LAN being connected to an outside computer or computer network through one or more gateways, comprising:

(a) one or more Control Centers, each Control Centers being connected to one or more gateways located between a LAN and an external computer communication network;

(b) means coupled to each of said gateways, to detect Executable Objects reaching said gateway, to analyze the header of each of said Executable Objects, and to determine the resources of the computer that the Executable Object needs to utilize;

(c) means coupled to each of said gateways, to store a user's Security Policy representing the resources, or combination of resources, that the user allows or does not allow an Executable Object to utilize within its LAN, wherein the Security Policy is received from and/or stored in each of said one or more Control Centers;

(d) means, provided at, or coupled to, each gateway:

1. to analyze the header of when an Executable Object which is detected at the gateway;
2. to determine the resources of the computer that the Executable Object needs to utilize;

-13-

3. to compare the resources of the computer that the Executable Object needs to utilize with the Security Policy and;

(i) means to allow the Executable Object to pass through the gateway and to reach the computer which has initiated its downloading, if the resources of the computer that the Executable Object needs to utilize are included in the list of the resources allowed for use by the Security Policy; and

(ii) means for preventing the Executable Object from passing through the gateway, thereby preventing it from reaching the computer which has initiated its downloading, if the resources of the computer that the Executable Object needs to utilize are included in the list of the resources prohibited for use by the Security Policy.

The computer system may also comprise, in addition to the means detailed under (d)3 above, also:

(iii) means for alerting the Security Officers that a given type of Executable Object has entered the gateway; and

(iv) means for storing information pertaining to a given Executable Object according to the Security Policy.

Brief Description of the Drawings

In the drawings:

Fig. 1 is a schematic representation of a system according to the invention;

Fig. 2 schematically shows an Executable Object; and

Fig. 3 illustrates the screening function of the gateway operated according to the invention.

Detailed Description of Preferred Embodiments

Looking now at Fig. 1, a possible system is schematically shown, which consists of a main Control Center (CC1), and a subordinate Control Center (CC2). Each CC is connected to a plurality of gateways. The main Control Center (CC1) receives data on the Security Policy (SP) from the operator, and immediately proceeds to update the information in gateways GW1 through GW3, and Control Center CC2 which, in turn, updates GW4 and GW5, including any additional limitations which are set in CC2. Each gateway services a plurality of workstations, typically personal computers. Two such workstations, indicated by PC, are shown in Fig. 1 as being connected to GW3, the remaining workstations not being shown, for the sake of simplicity.

Fig. 2 schematically shows an EO (EO1), which has a header from the analysis of which it can be seen that it needs, in order to function, to use resources x, y, z and w. EO1 is shown in Fig. 3, together with an

-15-

additional EO (EO2) and a gateway GW, as seen in the figure. The gateway detects that EO1 needs to utilize resources x and z, which are prohibited according to the Security Policy. Accordingly, EO1 is not allowed to pass the gateway. On the contrary, EO2, which only needs to utilize resources y and w, which are permitted by the Security Policy, is allowed to proceed and to pass the gateway, toward its destination (viz., the workstation which has asked for it).

When an applet enters the LAN it has to declare which workstation within the organization it has to reach. The allowability of the destination is to be checked, since it is possible that a given applet cannot reach one workstation, with a high security level, but can reach another workstation, with a lower security level. Furthermore, the system may change the levels of security on the basis of other considerations, such as the time of the day, the day of the week, etc.

All the above description of preferred embodiments has been provided for the sake of illustration, and is not intended to limit the invention in any way, except as defined by the claims. Many modifications may be effected in the invention. For instance, any number and distribution of Control Centers, Gateways and PCs can be provided, and different Security Policies can be provided by the users. Additionally, a variety of Executable Objects can be monitored, on different infranets and intranets, all without exceeding the scope of the invention.

Claims

1. A method for selectively preventing the downloading and execution of undesired Executable Objects in a computer, comprising the steps of:

(a) providing one or more Control Centers, each connected to one or more gateways located between a LAN and an external computer communication network;

(b) providing means coupled to each of said gateways, to detect Executable Objects reaching said gateway, to analyze the header of each of said Executable Objects, and to determine the resources of the computer that the Executable Object needs to utilize;

(c) providing means coupled to each of said gateways, to store a user's Security Policy representing the resources, or combination of resources, that the user allows or does not allow an Executable Object to utilize within its LAN, wherein the Security Policy is received from and/or stored in each of said one or more Control Centers;

(d) when an Executable Object is detected at the gateway:

1. analyzing the header of said Executable Object;
2. determining the resources of the computer that the Executable Object needs to utilize;
3. comparing the resources of the computer that the Executable Object needs to utilize with the Security Policy and;

-17-

- (i) if the resources of the computer that the Executable Object needs to utilize are included in the list of the resources allowed for use by the Security Policy, allowing the Executable Object to pass through the gateway and to reach the computer which has initiated its downloading; and
- (ii) if the resources of the computer that the Executable Object needs to utilize are included in the list of the resources prohibited for use by the Security Policy, preventing the Executable Object from passing through the gateway, thereby preventing it from reaching the computer which has initiated its downloading.

2. A method according to claim 1, further comprising, in addition to the means of claim 1(d)3:

- (iii) means for alerting the Security Officers that a given type of Executable Object has entered the gateway; and
- (iv) means for storing information pertaining to a given Executable Object according to the Security Policy.

-18-

3. A method according to claim 1, wherein the Executable Object is selected from Java Applets, Active-X, OCX, Win32 Executables, DLLs, or the like executable objects.
4. A method according to any one of claims 1 to 3, further comprising the steps of:
 - (4) when an undesirable Executable Object is detected at a gateway, providing an identifying value therefrom, and notifying all gateways thereof; and
 - (5) providing memory means and suitable identity verification means, coupled to each gateway, to identify undesirable Executable Objects already analyzed by another gateway, and from preventing it from passing the gateway.
5. A method according to claim 4, wherein a checksum or the like procedure is carried out on the Executable Object, to generate a substantially unique identification thereof.
6. A method according to claim 1, wherein when more than one Control Centers are provided, Security Policies are disseminated from one main Control Center to the remaining Control Centers, and wherein each Control Center, in turn, controls the operation of the gateways connected to it.

7. A method according to claim 1 or 6, wherein each Control Center and each group of gateways is provided with its own individual Security Policy, which may be the same or different from the Security Policy received from the main Control Center.
8. A method according to claim 7, wherein each subordinate Control Center may add additional limitations to the Security Policy received from the main Control Center, by the addition of resources to the list of those the use of which is not allowed, but it may not remove limitations from the list of limited resources contained in the Security Policy distributed by the main Control Center.
9. A computer system comprising one or more LANs, each LAN being connected to an outside computer or computer network through one or more gateways, comprising:
 - (a) one or more Control Centers, each Control Centers being connected to one or more gateways located between a LAN and an external computer communication network;
 - (b) means coupled to each of said gateways, to detect Executable Objects reaching said gateway, to analyze the header of each of said Executable Objects, and to determine the resources of the computer that the Executable Object needs to utilize;

-20-

(c) means coupled to each of said gateways, to store a user's Security Policy representing the resources, or combination of resources, that the user allows or does not allow an Executable Object to utilize within its LAN, wherein the Security Policy is received from and/or stored in each of said one or more Control Centers;

(d) means, provided at, or coupled to, each gateway:

1. to analyze the header of when an Executable Object which is detected at the gateway;
2. to determine the resources of the computer that the Executable Object needs to utilize;
3. to compare the resources of the computer that the Executable Object needs to utilize with the Security Policy and;

(i) means to allow the Executable Object to pass through the gateway and to reach the computer which has initiated its downloading, if the resources of the computer that the Executable Object needs to utilize are included in the list of the resources allowed for use by the Security Policy; and

(ii) means for preventing the Executable Object from passing through the gateway, thereby preventing it from reaching the computer which has initiated its downloading, if the resources of the computer that the Executable Object needs to utilize are included

-21-

in the list of the resources prohibited for use by the Security Policy.

10. A computer system according to claim 9, further comprising, in addition to the means of claim 9(d)3:

(iii) means for alerting the Security Officers that a given type of Executable Object has entered the gateway; and

(iv) means for storing information pertaining to a given Executable Object according to the Security Policy.

11. A method for selectively preventing the downloading and execution of undesired Executable Objects in a computer, essentially as described and illustrated.

12. A computer system comprising one or more LANs, each LAN being connected to an outside computer or computer network through one or more gateways, essentially as described and illustrated.

1/2

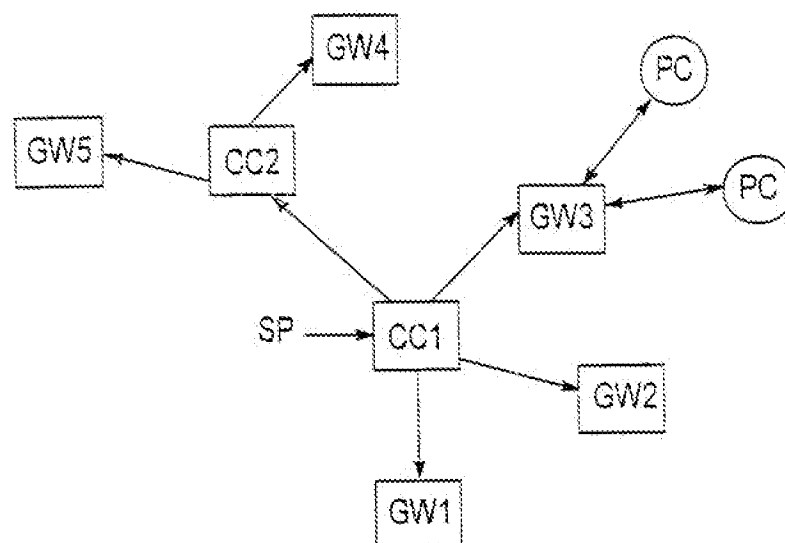


Fig. 1

2/2

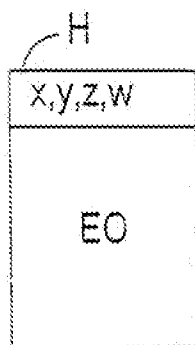


Fig. 2

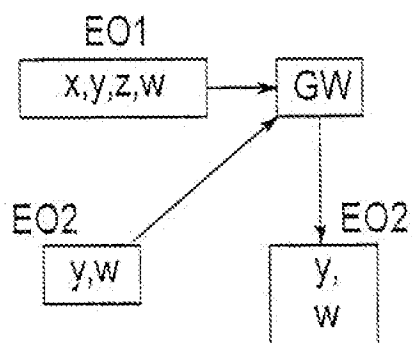


Fig. 3

INTERNATIONAL SEARCH REPORT

International Application No.
PCT/IL 98/00083

A. CLASSIFICATION OF SUBJECT MATTER

IPC 6 H04L29/06

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 6 H04L G06F

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

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	SHARON MACHLIS: "Screening for applets" COMPUTERWORLD, vol. 31, no. 6, 10 February 1997, USA, pages 51-52, XP002069848 see page 51, column 1, line 1 - line 29 see page 52, column 1, line 1 - column 2, line 1	1,9
A	DEAN D ET AL: "JAVA SECURITY: FROM HOTJAVA TO NETSCAPE AND BEYOND" PROCEEDINGS OF THE 1996 IEEE SYMPOSIUM ON SECURITY AND PRIVACY, OAKLAND, CA., MAY 6 - 8, 1996, no. SYMP. 17, 6 May 1996, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, pages 190-200, XP000634844 --- -/-	1,9,11

☒ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents:

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

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

"U" 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

"S" document member of the same patent family

Date of the actual completion of the international search

14 July 1998

Date of mailing of the international search report

29/07/1998

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentaan 2
NL - 6250 HV Rijswijk
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,
Fax: (+31-70) 340-3016

Authorized officer

Adkhis, F

INTERNATIONAL SEARCH REPORT

International Application No.

PCT/IL 98/00083

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT		
Category	Citation of document, with indication where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 414 833 A (HERSHEY PAUL C ET AL) 9 May 1995 see column 6, line 47 - column 7, line 51 see column 19, line 33 - line 34 see column 27, line 39 - line 69; figure 13 see column 30, line 29 - column 32, line 38	1, 9, 11, 12

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

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No.

PCT/IL 98/00083

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5414833 A	09-05-1995	NONE	

Form PCT/ISA/210 (patent family annex) (July 1992)



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : H04L 9/00, H04K 1/00	A1	(11) International Publication Number: WO 98/31124 (43) International Publication Date: 16 July 1998 (16.07.98)

(21) International Application Number: PCT/US98/01117
(22) International Filing Date: 6 January 1998 (06.01.98)
(30) Priority Data: 08/782,479 10 January 1997 (10.01.97) US
(71)(72) Applicants and Inventors: HANSON, Gordon, L. [US/US]; 16613 S.E. 254th Place, Kent, WA 98042 (US). HANSON, Kevin, L. [US/US]; 16613 S.E. 254th Place, Kent, WA 98042 (US).
(74) Agent: SMITH, Michael, S.; Christensen O'Connor Johnson & Kindness, Suite 2800, 1420 Fifth Avenue, Seattle, WA 98101 (US).

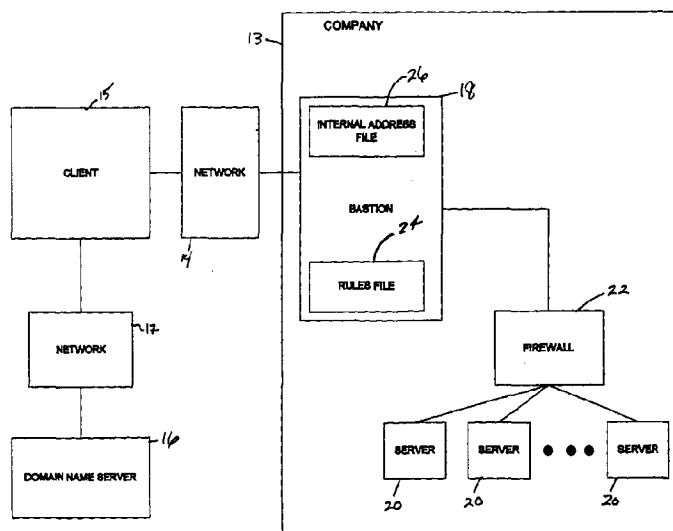
(81) Designated States: CA, JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published

With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: REVERSE PROXY SERVER



(57) Abstract

A method and system for securely accessing servers over an internetwork (14). Each server includes a processor and a memory. A first server (18) outside a company's firewall (22) connects the company to the internetwork. A user or client (15) sends a data packet with a server name to a second server (20) identified by the server name and located within the company's firewall. The location address of the first server is retrieved according to the domain/server name, a connection is made with the first server according to the retrieved location address of the first server and the data packet with domain/server name in the sent data packet to a list of at least one internal address, wherein the at least one internal address of the list identifies the location of the second server. If an internal address is found to match the server name, the first server sends the packet to the internal address.

FOR THE PURPOSES OF INFORMATION ONLY

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

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

REVERSE PROXY SERVER

Field of the Invention

This invention relates to client/server computer communication over an internetwork system and, more particularly, to improved access of firewall protected servers.

Background of the Invention

Communication networks are well-known in the computer communications field. By definition, a network is a group of computers and associated devices that are connected by communications facilities or links. Network connections can be of a permanent nature, such as via cables, or can be of a temporary nature, such as connections made through telephone or other communication links. Networks vary in size, from local area network (LAN) consisting of a few computers and related devices, to a wide area network (WAN) which interconnects computers and LANs that are geographically dispersed. An internetwork, in turn, is the joining of multiple computer networks, both similar and dissimilar, by means of gateways or routers that facilitate data transfer and conversion from various networks. A well-known abbreviation for internetwork is "Internet." As currently understood, the capitalized term "Internet" refers to the collection of networks and routers that use a Transmission Control Protocol/Internet Protocol (TCP/IP) to communicate with one another.

A representative section 10 of the Internet is shown in FIGURE 1 (Prior Art) in which a plurality of LANs 12 are interconnected by routers 11. The routers 11 are generally special purpose computers used to interface one LAN to another. Communication links within the LANs may be twisted wire pair, or coaxial cable,

-2-

while communication links between networks may utilize 56Kbps analog telephone lines, 1Mbps digital T-1 lines and/or 45Mbps T-3 lines. It will be appreciated that the Internet comprises a vast number of such interconnected networks and routers and that only a small representative section of the Internet is shown in FIGURE 1.

5 The rapid growth and development of the Internet has made the Internet an important business tool. A company's primary concern when connected to the Internet is security. The advantages of using the Internet can immediately be nullified by the possibility of internal company computers being compromised by an external entity. Ruinous results occur if data is stolen or computers are infected by viruses.

10 The immediate solution created is the firewall, a filter which can work either at the circuit or software level. This solution has been widely accepted by corporations and, with proper care and administration, effectively allows a company to securely utilize the Internet. The firewall's primary fault is that it blocks communication in both directions, input to the internal network of the company and output to the Internet.

15 Proxy servers act as relay stations between an internal network and the Internet for communication requests initiated inside the company's network, thus helping solve the known problem. The fundamental security basis to a proxy server is the trust of all internal computers. It is assumed that any communication requested by internal computers is not going to compromise the security of internal computers.

20 This is a valid assumption in most cases. Unfortunately, the opposite of this assumption is not valid. Computers on the Internet at large cannot be trusted without elaborate authentication and encryption, see U.S. Patent No. 5,550,984. This poses a major problem to companies on the Internet which, selectively, want to share internal company resources on the Internet.

25 For example, company A has a printer attached to their own network. Whenever someone wants to print a job to this printer, they just print over the network to the printer's server. Company A is also attached to the Internet with a firewall protecting their valuable computer secrets. Now, company A decides that it would like to allow a few allies, company B, C, and D, to be able to print confidential

30 documents directly to company A's printer. Instead of having to ship printed contracts via courier or U.S. Mail, the allies would securely send the document directly to the printer inside company A's network. There are a few immediate solutions to accomplish what company A wants. A first solution would require the physical movement of the printer to a new network which is open to the Internet.

35 Another solution would require that company A's firewall be opened to allow

connection from the Internet to the printer. Both solutions would require a new InterNIC valid IP address, something which recently is increasingly in short supply, and each computer inside company A would have to be reconfigured to use the new IP address for the printer.

- 5 Accordingly, there is a need for secure two-way data communication over the Internet. The present invention is directed to providing such communication security.

Summary of the Invention

- 10 In accordance with this invention, a method and system for securely accessing servers over an internetwork are provided. Each server includes a processor and a memory. The location address of a first server is determined. Then, the user sends a data packet with a server name to the first server according to the determined location address, wherein the server name is associated with a second server connected to the internetwork through the first server. The first server compares the server name in the sent data packet to at least one internal address, wherein the at least one internal address identifies the location of said second server. If an internal address is found to match the server name, the packet is sent to the found internal address.

- 15 In accordance with other aspects of the present invention, a client or user transmits a server name to a third server prior to sending said data packet. The user receives a location address from the third server in response to the transmitted server name, wherein the location address is the address identifying a first server associated with the server name and connected to the internetwork.

20 In accordance with still other aspects of the present invention, the third server is a server name server and the first server is a web server.

- 25 In accordance with further aspects of the present invention, communication between the first server and the second server is performed through a firewall.

 In accordance with yet other aspects of the present invention, the list of at least one internal address is encrypted. The encrypted list is decrypted then compared to the server name of the received data packet. The decrypted list is deleted at the completion of the comparison.

- 30 In accordance with still further aspects of the present invention, the user receives an error reply if no match was found in the compare of the server name with the list of internal addresses or if a reply data packet from the second server fails to be sent from the first server.

Brief Description of the Drawings

The foregoing aspects and many of the attendant advantages of this invention will become more readily appreciated as the same becomes better understood by reference to the following detailed description, when taken in conjunction with the accompanying drawings, wherein:

FIGURE 1 is a schematic diagram illustrating a network, such as the Internet;

FIGURE 2 is a block diagram of a preferred embodiment of a system configuration of the present invention;

FIGURE 3 is an example system functioning according to the invention of FIGURE 2;

FIGURES 4 and 5 are flow diagrams illustrating the steps for executing the invention shown in FIGURE 2; and

FIGURES 6 and 7 are flow diagrams illustrating an example packet request and reply according to steps shown in FIGURES 4 and 5.

Detailed Description of the Preferred Embodiment

As shown in FIGURE 2, a network system provides secure bi-directional data packet communication between a requester or client 15 and servers 20 protected by a firewall according to the present invention. A company 13 includes a number of internal servers 20 protected by a firewall 22 and a bastion server 18 connected to a network 14. Client 15 also connects to the network 14, and includes a domain name server 16 or connects to a domain name server 16 over a network 17. Networks 14 and 17 may be the same network. Client 15 can be a separate entity from company 13 or a separate entity in a different department of company 13. Examples of networks 14 or 17 are the Internet and an intranet.

FIGURE 3 illustrates an example of computer components and peripheral devices that may be used in place of like numbered block diagram components shown in FIGURE 2. As shown in FIGURE 3, a computer 20a, file server 20b, printer 20c and facsimile device 20d are examples of computers or peripheral devices that can act as or include a server 20 connected within Company A's firewall 22 (FIGURE 2). Company A's system 13a includes a bastion server 18a coupled between the network 14a and the Company A's firewall 22a. The firewall 22a provides discrete access to the servers 20a-d located inside Company A.

As shown in FIGURE 2, the bastion 18 is a server that includes a processor and memory like typical servers, but also includes an internal address file 26 and a rules file 24 stored in memory. The internal address file 26 includes the internal

addresses for each of the servers 20 the company wishes to allow external access to through the firewall 22. Any data packet sent by a client specifically identifying a server 20 located behind another entity's (company 13) firewall 22, first goes to the client's domain name server 16 for retrieval of the IP address assigned to the
5 bastion 18 that protects the destination server 20. The IP address is the bastion 18's location on the network 14. The client may include a domain name server, thereby reducing the step of having to retrieve from an external domain name server. The data packet is then sent to the retrieved destination IP address. The rules file 24 provides a predefined set of rules for maintaining secure communication of data
10 packets passing in both directions through bastion server 18. Bastion 18 limits communication with a single server 20 within firewall 22 without putting the company's system 13 at risk to rogue clients or unauthorized requests. The bastion 18 can perform this service because the internal address file 26 is the only location where internal IP addresses are stored. The limited secure communication performed by the
15 bastion 18 is described in more detail below with reference to the example illustrated in the Tables.

FIGURE 4 is an information flow diagram according to the invention shown in FIGURE 2. At block 40, the client uses a domain (server) name to request the IP address of a bastion coupled to the server identified by the server name through a
20 firewall. This request can be accomplished by various methods known in the art. Supplying the server name directly to a domain name server is one method and performing an automatic request from a domain name server at the time the client sends a data packet is another. In order to maintain a certain level of security, the client receives the server name from the company under confidence prior to execution
25 of any transactions. At block 41, if the client does not know the IP address assigned to the bastion associated with the server name or have the ability to force a packet with the server name to the IP address assigned the bastion, the domain name server (DNS) retrieves the IP address associated with the requested server name, see block 42. Within the DNS, all names of accessible servers within a company's firewall
30 are stored with direct reference to the company's IP address (IP address of the company's bastion). Upon finding the requested server name, the DNS returns the directly referenced IP address, see block 44. Thus, a company only requires only a single IP address for identifying all the servers within its system. This feature of needing a limited number of IP addresses is important because InterNIC, the Internet
35 IP address controlling entity, is limiting the number of IP addresses because available

addresses are running out. If a first DNS fails to include the requested information, other DNSs are accessed until the information is found or it is determined the information does not exist. DNSs are tied together in a distributed database system which provides a system of interconnected DNSs with databases that can
5 automatically update their internal files according to the frequency of particular requests.

Now that the client has the company's IP address, the client establishes a connection with the bastion, see block 46. After a connection is made, the data packet, which includes the server name, is sent to the bastion, see block 48. The
10 bastion then compares the server name to the internal address file, see block 52. At decision block 54, the bastion determines whether a match exists between the server name and an internal address located in the internal address file. If no match is found, the bastion sends a reply to the client indicating that the request packet cannot be delivered because the requested server is not listed or no longer listed in the internal
15 address file within the company or the request is incorrect for some other predefined reason, see block 56. If a match was found, the received packet is checked against rules contained within the rules file, see decision block 58. If the received data packet fails to pass any of the predefined rules in the rules file, a reply is sent to the client indicating so, see block 56. However, if the received data packet passes all the rules
20 contained within the rules file, a connection is made between the client and the server associated with the matched internal IP address and the data packet is delivered to the server. The rule checks include certain security- programs that operate upon received data packets and, particularly, data packets that are or include programs. Some unique rules and rules in the form of programs are described in more detail below with
25 respect to Table 3.

FIGURES 6 and 7 illustrate an example of two-way packet communication between a client and a firewall protected server over the Internet. The client's and bastion's IP addresses are 245.23.12.3 and 124.12.32.1, respectively. The client's goal is to establish two way data packet communication with a server protected by a
30 firewall and connected to the Internet through a bastion. As shown in FIGURE 6, at block 80, the client sends a request packet out on the Internet to the destination server, fujil.hde.com.

-7-

@	IN	SOA	bastion.hde.com. root.bastion.hde.com (
		8	;Serial
		21600	;Refresh 6 hours
		900	Retry 15 minutes
		172800	;Expire 48 hours
		3600)	;Minimum 1 hour
	IN	NS	bastion.hde.com
	IN	MX	1 bastion.hde.com
ns	IN	CNAME	bastion.hde.com
localhost	IN	A	127.0.0.1
bastion	IN	A	124.12.32.1
laser_printerA	IN	A	124.12.32.1
file_serverA	IN	A	124.12.32.1
jp_www	IN	A	124.12.32.1
fujil	IN	A	124.12.32.1

Table 1 - DNS zone file.

At block 82, a domain name server (DNS) searches for the domain name zone file associated with the server name that ends with hde.com. The DNS runs the DNS server program called "named" that reads the zone file, shown in Table 1. The zone file provides the translation of the requested server name, fujil, into the IP address, 124.12.32.1, for the bastion connecting the server identified by the server name to the Internet, see block 84. If no IP address matches the server name, fujil, an alternate DNS is accessed for locating an IP address associated with the server name. The alternate DNS is identified in the DNS zone file by the code line: "IN NS bastion.hde.com." NS is name server. At block 86, the client establishes a connection to the bastion with IP address 124.12.32.1. Port 80 of the bastion is the receiving port and the client's connecting port depends upon other client port connections. In this example the client connecting port is port 80.

After communication is established, the bastion receives the data packet(s) from the client at block 88. At block 90, the server name, fujil.hde.com, and the data packet are transmitted to the bastion. A program operating within the bastion

provides the necessary coordination for the received data packet. The server name, fuji1.hde.com, is first compared to a decrypted internal address file, internal.conf, see Table 2. The internal IP address file is decrypted into active memory upon data packet reception and the decrypted internal IP address file is deleted from memory upon completion of the comparison. Since the decrypted internal IP address file appears in active memory, i.e., RAM, for a very short period of time, it is virtually impossible for one to access this information in an attempt to discover the internal addresses. The internal address file presents a list of domain names and corresponding internal IP addresses. These internal addresses are known only to the bastion.

jp_www.hde.com = 204.96.95.251
fuji1.hde.com = 204.96.95.95
ftp.hde.com = 204.96.95.251

Table 2 - internal IP address file (internal.conf) after decryption.

At block 92, the server name, fuji1.hde.com, is found to match an internal server with address 204.96.95.95. The data packet then passes the check against the set of rules within the rules file at block 94, see Table 3. Then, a connection from the client through the bastion to the internal server is established through port 80 of the internal server and a predefined output port on the bastion, at block 96. After connection is established, the internal server receives and processes the client's request packet at block 98.

FIGURE 7 illustrates an example of a reply packet sent from the internal server through the bastion back to the client. At block 100, the internal server first sends the reply packet to the bastion. In this example, the reply data packet is a reply to the data packet received from the client. The reply packet may also be an unsolicited data packet. At block 102, the reply data packet passes a check against a set of rules for outgoing data packets. Similar to checks of received data packets, the outgoing data packet check is performed within the bastion. And finally at block 104, the outgoing data packet is sent to the client with IP address 245.23.12.3.

Destination IP: 204.95.95.0-94

FILE: viruses.dat

FILE: ps_error.dat

Flags: 0110001101

URL: fuji1.hde.com/target.html

JAVA Checks:

signature.class

security.class

test.class

ActiveX Checks:

security.ocx

Table 3 - rules file.

- Table 3 illustrates an example set of rules for checking data packets passing through the bastion. As shown in Table 3, the code line "Destination IP: 204.95.95.0-94" is an IP address limiter rule. If an internal address discovered in the
- 5 internal address file check of a received data packet is within the range of 204.95.95.0 through 204.95.95.94, the received data packet is denied access. The limiter rule may also be used to present a range of allowable IP addresses. By using a IP address limiter, a system operator can limit access to specific internal servers. Another check performed is a file compare of the received data packet against prespecified files, as
- 10 indicated by the code lines beginning with "FILE:". As shown in Table 3, the received data packet is checked against the indicated files "viruses.dat" and "ps_error.dat." These files compare the data packet to known viruses and data errors. If the data packet fails to pass the check with either one of these files, the data packet is refused and destroyed, because it most likely contains a virus. Similar file compare programs
- 15 may also be used in the line "FILE:". The code line "Flags:" prevents so called "Christmas tree" data packets from being relayed to the internal server by the bastion and causing problems with the internal server. As shown in Table 3, a received packet that exhibits the flags 0110001101 in its header will not be processed by the bastion.
- 20 The "URL:" code line identifies all URL addresses the internal server is denied access to. If an internal server attempts to address an address contained within the

"URL:" code line, the access is denied and an error message is returned to the client explaining the failed transaction. In the case that any one of the rules are broken and a specific service is denied, a log file is maintained within the bastion to record information regarding the denied transaction. The log file may include such things as why the denial was made, who it was from, and what was the destination of the packet. It can be appreciated to those of ordinary skill in the art that the rules file can include very specific rules relating to the type of system in which it is being used.

The "JAVA Checks:" code lines indicate JAVA class files that execute if the bastion receives a JAVA applet as or in a data packet. The JAVA class files listed are "signature.class", "security.class" and "test.class". Each perform specific checks of received JAVA applets. The "security.class" program ensures that the data within the data packet going to the client or internal server is not destructive for the intended recipient. The "security.class" program performs security operations similar to that performed by a complete, secure JAVA virtual machine. "Security.class" performs these protective illegal operation overrides by attaching itself to the applet being sent in the data packet. When the applet intended for the recipient is run at the destination client or server, "security.class" is run simultaneously. Since every system operable with JAVA applets includes a JAVA virtual machine, "security.class" sometimes performs redundant security checks to those performed by a complete JAVA virtual machine, thus protecting against a bad or incomplete JAVA virtual machine. The "signature.class" program performs a certification operation similar to those provided by VeriSign Corporation's programs. The "signature.class" program authenticates received JAVA applets by adding a signature program and/or time stamp to the received applet. When the applet intended for the destination client or server is run, the signature and/or time stamp program is also run to insure that in between transmission from the client server and the destination, the applet was not tampered with or altered in any manner. If the applet will not run and the destination client or server will be informed that the applet received was invalid.

The "test.class" program performs two checks of the applet being transmitted through the bastion server. First, it compares the applet against known virus code, similar to a virus checking program. Next, it communicates with a secure Java virtual machine located on the bastion. The Java virtual machine on the bastion executes the applet intended for the destination client or server. If any illegal operations are attempted by the applet, the Java virtual machine informs the "test.class" program and

the "test.class" program relays to the a program on the bastion that the Java applet is not secure and should not be transmitted to the destination client or server.

The "ActiveX Checks:" section includes the program "security.ocx". The "ActiveX Checks:" section is similar to the "Java Checks:" section except it applies to data packets which are ActiveX programs which are intended to be run on the destination client or server. The "security.ocx" program attaches to an ActiveX program destined for a client or server. This program is run at the destination client or server and behaves similar to a common virus checking program. It monitors the execution of the ActiveX program that is running on the destination client or server.

5

10 If the ActiveX program attempts an operation which is destructive to the host client or server, the "security.ocx" stops execution and warns the client or server user that the ActiveX program attempted an illegal operation.

It can be appreciated to those of ordinary skill in the art that the attachment of security programs and the examination of the data packets at the bastion or the destination client or server can be extended to provide various levels of security and protection against various types of data being transmitted. It also can be appreciated that resources are reserved at the bastion, if the execution of a security program is performed at the destination client or server.

15

While the preferred embodiment of the invention has been illustrated and described, it will be appreciated that various changes can be made therein without departing from the spirit and scope of the invention.

20

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. A method for securely sending data to servers over an internetwork, wherein each server comprises a processor and a memory, said method comprising:
determining the location address of a first server;
sending a data packet with a server name to said first server according to said determined location address, wherein said server name being associated with a second server connected to the internetwork through said first server;
comparing within said first server said server name in said sent data packet to a list of at least one internal address, wherein said at least one internal address of said list identifies the location of said second server; and
if an internal address is found to match the server name, sending the packet to said found internal address.
2. The method of Claim 1, further comprising:
transmitting a domain name to a third server prior to sending said data packet;
and
receiving a location address from said third server in response to said transmitted domain name, wherein said location address being the address identifying said first server associated with said domain name and connected to the internetwork.
3. The method of Claim 2, wherein said third server being a domain name server.
4. The method of Claim 1, wherein said first server being a web server.
5. The method of Claim 1, wherein communication between said first server and said second server connected to the internetwork through said first server is performed through a firewall.
6. The method of Claim 1, wherein said list of at least one internal address is encrypted.
7. The method of Claim 1, wherein comparing further comprises:
decrypting said encrypted list;
comparing said decrypted list to said server name; and

deleting said decrypted list at completion of said comparison.

8. The method of Claim 1, further comprising generating an error message, if no match was found in said compare of said server name with the list of at least one internal address.

9. The method of Claim 1, further comprising:
performing at least one of a signature, a security and a test check of the received data packet within said first server; and
sending said data packet to said found internal address, if all performed checks are successful.

10. The method of Claim 1, further comprising:
tagging a security check program onto the received data packet, if a specific program exists within the received data packet; and
performing the security check program simultaneously with the specific program of the received data packet at said second server.

11. A system for securely accessing servers over an internetwork, wherein each server comprises a processor and a memory, said system comprising:
a means for determining the location address of a first server;
a means for sending a data packet with a server name to said first server according to said determined location address, wherein said server name being associated with a second server connected to the internetwork through said first server; and
a means for comparing within said first server said server name in said sent data packet to a list of at least one internal address, wherein said at least one internal address of said list identifies the location of said second server, and if an internal address is found to match the server name, sending the packet to the internal address.

12. The system of Claim 11, further comprising:
a means for transmitting a domain name to a third server prior to sending said data packet; and
a means for receiving a location address from said third server in response to said transmitted domain name, wherein said location address being the address identifying said first server associated with said domain name and connected to the internetwork.

13. The system of Claim 12, wherein said third server is a domain name server.

14. The system of Claim 11, wherein said first server is a web server.

15. The system of Claim 11, wherein communication between said first and second server is performed through a firewall.

16. The system of Claim 11, wherein said list of at least one internal address is encrypted.

17. The system of Claim 11, wherein said first server further comprises:
a means for decrypting said encrypted list;
a means for determining from said decrypted list the location of the server not directly connected to the internetwork with said received server name; and
a means for deleting said decrypted list at completion of said determination.

18. The system of Claim 11, wherein said first server further comprises transmitting an error message to said third server if no internal address was determined to match said server name of the received data packet.

19. The system of Claim 11, further comprising:
a means for performing at least one of a signature, a security and a test check of the received data packet within said first server; and
a means for sending said data packet to said found internal address, if said all performed checks are successful.

20. The system of Claim 11, further comprising:
a means for tagging a security check program onto the received data packet, if a specific program exists within the received data packet; and
a means for performing the security check program simultaneously with the specific program of the received data packet at said second server.

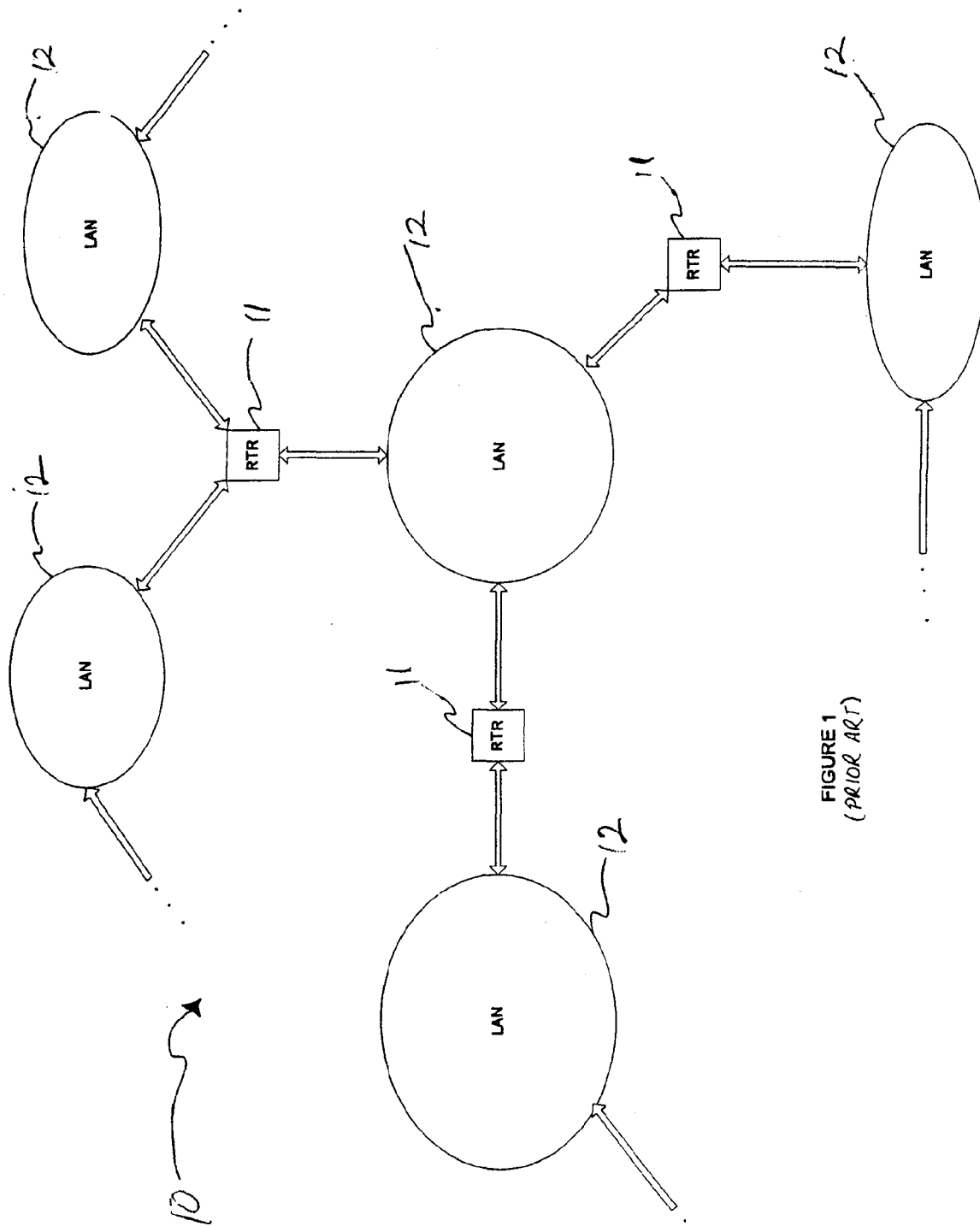


FIGURE 1
(PRIOR ART)

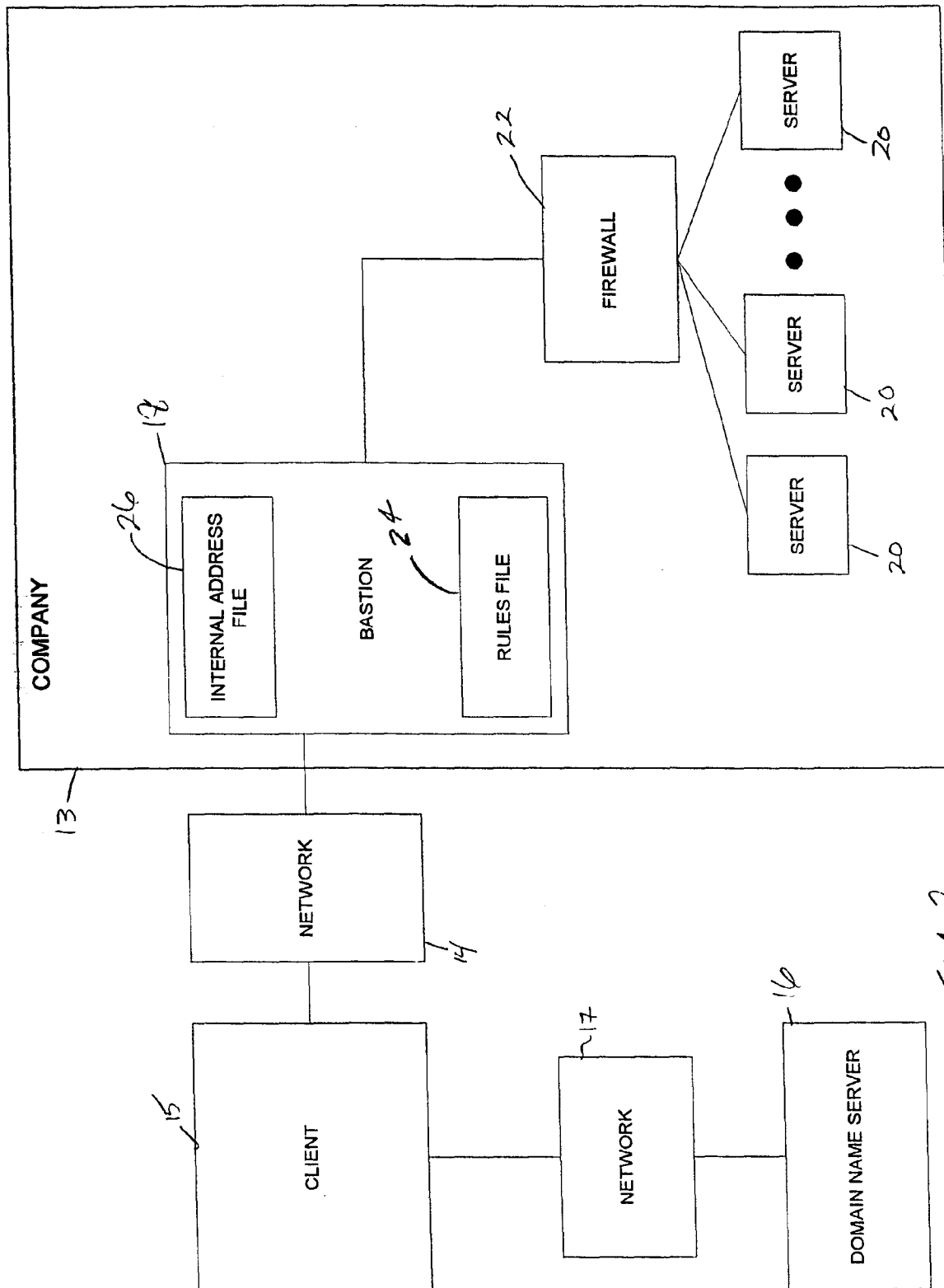
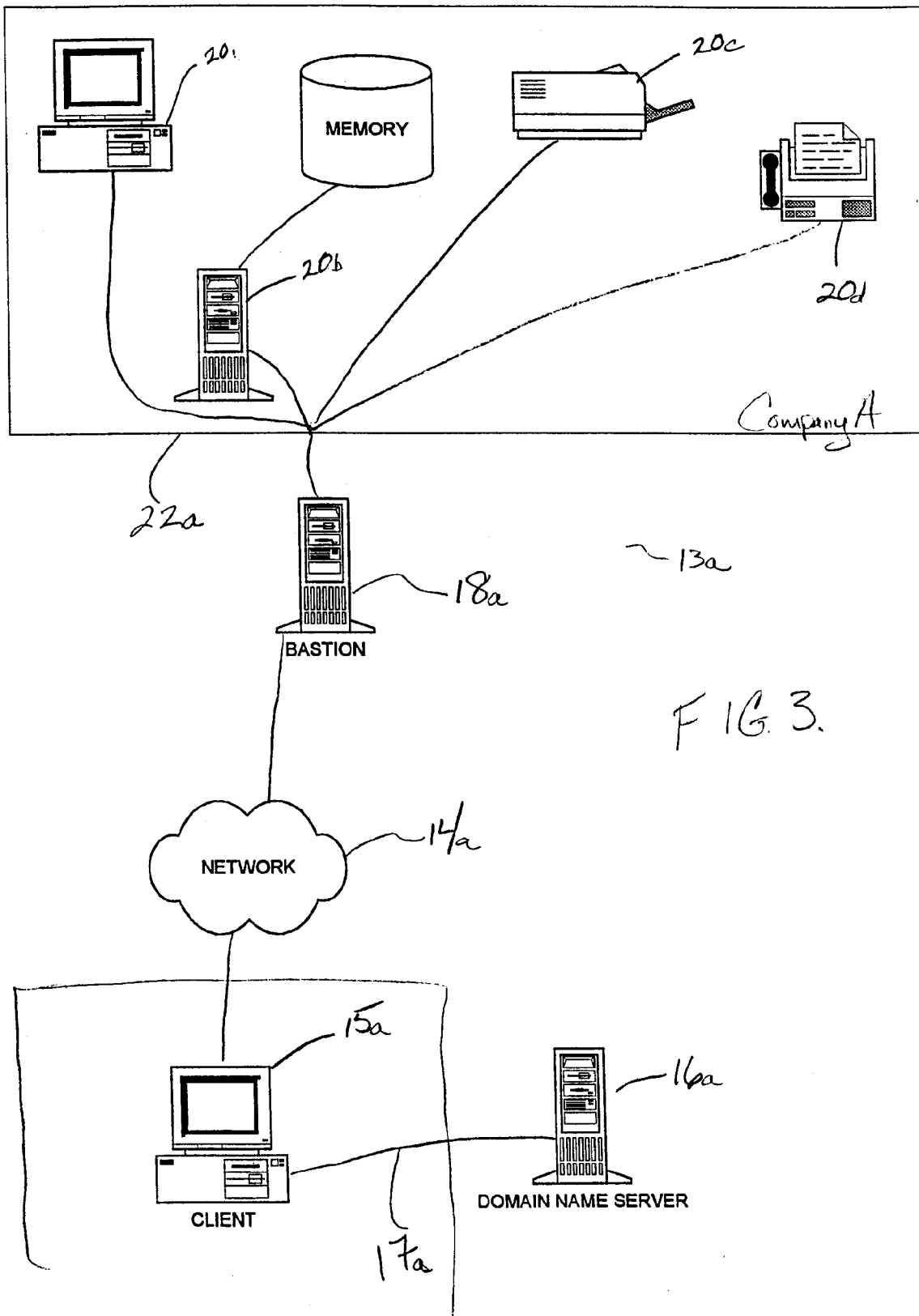
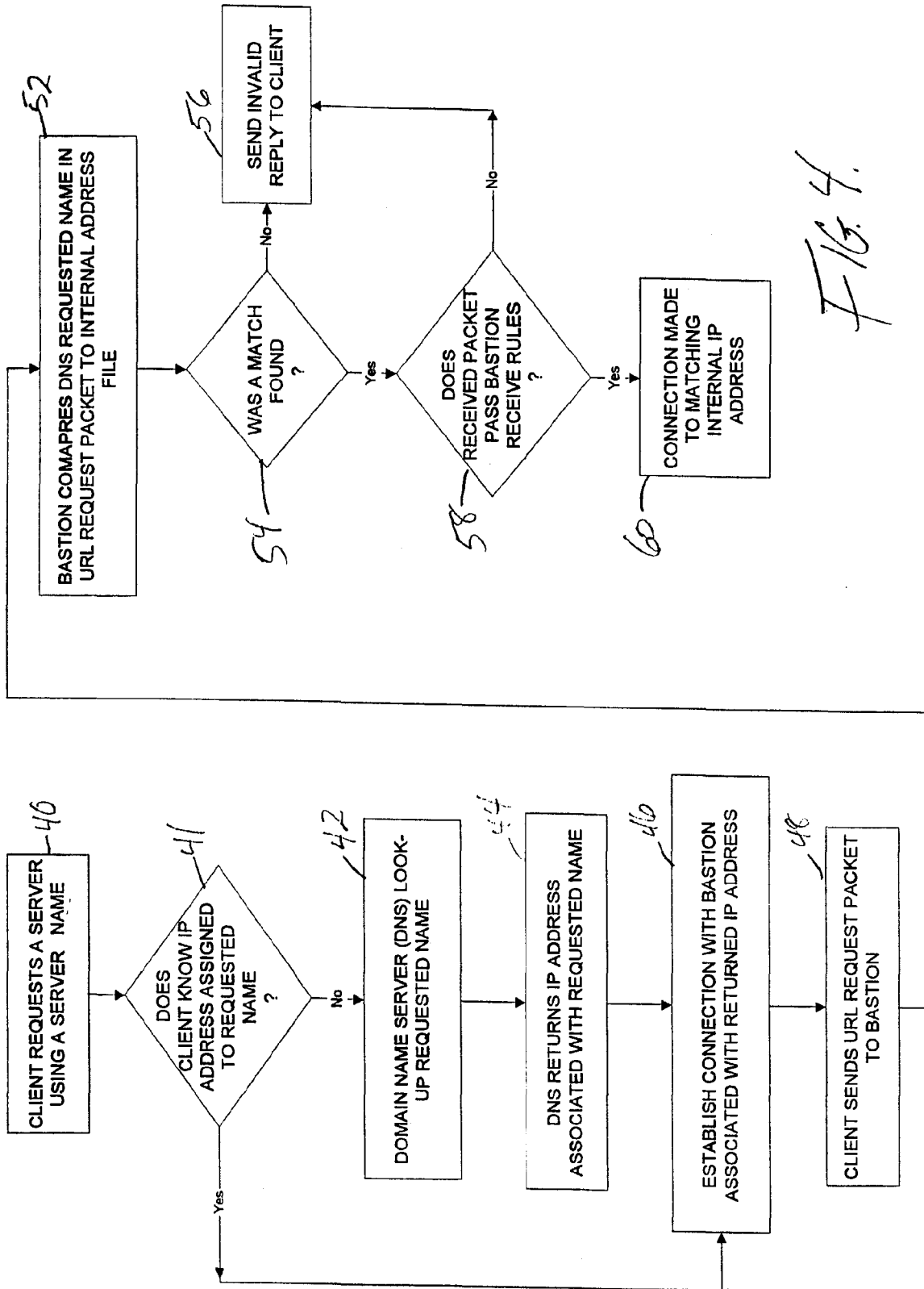


FIG 2





5/6

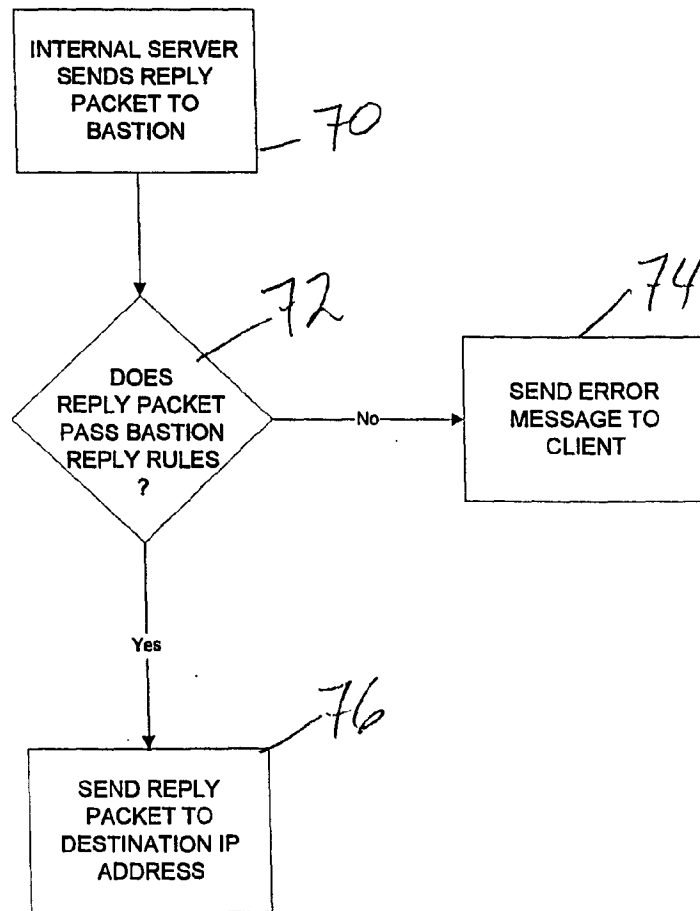
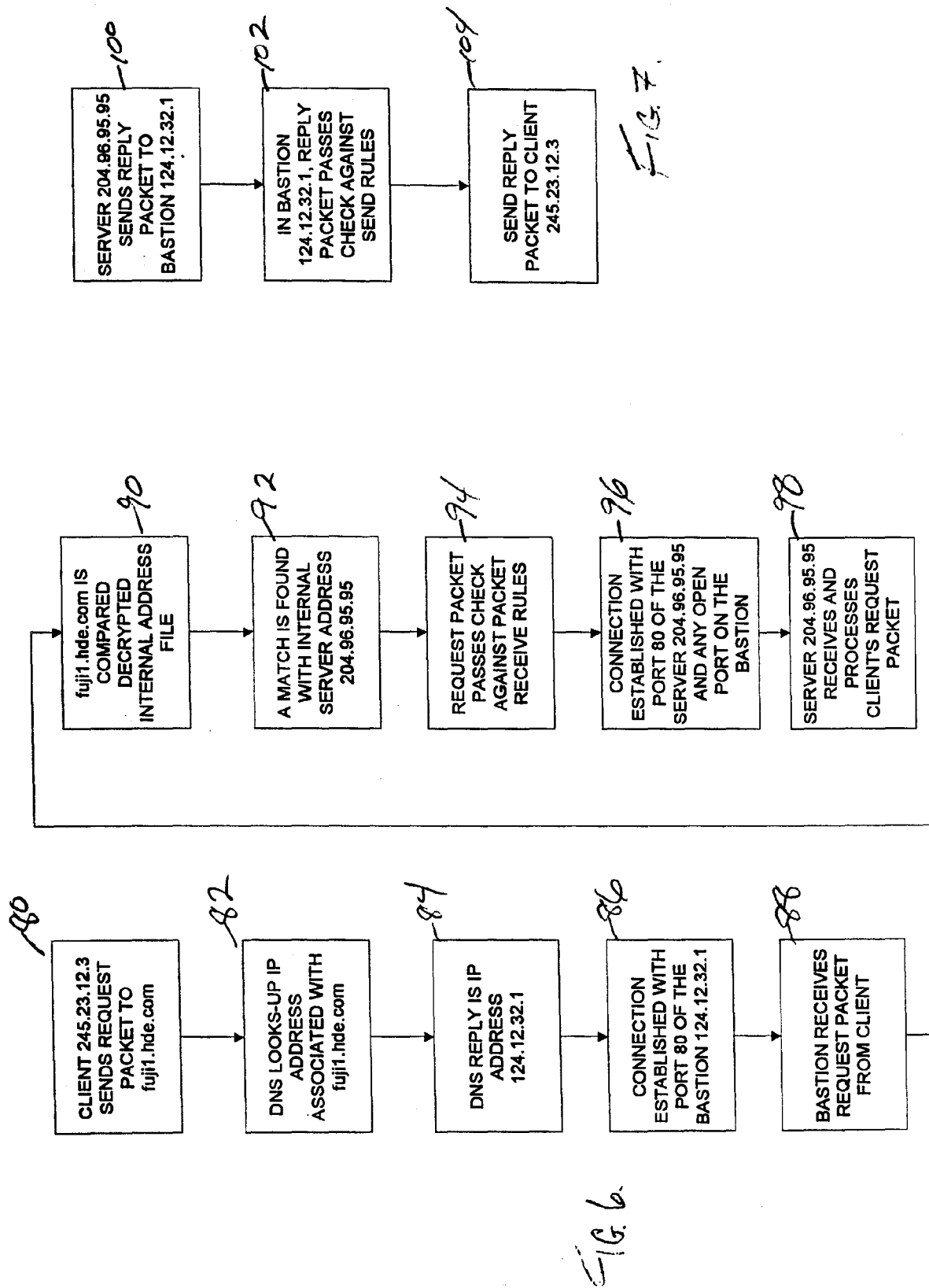


FIG. 5

6/6



INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/01117

A. CLASSIFICATION OF SUBJECT MATTER IPC(6) H 04 L 9/00; H 04 K 1/00 US CL 380/49, 23, 25 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. : 380/49, 23, 25 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) APS		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,583,940 A (VIDRASCU et al.) 10 December 1996, the whole document.	1-20
X	US 5,550,984 A (GELB) 27 August 1996, the whole document.	1-20
☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.		
* Special categories of cited documents		*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
A document defining the general state of the art which is not considered to be of particular relevance		*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
E earlier document published on or after the international filing date		*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
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)		*G* document member of the same patent family
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		
Date of the actual completion of the international search	Date of mailing of the international search report	
26 MAY 1998	25 JUN 1998	
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer <i>Liane Gorkungh</i> HRAYR A. SAYADIAN Telephone No. (703) 306-4169	



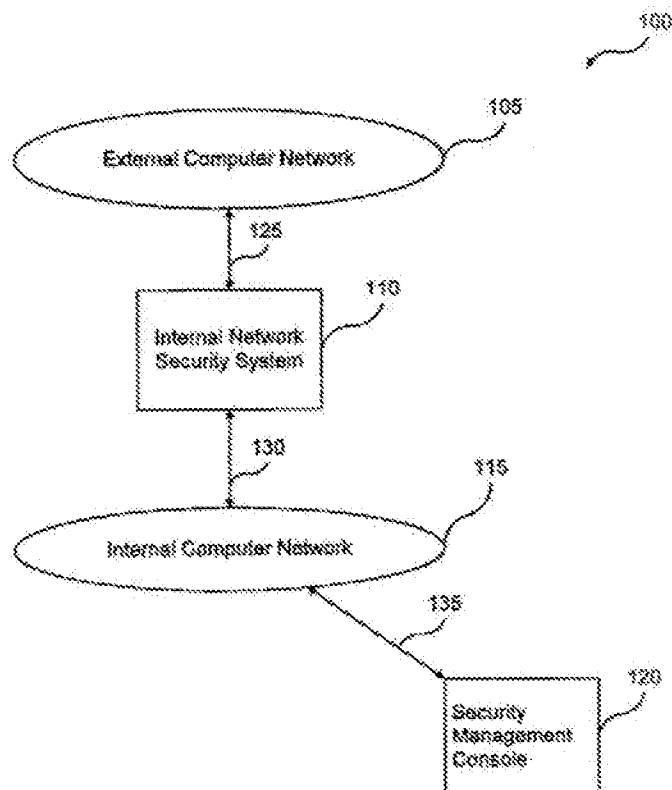
INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : G06K		(11) International Publication Number: WO 98/21683
A2		(43) International Publication Date: 22 May 1998 (22.05.98)
(21) International Application Number: PCT/IL97/01626		(81) Designated States: CA, IL, JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).
(22) International Filing Date: 6 November 1997 (06.11.97)		
(30) Priority Data:		
600230,639	8 November 1996 (08.11.96) US	
Not furnished	6 November 1997 (06.11.97) US	
(71) Applicant: FINJAN SOFTWARE, LTD. (IL/IL); 42945 Kefar-Haim (IL).		Published Without international search report and to be republished upon receipt of that report.
(72) Inventor: TOUBOUL, Shimon; 42945 Kefar-Haim (IL).		

(54) Title: SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES

(57) Abstract

A system protects a computer from suspicious Downloadables. The system comprises a security policy, an interface for receiving a Downloadable, and a comparator, coupled to the interface, for applying the security policy to the Downloadable to determine if the security policy has been violated. The Downloadable may include a Java™ applet, an ActiveX™ control, a JavaScript™ script, or a Visual Basic script. The security policy may include a default security policy to be applied regardless of the client to whom the Downloadable is addressed, or a specific security policy to be applied based on the client or the group to which the client belongs. The system uses an ID generator to compute a Downloadables ID identifying the Downloadable, preferably, by fetching all components of the Downloadable and performing a hashing function on the Downloadable including the fetched components. Further, the security policy may indicate several tests to perform, including (1) a comparison with known hostile and non-hostile Downloadables; (2) a comparison with Downloadables to be blocked or allowed per administrative override; (3) a comparison of the Downloadable security profile data against access control lists; (4) a comparison of a certificate embodied in the Downloadable against trusted certificates; and (5) a comparison of the URL from which the Downloadable originated against trusted and untrusted URLs. Based on these tests, a logical engine can determine whether to allow or block the Downloadable.



FOR THE PURPOSES OF INFORMATION ONLY

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

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

SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK
FROM HOSTILE DOWNLOADABLES

BACKGROUND OF THE INVENTION

5 1. Field of the Invention

This invention relates generally to computer networks, and more particularly provides a system and method for protecting a computer and a network from hostile Downloadables.

2. Description of the Background Art

10 The Internet is currently a collection of over 100,000 individual computer networks owned by governments, universities, nonprofit groups and companies, and is expanding at an accelerating rate. Because the Internet is public, the Internet has become a major source of many system damaging and system fatal application programs, commonly referred to as "viruses."

15 Accordingly, programmers continue to design computer and computer network security systems for blocking these viruses from attacking both individual and network computers. On the most part, these security systems have been relatively successful. However, these security systems are not configured to recognize computer viruses which have been attached to or configured as Downloadable application programs, commonly
20 referred to as "Downloadables." A Downloadable is an executable application program, which is downloaded from a source computer and run on the destination computer. Downloadable is typically requested by an ongoing process such as by an Internet browser or web engine. Examples of Downloadables include Java[™] applets designed for use in the Java[™] distributing environment developed by Sun Microsystems, Inc., JavaScript scripts also

developed by Sun Microsystems, Inc., ActiveX™ controls designed for use in the ActiveX™ distributing environment developed by the Microsoft Corporation, and Visual Basic also developed by the Microsoft Corporation. Therefore, a system and method are needed to protect a network from hostile Downloadables.

5

SUMMARY OF THE INVENTION

The present invention provides a system for protecting a network from suspicious Downloadables. The system comprises a security policy, an interface for receiving a Downloadable, and a comparator, coupled to the interface, for applying the security policy to
10 the Downloadable to determine if the security policy has been violated. The Downloadable may include a Java™ applet, an ActiveX™ control, a JavaScript™ script, or a Visual Basic script. The security policy may include a default security policy to be applied regardless of the client to whom the Downloadable is addressed, a specific security policy to be applied based on the client or the group to which the client belongs, or a specific policy to be applied
15 based on the client/group and on the particular Downloadable received. The system uses an ID generator to compute a Downloadable ID identifying the Downloadable, preferably, by fetching all components of the Downloadable and performing a hashing function on the Downloadable including the fetched components.

Further, the security policy may indicate several tests to perform, including (1) a
20 comparison with known hostile and non-hostile Downloadables; (2) a comparison with Downloadables to be blocked or allowed per administrative override; (3) a comparison of the Downloadable security profile data against access control lists; (4) a comparison of a certificate embodied in the Downloadable against trusted certificates; and (5) a comparison of the URL from which the Downloadable originated against trusted and untrusted URLs.

Based on these tests, a logical engine can determine whether to allow or block the Downloadable.

The present invention further provides a method for protecting a computer from suspicious Downloadables. The method comprises the steps of receiving a Downloadable, comparing the Downloadable against a security policy to determine if the security policy has been violated, and discarding the Downloadable if the security policy has been violated.

It will be appreciated that the system and method of the present invention may provide computer protection from known hostile Downloadables. The system and method of the present invention may identify Downloadables that perform operations deemed suspicious. The system and method of the present invention may examine the Downloadable code to determine whether the code contains any suspicious operations, and thus may allow or block the Downloadable accordingly.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating a network system, in accordance with the present invention:

FIG. 2 is a block diagram illustrating details of the internal network security system of FIG. 1;

FIG. 3 is a block diagram illustrating details of the security program and the security database of FIG. 2;

FIG. 4 is a block diagram illustrating details of the security policies of FIG. 3;

FIG. 5 is a block diagram illustrating details of the security management console of FIG. 1;

FIG. 6A is a flowchart illustrating a method of examining for suspicious Downloadables, in accordance with the present invention;

FIG. 6B is a flowchart illustrating details of the step for finding the appropriate security policy of FIG. 6A;

5 FIG. 6C is a flowchart illustrating a method for determining whether an incoming Downloadable is to be deemed suspicious;

FIG. 7 is a flowchart illustrating details of the FIG. 6 step of decomposing a Downloadable; and

FIG. 8 is a flowchart illustrating a method 800 for generating a Downloadable ID for
10 identifying a Downloadable.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1 is a block diagram illustrating a network system 100, in accordance with the present invention. The network system 100 includes an external computer network 105, such
15 as the Wide Area Network (WAN) commonly referred to as the Internet, coupled via a communications channel 125 to an internal network security system 110. The network system 100 further includes an internal computer network 115, such as a corporate Local Area Network (LAN), coupled via a communications channel 130 to the internal network computer system 110 and coupled via a communications channel 135 to a security
20 management console 120.

The internal network security system 110 examines Downloadables received from external computer network 105, and prevents Downloadables deemed suspicious from reaching the internal computer network 115. It will be further appreciated that a Downloadable is deemed suspicious if it performs or may perform any undesirable operation.

or if it threatens or may threaten the integrity of an internal computer network 115 component. It is to be understood that the term "suspicious" includes hostile, potentially hostile, undesirable, potentially undesirable, etc. Security management console 120 enables viewing, modification and configuration of the internal network security system 110.

5

FIG. 2 is a block diagram illustrating details of the internal network security system 110, which includes a Central Processing Unit (CPU) 205, such as an Intel Pentium® microprocessor or a Motorola Power PC® microprocessor, coupled to a signal bus 220. The internal network security system 110 further includes an external communications interface 210 coupled between the communications channel 125 and the signal bus 220 for receiving Downloadables from external computer network 105, and an internal communications interface 225 coupled between the signal bus 220 and the communications channel 130 for forwarding Downloadables not deemed suspicious to the internal computer network 115. The external communications interface 210 and the internal communications interface 225 may be functional components of an integral communications interface (not shown) for both receiving Downloadables from the external computer network 105 and forwarding Downloadables to the internal computer network 115.

Internal network security system 110 further includes Input/Output (I/O) interfaces 215 (such as a keyboard, mouse and Cathode Ray Tube (CRT) display), a data storage device 230 such as a magnetic disk, and a Random-Access Memory (RAM) 235, each coupled to the signal bus 220. The data storage device 230 stores a security database 240, which includes security information for determining whether a received Downloadable is to be deemed suspicious. The data storage device 230 further stores a users list 260 identifying the users within the internal computer network 115 who may receive Downloadables, and an

event log 245 which includes determination results for each Downloadable examined and runtime indications of the internal network security system 110. An operating system 250 controls processing by CPU 205, and is typically stored in data storage device 230 and loaded into RAM 235 (as illustrated) for execution. A security program 255 controls examination of incoming Downloadables, and also may be stored in data storage device 230 and loaded into RAM 235 (as illustrated) for execution by CPU 205.

FIG. 3 is a block diagram illustrating details of the security program 255 and the security database 240. The security program 255 includes an ID generator 315, a policy finder 317 coupled to the ID generator 315, and a first comparator 320 coupled to the policy finder 317. The first comparator 320 is coupled to a logical engine 333 via four separate paths, namely, via Path 1, via Path 2, via Path 3 and via Path 4. Path 1 includes a direct connection from the first comparator 320 to the logical engine 333. Path 2 includes a code scanner coupled to the first comparator 320, and an Access Control List (ACL) comparator 330 coupling the code scanner 325 to the logical engine 333. Path 3 includes a certificate scanner 340 coupled to the first comparator 320, and a certificate comparator 345 coupling the certificate scanner 340 to the logical engine 333. Path 4 includes a Uniform Resource Locator (URL) comparator 350 coupling the first comparator 320 to the logical engine 333. A record-keeping engine 335 is coupled between the logical engine 333 and the event log 245.

The security program 255 operates in conjunction with the security database 240, which includes security policies 305, known Downloadables 307, known Certificates 309 and Downloadable Security Profile (DSP) data 310 corresponding to the known Downloadables 307. Security policies 305 includes policies specific to particular users 260

and default (or generic) policies for determining whether to allow or block an incoming Downloadable. These security policies 305 may identify specific Downloadables to block, specific Downloadables to allow, or necessary criteria for allowing an unknown Downloadable. Referring to FIG. 4, security policies 305 include policy selectors 405, access control lists 410, trusted certificate lists 415, URL rule bases 420, and lists 425 of Downloadables to allow or to block per administrative override.

Known Downloadables 307 include lists of Downloadables which Original Equipment Manufacturers (OEMs) know to be hostile, of Downloadables which OEMs know to be non-hostile, and of Downloadables previously received by this security program 255. DSP data 310 includes the list of all potentially hostile or suspicious computer operations that may be attempted by each known Downloadable 307, and may also include the respective arguments of these operations. An identified argument of an operation is referred to as "resolved." An unidentified argument is referred to as "unresolved." DSP data 310 is described below with reference to the code scanner 325.

The ID generator 315 receives a Downloadable (including the URL from which it came and the userID of the intended recipient) from the external computer network 105 via the external communications interface 210, and generates a Downloadable ID for identifying each Downloadable. The Downloadable ID preferably includes a digital hash of the complete Downloadable code. The ID generator 315 preferably prefetches all components embodied in or identified by the code for Downloadable ID generation. For example, the ID generator 315 may prefetch all classes embodied in or identified by the Java™ applet bytecode to generate the Downloadable ID. Similarly, the ID generator 315 may retrieve all components listed in the .INF file for an ActiveX™ control to compute a Downloadable ID. Accordingly, the Downloadable ID for the Downloadable will be the same each time the ID

generator 315 receives the same Downloadable. The ID generator 315 adds the generated Downloadable ID to the list of known Downloadables 307 (if it is not already listed). The ID generator 315 then forwards the Downloadable and Downloadable ID to the policy finder 317.

5 The policy finder 317 uses the userID of the intended user and the Downloadable ID to select the specific security policy 305 that shall be applied on the received Downloadable. If there is a specific policy 305 that was defined for the user (or for one of its super groups) and the Downloadable, then the policy is selected. Otherwise the generic policy 305 that was defined for the user (or for one of its super groups) is selected. The policy finder 317
10 then sends the policy to the first comparator 320.

 The first comparator 320 receives the Downloadable, the Downloadable ID and the security policy 305 from the policy finder 317. The first comparator 320 examines the security policy 305 to determine which steps are needed for allowing the Downloadable. For example, the security policy 305 may indicate that, in order to allow this Downloadable, it
15 must pass all four paths, Path 1, Path 2, Path 3 and Path 4. Alternatively, the security policy 305 may indicate that to allow the Downloadable, the it must pass only one of the paths. The first comparator 320 responds by forwarding the proper information to the paths identified by the security policy 305.

20 Path 1

 In path 1, the first comparator 320 checks the policy selector 405 of the security policy 305 that was received from the policy finder 317. If the policy selector 405 is either "Allowed" or "Blocked," then the first comparator 320 forwards this result directly to the logical engine 333. Otherwise, the first comparator 320 invokes the comparisons in path2

and/or path 3 and/or path 4 based on the contents of policy selector 405. It will be appreciated that the first comparator 320 itself compares the Downloadable ID against the lists of Downloadables to allow or block per administrative override 425. That is, the system security administrator can define specific Downloadables as "Allowed" or "Blocked."

5 Alternatively, the logical engine 333 may receive the results of each of the paths and based on the policy selector 405 may institute the final determination whether to allow or block the Downloadable. The first comparator 320 informs the logical engine 333 of the results of its comparison.

10 Path 2

In path 2, the first comparator 320 delivers the Downloadable, the Downloadable ID and the security policy 305 to the code scanner 325. If the DSP data 310 of the received Downloadable is known, the code scanner 325 retrieves and forwards the information to the ACL comparator 330. Otherwise, the code scanner 325 resolves the DSP data 310. That is,
15 the code scanner 325 uses conventional parsing techniques to decompose the code (including all prefetched components) of the Downloadable into the DSP data 310. DSP data 310 includes the list of all potentially hostile or suspicious computer operations that may be attempted by a specific Downloadable 307, and may also include the respective arguments of these operations. For example, DSP data 310 may include a READ from a specific file, a
20 SEND to an unresolved host, etc. The code scanner 325 may generate the DSP data 310 as a list of all operations in the Downloadable code which could ever be deemed potentially hostile and a list of all files to be accessed by the Downloadable code. It will be appreciated that the code scanner 325 may search the code for any pattern, which is undesirable or suggests that the code was written by a hacker.

An Example List of Operations Deemed Potentially Hostile

- File operations: READ a file, WRITE a file;
- Network operations: LISTEN on a socket, CONNECT to a socket, SEND data,
5. RECEIVE data, VIEW INTRANET;
- Registry operations: READ a registry item, WRITE a registry item;
- Operating system operations: EXIT WINDOWS, EXIT BROWSER, START
PROCESS/THREAD, KILL PROCESS/THREAD, CHANGE PROCESS/THREAD
PRIORITY, DYNAMICALLY LOAD A CLASS/LIBRARY, etc.; and
- 10 • Resource usage thresholds: memory, CPU, graphics, etc.

In the preferred embodiment, the code scanner 325 performs a full-content inspection. However, for improved speed but reduced security, the code scanner 325 may examine only a portion of the Downloadable such as the Downloadable header. The code scanner 325 then
15 stores the DSP data into DSP data 310 (corresponding to its Downloadable ID), and sends the Downloadable, the DSP data to the ACL comparator 330 for comparison with the security policy 305.

The ACL comparator 330 receives the Downloadable, the corresponding DSP data and the security policy 305 from the code scanner 325, and compares the DSP data against
20 the security policy 305. That is, the ACL comparator 330 compares the DSP data of the received Downloadable against the access control lists 410 in the received security policy 305. The access control list 410 contains criteria indicating whether to pass or fail the Downloadable. For example, an access control list may indicate that the Downloadable fails

if the DSP data includes a WRITE command to a system file. The ACL comparator 330 sends its results to the logical engine 333.

Path 3:

3 In path 3, the certificate scanner 340 determines whether the received Downloadable was signed by a certificate authority, such as VeriSign, Inc., and scans for a certificate embodied in the Downloadable. The certificate scanner 340 forwards the found certificate to the certificate comparator 345. The certificate comparator 345 retrieves known certificates 309 that were deemed trustworthy by the security administrator and compares the found
10 certificate with the known certificates 309 to determine whether the Downloadable was signed by a trusted certificate. The certificate comparator 345 sends the results to the logical engine 333.

Path 4:

15 In path 4, the URL comparator 350 examines the URL identifying the source of the Downloadable against URLs stored in the URL rule base 420 to determine whether the Downloadable comes from a trusted source. Based on the security policy 305, the URL comparator 350 may deem the Downloadable suspicious if the Downloadable comes from an untrustworthy source or if the Downloadable did not come from a trusted source. For
20 example, if the Downloadable comes from a known hacker, then the Downloadable may be deemed suspicious and presumed hostile. The URL comparator 350 sends its results to the logical engine 333.

The logical engine 333 examines the results of each of the paths and the policy selector 405 in the security policy 305 to determine whether to allow or block the Downloadable. The policy selector 405 includes a logical expression of the results received from each of the paths. For example, the logical engine 333 may block a Downloadable if it fails any one of the paths, i.e., if the Downloadable is known hostile (Path 1), if the Downloadable may request suspicious operations (Path 2), if the Downloadable was not signed by a trusted certificate authority (Path 3), or if the Downloadable did come from an untrustworthy source (Path 4). The logical engine 333 may apply other logical expressions according to the policy selector 405 embodied in the security policy 305. If the policy selector 405 indicates that the Downloadable may pass, then the logical engine 333 passes the Downloadable to its intended recipient. Otherwise, if the policy selector 405 indicates that the Downloadable should be blocked, then the logical engine 333 forwards a non-hostile Downloadable to the intended recipient to inform the user that internal network security system 110 discarded the original Downloadable. Further, the logical engine 333 forwards a status report to the record-keeping engine 335, which stores the reports in event log 245 in the data storage device 230 for subsequent review, for example, by the MIS director.

FIG. 5 is a block diagram illustrating details of the security management console 120, which includes a security policy editor 505 coupled to the communications channel 135, an event log analysis engine 510 coupled between communications channel 135 and a user notification engine 515, and a Downloadable database review engine 520 coupled to the communications channel 135. The security management console 120 further includes computer components similar to the computer components illustrated in FIG. 2.

The security policy editor 305 uses an I/O interface similar to I/O interface 215 for enabling authorized user modification of the security policies 305. That is, the security policy editor 305 enables the authorized user to modify specific security policies 305 corresponding to the users 260, the default or generic security policy 305, the Downloadables to block per administrative override, the Downloadables to allow per administrative override, the trusted certificate lists 415, the policy selectors 405, the access control lists 410, the URLs in the URL rule bases 420, etc. For example, if the authorized user learns of a new hostile Downloadable, then the user can add the Downloadable to the Downloadables to block per system override.

The event log analysis engine 510 examines the status reports contained in the event log 245 stored in the data storage device 230. The event log analysis engine 510 determines whether notification of the user (e.g., the security system manager or MIS director) is warranted. For example, the event log analysis engine 510 may warrant user notification whenever ten (10) suspicious Downloadables have been discarded by internal network security system 110 within a thirty (30) minute period, thereby flagging a potential imminent security threat. Accordingly, the event log analysis engine 510 instructs the user notification engine 515 to inform the user. The user notification engine 515 may send an e-mail via internal communications interface 220 or via external communications interface 210 to the user, or may display a message on the user's display device (not shown).

FIG. 6A is a flowchart illustrating a method 600 for protecting an internal computer network 115 from suspicious Downloadables. Method 600 begins with the ID generator 315 in step 602 receiving a Downloadable. The ID generator 315 in step 604 generates a Downloadable ID identifying the received Downloadable, preferably, by generating a digital

hash of the Downloadable code (including prefetched components). The policy finder 317 in step 606 finds the appropriate security policy 305 corresponding to the userID specifying intended recipient (or the group to which the intended recipient belongs) and the Downloadable. The selected security policy 305 may be the default security policy 305.

5 Step 606 is described in greater detail below with reference to FIG. 6B.

The first comparator 320 in step 608 examines the lists of Downloadables to allow or to block per administrative override 425 against the Downloadable ID of the incoming Downloadable to determine whether to allow the Downloadable automatically. If so, then in step 612 the first comparator 320 sends the results to the logical engine 333. If not, then the

10 method 600 proceeds to step 610. In step 610, the first comparator 620 examines the lists of Downloadables to block per administrative override 425 against the Downloadable ID of the incoming Downloadable for determining whether to block the Downloadable automatically. If so, then the first comparator 420 in step 612 sends the results to the logical engine 333. Otherwise, method 600 proceeds to step 614.

15 In step 614, the first comparator 320 determines whether the security policy 305 indicates that the Downloadable should be tested according to Path 4. If not, then method 600 jumps to step 618. If so, then the URL comparator 350 in step 616 compares the URL embodied in the incoming Downloadable against the URLs of the URL rules bases 420, and then method 600 proceeds to step 618.

20 In step 618, the first comparator 320 determines whether the security policy 305 indicates that the Downloadable should be tested according to Path 2. If not, then method 600 jumps to step 620. Otherwise, the code scanner 235 in step 626 examines the DSP data 310 based on the Downloadable ID of the incoming Downloadable to determine whether the Downloadable has been previously decomposed. If so, then method 600 jumps to step 630.

Otherwise, the code scanner 325 in step 628 decomposes the Downloadable into DSP data. Downloadable decomposition is described in greater detail with reference to FIG. 7. In step 630, the ACL comparator 330 compares the DSP data of the incoming Downloadable against the access control lists 410 (which include the criteria necessary for the Downloadable to fail or pass the test).

In step 620, the first comparator 320 determines whether the security policy 303 indicates that the Downloadable should be tested according to Path 3. If not, then method 600 returns to step 612 to send the results of each of the test performed to the logical engine 333. Otherwise, the certificate scanner 622 in step 622 scans the Downloadable for an embodied certificate. The certificate comparator 345 in step 634 retrieves trusted certificates from the trusted certificate lists (TCL) 415 and compares the embodied certificate with the trusted certificates to determine whether the Downloadable has been signed by a trusted source. Method 600 then proceeds to step 612 by the certificate scanner 345 sending the results of each of the paths taken to the logical engine 333. The operations of the logical engine 333 are described in greater detail below with reference to FIG. 6C. Method 600 then ends.

One skilled in the art will recognize that the tests may be performed in a different order, and that each of the tests need not be performed. Further, one skilled in the art will recognize that, although path 1 is described in FIG. 6A as an automatic allowance or blocking, the results of Path 1 may be another predicate to be applied by the logical engine 333. Further, although the tests are shown serially in FIG. 6A, the tests may be performed in parallel as illustrated in FIG. 3.

FIG. 6B is a flowchart illustrating details of step 606 of FIG. 6A (referred to herein as method 606). Method 606 begins with the policy finder 317 in step 650 determining whether security policies 305 include a specific security policy corresponding to the userID and the Downloadable. If so, then the policy finder 317 in step 654 fetches the
5 corresponding specific policy 305. If not, then the policy finder 317 in step 652 fetches the default or generic security policy 305 corresponding to the userID. Method 606 then ends.

FIG. 6C is a flowchart illustrating details of a method 655 for determining whether to allow or to block the incoming Downloadable. Method 655 begins with the logical engine
10 333 in step 660 receiving the results from the first comparator 320, from the ACL comparator 330, from the certificate comparator 345 and from the URL comparator 350. The logical engine 333 in step 662 compares the results with the policy selector 405 embodied in the security policy 305, and in step 664 determines whether the policy selector 405 confirms the pass. For example, the policy selector 405 may indicate that the logical
15 engine 333 pass the Downloadable if it passes one of the tests of Path 1, Path 2, Path 3 and Path 4. If the policy selector 405 indicates that the Downloadable should pass, then the logical engine 333 in step 666 passes the Downloadable to the intended recipient. In step 668, the logical engine 333 sends the results to the record-keeping engine 335, which in turn stores the results in the event log 245 for future review. Method 655 then ends. Otherwise,
20 if the policy selector 405 in step 664 indicates that the Downloadable should not pass, then the logical engine 333 in step 670 stops the Downloadable and in step 672 sends a non-hostile substitute Downloadable to inform the user that the incoming Downloadable has been blocked. Method 655 then jumps to step 668.

FIG. 7 is a flowchart illustrating details of step 628 of FIG. 6A (referred to herein as method 628) for decomposing a Downloadable into DSP data 310. Method 628 begins in step 705 with the code scanner 325 disassembling the machine code of the Downloadable. The code scanner 325 in step 710 resolves a respective command in the machine code, and in
5 step 715 determines whether the resolved command is suspicious (e.g., whether the command is one of the operations identified in the list described above with reference to FIG. 3). If not, then the code scanner 325 in step 725 determines whether it has completed decomposition of the Downloadable, i.e., whether all operations in the Downloadable code have been resolved. If so, then method 628 ends. Otherwise, method 628 returns to step
10 710.

Otherwise, if the code scanner 325 in step 715 determines that the resolved command is suspect, then the code scanner 325 in step 720 decodes and registers the suspicious command and its command parameters as DSP data 310. The code scanner 325 in step 720 registers the commands and command parameters into a format based on command class
15 (e.g., file operations, network operations, registry operations, operating system operations, resource usage thresholds). Method 628 then jumps to step 725.

FIG. 8 is a flowchart illustrating a method 800 for generating a Downloadable ID for identifying a Downloadable. Method 800 begins with the ID generator 315 in step 810
20 receiving a Downloadable from the external computer network 105. The ID generator 315 in step 820 may fetch some or all components referenced in the Downloadable code, and in step 830 includes the fetched components in the Downloadable code. The ID generator 315 in step 840 performs a hashing function on at least a portion of the Downloadable code to generate a Downloadable ID. The ID generator 315 in step 850 stores the generated

Downloadable ID in the security database 240 as a reference to the DSP data 310. Accordingly, the Downloadable ID will be the same for the identical Downloadable each time it is encountered.

5 The foregoing description of the preferred embodiments of the invention is by way of example only, and other variations of the above-described embodiments and methods are provided by the present invention. For example, although the invention has been described in a system for protecting an internal computer network, the invention can be embodied in a system for protecting an individual computer. Components of this invention may be
10 implemented using a programmed general purpose digital computer, using application specific integrated circuits, or using a network of interconnected conventional components and circuits. The embodiments described herein have been presented for purposes of illustration and are not intended to be exhaustive or limiting. Many variations and modifications are possible in light of the foregoing teaching. The system is limited only by
15 the following claims.

1 WHAT IS CLAIMED IS:

2

- 3 1. A computer-based method, comprising the steps of:
4 receiving a Downloadable;
5 comparing the Downloadable against a security policy to determine if the
6 security policy has been violated; and
7 discarding the Downloadable if the security policy has been violated.

8

- 9 2. The method of claim 1, further comprising the steps of decomposing the
10 Downloadable into Downloadable security profile data, and comparing the
11 Downloadable security profile data against the security policy.

12

- 13 3. The method of claim 1, further comprising the steps of scanning for a
14 certificate and comparing the certificate against a trusted certificate.

15

- 16 4. The method of claim 1, further comprising the step of comparing the URL
17 from which the Downloadable originated against a known URL.

18

- 19 5. The method of claim 4, wherein the known URL is a trusted URL.

20

- 21 6. The method of claim 4, wherein the known URL is an untrusted URL.

22

- 23 7. The method of claim 1, wherein the Downloadable includes a Java™ applet.

24

25 8. The method of claim 1, wherein the Downloadable includes an ActiveX™
26 control.

27

28 9. The method of claim 1, wherein the Downloadable includes a JavaScript™
29 script.

30

31 10. The method of claim 1, wherein the Downloadable includes a Visual Basic
32 script.

33

34 11. The method of claim 1, wherein
35 the Downloadable is addressed to a client; and
36 the security policy includes a default security policy to be applied regardless of
37 the client to whom the Downloadable is addressed.

38

39 12. The method of claim 1, wherein
40 the Downloadable is addressed to a client; and
41 the security policy includes a specific security policy to be applied if the
42 Downloadable is addressed to the client.

43

44 13. The method of claim 1, wherein
45 the Downloadable is addressed to a client belonging to a group; and
46 the security policy includes a specific security policy to be applied if the client
47 belongs to a particular group.

48

49 14. The method of claim 1,
50 wherein the Downloadable is addressed to a client; and
51 further comprising, after discarding the Downloadable, the step of sending a
52 substitute non-hostile Downloadable to the client for informing the client.

53

54 15. The method of claim 1, further comprising, after discarding the Downloadable,
55 the step of recording the violation in an event log.

56

57 16. The method of claim 1, further comprising the step of computing a
58 Downloadable ID to identify the Downloadable.

59

60 17. The method of claim 16, further comprising the steps of fetching components
61 identified by the Downloadable and including the fetched components in the
62 Downloadable.

63

64 18. The method of claim 17, further comprising the step of performing a hashing
65 function on the Downloadable.

66

67 19. The method of claim 17, further comprising the step of fetching all
68 components identified by the Downloadable.

69

70 20. The method of claim 1 further comprising the step of examining the intended
71 recipient userID to determine the appropriate security policy.

72

73 21. The method of claim 1, further comprising the step of examining the
74 Downloadable to determine the appropriate security policy.

75

76 22. The method of claim 20, wherein the appropriate security policy includes the
77 default security policy.

78

79 23. The method of claim 26, further comprising the step of including a previously
80 received Downloadable as a known Downloadable.

81

82 24. The method of claim 23, wherein the security policy identifies a
83 Downloadable to be blocked per administrative override.

84

85 25. The method of claim 23, wherein the security policy identifies a
86 Downloadable to be allowed per administrative override.

87

88 26. The method of claim 1, further comprising the step of comparing the
89 Downloadable against a known Downloadable.

90

91 27. The method of claim 26, wherein the known Downloadable is hostile.

92

93 28. The method of claim 26, wherein the known Downloadable is non-hostile.

94

95 29. The method of claim 2, wherein the security policy includes an access control
96 list and further comprising the step of comparing the Downloadable security profile
97 data against the access control list.

98

99 30. The method of claim 1, further comprising the step of informing a user upon
100 detection of a security policy violation.

101

102 31. A system, comprising:

103 a security policy;

104 an interface for receiving a Downloadable; and

105 a comparator, coupled to the interface, for applying the security policy to the
106 Downloadable to determine if the security policy has been violated.

107

108 32. The system of claim 31, wherein the Downloadable includes a Java™ applet.

109

110 33. The system of claim 31, wherein the Downloadable includes ActiveX™
111 control.

112

113 34. The system of claim 31, wherein the Downloadable includes a JavaScript™
114 script.

115

116 35. The system of claim 31, wherein the Downloadable includes a Visual Basic
117 script.

118

119 36. The system of claim 31, wherein
120 the Downloadable is addressed to a client; and
121 the security policy includes a default security policy to be applied regardless of
122 the client to whom the Downloadable is addressed.

123
124 37. The system of claim 31, wherein
125 the Downloadable is addressed to a client; and
126 the security policy includes a specific security policy to be applied if the
127 Downloadable is addressed to the client.

128
129 38. The system of claim 31, wherein
130 the Downloadable is addressed to a client belonging to a group; and
131 the security policy includes a specific security policy to be applied if the client
132 belongs to a particular group.

133
134 39. The system of claim 31, further comprising an ID generator coupled to the
135 interface for computing a Downloadable ID identifying the Downloadable.

136
137 40. The system of claim 39, wherein the ID generator prefetches all components of
138 the Downloadable and uses all components to compute the Downloadable ID.

139
140 41. The system of claim 40, wherein the ID generator computes the digital hash of
141 all the prefetched components.

142

143 42. The system of claim 31, further comprising a policy finder for finding the
144 security policy.

145

146 43. The system of claim 42, wherein the policy finder finds the security policy
147 based on the user.

148

149 44. The system of claim 42 wherein the policy finder finds the security policy
150 based on the user and the Downloadable.

151

152 45. The system of claim 42, wherein the policy finder obtains the default security
153 policy.

154

155 46. The system of claim 31 wherein the comparator examines the security policy
156 to determine which tests to apply.

157

158 47. The system of claim 46 wherein the comparator compares the Downloadable
159 against a known Downloadable.

160

161 48. The system of claim 47, wherein the known Downloadable is hostile.

162

163 49. The system of claim 47, wherein the known Downloadable is non-hostile.

164

165 50. The system of claim 31, wherein the security policy identifies a Downloadable
166 to be blocked per administrative override.

167

168 51. The system of claim 31, wherein the security policy identifies a Downloadable
169 to be allowed per administrative override.

170

171 52. The system of claim 31, wherein
172 the Downloadable is addressed to a client; and
173 the comparator sends a substitute non-hostile Downloadable to the client for
174 informing the client.

175

176 53. The system of claim 31, further comprising a code scanner coupled to the
177 comparator for decomposing the Downloadable into Downloadable security profile
178 data.

179

180 54. The system of claim 53, further comprising an ACL comparator coupled to the
181 code scanner for comparing the Downloadable security profile data against an access
182 control list.

183

184 55. The system of claim 31, further comprising a certificate scanner coupled to the
185 comparator for examining the Downloadable for a certificate.

186

187 56. The system of claim 55, further comprising a certificate comparator coupled to
188 the certificate scanner for comparing the certificate against a trusted certificate.

189

190 57. The system of claim 31, further comprising a URL comparator coupled to the
191 comparator for comparing the URL from which the Downloadable originated against
192 a known URL.

193

194 58. The system of claim 57, wherein the known URL identifies an untrusted URL.

195

196 59. The system of claim 57, wherein the known URL identifies a trusted URL.

197

198 60. The system of claim 31, further comprising a logical engine for responding to
199 the results of the comparison.

200

201 61. The system of claim 31, wherein the logical engine responds according to the
202 security policy.

203

204 62. The system of claim 31, further comprising a record-keeping engine coupled
205 to the comparator for recording results in an event log.

206

207 63. A system, comprising:

208 means for receiving a Downloadable;

209 means for comparing the Downloadable against a security policy to determine
210 if the security policy has been violated; and

211 means for discarding the Downloadable if the security policy has been
212 violated.

213

- 214 64. A computer-readable storage medium storing program code for causing a
215 computer to perform the steps of:
216 receiving a Downloadable;
217 comparing the Downloadable against a security policy to determine if the
218 security policy has been violated; and
219 discarding the Downloadable if the security policy has been violated.
220
- 221 65. A computer-based method for generating a Downloadable ID to identify a
222 Downloadable, comprising the steps of:
223 selecting Downloadable code;
224 performing a function on the selected Downloadable code to generate the
225 Downloadable ID; and
226 storing the Downloadable ID.
227
- 228 66. The method of claim 65, wherein the function includes a hashing function

67. The method of claim 65, wherein the Downloadable code includes a reference to a Downloadable component, and further comprising the step of fetching the component.

5

68. The method of claim 67, wherein the component includes the first component referenced by the Downloadable code.

69. The method of claim 65, wherein the selected Downloadable code includes all
10 of the code included in and identified by the Downloadable.

70. The method of claim 67, further comprising the step of fetching all components referenced by the Downloadable.

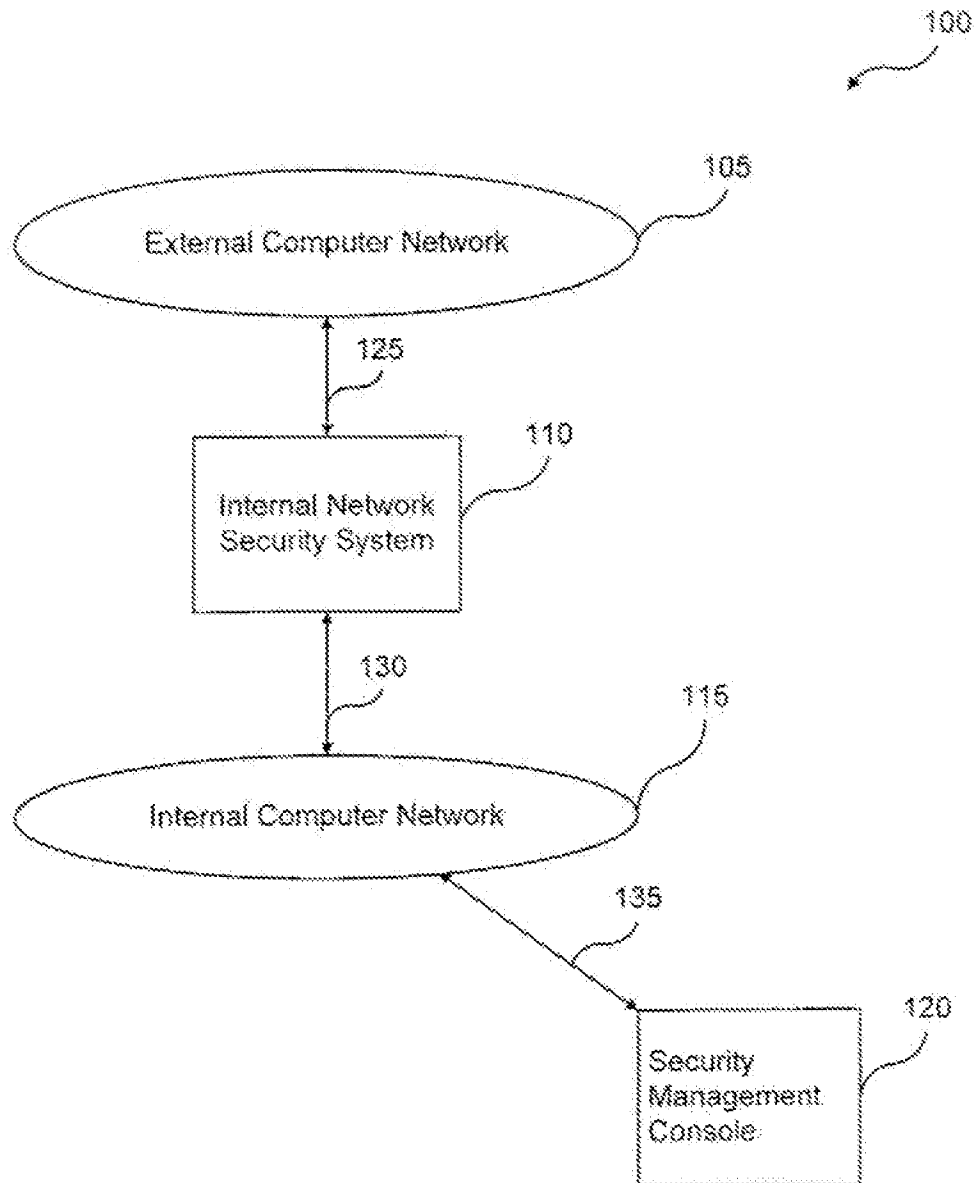


FIG. 1

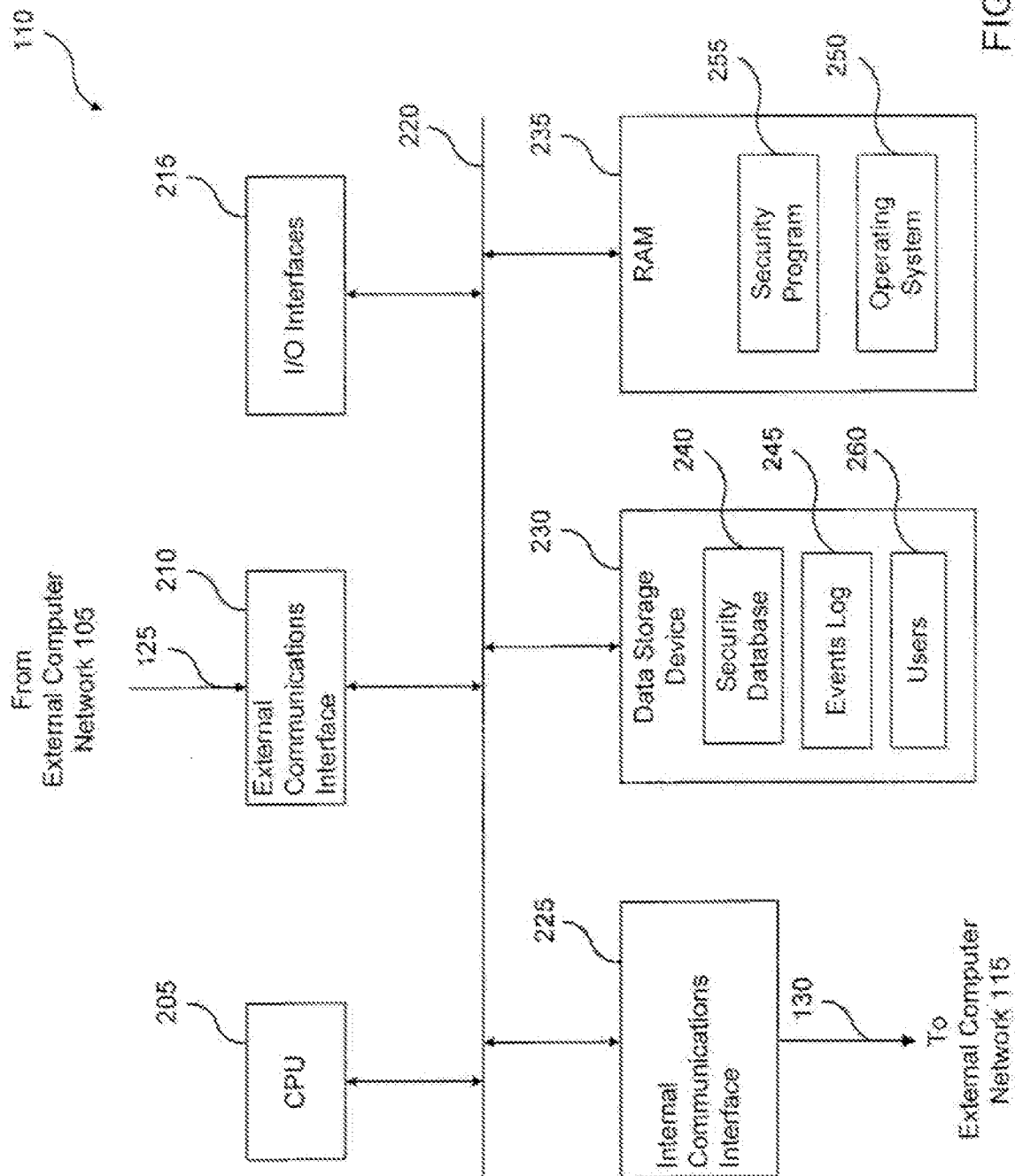


FIG. 2

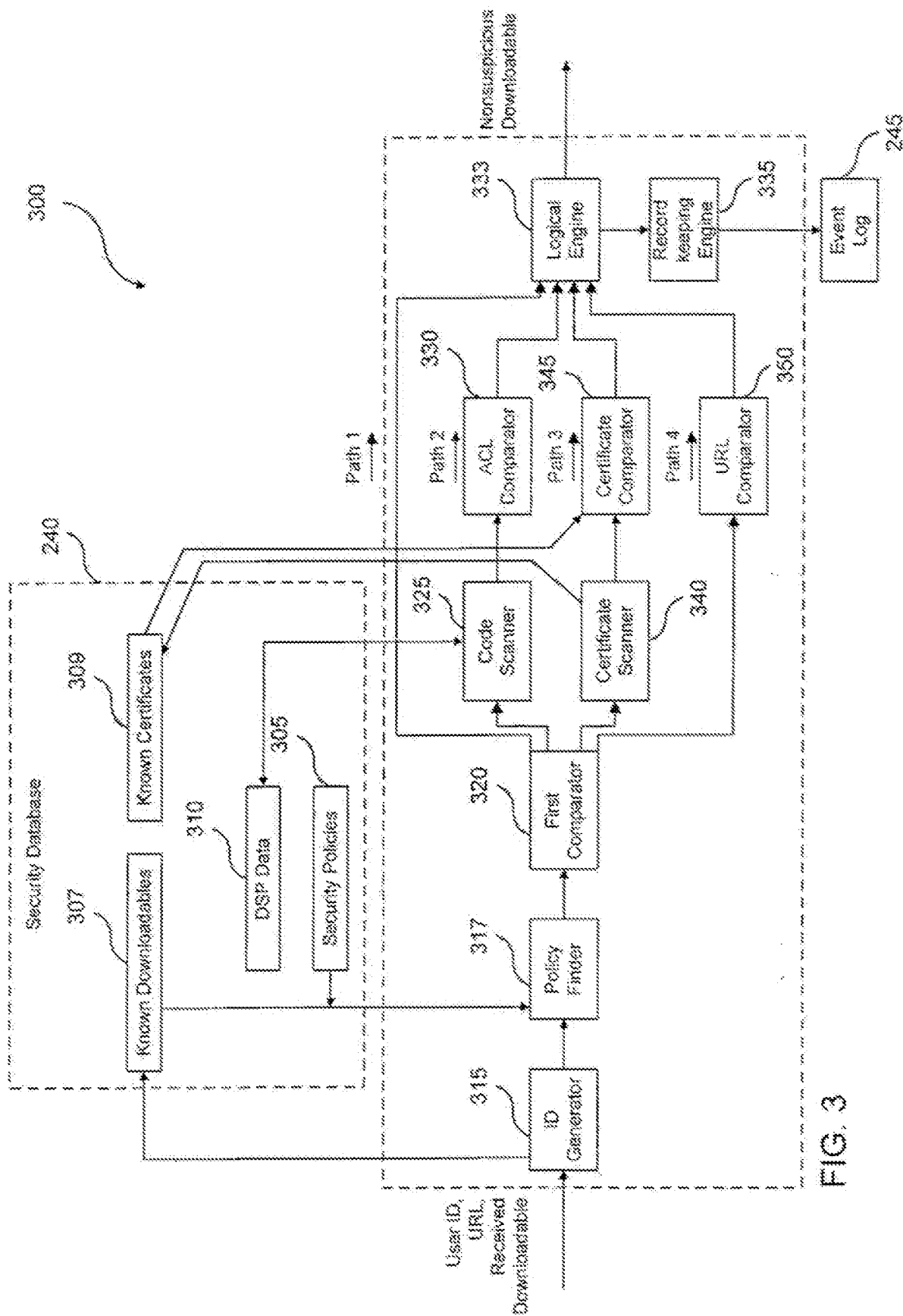


FIG. 3

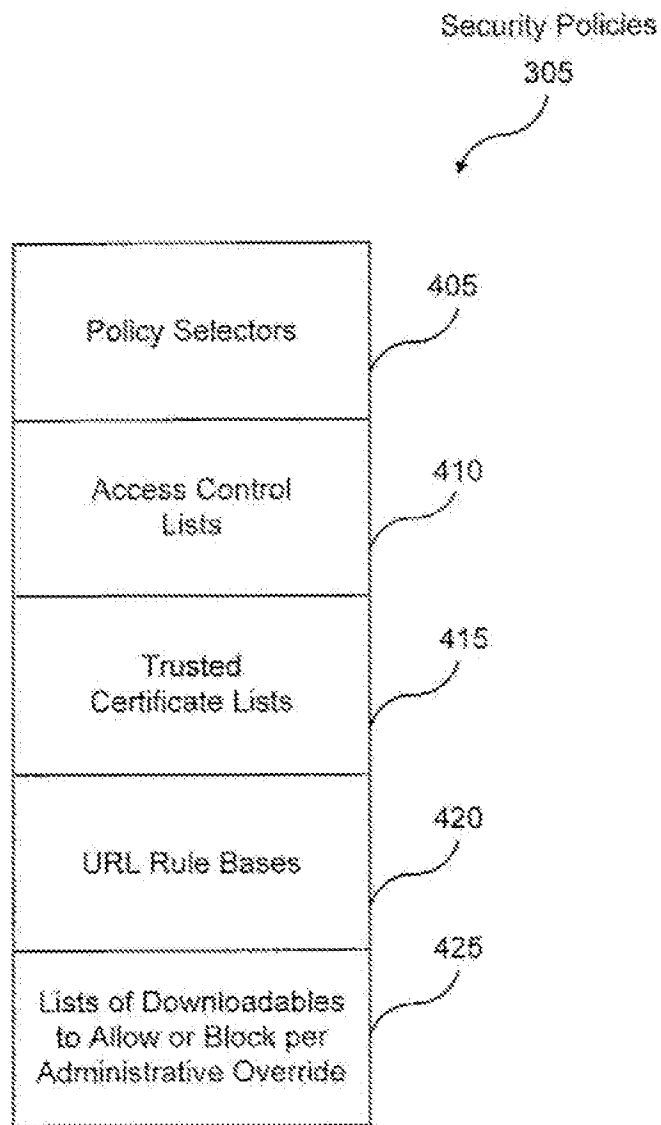


FIG. 4

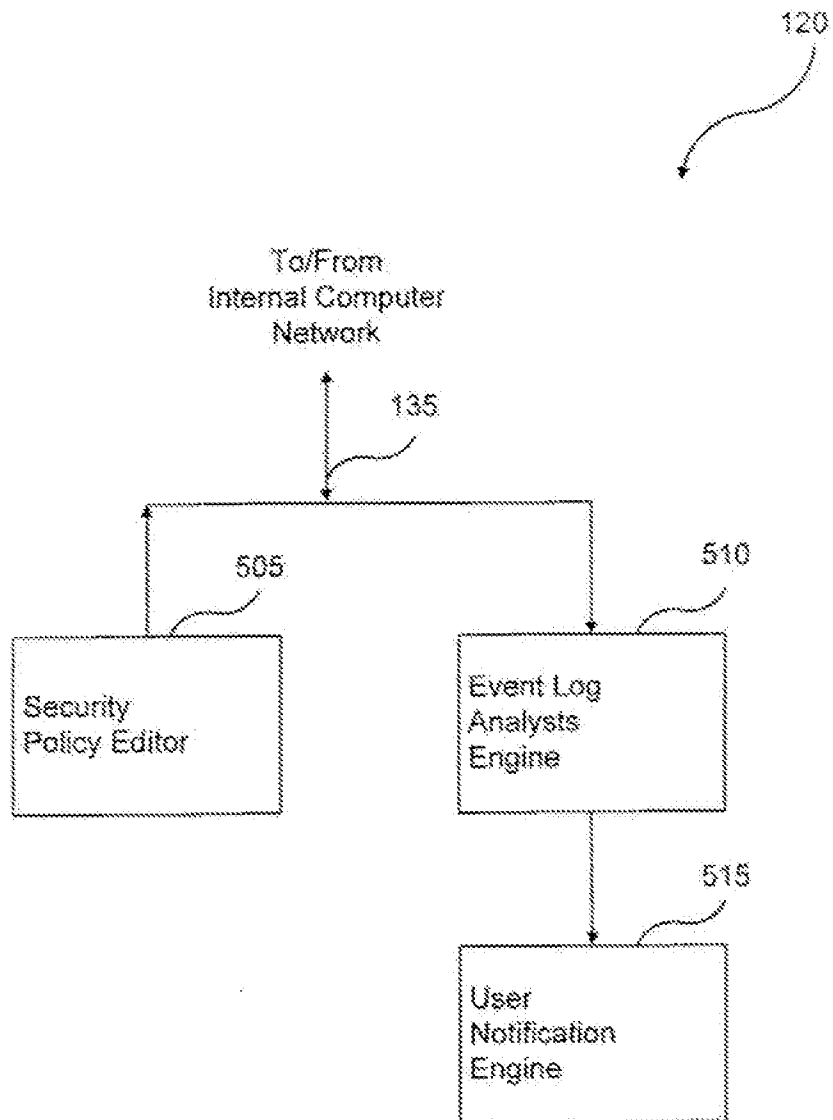


FIG. 5

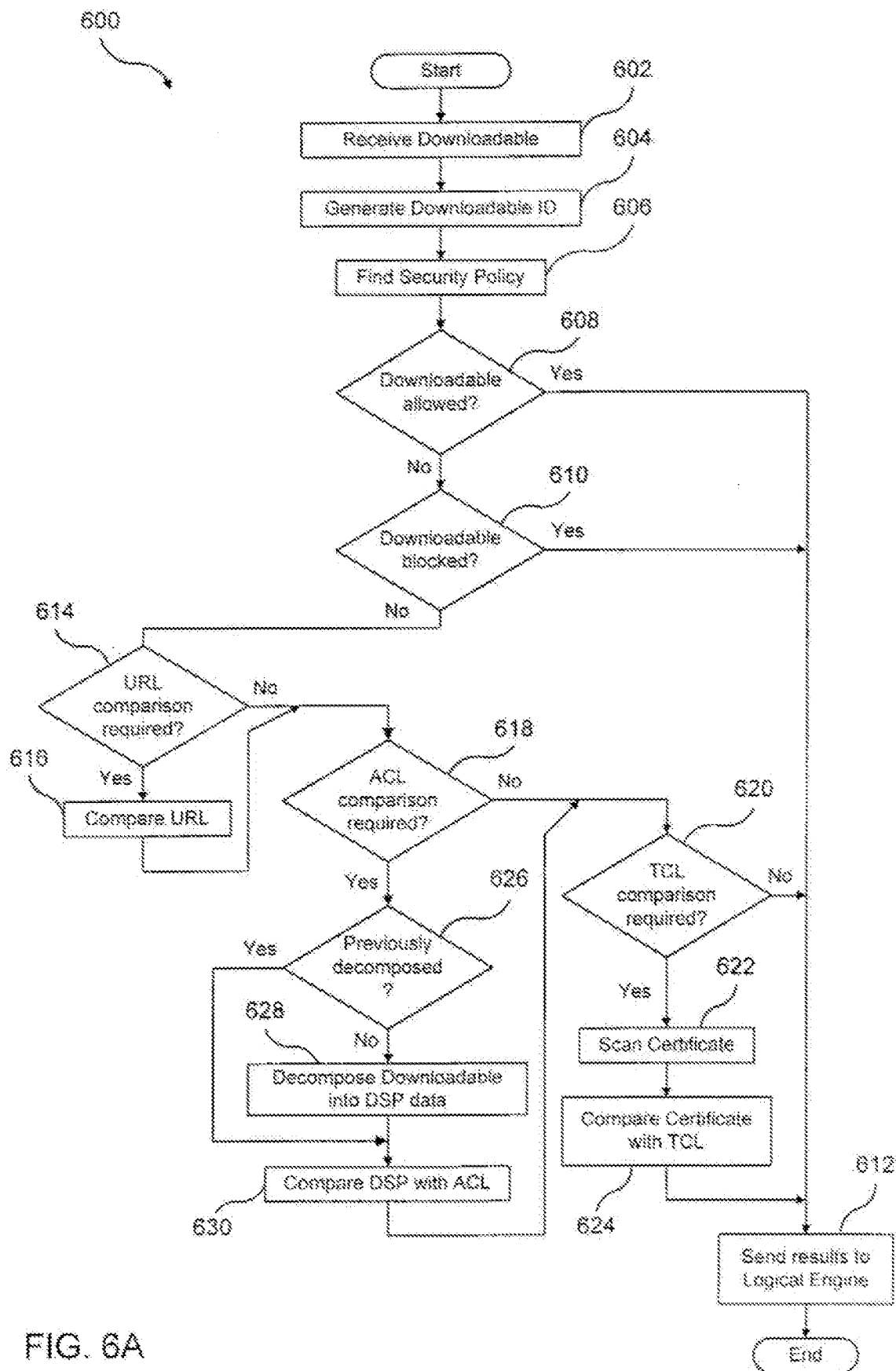


FIG. 6A

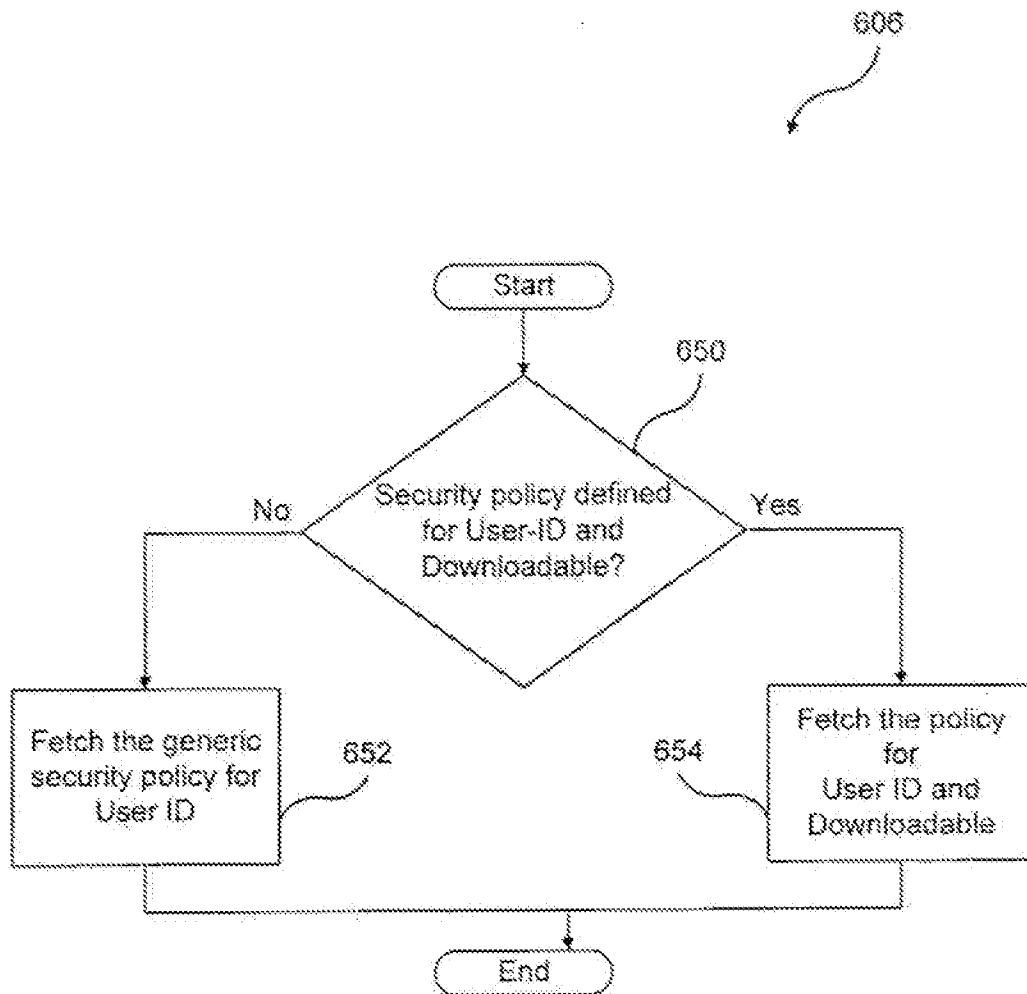


FIG. 6B

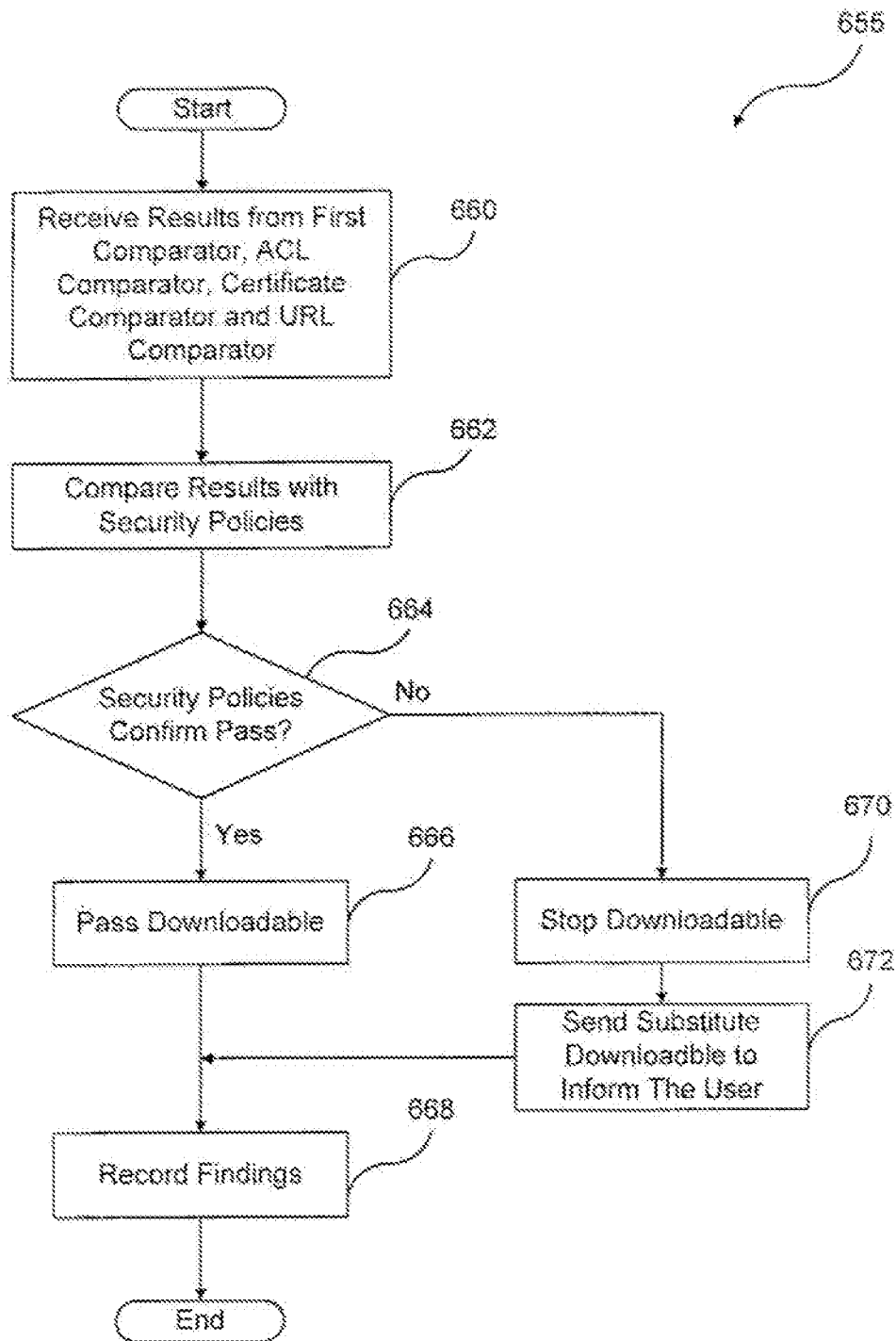


FIG. 6C

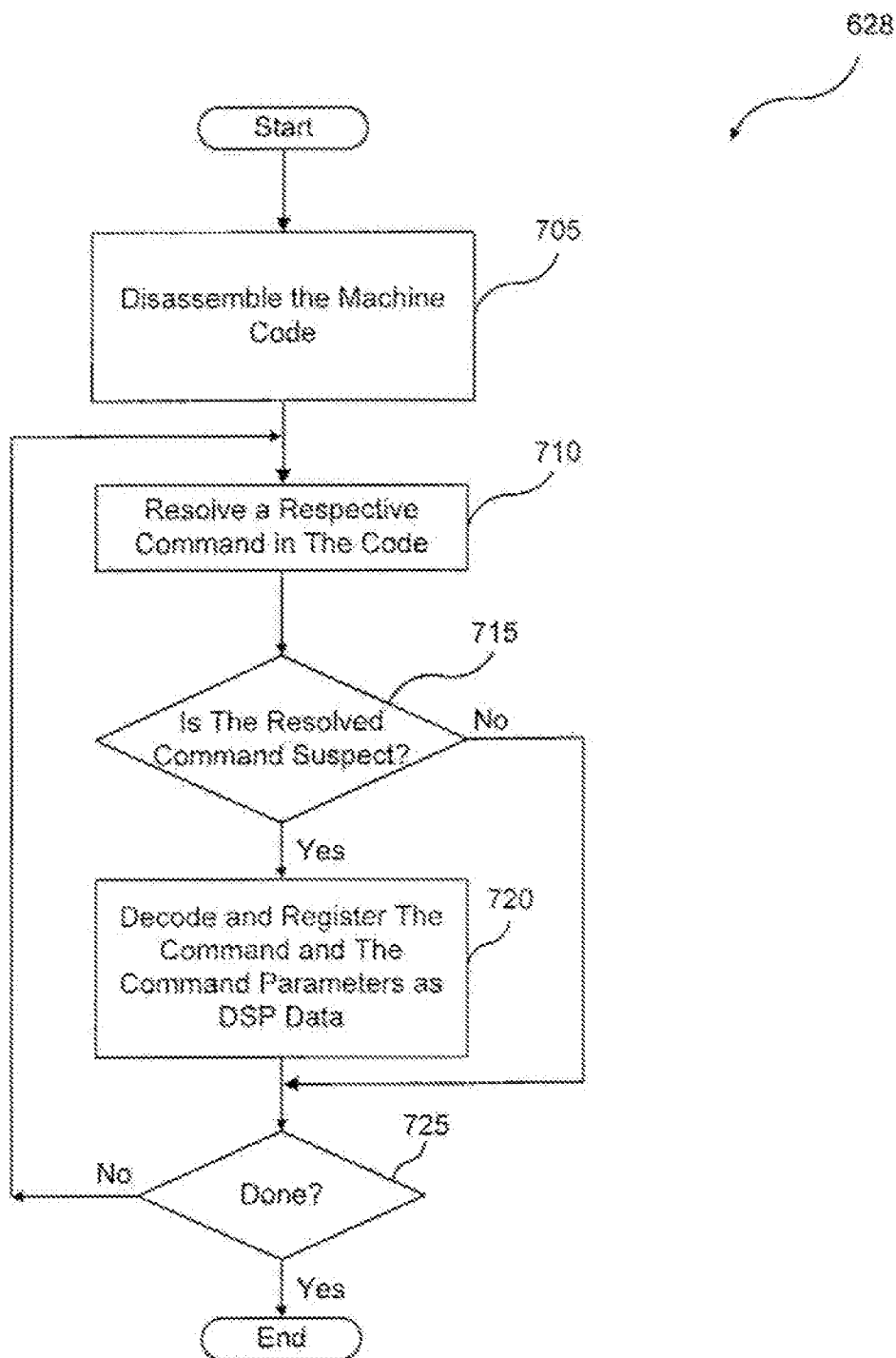


FIG. 7

10/10

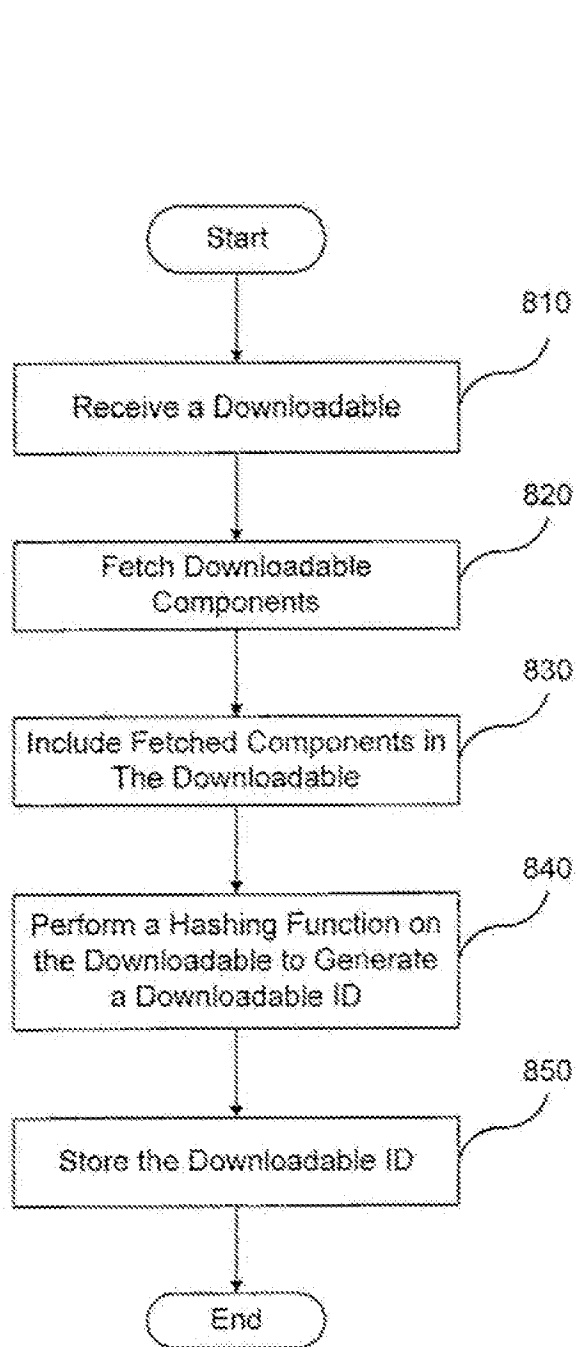


FIG. 8

**Espacenet****Bibliographic data: JPH1021134 (A) — 1998-01-23**

**GATE WAY DEVICE, CLIENT COMPUTER AND DISTRIBUTED FILE SYSTEM
CONNECTING THEM**

Inventor(s): DOI KATSUYOSHI; TAKEZAWA SO ± (DOI KATSUYOSHI, ;
TAKEZAWA SO)

Applicant(s): SHARP KK ± (SHARP CORP)

Classification: - **international:** **G06F12/00; G06F13/00; G06F17/30; G06F3/06;
G06F9/50;** (IPC1-7): G06F12/00; G06F13/00;
G06F3/06
- **cooperative:**

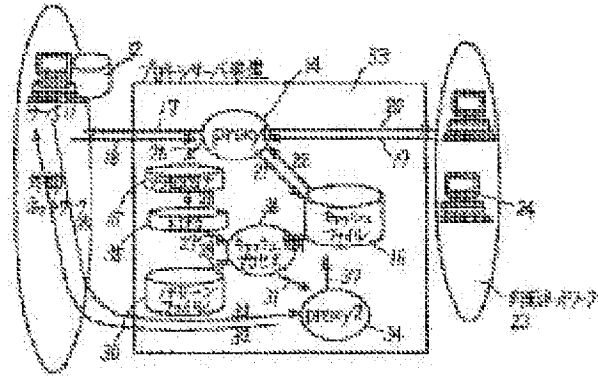
Application number: JP19960174561 19960704

Priority number (s): JP19960174561 19960704

Also published as: JP3481054 (B2)

Abstract of JPH1021134 (A)

PROBLEM TO BE SOLVED: To provide a gate way device improved in a cache hit rate while keeping a cache file fresh. SOLUTION: This system includes a first proxy process 14 for storing the acquired file object in a cache file 16 and storing an access log 15, a cache updating process 36 for extracting an access log from the access log 15 in accordance with a prescribed rule and a second proxy process 36 for acquiring the file object of the extracted access log from a server computer 11 and storing it in the cache file 16. When the first proxy process 14 is hit to a cache, the cache updating process 36 becomes more effective when second proxy process 34 updates cache contents based on the access log of the file object hit to the cache.



(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平10-21134

(43)公開日 平成10年(1998)1月23日

(51)Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 12/00	5 4 5		G 0 6 F 12/00	5 4 5 A
	5 1 4			5 1 4 K
3/06	3 0 1		3/06	3 0 1 M
13/00	3 5 7		13/00	3 5 7 Z

審査請求 未請求 請求項の数12 O L (全 41 頁)

(21)出願番号 特願平8-174561

(22)出願日 平成8年(1996)7月4日

(71)出願人 000005049

シャープ株式会社

大阪府大阪市阿倍野区長池町22番22号

(72)発明者 土居 克良

大阪府大阪市阿倍野区長池町22番22号 シ
ャープ株式会社内

(72)発明者 竹澤 創

大阪府大阪市阿倍野区長池町22番22号 シ
ャープ株式会社内

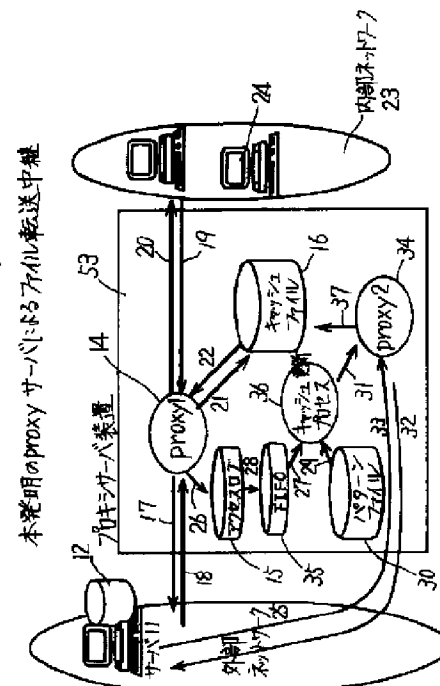
(74)代理人 弁理士 深見 久郎

(54)【発明の名称】 ゲートウェイ装置、クライアント計算機およびそれらを接続した分散ファイルシステム

(57)【要約】

【課題】 キャッシュファイルを新鮮に保ちながら、キャッシュヒット率の向上したゲートウェイ装置を提供すること。

【解決手段】 取得したファイルオブジェクトをキャッシュファイル16に格納し、アクセスログ15を格納する第1のproxyプロセス14に加えて、アクセスログ15から所定の規則に従ってアクセスログを抽出するキャッシュ更新プロセス36と、この抽出されたアクセスログのファイルオブジェクトをサーバ計算機11から取得してキャッシュファイル16に格納する第2のproxyプロセス34を含む。キャッシュ更新プロセス36は、第1のproxyプロセス14がキャッシュにヒットした場合にそのキャッシュにヒットしたファイルオブジェクトのアクセスログに基づいて第2のproxyプロセス34がキャッシュ内容を更新するとさらに効果的である。



【特許請求の範囲】

【請求項1】 クライアント計算機が接続される第1のネットワークと、サーバ計算機が接続される第2のネットワークとの間に介在するように設けられるゲートウェイ装置であって、

前記第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に応答して、当該ファイルオブジェクトにより指定される前記第2のネットワークのサーバ計算機に対して、当該ファイルオブジェクトに対するアクセス要求を行なうネットワークファイル中継手段を含み、

前記ネットワークファイル中継手段は、取得したファイルオブジェクトを、所定の有効期限が経過するまで一時的に蓄積するとともに、最終変更時刻をファイルオブジェクト単位で記録するためのキャッシュファイル手段と、

過去の一定期間内に行なわれたファイルオブジェクトの転送記録を蓄積するファイル転送記録手段と、ファイルオブジェクトに対するアクセス要求に応答して、当該ファイルオブジェクトの最終変更時刻を参照し、有効期限内のファイルオブジェクトが前記キャッシュファイル手段に存在する場合は当該ファイルオブジェクトをクライアント計算機に転送し、有効期限内のファイルオブジェクトが前記キャッシュファイル手段に存在しない場合には当該ファイルオブジェクトにより指定されるサーバ計算機の当該ファイルオブジェクトを取得してクライアント計算機に転送するための転送手段とを含み、

前記ゲートウェイ装置はさらに、前記ファイル転送記録手段から所定の規則に従って転送記録を抽出し、当該転送記録のファイルオブジェクトを前記第2のネットワークのサーバ計算機から取得して前記キャッシュファイル手段に格納するためのキャッシュ更新手段を含むゲートウェイ装置。

【請求項2】 前記キャッシュ更新手段はさらに、特定の文字列パターンを予め記憶するための文字列パターン記憶手段を含み、

前記キャッシュ更新手段は、前記特定の文字列パターンと一致するファイルオブジェクト名称を含む転送記録を前記ファイル転送記録手段から抽出し、当該ファイルオブジェクトを前記第2のネットワークのサーバ計算機から取得して前記キャッシュファイル手段に格納する、請求項1記載のゲートウェイ装置。

【請求項3】 前記ネットワークファイル中継手段は、各々が第1のネットワークのクライアント計算機からのファイルオブジェクトのアクセス要求を処理するための第1の転送プロセスを複数個、同時並列的に起動させ、前記キャッシュ更新手段は、各々が1つの前記ファイルオブジェクトの取得を行なうための第2の転送プロセスを複数個、同時並列的に起動させ、前記第1の転送プロ

セスの数を検知して、前記第1の転送プロセスの数と前記第2の転送プロセスの数との和が予め定められる上限以下となるように前記第2の転送プロセスの新たな起動を制御する、請求項1記載のゲートウェイ装置。

【請求項4】 前記キャッシュ更新手段は、前記ファイル転送記録手段の中から、キャッシュにヒットしたファイルオブジェクトの転送記録を抽出し、当該ファイルオブジェクトを前記第2のネットワークのサーバ計算機から取得して前記キャッシュファイル手段に格納する、請求項1記載のゲートウェイ装置。

【請求項5】 前記キャッシュ更新手段はさらに、前記ファイル転送記録手段の末尾から転送記録を順次読出し格納し、先に格納された転送記録から順に出力するFIFOバッファ手段を含み、

前記キャッシュ更新手段は、前記FIFOバッファ手段から転送記録を読出して、当該転送記録のファイルオブジェクトを前記第2のネットワークのサーバ計算機から取得して前記キャッシュファイル手段に格納する、請求項1記載のゲートウェイ装置。

【請求項6】 前記キャッシュ更新手段は、前記ファイル転送記録手段から抽出した転送記録のファイルオブジェクトの前の更新が所定時間以内であれば、前記第2のネットワークのサーバ計算機へのアクセスを行なわない、請求項1記載のゲートウェイ装置。

【請求項7】 クライアント計算機と第1のネットワークによって接続され、かつサーバ計算機と第2のネットワークによって接続される複数のキャッシュサーバ計算機と第3のネットワークによって接続されるゲートウェイ装置であって、

前記第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行なうための演算手段と、当該演算結果に基づいて前記第3のネットワークの複数のキャッシュサーバ計算機の中からアクセス要求を出力するキャッシュサーバ計算機を選択するための中継制御手段とを含むゲートウェイ装置。

【請求項8】 クライアント計算機とゲートウェイ装置とが第1のネットワークで接続され、サーバ計算機とキャッシュサーバ計算機とが第2のネットワークで接続され、複数の前記ゲートウェイ装置と複数の前記キャッシュサーバ計算機とが第3のネットワークで接続される分散ファイルシステムであって、

前記ゲートウェイ装置は、前記第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行なうための演算手段と、

当該演算結果に基づいて前記第3のネットワークの複数のキャッシュサーバ計算機の中からアクセス要求を出力

するキャッシュサーバ計算機を選択するための中継制御手段とを含む分散ファイルシステム。

【請求項9】 クライアント計算機が接続される第1のネットワークと、サーバ計算機が接続される第2のネットワークとの間に介在するように設けられるゲートウェイ装置であって、

前記第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行ない、該演算結果に基づいてゲートウェイ装置を選択し、該選択が他のゲートウェイ装置であれば、当該アクセス要求を第3のネットワークを介して前記他のゲートウェイ装置へ出力するための中継制御手段と、

前記第2のネットワークのサーバ計算機から取得したファイルオブジェクトを、所定の有効期限が経過するまで一時的に蓄積するとともに、最終変更時刻をファイルオブジェクト単位で記録するためのキャッシュファイル手段と、

過去の、一定時間内に行なわれたファイルオブジェクトの転送記録を蓄積するためのファイル転送記録手段と、前記中継制御手段が自身のゲートウェイ装置を選択した場合と前記第3のネットワークを介して他のゲートウェイ装置からアクセス要求を受信した場合に、ファイルオブジェクトに対するアクセス要求に回答して、当該ファイルオブジェクトの最終変更時刻を参照し、有効期限内のファイルオブジェクトが前記キャッシュファイル手段に存在する場合は当該ファイルオブジェクトをクライアント計算機に転送し、有効期限内のファイルオブジェクトが前記キャッシュファイル手段に存在しない場合には、当該ファイルオブジェクトにより指定されるサーバ計算機の当該ファイルオブジェクトを取得してクライアント計算機に転送するための転送手段とを含むゲートウェイ装置。

【請求項10】 サーバ計算機と複数のキャッシュサーバ計算機とが第2のネットワークで接続され、前記複数のキャッシュサーバ計算機と第1のネットワークによって接続されるクライアント計算機であって、キャッシュサーバ計算機に対するファイルオブジェクトのアクセス要求を出力し、当該ファイルオブジェクトを取得するためのファイル取得手段と、

キャッシュサーバ計算機へのファイルオブジェクトに対するアクセス要求を出力する際、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行ない、当該演算結果に基づいて前記複数のキャッシュサーバ計算機の中からアクセス要求を出力するキャッシュサーバ計算機を選択するための中継制御手段とを含むクライアント計算機。

【請求項11】 前記中継制御手段はさらに、第3のネットワークを介して他のクライアント計算機からのファ

イルオブジェクトに対するアクセス要求を受信するための受信手段を含み、

前記中継制御手段は、前記他のクライアント計算機からのアクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行ない、当該演算結果に基づいて前記複数のキャッシュサーバ計算機の中からアクセス要求を出力するキャッシュサーバ計算機を選択する、請求項10記載のクライアント計算機。

【請求項12】 クライアント計算機と第1のネットワークによって接続され、かつサーバ計算機と第2のネットワークによって接続される複数のキャッシュサーバ計算機と第3のネットワークによって接続されるゲートウェイ装置であって、

前記第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行ない、該演算結果に基づいてゲートウェイ装置を選択し、該選択が他のゲートウェイ装置であれば、当該アクセス要求を第3のネットワークを介して前記他のゲートウェイ装置へ出力するための中継制御手段と、

前記第2のネットワークのサーバ計算機から取得したファイルオブジェクトを、所定の有効期限が経過するまで一時的に蓄積するとともに、最終変更時刻をファイルオブジェクト単位で記録するためのキャッシュファイル手段と、

過去の、一定時間内に行なわれたファイルオブジェクトの転送記録を蓄積するためのファイル転送記録手段と、前記中継制御手段が自身のゲートウェイ装置を選択した場合と前記第3のネットワークを介して他のゲートウェイ装置からアクセス要求を受信した場合に、ファイルオブジェクトに対するアクセス要求に回答して、当該ファイルオブジェクトの最終変更時刻を参照し、有効期限内のファイルオブジェクトが前記キャッシュファイル手段に存在する場合は当該ファイルオブジェクトをクライアント計算機に転送し、有効期限内のファイルオブジェクトが前記キャッシュファイル手段に存在しない場合には当該ファイルオブジェクトにより指定されるサーバ計算機の当該ファイルオブジェクトを取得してクライアント計算機に転送するための転送手段とを含むゲートウェイ装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ネットワーク上に分散した複数のサーバ計算機と、複数のクライアント計算機とがゲートウェイ装置（計算機）を介して通信回線で相互に接続されている分散ファイルシステムに関し、特に、複数のクライアント計算機がゲートウェイ装置を介してサーバ計算機内部の記憶装置内のファイルオブジェクトにアクセスする場合の分散ファイルシステムのフ

ファイル中継を行なうゲートウェイ装置、ゲートウェイ装置にアクセス要求を出力するクライアント計算機、およびそれらを接続した分散ファイルシステムに関する。

【0002】

【従来の技術】以下の説明で「サーバ計算機」とは、ネットワーク上において何らかのサービスを提供している計算機をいい、「クライアント計算機」とは、そうしたサーバ計算機のサービスを要求し、受ける計算機をいうものとする。また「ファイルオブジェクト」とは、分散ファイルシステムの利用するネットワークプロトコルと、ネットワークアドレス（サーバ計算機の名称）と、ファイル名称と、ファイルの実体との組をいうものとする。ここで「ファイルオブジェクトの名称」とは、ネットワークプロトコルとネットワークアドレスとファイル名称との組のことをいう。

【0003】従来、上述したような分散ファイルシステムにおいては、複数のクライアント計算機からのサーバ計算機上のファイルオブジェクトに対する読出要求は、一旦途中のゲートウェイ計算機で中継されていた。ゲートウェイ計算機がサーバ計算機から読出したファイルオブジェクトは、クライアント計算機に中継される。それと同時にそれらファイルオブジェクトは、ゲートウェイ計算機内部のキャッシュファイル（固定ディスク、半導体メモリなど）に蓄積される。そうしたゲートウェイ計算機の一例が特開平4-313126号公報（発明の名称「分散ファイルシステムのファイル入出力方式」）に開示されている。

【0004】図15は、この従来の分散ファイルシステムを示している。このシステムは、ゲートウェイ計算機110と、このゲートウェイ計算機110を介して相互に接続されたサーバ計算機100および複数のクライアント計算機120を含む。サーバ計算機100は、複数のクライアント計算機120によって共有されている。

【0005】サーバ計算機100は、ファイルを格納するディスク装置136と、ネットワークを介してディスク装置136のファイルの内容を入出力するためのファイル入出力応答手段134と、ディスク装置136内のブロック情報をネットワークを介してクライアント計算機120に応答するためのブロック情報応答手段132とを含む。

【0006】クライアント計算機120は、サーバ計算機100のブロック情報応答手段132に対してブロック情報要求を送り、ブロック情報を受けるためのブロック情報要求手段152と、ゲートウェイ計算機110を介してサーバ計算機100のファイル入出力応答手段134に対してファイル入出力要求を送り、当該ファイルを受信するためのファイル入出力要求手段154とを含む。

【0007】ゲートウェイ計算機110は、クライアント計算機120（複数）とサーバ計算機100との間に

介在し、サーバ計算機100内のディスク装置136と、各クライアント計算機120との間のキャッシュメモリとして機能するものであり、キャッシュした内容を格納するためのディスクキャッシュ144と、ファイル入出力要求手段154およびファイル入出力応答手段134の間に介在するキャッシュ管理手段142とを含む。

【0008】このシステムは次のように動作する。まず、クライアント計算機120がサーバ計算機100内のディスク装置136内のファイルにアクセスして1つのブロックを読込む場合の動作について説明する。ブロック情報要求手段152は、ゲートウェイ計算機110を介してサーバ計算機100に対して読込対象のブロックにかかるブロック情報の取得を要求するブロック情報要求メッセージを発行する。

【0009】サーバ計算機100内のブロック情報応答手段132は、このメッセージに回答して、ディスク装置136から該当するブロック情報を取出し、そのブロック情報を有するブロック情報応答メッセージをゲートウェイ計算機110を介してクライアント計算機120内のブロック情報要求手段152に返却する。

【0010】クライアント計算機120のファイル入出力要求手段154は、返却されたブロック情報に基づいて読込対象のブロックの読込を要求するファイルアクセス要求を、ゲートウェイ計算機110内のキャッシュ管理手段142に対して発行する。

【0011】このファイルアクセス要求を受取ったキャッシュ管理手段142は、そのファイルアクセス要求にかかるブロック情報とディスクキャッシュ144に記憶されている読込対象のブロックにかかるブロック情報との比較を行なう。そしてディスクキャッシュ144に、該当するブロック情報が存在しない場合、または両方のブロック情報の内容（更新時間）が異なる場合（読込対象のブロックのキャッシュが有効でない場合）には、キャッシュ管理手段142は上述のファイルアクセス要求をサーバ計算機100のファイル入出力応答手段134に対して発行する。

【0012】サーバ計算機100のファイル入出力応答手段134は、このファイルアクセス要求に回答して、ディスク装置136内のファイルをアクセスして、該当するブロックを読込み、ゲートウェイ計算機110にそのブロックを転送する。

【0013】キャッシュ管理手段142は、このブロックをディスクキャッシュ144に格納し、そのブロックにかかるブロック情報のディスクキャッシュ144への設定または更新を行なう。キャッシュ管理手段142は同時に、そのブロックをクライアント計算機120のファイル入出力要求手段154に対して転送する。

【0014】以上でディスクキャッシュ144内に有効なファイルが存在しない場合のこのシステムの動作につ

いて説明した。

【0015】次回に同一ファイルに対するアクセス要求があった場合には、ゲートウェイ計算機110のキャッシュ管理手段142は、ディスクキャッシュ144から当該ブロックを取出し、アクセス要求を発行したクライアント計算機120のファイル入出力要求手段154に対して当該ブロックを転送する。したがってこの場合、ゲートウェイ計算機110からサーバ計算機100に対する転送要求は発行されない。したがって中継アクセスの高速化が図られる。

【0016】類似の技術が、特開昭63-200244号公報（発明の名称「ファイル・アクセス・システム」）や、特開昭63-20184号公報（発明の名称「遠隔ファイル・アクセス・システム」）にも開示されている。いずれの技術においても、あるファイルオブジェクトに対するアクセス要求が発行された場合に、当該ファイルオブジェクトの内容が更新されているかどうかを確認するために、サーバ計算機の保持するファイルオブジェクトの変更時刻をキャッシュファイルの変更時刻と比較する。そしてキャッシュファイルの内容が古い場合には、クライアント計算機は更新されたサーバ計算機の内容を読出すようになっている。しかし、これら2件の公報に記載された技術では、キャッシュファイルはクライアント計算機内に存在しており、特開平4-313126号公報に開示されたシステムのようにシステム内にゲートウェイ計算機が含まれているわけではない。

【0017】しかし、これら3件の従来技術文献は、サーバ計算機のファイルオブジェクトと、キャッシュされたファイルオブジェクトとの内容が食い違わないような工夫をしているという共通点を持っている。

【0018】このように本来同一であるべきファイルの食い違いを防ぐことをインテグリティ（Integrity）を保つという。また、食い違いの起きたファイルのデータをステールデータ（stale data）と呼ぶ。

【0019】上述した従来の技術の基本は、ネットワーク経由でサーバ計算機の、該当するファイルオブジェクトの変更時刻を調べ、対応するキャッシュファイルの内容の変更時刻と比較するアルゴリズムを使用している。そのため、サーバ計算機に対するネットワーク経由の問合せを必ず行なわなければならない。ネットワークの実効的な転送速度が小さい場合や、該当するファイルオブジェクトを有するサーバ計算機にかかっている負荷が高くて即座に回答できない場合には、そうした問合せ自体にかなり時間がかかってしまうことがある。そのためネットワークファイルシステムによっては、このようなファイルキャッシュのシステムを採用していない場合もある。すなわちこうしたシステムでは、サーバ計算機の保持するファイルオブジェクトが更新されているにもかかわらず、ゲートウェイ計算機の持つキャッシュ内の対応するファイルオブジェクトは必ずしも更新されていない

い。このようなネットワークファイルシステム方式の代表的なものとして、いわゆるインターネットにおけるHTTP（Hyper Text Transfer Protocol）を使った広域分散型マルチメディア情報システムがある。

【0020】インターネットは、その基本プロトコルとしてTCP/IPプロトコルを利用した、グローバルなネットワークである。インターネット上に構築された地域分散型マルチメディア情報提供システムは、World Wide Web（WWW）と呼ばれる。WWWは、ネットワーク上に分散したファイルオブジェクトを扱うことができる。これらのファイルオブジェクトは、単なるテキストデータにとどまらず、画像データ、音声データ、ビデオ画像データなどさまざまな種類のものを含む。WWWは、情報提供側にとっても、情報利用者（ユーザ）にとっても魅力的であるため、ネットワーク上のWWWに関するトラフィックが爆発的に増加している。WWWシステムでは、クライアント計算機のユーザは、グラフィカルユーザインタフェースを持ったブラウザソフトウェアをクライアント計算機上で実行させるだけで、ネットワーク上に分散したサーバ計算機の保持するさまざまなファイルオブジェクトで構成された情報を次々とアクセスすることができる。WWWは、このような操作の簡単さのために近時めざましい普及をみせている。

【0021】そしてこのWWWシステムは、TCP/IPプロトコルの上に構築されたHTTPプロトコルでファイルオブジェクトを転送している。

【0022】HTTPプロトコルを実施する上においては、特開平4-313126号公報に記載されたようなゲートウェイ計算機によりファイルオブジェクトを中継転送することが広く行なわれている。データは、ゲートウェイ装置にキャッシュされる。またWWWシステムでは、ファイルオブジェクトの中継転送においてステールデータが発生することは容認されている。すなわちゲートウェイ計算機にキャッシュされている内容が、サーバ計算機の当該ファイルオブジェクトと一致していない場合を許容している。

【0023】キャッシュファイル内のステールデータの率をステールデータ率と呼ぶこととする。何らかの方式でキャッシュファイルの内容を更新しない限り、ステールデータ率は増加する。一般にキャッシュファイルのステールデータ率が増加すると、クライアント計算機を操作しているユーザは、内容が古いと自立的に判断して、ファイルオブジェクトをサーバ計算機から再度ロードすることを試みる。この場合、キャッシュファイルの内容を利用しないプロトコルが用いられる。この結果、ゲートウェイ計算機内のキャッシュファイルシステムが意味を持たなくなる傾向がある。

【0024】一方で、このようなステールデータの発生を容認するプロトコルの利点もある。クライアント計算機が、サーバ計算機のファイルオブジェクトのアクセス

をゲートウェイ計算機に中継依頼してアクセスを行なった場合、ゲートウェイ計算機のキャッシュファイルにヒットした場合には、キャッシュファイルからファイルオブジェクトが読出されてクライアント計算機に転送される。サーバ装置に対する当該ファイルオブジェクトの変更時刻の問合せは不要である。そのため短時間にファイルオブジェクトがキャッシュから取出され、クライアントに返送される。

【0025】HTTPプロトコルのようにステールデータを容認するネットワークファイルプロトコルは、全地球的な広域ネットワークで利用される上で有利な点を有している。すなわち、ゲートウェイ装置のキャッシュに当該ファイルオブジェクトがキャッシュされていれば、サーバ装置に対するアクセスが発生しない。その結果応答時間短縮を図ることができるという有利な点を持っている。

【0026】上述のようにネットワーク上のファイルオブジェクトのアクセスを高速化し、かつインターネットのトラフィックを低減させるゲートウェイ計算機は特に「proxyサーバ装置」とも呼ばれている。

【0027】「proxy」とは「代理人」という意味である。proxyサーバ装置は、その名のとおり、クライアント計算機がネットワーク上のサーバ計算機のファイルオブジェクトにアクセスする場合に、そのアクセス要求をサーバ計算機を代理して受付け、ファイルオブジェクトの転送を中継する機能を果たす。proxyサーバ装置の概念を図16を参照して説明する。

【0028】proxyサーバ装置13は、たとえばある企業内に用いられた内部ネットワーク23と、その企業外の外部ネットワーク（インターネットなど）25との間に介在して設けられるものである。内部ネットワーク23は複数のクライアント計算機24を含み、外部ネットワーク25は、同じく複数のサーバ計算機11を含んでいる。各サーバ計算機11は、ファイルオブジェクト12を保有している。

【0029】proxyサーバ装置13は、キャッシュファイル16と、クライアント計算機からリード要求19を受け、キャッシュファイル16内に当該ファイルオブジェクトが存在する場合にはそのファイルオブジェクトをデータ20として返送し、キャッシュファイル16内に当該ファイルオブジェクトの有効なものが存在しない場合には外部ネットワーク25を介してサーバ計算機11に対するリード要求17を発行し、対応するファイルオブジェクトをデータ18として受けてクライアント計算機24に対して返送するproxyプロセス14と、proxyプロセス14によるファイルオブジェクトの転送記録を記録するアクセスログ15を含む。proxyプロセス14は、サーバ計算機11からファイルオブジェクトを取得したときには、当該ファイルオブジェクトをキャッシュファイル16にも新たに書込む機

能を有している。

【0030】このproxyサーバ装置13は次のように動作する。まず、内部ネットワーク23内のクライアント計算機24が、外部ネットワーク25上のサーバ計算機11のファイルオブジェクト12に対するリード要求19をproxyサーバ装置13内のproxyプロセス14に対して発行する。proxyプロセス14はこのリード要求19に応答して、キャッシュファイル16をアクセスし、キャッシュファイル中に当該ファイルオブジェクトのキャッシュデータがあるか否かを判定する。当該ファイルオブジェクトがキャッシュされていれば、proxyサーバ装置13固有のキャッシュファイル有効期限と、キャッシュされたファイルオブジェクトの最終変更時刻とを比較する。キャッシュが有効期限内であれば、キャッシュファイルからファイルオブジェクト読出（22）、そのデータ20を内部ネットワーク23のリード要求19を発行したクライアント計算機24に対して返送する。

【0031】キャッシュファイル16内に該当ファイルオブジェクトが存在しない場合、および存在していたとしても有効期限が過ぎている場合には、キャッシュファイル16内に有効なファイルオブジェクトが存在しないものと判定される。その場合、proxyプロセス14は、外部ネットワーク25のサーバ計算機11に対して、ファイルオブジェクト12の読出要求17を発行する。これに応答してサーバ計算機11は、ファイルオブジェクト12をデータ18としてproxyプロセス14に返送する。proxyプロセス14は、ログファイル15にアクセスログとして、ファイルオブジェクト名称（すなわちネットワークプロトコル名称と、サーバ計算機のネットワークアドレス名称と、ファイル名称との組）と、ファイルオブジェクトのデータの実体（すなわちファイルオブジェクトそのもの）と、書込時刻とを書込む（26）。

【0032】proxyプロセス14はさらに、内部ネットワーク23内のクライアント計算機24にデータ20を転送し、かつキャッシュファイル16に、当該ファイルオブジェクトを書込む（21）。

【0033】したがってキャッシュファイル16内には、ファイルオブジェクト名称と、ファイルオブジェクトの実体と、ファイルオブジェクトをサーバ装置から取得した時刻（最終変更時刻）とが記録されている。

【0034】proxyサーバ装置は、ヨーロッパ原子核物理研究所CERNで開発されたhttpdや、日本の電子技術総合研究所の佐藤豊氏の開発されたDc1c Gateなどのソフトウェアを、ネットワーク接続されたUNIXの動くコンピュータ上で実行することにより実現されることが一般的である。

【0035】proxyサーバ装置の物理構成を図17に示す。proxyサーバ装置は、UNIXの動作する

ワークステーション200により構成される。ワークステーション200は、CPU（中央処理装置）202と、CPU202に対して内部バス204により接続されるメモリ206と、ファイル用のI/O（入出力）装置208と、ルータ210を介して外部ネットワークに接続される第1ネットワークI/Oインタフェース212と、内部ネットワークに接続される第2ネットワークI/Oインタフェース214を含む。I/O装置208には、キャッシュファイルの蓄積、アクセスログ、および各種ワークファイルの記憶場所として使用されるファイル部216が接続されている。

【0036】このワークステーション200は、CPU202が、上述したhttpdやDeleGateというソフトウェアを実行することによりproxyサーバ装置13を実現する。

【0037】図18は、proxyサーバ装置の他の構成例を示す図である。このproxyサーバ装置はワークステーション300からなっている。ワークステーション300は、概略、図17のワークステーション200と同様の構成であるが、図17の第2ネットワークI/Oインタフェース214に代えて、モデム装置304に接続されるシリアルポート302を有している点が異なっている。そしてこのワークステーション300は、モデム装置304および公衆電話回線網306を介して内部ネットワークに接続されている。

【0038】図18において、図17と同一の部品には同一の参照番号を付している。それらの名称および機能も同一である。したがって、ここではそれらについての詳しい説明は繰返さない。

【0039】図19にproxyサーバ装置のさらに他の構成例を示す。図19のproxyサーバ装置も、同じワークステーション400により実現される。このワークステーション400は、概略、図17に示されるワークステーション200と同様の構成であるが、第1ネットワークI/Oインタフェース212および第2ネットワークI/Oインタフェース214に代えて、内部ネットワーク23に接続された1つのネットワークI/Oインタフェース402を有している点で異なっている。内部ネットワーク23は、ルータ210を介して外部ネットワーク25に接続されており、また複数個のクライアント計算機24にも接続されている。ワークステーション400は、内部ネットワーク23およびルータ210を介して外部ネットワーク25のサーバ計算機11と通信するとともに、内部ネットワーク23およびクライアント計算機24を介してユーザ31と通信するこ

$$Vave(kbps) = 64kbps \times (1 - Hitrate) + 10000kbps \times Hitrate$$

【0048】キャッシュファイルが存在しない場合、Hitrateは0である。したがって平均速度(kbps)は、数1から外部ネットワークのアクセス速度64kbpsと等しくなる。また、Hitrateが1であ

とができる。

【0040】図19と図17とにおいて、同一の部品には同一の参照符号および名称を与えている。それらの機能も同一である。したがって、ここではそれらについての詳しい説明は繰返さない。

【0041】proxyサーバ装置は、既に述べたように中継データのキャッシュ機構を有する。すなわち、proxyサーバ装置はその内部に存在するキャッシュファイル装置に、中継するファイルオブジェクトのデータをキャッシュする。

【0042】proxyサーバ装置のキャッシュファイルには、proxyサーバ装置固有の有効期限が定められている。ファイルオブジェクトがキャッシュに書込まれてから有効期限内に、同じサーバ計算機上のファイルオブジェクトへのアクセス要求をクライアント計算機から受取った場合には、キャッシュファイルに以前に書込まれたファイルオブジェクトを取出してクライアント計算機に転送する。これによりサーバ計算機に対するアクセスは発生しない。

【0043】このようなゲートウェイ装置によるキャッシュの効果を以下具体的に説明する。proxyサーバ装置はある会社組織に設置し、外部のネットワークとは64kbpsの速度で接続するものとする。また、このproxyサーバ装置は一方で、10Mbpsの速度の会社組織内のローカルエリアネットワークに接続されるものとする。外部と内部のネットワークの速度差は1桁以上ある。このような例はよく見られる例であり、その典型的な例が図19に示す構成である。

【0044】このとき、proxyサーバ装置で中継ファイルオブジェクトをキャッシュするものとして、キャッシュのヒット率をHitrateとする。Hitrateは、ファイルオブジェクトに対するアクセス要求に対してキャッシュが100%ヒットするとき1であり、0%ヒットするとき0になる指数である。

【0045】なお、proxyサーバ装置におけるキャッシュファイルからのデータ読出およびキャッシュファイルへのデータ書込速度は、ネットワーク速度に比べて十分高速であるものとする。すなわち、キャッシュファイルに対するファイルオブジェクトの入出力に要する時間は無視できるものとする。

【0046】このとき、外部ネットワークにあるファイルオブジェクトを内部からアクセスするときの平均アクセス速度Vaveは次の数1によって表わされる。

【0047】

【数1】

れば、すなわち100%キャッシュファイルにヒットするのであれば、平均速度(kbps)は、内部ネットワークの速度と等しく10,000kbps(=10Mbps)となる。つまり、ヒット率が高いほどクライアン

ト計算機から見た平均アクセス速度は高くなる。ヒット率と平均アクセス速度との関係を示す数1をグラフ化したものを図20に示す。

【0049】ところで、キャッシュヒット率は、キャッシュファイルの量に依存している。すなわち、ファイルオブジェクトをなるべくたくさんキャッシュファイルの中に蓄積しておけば、再びアクセスされたファイルオブジェクトやキャッシュファイル中に含まれる確率は高く、ヒット率は高くなる。そうした関係を示すものとして、キャッシュヒット率と、キャッシュファイルの蓄積量との関係を図21にグラフとして示す。図21に示すように、キャッシュファイルサイズを小さくすればキャッシュヒット率は0に近く、キャッシュファイルサイズを大きくすればキャッシュヒット率は徐々に上昇する。

【0050】proxyプロセスにおいては、キャッシュファイルの内容を維持する方式として、最初のファイルアクセスによる書込が起きてから一定期間経過したファイルオブジェクトを無効としていく制御方式が取られている。この一定期間を有効期限と呼んでいる。この有効期限は、proxyサーバ装置を管理する者が実情に合わせて適切な値を設定する。有効期限を長く取れば、キャッシュファイルに蓄積されるデータ量は増加する。その結果、キャッシュのヒット率は高くなる。すなわち、キャッシュのヒット率を上げるには、キャッシュの有効期限を長く取ればよい。

【0051】なお、キャッシュファイルを格納するファイル装置の記憶容量は有限であるから、有効期限を無制限に長くするわけにはいかない。有効期限が過ぎたキャッシュファイルは、何らかのタイミングで消去される。このようにキャッシュファイルを消去するまでの時間を、この明細書では消去時間と呼ぶことにする。

【0052】本願発明者の実験によれば、キャッシュ有効期限とキャッシュファイルに蓄積されるファイル量とほぼ比例する。この関係を図22に示す。これは、内部ネットワークから外部ネットワークへのアクセス量が、日によらずほぼ一定であるためである。したがって次のような関係が成り立つ。

【0053】

【数2】

$$\begin{aligned} \text{キャッシュ有効期限} &\propto \text{キャッシュファイル蓄積量} \\ &\propto \text{キャッシュヒット率} \end{aligned}$$

【0054】この関係から、図22に示したように、キャッシュ有効期限を増やすとキャッシュのヒット率が高まることがわかる。

【0055】キャッシュ有効期限とファイルオブジェクトの量との関係は、クライアント計算機を利用する内部ユーザが、外部ネットワーク上のサーバ計算機のファイルオブジェクトをアクセスする頻度によって異なる。キャッシュ有効期限とキャッシュファイル蓄積量とは、ほ

ぼ比例するものの、その比例係数はネットワーク速度、内部クライアント計算機ユーザ数などの使用環境により変動する。

【0056】なお、キャッシュファイル1日程度の有効期限では、キャッシュヒット率は12%程度である。有効期限を14日にするとキャッシュヒット率は41%程度になることが経験されている。有効期限14日のときにはファイルオブジェクトが700MB程度キャッシュファイルに蓄積されていることが観察されている。

【0057】また、近年ローカルネットワークの規模が大きくなり、接続されるクライアント計算機の数が増大するにつれ、またはクライアント計算機の読出要求が増大するにつれて、単一のproxyサーバ計算機では、各クライアント計算機からの要求に十分に 대응することが難しくなっている。そこで、複数のproxyサーバ計算機を組合せて、ファイルオブジェクトに対するアクセス処理の効率の向上を図る方式が提案されており実施されている。

【0058】図23は、複数のproxyサーバ計算機を用いたシステムの構成例である。proxyサーバ計算機501は、広域ネットワークとローカルネットワーク間の中継を行ない、ファイルオブジェクトをキャッシュする。このproxyサーバ計算機501は、以後2次キャッシュproxyサーバ計算機と呼ぶ。これに対して、proxyサーバ計算機503～508は、1次キャッシュproxyサーバ計算機と呼ばれ、さらに分割されたローカルネットワークに接続されるクライアント計算機24と2次キャッシュproxyサーバ計算機501間の中継およびファイルオブジェクトのキャッシュを行なう。

【0059】たとえば、クライアント計算機24から1次キャッシュproxyサーバ計算機504に対してファイルオブジェクトに対する読出要求を出力したとする。このとき、1次キャッシュproxyサーバ計算機504のキャッシュファイル510内に有効なファイルオブジェクトが存在する場合、1次キャッシュproxyサーバ計算機504は、キャッシュファイル510よりファイルオブジェクトを取出し、読出要求を出したクライアント計算機24に送信する。

【0060】また、1次キャッシュproxyサーバ計算機504に有効なファイルオブジェクトが存在しなかった場合、クライアント計算機24からの読出要求は2次キャッシュproxyサーバ計算機501に中継される。2次キャッシュproxyサーバ計算機501は、同様にキャッシュファイル502内の有効なファイルオブジェクトの有無をチェックし、当該ファイルオブジェクトが存在すれば1次キャッシュproxyサーバ計算機504に送出し、存在しなければ読出要求を広域ネットワーク61を介してサーバ計算機11に中継する。

【0061】以降の処理は、単体のproxyサーバ計

算機と同様であるので、ここではその詳しい説明は繰返さない。

【0062】この方式によって、各々の1次キャッシュproxyサーバ計算機503～508は、クライアント計算機を分担して受け持つことができ、負荷の分散が図れる。また、2次キャッシュproxyサーバ計算機501も、各1次キャッシュproxyサーバ計算機503～508でキャッシュされていなかったファイルオブジェクトに関してのみ処理を行なえばよいので、負荷

の軽減が図れる。具体的には、全体でN台のクライアント計算機24が存在する場合、従来方式ではproxyサーバ計算機は、N台のクライアント計算機の要求を処理する必要があったが、この方式では数3で算出される台数分のクライアント計算機からの要求を処理すればよいことになる。

【0063】

【数3】

$$(1 - (\text{一次キャッシュproxyサーバ計算機のキャッシュヒット率})) \times N \text{ 台}$$

【0064】しかしながら、この方式では1台の2次キャッシュproxyサーバ計算機501が広域ネットワーク61に接続されるサーバ計算機11との中継をすべて行なっており、本質的な負荷の分散が達成されているとはいえない。

【0065】また、キャッシュファイルの使用にも無駄が多い。すなわち、1次キャッシュproxyサーバ計算機群503～508の有するファイルオブジェクトの内容はすべて2次キャッシュproxyサーバ計算機501も保持していると考えられる。これは、中継経路上にある2次キャッシュproxyサーバ計算機501と1次キャッシュproxyサーバ計算機503～508では必ずファイルオブジェクトがキャッシュされるからである。また、各1次キャッシュproxyサーバ計算機同士でも、別の1次キャッシュproxyサーバ計算機に接続されるクライアント計算機が同一のファイルオブジェクトを讀出している場合、同一のファイルオブジェクトを重複して持っている可能性が高い。

【0066】したがって、分散ファイルシステム全体でのキャッシュ容量は2次キャッシュproxyサーバ計算機501のキャッシュファイル502の容量であるといえるので、1台の2次キャッシュproxyサーバ計算機だけではキャッシュファイルの容量には限界があるといえる。

【0067】このような問題を解決するために、図24のような方式が提案され実施されている。

【0068】この方式では、図23の2次キャッシュproxyサーバ計算機501に相当するproxyサーバ計算機は存在しない。各proxyサーバ計算機601～606は、クライアント計算機群を分担して受け持っている。たとえば、proxyサーバ計算機602に接続されたクライアント計算機24からの読出要求は、proxyサーバ計算機602に受信される。ここで、proxyサーバ計算機602のキャッシュファイル608に有効なファイルオブジェクトが存在する場合、当該ファイルオブジェクトを取出し、読出要求を出したクライアント計算機24に送出する。

【0069】proxyサーバ計算機602のキャッシュファイル608に有効なファイルオブジェクトが存在

しなかった場合、proxyサーバ計算機602はローカルネットワーク600に接続される他のproxyサーバ計算機601、603～606に対し、それぞれに接続されるキャッシュファイル607、609～612内に有効なファイルオブジェクトを保持しているかどうかを問合せ。有効なファイルオブジェクトを保持しているproxyサーバ計算機が存在する場合には、たとえば、proxyサーバ計算機605がキャッシュファイル611に有効なファイルオブジェクトを保持している場合、proxyサーバ計算機605はproxyサーバ計算機602に対してファイルオブジェクトを送信し、proxyサーバ計算機602は当該ファイルオブジェクトを読出要求を出したクライアント計算機24に送出する。

【0070】ローカルネットワーク600に接続されたproxyサーバ計算機601、603～606のキャッシュファイル607、609～612に有効なファイルオブジェクトが存在しなかった場合、proxyサーバ計算機602は広域ネットワーク61を経由して、サーバ計算機11にクライアント計算機24からの読出要求を中継する。

【0071】以降の処理は、単体のproxyサーバ計算機と同様であるので、ここでの詳細な説明は繰返さない。

【0072】この方式により、同一のファイルオブジェクトを複数のproxyサーバ計算機が保持することがなくなり、キャッシュファイルの有効利用が図れる。また、分散ファイルシステム全体でのキャッシュ容量は、全proxyサーバ計算機601～606のキャッシュファイル607～612の容量の合計となり、キャッシュの大容量化が図れる。したがって、高いキャッシュヒット率が期待でき、実行転送速度の向上が期待できる。

【0073】また、図25のような方式も提案され実施されている。図23の方式との違いは、2次キャッシュproxyサーバ計算機705～707が複数台用意され、1次キャッシュproxyサーバ計算機711～716が2次キャッシュproxyサーバ計算機705～707にクライアント計算機群からの読出要求を中継する場合、その読出要求の要求先であるサーバ計算機70

1, 702のアドレス名称によって、たとえば、名称がxxx.eduならばeduドメインサーバ計算機702に対する読出要求と判断して、proxyサーバ計算機705にアクセス要求を出力する。また、名称がyy.y.comならばcomドメインサーバ計算機702に対する読出要求と判断してproxyサーバ計算機706にアクセス要求を出力する。このように、末尾のドメイン名称(edu, com等)による簡単なルールでアクセス要求を中継する2次キャッシュproxyサーバ計算機705~707を選択している。

【0074】その他の処理は単一のproxyサーバ計算機と同一であるので、ここでの詳細な説明は繰返さない。

【0075】これにより、同一の2次キャッシュproxyサーバ計算機に読出要求が集中することがなくなり、負荷の分散が図れる。この方式においても、同一のファイルオブジェクトを複数の2次キャッシュproxyサーバ計算機が保持する必要はなく、キャッシュファイルの有効利用が図れる。また、分散ファイルシステム全体でのキャッシュ容量は全2次キャッシュproxyサーバ計算機のキャッシュファイルの容量の合計となり、キャッシュの大容量化が図れる。したがって、高いキャッシュヒット率が期待でき、実行転送速度の向上が期待できる。

【0076】

【発明が解決しようとする課題】上述したproxyサーバ装置においては、キャッシュ有効期限をある程度の長さで設定することによりキャッシュファイルに蓄積されるファイルオブジェクトの容量が増え、キャッシュヒット率が高まり効率的なファイルオブジェクトの転送が可能になる。

【0077】しかし、一方、サーバ計算機のファイルオブジェクトの内容は、修正変更されることがあるため、キャッシュファイルの有効期限が長いと、その期間内はサーバ計算機内のファイルオブジェクトの内容が変更されているにもかかわらず、内部ネットワークのクライアント計算機のユーザはキャッシュファイルに蓄積された古い内容のファイルオブジェクトを読出すことになる。

【0078】たとえば、天気図や電子新聞等のファイルオブジェクトは、同一のファイルオブジェクト名称でありながら数時間おきに情報更新されるものが多い。したがって、たとえば、キャッシュ有効期限を24時間とすると、3時間おきに更新される天気図、電子新聞などの情報は7/8以上の確率で古くて役に立たない情報となってしまう。

【0079】図26は、このような従来のproxyサーバ装置におけるキャッシュ制御のデータの流れをタイムチャートにしたものである。A, B, C, D, E等の名称は同一ファイルオブジェクト名称でありながら内容が異なるデータであることを示している。クライアント

計算機がproxyサーバ装置経由でアクセス(ユーザアクセス(1))すると、サーバ計算機のファイルオブジェクトの内容Bがproxyサーバ計算機によってキャッシュされ、クライアント計算機にはBが転送される。次に、クライアント計算機からユーザアクセス

(2)が送信されると、proxyサーバ装置にキャッシュされている内容Bは24時間有効なので、キャッシュヒットしてファイルオブジェクトの内容Bがクライアント計算機に転送される。しかしこのときには、既にサーバ計算機内では内容がEとなっているのに、クライアント計算機はBという内容の古いデータを見ていることになる。

【0080】このようなキャッシュファイルの内容のうち、外部ネットワークに接続されたサーバ計算機内に存在する元のデータが変更されている割合をステールデータ率(stale data rate)と呼ぶ。したがって、キャッシュ有効期限を増やすとキャッシュファイルに蓄積されるファイルオブジェクトの容量が増加してキャッシュヒット率が高くなるが、キャッシュファイル内の内容のステールデータ率が増加することになる。

【0081】上述したCERNのhttpdソフトウェアで構成されるproxyサーバ装置であれば、特定のサーバ計算機のファイルオブジェクトのキャッシュ有効期限を小さくできるが、この方法ではクライアント計算機がファイルオブジェクトにアクセスを行なうと、キャッシュ有効期限が短いためキャッシュにミスヒットしてからサーバ計算機にファイルオブジェクトに対するアクセスを行なうことになる。したがって、クライアント計算機のユーザは、proxyサーバ計算機によるキャッシュの恩恵を受けられず、広域ネットワークによる低速なファイルオブジェクト転送が終了するまで待たされることになる。

【0082】したがって上述した従来の技術では、キャッシュファイルの中のファイルオブジェクトの内容の新鮮さとキャッシュヒット率の維持はそれぞれ相反する問題となる。

【0083】一般に、キャッシュファイルのステールデータ率が増加すると、クライアント計算機のユーザは内容が古いはずであると自主的に判断してファイルオブジェクトをサーバ計算機から直接ロードするようになる。この場合キャッシュファイルを通り越して、実際にサーバ計算機からロードすることを指定するプロトコルが利用される。その結果、proxyサーバ計算機のキャッシュファイルが意味を持たなくなることになる。

【0084】また、従来よりproxyサーバ計算機の負荷を分散する方法として、上述したように図23~図25の方式が提案されている。しかしいずれの場合にも十分に負荷が分散されているとはいえない。図23に示す方式では、1台の2次キャッシュproxyサーバ計算機501が広域ネットワークに接続されているサーバ

計算機との中継をすべて行なっているため、負荷の分散が達成されているとはいえない。

【0085】図24の方式では、proxyサーバ計算機601～606と広域ネットワーク61との間の経路が多重化されていない。また、ローカルネットワーク600に接続されるproxyサーバ計算機601～606は、この方式に対応した特殊なプロトコルを理解するものに置き換える必要がある。

【0086】図25の方式では、1次キャッシュproxyサーバ計算機711～716は、ドメイン名等の単純なルールで分散を行なうので、2次キャッシュproxyサーバ計算機705～707間での負荷の偏りが生じる。

【0087】本発明は、上記問題点を解決するためになされたもので、請求項1に記載の発明の目的は、キャッシュファイルの内容を新鮮に保つとともに、キャッシュヒット率も向上させることである。

【0088】請求項2に記載の発明の目的は、キャッシュファイルの内容のうちサーバ計算機における更新頻度の高いファイルオブジェクトを新鮮に保つとともに、キャッシュヒット率も向上させることである。

【0089】請求項3に記載の発明の目的は、通常のユーザの利用に悪影響を与えることなく、キャッシュファイルの内容を新鮮に保つとともに、キャッシュヒット率も向上させることである。

【0090】請求項4から6に記載の発明の目的は、キャッシュファイルの内容を新鮮に保つとともに、キャッシュヒット率も向上させることである。

【0091】請求項7または8に記載の発明の目的は、ファイルオブジェクトに対するアクセス要求を分散して経路を有効に利用し、スループットを向上させることである。

【0092】請求項9から11に記載の発明の目的は、ファイルオブジェクトに対するアクセス要求を分散して経路を有効に利用し、スループットを向上させ、さらにローカルネットワーク上の通信量を削減することである。

【0093】請求項12に記載の発明の目的は、ファイルオブジェクトに対するアクセス要求を分散して経路を有効に利用し、スループットを向上させることである。

【0094】

【課題を解決するための手段】請求項1に記載の発明は、クライアント計算機が接続される第1のネットワークと、サーバ計算機が接続される第2のネットワークとの間に介在するように設けられるゲートウェイ装置であって、第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に応答して、当該ファイルオブジェクトにより指定される第2のネットワークのサーバ計算機に対して、当該ファイルオブジェクトに対するアクセス要求を行なうネットワーク

中継手段を含み、ネットワークファイル中継手段は、取得したファイルオブジェクトを所定の有効期限が経過するまで一時的に蓄積するとともに、最終変更時刻をファイルオブジェクト単位で記録するためのキャッシュファイル手段と、過去の一定期間内に行なわれたファイルオブジェクトの転送記録を蓄積するファイル転送記録手段と、ファイルオブジェクトに対するアクセス要求に応答して当該ファイルオブジェクトの最終変更時刻を参照し、有効期限内のファイルオブジェクトがキャッシュファイル手段に存在する場合は当該ファイルオブジェクトをクライアント計算機に転送し、有効期限内のファイルオブジェクトがキャッシュファイル手段に存在しない場合には当該ファイルオブジェクトにより指定されるサーバ計算機の当該ファイルオブジェクトを取得してクライアント計算機に転送するための転送手段とを含み、ゲートウェイ装置はさらに、ファイル転送記録手段から所定の規則に従って転送記録を抽出し、当該転送記録のファイルオブジェクトを第2のネットワークのサーバ計算機から取得してキャッシュファイル手段に格納するためのキャッシュ更新手段を含む。

【0095】ネットワークファイル中継手段によって転送されたファイルオブジェクトは、所定の規則に従ってキャッシュ更新手段により更新される。キャッシュ更新手段は、有効期限が切れたものはもちろん、有効期限内のファイルオブジェクトも第2のネットワークのサーバ計算機から取得してキャッシュファイル手段に格納する。そのため、キャッシュの有効期限を長めに取っていても、キャッシュの有効期限よりも短い間隔で変更されるサーバ計算機のファイルオブジェクトがキャッシュに反映される確率は高くなる。

【0096】請求項2に記載のゲートウェイ装置は、請求項1に記載のゲートウェイ装置であって、キャッシュ更新手段はさらに特定の文字列パターンを予め記憶するための文字列パターン記憶手段を含み、キャッシュ更新手段は特定の文字列パターンと一致するファイルオブジェクト名称を含む転送記録をファイル転送記録手段から抽出し、当該ファイルオブジェクトを第2のネットワークのサーバ計算機から取得してキャッシュファイル手段に格納する。

【0097】キャッシュ更新手段は、特定の文字列パターンと一致するファイルオブジェクトのみ更新アクセスを行なうので、予めファイルオブジェクトの更新頻度の高いファイルオブジェクト名称を登録しておけば不要なキャッシュ更新アクセスを削減でき、キャッシュファイルの内容を新鮮に保つとともにキャッシュヒット率も向上させることができる。

【0098】請求項3に記載のゲートウェイは、請求項1に記載のゲートウェイ装置であって、ネットワークファイル中継手段は、各々が第1のネットワークのクライアント計算機からのファイルオブジェクトのアクセス要

求を処理するための第1の転送プロセスを複数個、同時並列的に起動させ、キャッシュ更新手段は各々が1つのファイルオブジェクトの取得を行なうための第2の転送プロセスを複数個、同時並列的に起動させ、かつ第1の転送プロセスの数を検知して、第1の転送プロセスの数と第2の転送プロセスの数との和が予め定められる上限以下となるように第2の転送プロセスの新たな起動を制御する。

【0099】第1のプロセスが多数発生しているときは第2のプロセスの起動を抑制し、第1のプロセスが僅かしか発生していないときには第2のプロセスを多数起動させることができるので、通常のユーザによるファイルオブジェクトのアクセス要求が、キャッシュ更新手段による悪影響を受けることがない。また、通常のユーザによるアクセス要求が少ないときには随時更新処理が行なえるので、キャッシュの新鮮さを保つことができる。

【0100】請求項4に記載のゲートウェイ装置は、請求項1に記載のゲートウェイ装置であって、キャッシュ更新手段はファイル転送記録手段の中から、キャッシュファイル手段からキャッシュヒットしたファイルオブジェクトの転送記録を抽出し、当該ファイルオブジェクトを第2のネットワークのサーバ計算機から取得してキャッシュファイル手段に格納する。

【0101】ユーザによるアクセス要求がキャッシュにヒットした場合、キャッシュ更新手段はその直後にサーバ計算機へファイルオブジェクトのアクセス要求を行なう。したがって別ユーザが同一ファイルオブジェクトをアクセスした場合、最新のファイルオブジェクトが既にキャッシュに格納されていることになる。

【0102】請求項5に記載のゲートウェイ装置は、請求項1に記載のゲートウェイ装置であって、キャッシュ更新手段はさらにファイル転送記録手段の末尾から転送記録を順次読出して格納し、先に格納された転送記録から順に出力するFIFOバッファ手段を含み、キャッシュ更新手段はFIFOバッファ手段から転送記録を読出して、当該転送記録のファイルオブジェクトを第2のネットワークのサーバ計算機から取得してキャッシュファイル手段に格納する。

【0103】FIFOバッファ手段は、ファイル転送記録手段の末尾から転送記録を順次格納し出力するので、キャッシュ更新手段の処理が容易になる。

【0104】請求項6に記載のゲートウェイ装置は、請求項1に記載のゲートウェイ装置であって、キャッシュ更新手段はファイル転送記録手段から抽出した転送記録のファイルオブジェクトの前の更新が所定時間以内であれば、第2のネットワークのサーバ計算機へのアクセスは行なわない。

【0105】キャッシュ更新手段は、所定時間以内に再び同じファイルオブジェクトに対するアクセス要求があった場合に後のアクセス要求に対するキャッシュ更新処

理を行なわない。このことによって、不要なキャッシュ更新アクセスを削除し、無駄なトラフィックを抑制して回線を有効に使うことができる。

【0106】請求項7に記載の発明は、クライアント計算機と第1のネットワークによって接続され、かつサーバ計算機と第2のネットワークによって接続される複数のキャッシュサーバ計算機と第3のネットワークによって接続されるゲートウェイ装置であって、第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行なうための演算手段と、当該演算結果に基づいて第3のネットワークの複数のキャッシュサーバ計算機の中からアクセス要求を出力するキャッシュサーバ計算機を選択するための中継制御手段とを含む。

【0107】中継制御手段は、複数のキャッシュサーバ計算機の中からファイルオブジェクトに対するアクセス要求を出力するキャッシュサーバ計算機を選択する場合に、演算結果が一意的に定まる演算を行なうことによってキャッシュサーバ計算機を一意的に定める。したがって同一のファイルオブジェクトに対するアクセス要求は同一の経路によって転送され、異なるファイルオブジェクトに対するアクセス要求は分散して経路を転送することになるので、経路が有効に利用され、スループットが向上することになる。

【0108】請求項8に記載の発明は、クライアント計算機とゲートウェイ装置とが第1のネットワークで接続され、サーバ計算機とキャッシュサーバ計算機とが第2のネットワークで接続され、複数のゲートウェイ装置と複数のキャッシュサーバ計算機とが第3のネットワークで接続される分散ファイルシステムであって、ゲートウェイ装置は、第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行なうための演算手段と、当該演算結果に基づいて第3のネットワークの複数のキャッシュサーバ計算機の中からアクセス要求を出力するキャッシュサーバ計算機を選択するための中継制御手段とを含む。

【0109】各ゲートウェイ装置は、アクセス要求に含まれる所定のデータに同一の演算を行なうため複数のキャッシュサーバ計算機の中からキャッシュサーバ計算機を選択する際、同一のファイルオブジェクトであれば同一のキャッシュサーバ計算機を選択することになる。したがって同一のファイルオブジェクトが複数のキャッシュサーバ計算機内に含まれるということがなくなり、キャッシュの有効利用が可能となる。

【0110】請求項9に記載の発明は、クライアント計算機が接続される第1のネットワークと、サーバ計算機が接続される第2のネットワークとの間に介在するよう

に設けられるゲートウェイ装置であって、第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行ない、この演算結果に基づいてゲートウェイ装置を選択し、この選択が他のゲートウェイ装置であれば、アクセス要求を第3のネットワークを介して他のゲートウェイ装置へ出力するための中継制御手段と、第2のネットワークのサーバ計算機から取得したファイルオブジェクトを所定の有効期限が経過するまで一時的に蓄積するとともに、最終変更時刻をファイルオブジェクト単位で記録するためのキャッシュファイル手段と、過去の一定時間内に行われたファイルオブジェクトの転送記録を蓄積するためのファイル転送記録手段と、中継制御手段が自身のゲートウェイ装置を選択した場合と第3のネットワークを介して他のゲートウェイ装置からアクセス要求を受信した場合に、ファイルオブジェクトに対するアクセス要求に応答して、当該ファイルオブジェクトの最終変更時刻を参照し、有効期限内のファイルオブジェクトがキャッシュファイル手段に存在する場合は当該ファイルオブジェクトをクライアント計算機に転送し、有効期限内のファイルオブジェクトがキャッシュファイル手段に存在しない場合には当該ファイルオブジェクトにより所定されるサーバ計算機の当該ファイルオブジェクトを取得してクライアント計算機に転送するための転送手段とを含む。

【0111】このゲートウェイ装置は、複数のキャッシュサーバ計算機の中からキャッシュサーバ計算機を選択する機能と、ファイルオブジェクトをキャッシュする機能とを有することによって、ローカルネットワーク上での通信量を削減することを可能にしている。

【0112】請求項10に記載の発明は、サーバ計算機と複数のキャッシュサーバ計算機とが第2のネットワークで接続され、複数のキャッシュサーバ計算機と第1のネットワークによって接続されるクライアント計算機であって、キャッシュサーバ計算機に対するファイルオブジェクトのアクセス要求を出力し、当該ファイルオブジェクトを取得するためのファイル取得手段と、キャッシュサーバ計算機へのファイルオブジェクトに対するアクセス要求を出力する際、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行ない、当該演算結果に基づいて複数のキャッシュサーバ計算機の中からアクセス要求を出力するキャッシュサーバ計算機を選択するための中継制御手段とを含む。

【0113】クライアント計算機に、複数のキャッシュサーバ計算機の中からキャッシュサーバ計算機を選択する機能を含ませることによって、ローカルネットワーク上での通信量を削減し、クライアント計算機の有効利用を図っている。

【0114】請求項11に記載のクライアント計算機

は、請求項10に記載のクライアント計算機であって、中継制御手段はさらに、第3のネットワークを介して他のクライアント計算機からのファイルオブジェクトに対するアクセス要求を受信するための受信手段を含み、中継制御手段は他のクライアント計算機からのアクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行ない、当該演算結果に基づいて複数のキャッシュサーバ計算機の中からアクセス要求を出力するキャッシュサーバ計算機を選択する。

【0115】このクライアント計算機は、他のクライアント計算機からのアクセス要求を受信して、当該アクセス要求に含まれる所定のデータに基づいて複数のキャッシュサーバ計算機の中からキャッシュサーバ計算機を選択している。したがって同じローカルネットワークに接続されるクライアント計算機の中の一部にキャッシュサーバ計算機を選択する機能を持たせて、クライアント計算機の有効利用を図っている。

【0116】請求項12に記載の発明は、クライアント計算機と第1のネットワークによって接続され、かつサーバ計算機と第2のネットワークによって接続される複数のキャッシュサーバ計算機と第3のネットワークによって接続されるゲートウェイ装置であって、第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行ない、当該演算結果に基づいてゲートウェイ装置を選択し、この選択が他のゲートウェイ装置であれば、当該アクセス要求を第3のネットワークを介して他のゲートウェイ装置へ出力するための中継制御手段と、第2のネットワークのサーバ計算機から取得したファイルオブジェクトを、所定の有効期限が経過するまで一時的に蓄積するとともに、最終変更時刻をファイルオブジェクト単位で記録するためのキャッシュファイル手段と、過去の一定時間内に行われたファイルオブジェクトの転送記録を蓄積するためのファイル転送記録手段と、中継制御手段が自身のゲートウェイ装置を選択した場合と第3のネットワークを介して他のゲートウェイ装置からアクセス要求を受信した場合に、ファイルオブジェクトに対するアクセス要求に応答して当該ファイルオブジェクトの最終変更時刻を参照し、有効期限内のファイルオブジェクトがキャッシュファイル手段に存在する場合には当該ファイルオブジェクトをクライアント計算機に転送し、有効期限内のファイルオブジェクトがキャッシュファイル手段に存在しない場合には当該ファイルオブジェクトにより指定されるサーバ計算機の当該ファイルオブジェクトを取得してクライアント計算機に転送するための転送手段とを含む。

【0117】ゲートウェイ装置に、キャッシュサーバ計算機を選択する機能と、ファイルオブジェクトをキャッシュする機能とを持たせることによって、ローカルネッ

トワーク上の通信量を削減し、キャッシュサーバ計算機の負荷を軽減することを可能にしている。

【0118】

【発明の実施の形態】

〔実施の形態1〕本発明が適用されるゲートウェイ装置（proxyサーバ装置と呼ぶ）は、ネットワークインタフェースを備えたUNIX OSの動作する計算機によって、以下の説明および図面を参照すれば容易に実現可能である。そうしたproxyサーバ装置の構成の一例は、図19に示したとおりである。図19のproxyサーバ装置については既に説明したので、その構成についての詳細な説明はここでは繰返さない。

【0119】通常は、ユーザ31がクライアント計算機24を使用してproxyサーバ装置（図19におけるproxyサーバ装置400）に対して、サーバ計算機11のファイルオブジェクト10に取得するようネットワーク23を経由して依頼を出す。ファイルオブジェクト12は、前述したようにプロトコル名称と、サーバ計算機のネットワークアドレスと、ファイルオブジェクトの名称と、ファイルデータの実体との組からなっている。

【0120】図1に本願発明に係るproxyサーバ装置の第1の実施の形態における装置53のシステム構成を示す。proxyサーバ装置53は、内部ネットワーク23および外部ネットワーク25に接続された第1のproxyプロセス14と、第1のproxyプロセス14が使用するキャッシュファイル16と、第1のproxyプロセス14が、アクセスログ（転送記録）としてファイルの名称を記録するアクセスログ15と、アクセスログ15の末尾から順次取出して格納し、先に格納されたアクセスログから順に出力するFIFOバッファ35と、予め作成されたアクセスパターンリストを格納するパターンファイル30と、FIFOバッファ35からアクセスログを取出してパターンファイル30に格納されたアクセスパターンと比較し一致する場合にはこのアクセスログをメモリバッファURLBUF（URLについては後述する）に格納するキャッシュ更新プロセス36と、URLBUFに格納されたアクセスログに含まれるファイルオブジェクト名称をもとに、当該ファイルオブジェクトをサーバ計算機11から取得してキャッシュファイル16に格納する第2のproxyプロセス34とを含む。

【0121】第1のproxyプロセス14と、第2のproxyプロセス34と、キャッシュ更新プロセス36とはいずれも、たとえば図19に示すメモリ206内にプログラムとして格納され、CPU202によってこれを実行することにより実現される。

【0122】第1のproxyプロセス14が、特許請求の範囲に記載の転送手段に相当し、キャッシュファイル16がキャッシュファイル手段に相当し、アクセスロ

グ15がファイル転送記録手段に相当する。また、キャッシュ更新プロセス36と、パターンファイル30と、FIFOバッファ35と、第2のproxyプロセス34とによりキャッシュ更新手段が構成される。

【0123】第2のproxyプロセス34は、本願発明に係るゲートウェイ装置に特有なものであるが、次のような条件に従って起動される。

【0124】（1）第2のproxyプロセス34は、第1のproxyプロセス14と共通のキャッシュファイル16を使用する。

【0125】（2）第2のproxyプロセス34のキャッシュ有効期限は0に設定される。すなわち、第2のproxyプロセスを経由するサーバ計算機11へのアクセスは、必ず外部ネットワーク25内のサーバ計算機11に対して行なわれ、ファイルオブジェクトを取得してキャッシュファイル16に格納する。

【0126】図1に示すproxyサーバ装置53の動作について以下説明する。内部ネットワーク23内のクライアント計算機24が、このproxyサーバ装置53に対してネットワークアクセス要求19を与えるものとする。第1のproxyプロセス14が、このネットワークアクセス要求19を受け取り、まずI/Oインタフェース202（図17、図18、図19参照）を経由してファイル装置216内に存在するキャッシュファイル16をアクセスする（21）。また第1のproxyプロセス14は、このネットワークアクセス要求とキャッシュファイル16内のファイルオブジェクト名称とを比較し（22）、一致しているか否かを判定する。一致したものがあれば第1のproxyプロセス14はさらに、当該ファイルオブジェクトのキャッシュファイル16内の最終変更時刻を抽出し（22）、現在の時刻と比較する。そして現在の時刻が、当該ファイルオブジェクトの最終変更時刻を起点とする有効期限内であれば、キャッシュファイル16から当該ファイルオブジェクトを抽出し（22）、ネットワークI/Oインタフェース402を経由してクライアント計算機24にファイルオブジェクトを返す（20）。

【0127】該当するファイルオブジェクトがキャッシュファイル16内に存在していない場合、または存在していてもそのキャッシュ有効期限が切れている場合には、第1のproxyプロセス14は、ネットワークI/Oインタフェース402を経由して外部ネットワーク25のサーバ計算機11にファイルオブジェクト転送要求17を送る。このときのサーバ計算機11は、当該ファイルオブジェクト内のネットワークアドレス名称により特定される。

【0128】該当するサーバ計算機11は、要求されたファイルオブジェクトを第1のproxyプロセス14に返送する（18）。すなわちこのファイルオブジェクトが第1のproxyプロセス14により取得される。

【0129】第1のproxyプロセス14は、このファイルオブジェクトをキャッシュファイル16にその最終変更時刻とともに書込み(21)、アクセスログ15にこのファイルオブジェクトの名称を記録する(26)。

【0130】一方、本願発明のproxyサーバ装置53は、キャッシュ更新プロセス36と第2のproxyプロセス34とをCPU202により処理走行させてファイルオブジェクトを更新する点において従来のproxyサーバ装置と相違する。

【0131】キャッシュ更新プロセス36および第2のproxyプロセス34は次のように動作する。第1のproxyプロセス14によってアクセスログファイル15に格納されたアクセスログの末尾からFIFOバッファ35に順に格納される(28)。FIFOバッファ35は、先に格納されたアクセスログから順に出力する(27)。キャッシュ更新プロセス36は、FIFOバッファ35からアクセスログを1行ずつ抽出し、メモリバッファに格納する。キャッシュ更新プロセス36は、メモリバッファからアクセスログを抽出し、それぞれのアクセスログをもとにキャッシュにヒットしているか否かを判定する。キャッシュにヒットしていればさらに、パターンファイル30からアクセスパターンリストを抽出し(29)、アクセスログに含まれるファイルオブジェクト名称がアクセスパターンリストと一致しているものを抽出する。またキャッシュ更新プロセス36は、抽出したファイルオブジェクトのうちファイルオブジェクト名称が特定のパターンに一致するものは除外して、これをメモリバッファURLBUFに格納する。

【0132】第2のproxyプロセス34は、URLBUFが空でないならば、URLBUFに格納されたファイルオブジェクト名称をもとに(31)、サーバ計算機11にアクセスし(32)、当該ファイルオブジェクトを取得して(33)キャッシュファイル16に格納する(37)。これによりキャッシュファイル16内のファイルオブジェクトが最新の内容に維持される。

【0133】このように、第2のproxyプロセス34を起動すると、大きく次の2つの効果を得ることができる。

【0134】(1) 第1のproxyプロセス14では、キャッシュファイルの有効期限がある値に設定されており、ファイルオブジェクトのキャッシュ更新処理をこの第1のproxyプロセス14を用いて行なうとすると、キャッシュファイル16にヒットしてしまう可能性が高い。これでは、外部ネットワーク25のサーバ計算機11のファイルオブジェクトに対するアクセスが行なわれないので、キャッシュファイル16の更新が実現できない。

【0135】一方、第2のproxyプロセス36は、キャッシュ有効期限が0になっており、第2のprox

yプロセス36に対してアクセス要求を出せば、このキャッシュファイルはミスヒットしたことになるので、外部ネットワーク25を介してサーバ計算機11に対するファイルオブジェクトのアクセスが行なわれ(32)、当該ファイルオブジェクトは第2のproxyプロセス34に転送される(33)ことによって、キャッシュファイル16中のファイルオブジェクトは最新の状態に更新される。

【0136】(2) 第1のproxyプロセス14は、アクセスログを残すが、このアクセスログをもとにキャッシュ更新アクセスをするべきファイルオブジェクトが抽出される。したがって、もしキャッシュ更新プロセス36が第1のproxyプロセス14を使用してサーバ計算機11のファイルオブジェクトのアクセスを行なうと、その結果として生成されるアクセスログがアクセスログファイル15に記録されてしまう。キャッシュ更新プロセス36は、次の更新アクセス動作においてアクセスログファイル15に格納されたアクセスログをもとにアクセスすべきファイルオブジェクト名称を抽出する。したがって、再度同じファイルオブジェクトをサーバ計算機11から取得することになり、同一のファイルオブジェクトが多重に出現するループ動作が起きてしまうことになる。これを避けるために第2のproxyプロセス34を動作させ、第2のproxyプロセス34を利用したキャッシュ更新アクセス動作を行なう。

【0137】本発明を非常に有効に適用できる例として、インターネット上のWWWシステムがある。WWWシステム上でのproxyサーバ装置において、本願発明のキャッシュ更新プロセスを行なう手順を以下に説明する。

【0138】WWWシステムでは、ネットワーク上に分散したファイルオブジェクト名称はUniform Resource Locator(URL)と呼ばれる形式で表現され、特定される。URLの一例を次に示す。

【0139】

【数4】

<http://www.xxx.co.jp/test/index.html>

【0140】数4においては「http」は使用するプロトコルを示す。「www.xxx.co.jp」は、ネットワーク上のHTTPサーバ計算機のアドレスを示すものであり、ネットワーク上で同一のものはないよう選ばれている。また「/test/index.html」はサーバ計算機内のファイル名称を示す。

【0141】第1のproxyプロセス14として一般に利用されているDeleGateを使用する場合は、そのアクセスログの例は次のように、クライアント計算機名称、時刻、“HTTPプロトコル(ファイル取得要求)”、その他、およびキャッシュヒット状況(キャッシュヒットした場合H)などとなる。以下に、そのアクセスログの例を示す。

【0142】

【数5】

クライアント計算機名称 [時刻] "GET http://naragw.sharp.co.jp/HTTP/1.0"その他H
 クライアント計算機名称 [時刻] "GET http://naragw.sharp.co.jp/sharpcolor.gif HTTP/1.0"その他H
 クライアント計算機名称 [時刻] "GET http://naragw.sharp.co.jp/isg.gif HTTP/1.0"その他H
 クライアント計算機名称 [時刻] "GET http://naragw.sharp.co.jp/Zaurus.gif HTTP/1.0"その他H
 クライアント計算機名称 [時刻] "GET http://naragw.sharp.co.jp/Shoin.gif HTTP/1.0"その他H
 クライアント計算機名称 [時刻] "GET http://naragw.sharp.co.jp/Prostation.gif HTTP/1.0"その他H
 クライアント計算機名称 [時刻] "GET http://naragw.sharp.co.jp/S2.gif HTTP/1.0"その他H

【0143】数5に示すアクセスログは、図2に示すような、出願人がインターネット上で公開している情報ページへアクセスした場合のアクセスログに相当する。図2に示されるページは、テキストと、6個のグラフィックデータとから構成されているので、このページに対するアクセスを1回行なうと、6個のグラフィックデータに対するアクセスを含む7つのアクセスログが形成される。すなわち、上述のアクセスログにおいて第1行目はこの情報ページのテキストに対するアクセス要求であり、残りの6行は表紙に貼り込まれた出願人会社のロゴのグラフィックデータ (sharpcolor. gi

f)、出願人会社内の情報システム事業本部と呼ばれる事業部のロゴ (isg. gif)、出願人会社の4つの製品の写真のグラフィックデータ (Zaurus. gif, Shoin. gif, Prostation. gif, S2. gif) の取得要求を示している。

【0144】HTTPプロトコルは、コマンド文字列 (GET等) と、URLと、プロトコルバージョン (「HTTP/1.0」等) から構成されている。以下はその一例である。

【0145】

【数6】

"GET http://www.xxx.co.jp/test/index.html HTTP/1.0"

【0146】HTTPプロトコルについては、最後に掲げる表内に示された参考文献のうち、参考文献3および参考文献4に記載されている。キャッシュ更新プロセスの実施手順について以下に説明するが、そのための前提条件は次の(1)および(2)となっている。

【0147】(1) 第1のproxyプロセス14は、キャッシュ有効期限M (時間) で起動する。たとえばM=24時間とする。

【0148】(2) 第2のproxyプロセス34は、キャッシュファイルを第1のproxyプロセス14のものと共有し、キャッシュ有効期限を0に設定する、という2つの条件に従って起動される。

【0149】このような条件の下に、本願発明のキャッシュ更新プロセスを毎日一定時刻に起動し、一定時刻に終了する。毎日一定時刻にキャッシュ更新プロセスを自動起動するには、UNIX OSに備わっているタイマ機能 (cron) を使用する。これは、UNIX OSに関連してよく知られている機能であるので、その詳細についてはここでは説明しない。なお、UNIX OS以外のOSでも、同様の機能が提供されていることが多い。

【0150】図3は、本願発明のキャッシュ更新プロセスの処理手順を示すフローチャートである。このプロセ

スを実現するためのプログラムは、たとえば図19のメモリ206内に配置され、CPU202によって実行される。このプロセスとネットワークおよびファイル装置216との入出力には、それぞれネットワークI/Oインタフェース402とI/Oインタフェース208とが使用される。

【0151】まず、ステップS1において第1のproxyプロセス14が、アクセスログファイル15に格納されるアクセスログの末尾を1行読出す。読出せない場合は、アクセスログが読めるまで待機する。UNIXでは、このようなアクセスログの末尾を呼出す機能は、OSとともに提供されているコマンドtailで可能であり、その標準出力から1行読出し、メモリ内に確保された文字列バッファであるLINEバッファに1行格納するには、以下のように行なう。ここで、第1のproxyプロセス14が格納したアクセスログが/usr/local/etc/telegated/logs/10001.httpであるとすると、これは図1のアクセスログファイル15に格納されるアクセスログに相当する。

【0152】

【数7】

tail -f /usr/local/etc/delegated/logs/10001.http | read LINE

【0153】数7のtail -fによる1行読出しは、図1の27に相当するが、これはソフトウェアによるFIFOバッファ35を構成しており、読出ポインタおよ

び書込ポインタをそれぞれ有している。

【0154】

【数8】


```

green [時刻] "GET http://naragw.sharp.co.jp/HTTP/1.0"その他H
green [時刻] "GET http://naragw.sharp.co.jp/sharpcolor.gifHTTP/1.0"その他H
rp--> green [時刻] "GET http://naragw.sharp.co.jp/isg.gifHTTP/1.0"その他H
green [時刻] "GET http://naragw.sharp.co.jp/Zaurus.gifHTTP/1.0"その他H
green [時刻] "GET http://naragw.sharp.co.jp/Shoin.gifHTTP/1.0"その他H
green [時刻] "GET http://naragw.sharp.co.jp/Prostation.gifHTTP/1.0"その他H
green [時刻] "GET http://naragw.sharp.co.jp/ZAURUS/index.htmlHTTP/1.0"その他H
wp--> green [時刻] "GET http://naragw.sharp.co.jp/S2.gifHTTP/1.0"その他H

```

【0155】すなわち、第1のproxyプロセス14が書込むアクセスログは、1個のファイルオブジェクト転送ごとに1行発生するが、このとき書込ポインタ（wp）を1行進める。また、ソフトウェアによるFIFOバッファ35がtail-1でアクセスログを1行読出せば、読出ポインタ（rp）を1行進める。第1のproxyプロセス14による書込速度（行/秒）と、キャッシュ更新プロセス36におけるtail-1による1行読出速度（行/秒）が一致していなくてもよい。第1のproxyプロセス14によるアクセスログ書込速度が大きくなると、書込ポインタ（wp）は先に進むが、キャッシュ更新プロセス34によってこれを読出ポインタ（rp）が追いつけながらアクセスログが読出される。アクセスログ書込速度が小さいか、読出側のキャッシュ更新プロセス34の処理が進行すれば、やがて読出ポインタ（rp）が書込ポインタ（wp）に追いつくことになる。これが、ソフトウェアFIFOバッファとしての動作である。

【0156】次に、LINEバッファに格納された1行のアクセスログ情報は過去にアクセスされたファイルオブジェクトの記録であるので、以下のような条件でキャッシュ更新アクセスを行なうべきファイルオブジェクト名称を含むものを選択する。この選択処理を、以後フィルタリングと呼び、図3に示すフローチャートのステップS2に相当する。このフィルタリングのルールは以下のとおりである。

【0157】（1） HTTPコマンド文字列がファイルオブジェクトに対するリード要求（GET）であるもの。

【0158】（2） ファイルオブジェクト名称のプロトコル部分がhttpであるもの。

（3） キャッシュにヒットしたアクセスであるもの。

【0159】（4） ファイルオブジェクト名称がキャッシュ更新アクセスに適さないものを除外する。

【0160】（5） 特定の文字列パターンを含むアクセスリストとファイルオブジェクト名称が一致するもの。

【0161】以上の（1）～（5）の条件を用いてフィルタリングを行ない、ファイルオブジェクト名称のフィールドだけを取り出して、キャッシュ更新アクセスでファ

イルオブジェクト名称を抽出する。

【0162】ファイルオブジェクト名称のプロトコル部分がhttpのもののみ抽出するのは、第1のproxyプロセス14および第2のproxyプロセス34として一般的に利用されているDeleGateソフトウェアがキャッシュファイルを生成するプロトコルはhttpプロトコルのみであるので、そのファイルオブジェクトのみはキャッシュ更新アクセスする意味があるからである。それ以外のプロトコルによるアクセスはファイルオブジェクトがキャッシュされないので、キャッシュ更新アクセスをする意味がない。

【0163】キャッシュにヒットしたもののみを抽出するのは、キャッシュにヒットしていないファイルオブジェクトはサーバ計算機11へのアクセスが行なわれたものと解釈されるので、その直後にキャッシュ更新アクセスを行なう必要はないと判断されるからである。

【0164】ファイルオブジェクト名称が、キャッシュ更新アクセスに適さないものとしては、URL中に「？」を含むものなどがある。「？」を含むURL表記は、サーバ計算機11に対してクライアント計算機24を使用するユーザが文字列を手入力する場合などに使われるため、この文字を含むURL表記を有するファイルオブジェクトをキャッシュ更新アクセスの対象とすると、予期せぬ結果を招くおそれがある。

【0165】特定の文字列パターンを含むアクセスリストと一致するもののみを選ぶのは、サーバ計算機において頻繁に内容更新が行なわれるものである天気図、電子新聞等を提供しているサーバ計算機など特定のものとすることが多い。そこで、そのようなサーバ計算機のファイルオブジェクトを指定するような文字列パターンを格納したアクセスリストと一致したものだけをキャッシュ更新アクセスの対象とすることによって、不要なキャッシュ更新アクセスを極力削減するためのものである。

【0166】このような文字列パターンを格納したパターンファイル30を、ここでpattern.txtとし、その内容を数9に示すものとする。

【0167】

【数9】

weather
news
http://www.asahi.com
http://www.sjmercury.com
http://www.cnn.com
http://naragw.sharp.co.jp/ZAURUS/
daily

【0168】UNIXでは、このようなファイルオブジェクト名称の抽出作業は、OSとともに提供されているコマンド群である数10を用いて行なうことができる。

【0169】
【数10】

/usr/bin/fgrep によるストリングパターンの抽出(パターンファイル指定可能)
/usr/bin/grep によるストリングパターンの抽出
/usr/bin/grep -v によるストリングパターンの否定抽出
/usr/bin/awk による特定のフィールドの切りだし

【0170】たとえば、次のようなコマンドを入力することにより、ステップS2の処理を経たものがメモリ上の文字列バッファURLBUFに出力される。

【0171】
【数11】

```
URLBUF='echo $LINE|grep -v "\?"|grep "\GET"|grep 'H$'|  
awk '{print $7}'|fgrep -f pattern.txt'
```

【0172】なお、上述したフィルタリングルールに反していれば、文字列バッファURLBUFは空になる。この間の経過を以下に説明する。第1のproxyプロセス14により作成されたアクセスログファイル15の

形式は既に述べたように次のとおりである。

【0173】
【数12】

クライアント計算機名称——[21/Aug/1995:09:27:43+0900] "GET http :
//naragw.sharp.co.jp/S2.gifHTTP/1.0" その他 キャッシュヒット状況

【0174】キャッシュヒット時には“H”という文字がキャッシュヒット状況として行の末尾に出力される。

【0175】まず、grep -v "\?" に入力することで、?を含まない行を抽出し標準出力に出力する。それをさらにgrep "\GET" でその行がファイルオブジェクト取得要求であるGETコマンドである行のみ抽出し、標準出力に出力する。さらにgrep 'H\$' に入力することで行末が「H」で終了する、すなわちキャッシュにヒットしたものを選び標準出力に出力する。ここで、\$は行の末尾を表現する正規表現である。

【0176】さらに空白がフィールド間の区切であると考え、7番目のフィールドがファイルオブジェクト名称に相当するので、awk '{print \$7}' を使ってファイルオブジェクト名称のみを抽出している。

【0177】さらに、ファイルオブジェクト名称部分が、予め用意されたパターンファイルpattern.txtの中のいずれかの行に一致するか否かをfgrep -f pattern.txtによりマッチングを行なってその結果を文字列バッファURLBUFに出力する。

【0178】ステップS3では、文字列バッファURLBUFが空でないか否かを判定する。もし文字列バッファ

URLBUFが空の場合には、ステップS1に戻り、空でない場合にはステップS4へ進む。

【0179】ステップS4では、第2のproxyプロセス34によって、キャッシュ更新プロセス36で抽出されたファイルオブジェクトをサーバ計算機11から取得する。文字列バッファURLBUFに格納されたキャッシュ更新アクセスの対象であるファイルオブジェクト名称に基づいて、第2のproxyプロセス34はサーバ計算機11へアクセスするための子プロセスを起動する。子プロセスをキャッシュ更新子プロセスと呼ぶ。キャッシュ更新子プロセスは、第2のproxyプロセス34を通じてネットワーク接続をサーバ計算機11に対して行なう。これにより、サーバ計算機11に格納されるファイルオブジェクトへのアクセスが発生し、キャッシュファイル16の内容が更新される。この処理の詳細は後述する。

【0180】ステップS5では、現在の時刻を調べ、指定時刻が過ぎていればプロセスを終了する。指定時刻が過ぎていなければステップS1に戻り、以上の処理を繰返す。

【0181】以上が、本願発明の第1の実施の形態におけるproxyサーバ装置におけるアルゴリズムであ

る。図7は、proxyサーバ計算機53のキャッシュ有効期限が24時間で、サーバ計算機11のファイルオブジェクトが同一名称でありながら、内容が3時間おきに更新される場合のファイルオブジェクトのデータがキャッシュされるようすを示している。

【0182】まず、ユーザアクセス(1)が発生したときに、キャッシュファイル16に該当するファイルオブジェクトがなかったとすると、サーバ計算機11からデータBがアクセスされ、キャッシュファイル16に書込まれるとともに、クライアント計算機24に転送される。

【0183】次に、ユーザアクセス(2)が発生すると、キャッシュにヒットしたデータBがクライアント計算機24に転送される。このとき、サーバ計算機11はデータDになっているので、クライアント計算機24は3時間程度古いステールデータを読んだことになる。しかし、この動作の結果アクセスログファイル15にファイルオブジェクト名称とキャッシュにヒットしたという情報を含むアクセスログが記録されるので、図3を用いて説明した動作に従い、proxyサーバ装置53が自発的にサーバ装置11にアクセスし、データEを読み出しキャッシュファイル16を更新する。

【0184】さらに、ユーザアクセス(3)が起きると、キャッシュされたデータEがクライアント計算機24に転送される。この動作をきっかけにしてアクセスログファイル15にファイルオブジェクト名称とキャッシュにヒットしたという情報を含むアクセスログが記録されるので、図3を用いて説明した動作に従い、proxyサーバ装置53が自発的にサーバ計算機11にアクセ

スし、データIを読み出しキャッシュファイル16を更新する。

【0185】したがって、ユーザアクセス(4)が起きると、キャッシュされたデータIがクライアント計算機24に転送される。

【0186】以上の動作からわかるように、クライアント計算機24はステールデータを読んでいることには変わりがないが、図26の場合に比べると最新に近い情報を読んでいることになる。ユーザによるアクセスの間隔が3時間以内であれば、ステールデータは発生する確率が非常に低くなり、サーバ計算機11の最新の情報を得ることが可能になることが理解できる。

【0187】なお、図3のステップS4のネットワークアクセス部の実現方法としては、キャッシュ更新プロセスから直接HTTPプロトコルを発生させればよいが、UNIXでは広く一般に利用されているlynx等のWWWクライアントプログラムを利用すれば、HTTPを発生するキャッシュ更新プロセスを新たに製作しなくてもHTTPに従ったネットワークアクセスを行なうキャッシュ更新プロセス操作を実現できる。その方法を以下に説明する。

【0188】lynxはUNIX OSの上で動作するプログラムであるので、以下の説明ではUNIXの記述を使用する。

【0189】lynxでは次のコマンドを指定すれば、指定URLを読み出し、temp_fileというファイルに書込むことができる。

【0190】

【数13】

```
lynx -source http://www.xxx.co.jp/test/index.html > temp_file
```

【0191】さらに、proxyプロセス経由のアクセスも、UNIX OSの環境変数http_proxyを設定することにより、lynxから利用可能である。また、proxyプロセスがDeleGateのようなproxyソフトウェアであれば、URL表記にproxyを指定することによりproxy経由のアクセスが可能である。これについては参考文献12を参照されたい。

【0192】後者の方式を採用する場合について以下に説明する。proxyプロセスが、ネットワークアドレ

```
lynx -source http://proxyserver:10001/_ _ http://www.xxx.co.jp/
test/index.html > temp_file
```

【0194】したがって、本願発明を実施する際には、同様にしてproxyプロセスを第2のproxyプロセス34のポート番号に設定すればよい。

```
lynx -source http://proxyserver:10001/_ _ $URLBUF > temp_file
```

【0196】本願発明では、このようなファイルオブジ

スがproxyserverであるゲートウェイ計算機のTCP/IPの10001番のポート番号を利用して、このproxyプロセス経由でサーバ計算機11のファイルオブジェクトhttp://www.xxx.co.jp/test/index.htmlにアクセスするためには次の数14のコマンドを指定すればよい。

【0193】

【数14】

【0195】

【数15】

エクト(URL)のキャッシュ更新アクセスにより、第

1のproxyプロセス14の管理するキャッシュファイル16を最新状態に保つことが目的である。したがって、temp_file自体は利用しないので捨ててよい。OSがUNIXであれば、空ファイルとして/dev/nullが準備されているので、このファイルに書込むようにすれば実際にはファイルの生成は行なわないので効率的である。

【0197】次に、第1のproxyプロセス14および第2のproxyプロセス34のキャッシュ管理手法について説明する。proxyプロセスのキャッシュ管理手法は、参考文献5、参考文献12にあるような公知の技術であるキャッシュ制御機能を持ったDeleGateなどにおけるものである。DeleGateにおけるキャッシュ管理手法は、ソースコードの形で公開されているので以下にその手法を述べる。なお、本願発明は

```
lynx -source http://proxyserver:10001/_-$_URLBUF > temp_file
```

【0200】なお、第2のproxyプロセス34とキャッシュ更新プロセス36の通信はTCP/IP、たとえばUNIXのメッセージ通信で行なわれる。

【0201】たとえば、文字列バッファURLBUFにhttp://www.sharp.co.jp/sample/test.htmlという文字列が格納されているとする。キャッシュ更新プロセス36が数16のコマンドを実行すると、キャッシュ更新プロセス36はURL取得要求を第2のproxyプロセス34に出力する。第2のproxyプロセス34は、子プロセスを生成し、この子プロセスにURL取得の処理を任せて自身のプロセスは再びキャッシュ更新プロセス36からのURL出力要求を待つ。キャッシュ更新プロセス34による数16の実行によりURL取得を実行する第2のproxyプロセス34が生成した子プロセスは、図4のフローチャートに示す処理を行なう。

【0202】以下、第2のproxyプロセス34が生成した子プロセスの動作を図4のフローチャートを用い

/cache/プロトコル名称/サーバ計算機名称/ファイルオブジェクトの実体部分

【0207】したがって、http://www.sharp.co.jp/sample/test.htmlは数19に示すファイルとなる(S11)。

/cache/http/www.sharp.co.jp/sample/test.html

【0209】proxyプロセスはまず、数19に示すファイル名称に変換し、そのファイルが存在するかどうかOSのopen()のシステムコールにより既存のファイルがオープンできるかどうかにより調べる(S12)。ファイルが存在しなければ、ファイル新規作成モードで数19に示すファイル名のキャッシュファイルを作成する(S13)。また、キャッシュファイルに対して排他制御を行なって、他のプロセスが書込めないよう

これと同様のキャッシュ管理手法が使われているキャッシュ機能付proxyサーバソフトウェアであれば、第1のproxyプロセス14および第2のproxyプロセス34として適用可能である。

【0198】上述したように、第2のproxyプロセス34がproxyserverという名称のシステムで動作しており、TCP/IPの10001番でURL取得要求を受付けるものとする。このとき、文字列バッファURLBUFに指定URL表記が格納されている場合に、キャッシュ更新プロセス36は、数16に記載のコマンドを実行することでtemp_fileに指定URLのファイルオブジェクトが格納されると述べた。

【0199】

【数16】

て説明する。

【0203】proxyプロセスでは、キャッシュファイル16を持っているが、それは通常のUNIXファイルシステムの一部として作られる。キャッシュ用ディレクトリとして/cacheというパーティションを使用するとすると、数17に示すようにプロトコル名称がhttp、サーバ計算機名称www.sharp.co.jp、ファイルオブジェクトの実体部分がsample/test.htmlとなる。

【0204】

【数17】

http://www.sharp.co.jp/sample/test.html

【0205】このとき、キャッシュファイルは次の数18に示す名称規則でキャッシュファイルを生成しようとする。

【0206】

【数18】

/cache/プロトコル名称/サーバ計算機名称/ファイルオブジェクトの実体部分

【0208】

【数19】

に排他ロックする(S14)。

【0210】次に、ネットワークを経由してURLで指定されたファイルオブジェクトをサーバ計算機11から取得する(S15)。取得したファイルオブジェクトをクライアント計算機24へ転送するとともにファイルオブジェクトを数19に示すファイル名称のキャッシュファイルに書込む(S16)。続いて、キャッシュファイルをクローズして排他ロックを解除する(S17)。

【0211】上記処理では、キャッシュファイルを排他ロックしているが、この排他制御はOSがUNIXであればシステムコールである`flock()`関数を使って排他ロックできる。排他制御を使用する理由は、`proxy`プロセスは1つのファイルオブジェクトの転送ごとに子プロセスを起動してその子プロセスにキャッシュ制御を行なわせるため、同じキャッシュファイルシステムに対して子プロセス間で競合が発生する場合があるからである。

【0212】なお、キャッシュファイルに書込んだとき、ファイル変更時刻`Tm`がファイル(数19)の属性として記録される。ファイル更新時刻`Tm`は、UNIX OSのファイルシステムを管理するOSがファイルの属性として必ず付加するものである。すなわち、キャッシュファイルは通常のOSのファイルシステムをそのまま使用しており、何ら特殊なファイルシステムではない。

【0213】次に、キャッシュファイルが既に存在した場合を説明する(S19～S26)。

【0214】キャッシュファイル/`cache/http/www.sharp.co.jp/sample/test.html`の最終変更時刻`Tm`をOSから読出す。最終変更時刻は、UNIXであれば、`fstat()`システムコールで得られるもので、過去から未来に向かって単調に増加する関数である(S19)。

【0215】現在時刻をシステムコールから読出し`Tnow`という変数に格納する(S20)。現在時刻`Tnow`は、OSのシステムコールから得られる。最終変更時刻`Tm`と、現在時刻`Tnow`を比較し、キャッシュ有効期限内かどうかを判定する(S21)。キャッシュ有効期限内であれば、キャッシュファイルを共有排他ロックして他のプロセスからの書込はできないが、読出はできるようにする(S22)。そして、既にあるキャッシュファイルを読出し、クライアント計算機24に転送する(S23)。さらに、キャッシュファイルをクローズし、共有排他ロックを解除する(S17)。

【0216】また、キャッシュ有効期限を過ぎている場合は、キャッシュファイルに対して排他制御を行なって、他のプロセスが書込めないように排他ロックする(S24)。次に、ネットワーク経由でURLで指定されたファイルオブジェクトをサーバ計算機11から取得する(S25)。取得したファイルオブジェクトをクライアント計算機24へ転送するとともに、ファイルオブジェクトをキャッシュファイル(数19)に書込む(S26)。そして、キャッシュファイルをクローズして排他ロックを解除する(S17)。

【0217】以上が、一般にソースコードが公開され、よく知られている`proxy`プロセスのキャッシュ制御手順であるが、本願発明ではこの性質を利用して、キャッシュファイル16を第1の`proxy`プロセス14と

第2の`proxy`プロセス34で共有している。第2の`proxy`プロセス34は、キャッシュ有効期限0であるから、図3のフローチャートのS4により、第2の`proxy`プロセス34経由のサーバ計算機11へのアクセスがキャッシュファイル16の更新が必ず行なえることがわかる。

【0218】第1の実施形態においては、クライアント計算機24の利用が始まる7時00分より起動し、クライアント計算機24の利用が少なくなる夜21時00分には終了するよう実施した。これにより、クライアント計算機24の利用のない時間帯には、キャッシュ更新プロセス36を排除し、当制御方式を採用した計算機のメモリ資源を無駄にしないようにしている。

【0219】以上説明したように、`proxy`サーバ計算機53はユーザが行なったファイルオブジェクトに対するアクセス要求がキャッシュヒットすれば、その直後にサーバ計算機11から当該ファイルオブジェクトを取得してキャッシュファイル16を更新するので、別のユーザが同一ファイルオブジェクトをアクセスした場合に、最新のファイルオブジェクトが既にキャッシュファイル16に格納されていることになる。したがって、キャッシュ有効期限を長めに取っていても、キャッシュの有効期限よりも短い間隔で変更されるサーバ計算機のファイルオブジェクトがキャッシュに反映される確率が高くなる。

【0220】また、同一ユーザが時間を置いてから同一ファイルオブジェクトをアクセスし直すと、キャッシュの内容が最新のものに更新されていることになる。また、特定のパターンに一致するファイルオブジェクトのみ更新アクセスの対象となるので、予めファイルオブジェクトの更新頻度の高いサーバ計算機を登録しておけば不要なキャッシュ更新アクセスを削減でき、無駄なトラフィックを抑制して回線を有効に利用することができる。

【0221】〔実施の形態2〕次に、本願発明のゲートウェイ装置の第2の実施形態を説明する。この実施形態2は、実施形態1と同様であるが、実施形態1に対応するフローチャート(図3)のステップS4を改良したものである。実施形態1では、ステップS4においてアクセスログファイル15からファイルオブジェクト名称を文字列格納バッファ`URLBUF`に格納して、第2の`proxy`プロセス34を介してサーバ計算機11にアクセスしているが、第2の実施形態では処理の高速化を図るため複数のファイルオブジェクトに対するアクセスを、順不同で同時並列的に実行するものである。

【0222】本発明が実施できるTCP/IPのネットワークでは、通信はパケット単位で行なわれるため、外部ネットワークの複数のサーバ計算機との同時通信が見掛け上可能である。したがって、ステップS4におけるキャッシュ更新子プロセスを複数同時に実施することが

できる。

【0223】内部ネットワーク23には複数のクライアント計算機24が接続されるので、proxyサーバ装置53によるファイルオブジェクトの中継転送要求は、外部ネットワーク25上の異なる複数のサーバ計算機11へのアクセスが入り交じって行なわれる。したがって、ファイルオブジェクトに対するアクセスは外部ネットワーク25上のさまざまな経路を通じて行なわれるため、実際のファイルオブジェクトの転送速度は、経路の途中のボトルネックによって決まる。

【0224】そのうちの1つのボトルネックは、proxyサーバ装置53と外部ネットワーク25を結ぶ最初の通信路(図19における通信路127)である。さらには、外部ネットワーク25上への複数のサーバ計算機11への経路にはさまざまなボトルネックが存在する。このため、外部ネットワーク25からproxyサーバ計算機53への転送速度は、この通信路の最大転送速度より小さい転送速度となるのが常である。たとえば、proxyサーバ装置53と外部ネットワーク25を結ぶ通信路(図19における通信路127)の最大転送速度が64kbpsであっても、海外にあるサーバ計算機からの転送速度はその10分の1以下であることがしばしばである。

【0225】したがって、proxyサーバ装置53と外部ネットワーク25を結ぶ最初の通信路127の最大転送容量に近い転送を行なうように、同時に複数のネットワーク接続を実施することで転送のスループットを上げることが可能になる。すなわちこのネットワーク接続を1個の子プロセスに対応させるとして、複数のキャッシュ更新子プロセスを起動し、この通信路127の転送容量が100%近く使用されるように、複数のネットワーク接続を並列に実施した方が処理が高速化される。

【0226】そこで、同時に複数のキャッシュ更新アクセスを実行するように、キャッシュ更新子プロセスの最大並列実行数をMAXPROCESS数と定める。これは、以下に説明するフローチャートではメモリ(図19

のメモリ206)に置かれる定数である。この値をあまり大きく取ると、OSのプロセス手順などの管理資源(メモリ)を消費することになるので、システムの動作状況によって決める必要がある。この実施形態2では10とした。

【0227】以下、具体的な処理手順を図5を参照しながら説明する。まず、変数としてメモリ206中にprocesses変数を定義する。バックグラウンドで、現在走行しているキャッシュ更新子プロセスの数を求めてprocesses変数に代入する。プロセスの数はOSのプロセス管理テーブルから得られる(S421)。次にprocesses変数と指定された最大子プロセス数MAXPROCESSを比較する。もし、processes $\geq$ MAXPROCESSならばS423に移り、一定時間休止(sleep)する(たとえば10秒)。これは、先に走行しているキャッシュ更新子プロセスの処理を終了するのを待つために行なわれる(S423)。

【0228】次に、指定の終了時刻になったかどうかの判定を行なう。指定時刻を過ぎていれば、キャッシュ更新プロセスは処理を終了する。指定時刻を過ぎていなければ、S421以下の処理を繰返す(S424)。

【0229】また、S423でprocesses<MAXPROCESSならば、第2のproxyプロセス34にアクセス要求を出力することによって、第2のproxyプロセス34はキャッシュ更新子プロセスをバックグラウンドプロセスとして1つ起動する(S425)。UNIX OSにおいて、プロセスをバックグラウンドで実行させるには、コマンドラインに&の記号を付けて起動すればよい。文字列バッファURLBUFには、S2により取得されたファイルオブジェクト名称が格納されているので、数20に示すコマンドを実行すれば、キャッシュ更新子プロセスがバックグラウンドプロセスとして実施される。

【0230】

【数20】

```
lynx -source http://proxyserver.10001/_-$_URLBUF > temp_file &
```

【0231】最後に、指定の終了時刻になったかどうかを判定し、指定時刻を過ぎていればキャッシュ更新プロセスは処理を終了し、指定の終了時刻を過ぎていなければS1以下の処理を繰返す(S426)。

【0232】以上のような処理により、MAXPROCESS以下のプロセスで同時並列実行が可能になる。

【0233】この実施形態2により、キャッシュ更新処理速度を速め、常にキャッシュ更新処理を待つ状態にすることでキャッシュ更新処理の遅滞の拡大を防止することが可能となる。なお、パターンファイル30を用意して、特定ファイルオブジェクトだけを更新対象にしているS2の処理も効果を上げている。

【0234】以上説明したように、実施の形態2では、キャッシュ更新処理を同時に行なうことで、キャッシュ更新の開始から終了までが短縮されるので、第1のproxyプロセス14によるユーザからのアクセス処理の頻度に追従してキャッシュ更新処理を進めることが可能となる。

【0235】〔実施の形態3〕実施の形態3においては、実施の形態2と同様に、キャッシュ更新子プロセスの並列実行を行なうが、さらにプロセスの制御を行なうことにより、内部ネットワーク23のクライアント計算機24を使用するユーザが第1のproxyプロセス14を使用して外部ネットワーク25のサーバ計算機11

に頻繁にアクセスしている時間帯に、キャッシュ更新プロセス36を実行しても、キャッシュ更新子プロセスの数が抑制される。すなわち、キャッシュ更新子プロセスには低い優先権が与えられることで、クライアント計算機24を使用するユーザのネットワーク利用を妨げないようにしたものである。

【0236】また、第1のproxyプロセス14を利用するユーザのネットワークアクセスは、通常1つのページが複数のファイルオブジェクトから構成されているため（たとえば、図2に示す情報ページ）、1つのページのアクセスにおいては、複数のファイルオブジェクトのアクセスが時間的に連続して発生する。したがって、キャッシュ更新アクセスの子プロセスの起動において、休止時間を設けて子プロセス起動を遅延させることにより、ユーザのページアクセスが終了してからキャッシュ更新アクセスが起きる確率を高めることができる。

【0237】また、同一のファイルオブジェクトに対するキャッシュ更新アクセスの条件が揃ったとき、すなわち、実施形態1および2のURLBUFが空でないとき（図3および図4のS3がycsの場合）一定時間の無効時間を設け、前回キャッシュが更新されてから所定時間以内であればサーバ計算機11に対するキャッシュ更新アクセスを行なわないことで、クライアント計算機24が頻繁にファイルオブジェクトアクセスを行なっているときに、キャッシュ更新アクセスでサーバ計算機11に対するアクセスが頻繁に発生することを防止するものである。

【0238】この処理の簡単な実現方法として、第2のproxyプロセス34のキャッシュ有効期限を0では

processes <= キャッシュ更新子プロセスの数 + 第1proxyのネットワークアクセス
中継子プロセスの数

【0243】processes変数と指定された最大子プロセスMAXPROCESSを比較し、processes ≥ MAXPROCESSならばS433へ移り、一定時間休止（sleep）する（たとえば10秒）。これは、先に起動されたキャッシュ更新子プロセス数の処理が終了するのを待つものである。

【0244】次に、指定の終了時刻になったか否かを判定し、指定時刻を過ぎていればキャッシュ更新プロセスは処理を終了する。指定の終了時刻になっていなければ、S431に戻って処理を続行する（S434）。また、S432でprocesses < MAXPROCESSならば、抽出したファイルオブジェクト名称（URLBUFに格納されたもの）をもとに、第2のproxyプロセス36を用いてアクセスするキャッシュ更新子

(sleep \$DELAYTIME; lynx -source http:// proxyserver:10001/ - - \$URLBUF
> temp_file)&

【0247】最後に、指定の終了時刻になったかどうか

なく、無効時間を設定して起動すればよい。この無効時間として、30分または1時間等とし、サーバ計算機11の変更頻度の平均の最小時間程度にすればよい。これにより、無効時間以内に実施形態1、2または3の方式でキャッシュ更新アクセスを行なっても、キャッシュにヒットすることになるのでサーバ計算機11にアクセスが発生しなくなる。

【0239】図6は、実施の形態3の処理を示すフローチャートである。図6は、実施形態2の処理を示すフローチャートである図5のS421とS425をS431とS435に置き換えたものである。

【0240】図6に示す処理を行なうに先立って、最大ネットワーク接続個数MAXPROCESSという定数を定める。この定数は、キャッシュ更新子プロセスの数と第1のproxyプロセス14がネットワークアクセスの中継のために起動した子プロセスの数を加算した値の最大を示す定数である。したがって、proxyサーバ装置53は、なるべくのMAXPROCESS以下にサーバ計算機11へのネットワークアクセスを制限しようとする。図6のS1～S3の処理は、実施形態1、2と同様であるのでここでの説明は繰返さない。

【0241】バックグラウンドで走行しているキャッシュ更新子プロセスの数と、第1のproxyプロセス14のネットワークアクセス中継子プロセスの数を求めて加算してprocesses変数（数21）とする（S431）。

【0242】

【数21】

プロセスをバックグラウンドプロセスとして1つ起動する。

【0245】このとき、子プロセスは数22に示すように、休止時間をDELAYTIME変数（単位は秒）としてOSの休止機能をsleepを使ってDELAYTIME秒間休止してからキャッシュ更新アクセスを行なうようにする。1ページWWWデータは、図2に示すように複数の画像ファイルオブジェクトからなる場合、キャッシュヒット時は10秒以下でproxyサーバ装置53からクライアント計算機24に転送可能であることが経験的に知られているので、DELAYTIMEは10秒程度としている。

【0246】

【数22】

を判定し、指定時刻を過ぎていればキャッシュ更新プロ

セスは処理を終了し、指定終了時刻を過ぎていなければ、S1へ戻り処理を繰返す(S436)。

【0248】以上のように、キャッシュ更新プロセスの個数と第1のproxyプロセス14を利用しているユーザのネットワークアクセスの中継子プロセス数の合計を制限しながらキャッシュ更新プロセスを制御する。この結果、第1のproxyプロセス14を利用しているユーザによるネットワークアクセスの数が増えると、キャッシュ更新プロセスは起動されにくくなる。したがって、ユーザが第1のproxyプロセス14を利用している時間帯に、キャッシュ更新プロセス36を実行しても、キャッシュ更新プロセスには低い優先権が与えられることで、ユーザのネットワーク活動を妨げないようにすることができる。

【0249】また、ユーザのファイルオブジェクトに対するアクセスがキャッシュにヒットした場合は、通常1ページの転送は、数秒以内に終了するが、キャッシュ更新プロセスはDELAYTIME(秒)遅延してから起動されるので、DELAYTIMEを10秒程度に設定すればユーザのファイルオブジェクトアクセスとキャッシュ更新アクセスは時間的にずれて実施される(すなわちパイプライン)ので、proxyサーバ計算機53への負担を減らし、ユーザに快適なネットワーク利用を提供することが可能となる。

【0250】以上説明したように、実施の形態3ではユーザが第1のproxyプロセス14を利用している時間帯に、キャッシュ更新プロセス36を実行しても、キャッシュ更新プロセスには低い優先権が与えられることで、ユーザのネットワーク利用を妨げない効果がある。また、遅延時間を導入することで、キャッシュ更新処理はユーザの連続したファイルオブジェクト転送処理が終了してから行なわれるので、proxyサーバ計算機53の処理負担を軽減し、ユーザへのファイルオブジェクト転送を最優先できる。また、無効時間が設定されるため、クライアント計算機24が無効時間以内に同一ファイルオブジェクトに対してアクセスを行ない、キャッシュ更新アクセスの条件が揃ってもキャッシュ更新アクセスはキャッシュにヒットすることになり、実質的にサーバ計算機11に対するアクセスが発生しない。これにより、クライアント計算機24が同一ファイルオブジェクトを頻繁にアクセスしても、時間のかかるサーバ計算機11へのキャッシュ更新アクセスが抑制されるのでサーバ計算機11への負担の軽減とキャッシュ更新処理の効率化が図れる。

【0251】[実施の形態4] 実施の形態1～3は、proxyサーバ計算機に関するものであったが、実施の形態4ではファイルオブジェクトの中継制御のみを行なうゲートウェイ装置(redirector計算機)について説明する。図8は、キャッシュサーバ計算機(proxyサーバ計算機)65～67およびredire

ctor計算機72～77をローカルネットワーク71で接続する構成を示す図である。

【0252】redirector計算機72～77は、proxyサーバ計算機の1種でもあるが、実施の形態4のredirector計算機72～77は、proxyサーバ計算機65～67への中継のみを行ないキャッシュは行なわない。

【0253】以下、redirector計算機72～77の動作について詳細に説明する。

【0254】redirector計算機72～77は、複数のproxyサーバ計算機65～67とクライアント計算機群との間に介在し、proxyサーバ計算機65～67とはローカルネットワーク71で接続され、クライアント計算機群とはローカルネットワーク78で接続される。また、ファイルオブジェクトを提供するサーバ計算機11とproxyサーバ計算機65～67とは、広域ネットワーク61で接続される。

【0255】まず、クライアント計算機24からのファイルオブジェクトに対する読出要求をローカルネットワーク78を介してredirector計算機73内のredirectorプロセスが受信する。redirector計算機は、上述した図17～19に示すワークステーション200～400で実現可能である。すなわち、redirectorプロセスに相当するプログラムをメモリ206内に格納し、そのプログラムをCPU202が実行することによってこのredirector計算機の機能が実現される。redirectorプロセスは、受信した読出要求の内容に基づいてハッシュ関数の計算を行ないその計算結果に従って中継するproxyサーバ計算機65～67を選択する。そしてredirectorプロセスは、読出要求を選択されたproxyサーバ計算機65～67内のproxyプロセスに中継する。

【0256】本発明を非常に有効に適用できる例として、前述したインターネット上のWWWシステムがある。図9は、WWWシステム上でのredirector計算機の処理手順を示すフローチャートである。

【0257】WWWシステムではネットワーク上に分散したファイルオブジェクト名称はUniform Resource Locator(URL)と呼ばれる形式で表現され特定される。

【0258】たとえば、redirector計算機73が、クライアント計算機24からの読出要求を受信したとすると、redirector計算機73は要求に含まれるURLを取出す(S31)。取出されたURLに基づいてハッシュ関数による計算を行なう。ここで、ハッシュ関数にはさまざまなものが考えられるが、実施の形態4ではURL形式で表現されたファイルオブジェクト名称に含まれる文字の文字コードをすべて加算し、proxyサーバ計算機65から67の台数で割った剰余を使用した(S33)。

【0259】このハッシュ関数によって算出された値によって、読出要求を中継するproxyサーバ計算機65～67の番号を決定する(S34)。そして、選択されたproxyサーバ計算機65～67に読出要求が中継される(S35)。

【0260】同一のファイルオブジェクトの読出要求は、すべて同一のproxyサーバ計算機65～67で中継されることが、キャッシュの有効利用の観点からは望ましい。なぜなら、同一のファイルオブジェクトを異なるproxyサーバ計算機で重複して保持すると、キャッシュ容量が減少し、キャッシュファイルの有効利用が図れなくなるからである。そのためには、ハッシュ関数がすべてのredirector計算機72～77内のredirectorプロセスで同一のものである必要がある。

【0261】上述したハッシュ関数は、各proxyサーバ計算機65～67が均等に選択されるため、それぞれのproxyサーバ計算機65～67に対する負荷は均等になるが、適切にハッシュ関数を選ぶことにより、各proxyサーバ計算機65～67の能力に応じて意図的に均等でなく負荷を分散することも容易に実現できる。すなわち、処理能力の高いproxyサーバ計算機に対しては、選択される可能性を高くし、処理能力の低

いproxyサーバ計算機に対しては、選択される可能性を低くすることによって各proxyサーバ計算機の処理速度を均等にするものである。

【0262】このようにして算出された番号により示されるproxyサーバ計算機が、選択されたproxyサーバ計算機となり、redirectorプロセスは選択されたproxyサーバ計算機内のproxyプロセスにクライアント計算機24からの読出要求を中継する。proxyサーバ計算機65～67内のproxyプロセスは、redirector計算機73内のredirectorプロセスにより中継されたクライアント計算機24からの読出要求を広域ネットワーク61を介してサーバ計算機11に中継し、転送されたファイルオブジェクトをキャッシュする。

【0263】redirectorプロセスは、広く一般に利用されているDeleGateを利用して実施することが可能である。実施の形態4のredirectorプロセスは、DeleGateの中継記法とフィルタ機能を利用して実現されている。以下に、中継記法の例を示す。

【0264】

【数23】

`http://proxyserver:10080/_ - http://www.xxx.co.jp/test/index.html`

【0265】数23の記述のうち、最初のhttpは使用するプロトコル名を示し、proxyserverはネットワーク上のproxyサーバ計算機のアドレスを示すものである。また、10080はTCP/IP接続のポート番号と呼ばれ、proxyサーバ計算機上のどのプロセスに対し読出要求を送るのかを示す。――は、これがDeleGateの中継記法であることを示す。残りの`http://www.xxx.co.jp/test/index.html`が、本来の読出要求のURLである。

【0266】DeleGateが上記読出要求を受信すると、proxyサーバ計算機proxyserverのポート番号10080で示されるproxyプロセスに対し、URLで示されるファイルオブジェクトの読出要求を中継する。すなわち、読出要求の中で、中継するproxyサーバ計算機を指定することが可能である。

【0267】フィルタ機能は、外部プログラムを利用してDeleGate機能を拡張するための仕組みである。DeleGateはクライアント計算機もしくは他のproxyサーバ計算機からの読出要求を受信すると、受信した読出要求を一旦外部プログラムに引渡し、この外部プログラムによって処理された読出要求を再び受取って中継等の処理を行なうことが可能である。

`http://proxyserver:10080/_ - http://www.xxx.co.jp/test/index.html`

【0268】図11は、redirector計算機の内部構成を示す図である。実施の形態4において、DeleGateによって実現されるredirectorプロセス81が、たとえば、数24に示すURLを含む読出要求を受信したとする。

【0269】

【数24】

`http://www.xxx.co.jp/test/index.html`

【0270】これは、`www.xxx.co.jp`で表わされるサーバ計算機11にこの読出要求を送信することを示している。

【0271】しかし、実施の形態4ではDeleGateのフィルタ機能により、外部プログラムとして実現されたproxy切替制御部82においてURLをもとにハッシュ関数の計算を行ない(図9のS32～S33)、その演算結果によってproxyサーバ計算機proxyserver1が選択されたとすると、上記読出要求内のURLを数25のように書替える(図9のS34)。

【0272】

【数25】

【0273】外部プログラムであるproxy切換制御部82によって書替えられた読出要求を受取ったredirectorプロセス81は、proxyserver1で表わされるproxyサーバ計算機にwww. xxx. co. jpで表わされるサーバ計算機11に対する読出要求を中継する(図9のS35)。

【0274】したがって、ハッシュ関数で計算された結果に基づいてURL中のproxyサーバ計算機のアドレスをproxyserver1に変えることにより、元の読出要求のURLに基づいて中継するproxyサーバ計算機を選択することができる。実際に、2台のproxyサーバ計算機を用意し、ハッシュ関数はURLに含まれるファイルオブジェクト名称の文字の文字コードをすべて加算し(図8のS32)、proxyサーバ計算機の台数2で割った剰余を使用した(図8のS33)。すなわち、ハッシュ関数の計算結果が0であった場合proxyserver0により中継し、1であった場合はproxyserver1で中継を行なうものとする。

【0275】図10は、図2に示す同一ページのファイルオブジェクトが、2台のproxyサーバ計算機に分散されるようすを示したものである。図10の上段は、本来の読出要求のURL、ファイルオブジェクト名称の文字コードの総計(check sum)、文字コードの総計をproxyサーバ計算機の台数である2で割った余りおよび選択されたproxyサーバ計算機のアドレスを示している。図10の下段は、proxy切換制御部82で書替えられたURLを示している。この例では、ハッシュ関数の結果が0の方が多くなっており、proxyserver0への中継が多くなってしまいが、これはサンプル数が少ないことによるものである。実際の運用では膨大なアクセスが、ハッシュ関数による分散が確率的に均等になるという性質のもとに、ほぼ2分の1ずつ均等に分散されることが確認されている。

【0276】以上説明したように、複数あるproxyサーバ計算機が別々な経路で広域ネットワークに接続されている場合、クライアント計算機からのアクセス要求を複数あるproxyサーバ計算機に分散することは、複数ある経路に分散することでもある。したがって経路の有効利用が可能となり、スループットが向上する。proxyサーバ計算機の処理能力には限界があるが、クライアント計算機からのアクセス要求が複数のproxyサーバ計算機に分散されることにより、proxyサーバ計算機1台で処理されるアクセス要求数が減少し、スループットが向上する。

【0277】また、同一ページ内に複数のファイルオブジェクトが含まれている場合、それらのファイルオブジェクトに対するアクセス要求も複数のproxyサーバ計算機によって分散して処理されるので、単一ページのアクセス速度も向上する。

【0278】また、どのクライアント計算機からの中継要求であっても、同一URLに対するハッシュ関数が同じであるので、常に同じproxyサーバ計算機が選択されることになり、同一URLのアクセスが2回以上あればキャッシュによるアクセスの高速化が図れる。さらに、ハッシュ関数が一意にきまるものであれば任意の関数が選択できるので、proxyサーバ計算機の数もまた任意の台数が使用できる。

【0279】実施の形態4において、ハッシュ関数内で使われているURLに含まれる文字の文字コードをすべて加算した値(check sum)は、平均して数千程度になることが確かめられている。たとえば、シャープ奈良工場における実施においては、1996年3月13日に実測された30481件のファイルオブジェクトに対するアクセス要求のURLに含まれる文字や文字コードの合計の平均は4438(小数点以下四捨五入)であった。

【0280】前述したとおり、実施の形態4ではこのアクセス要求のURLに含まれる文字や文字コードの合計をproxyサーバ計算機の台数で割った剰余をハッシュ関数として使用し、proxyサーバ計算機を選択に使用している。したがって、図10の例ではproxyサーバ計算機は2台であるが、この台数が数千程度であっても、ハッシュ関数によるproxyサーバ計算機を選択はほぼ均等に行なわれると考えられる。

【0281】また、同一のハッシュ関数であれば、redirector計算機の数も任意である。したがって、本方式ではスケーラビリティも非常に優れている。

【0282】また、本方式ではキャッシュ付のproxyサーバ計算機は既存のものをそのまま使用することが可能である。したがって、図24の従来の方式に比べ導入が容易で経済的である。また、クライアント計算機に関しても、既存のクライアント計算機を使用して、redirector計算機を既存のproxyサーバ計算機に見立てて設定を行なうだけでよく、特殊なクライアント計算機を用意する必要はない。

【0283】〔実施の形態5〕実施の形態5は、実施の形態4において独立した計算機上で実行していたredirectorプロセスを、クライアント計算機24上で実行する。図12は、実施の形態5における装置の内部構成を示す図であり、クライアント計算機24内のクライアントプロセス84(図8のクライアント計算機24と同じ動作をするプロセス)からの読出要求は、ローカルネットワーク上には送出されず、クライアント計算機24内部のredirectorプロセス85に直接送信される。この送信は、たとえばUNIXのメッセージ通信によって行なわれる。

【0284】redirectorプロセス85がアクセス要求を受信した後の処理は、上述したredirector計算機72～77内で動作するredirec

torプロセス81と同様であり、ローカルネットワークを介する他のクライアント計算機からのアクセス要求を受信して処理することも可能である。したがって、ここでの詳細な説明は繰返さない。また、このredirectorプロセス85の機能をクライアントプロセス84のプログラムに直接組込むことも可能である。

【0285】図8に示す実施の形態4のクライアント計算機24とredirector計算機72～77との通信が、実施の形態5では同じクライアント計算機24内部で処理されるため、ローカルネットワーク上での通信量が削減できる。したがって、独立したredirector計算機を用意する必要がなく、クライアント計算機24を利用してredirector計算機の機能を実現できるためコストの削減が可能となる。

【0286】〔実施の形態6〕実施の形態6では、実施の形態4において独立した計算機上で実行していたredirectorプロセスを、proxyサーバ計算機上で実行する。図13は、実施の形態6における装置の内部構成を示す図であり、たとえば、クライアント計算機24bから読出要求があった場合に、その読出要求はproxyサーバ計算機99内のredirectorプロセス91で受信される。その後、redirectorプロセス91は、図9を用いて説明したようにアクセス要求を中継するproxyサーバ計算機を選択し、その選択されたproxyサーバ計算機が98であれば、proxyサーバ計算機98のproxyプロセス86に対してアクセス要求を中継する(92)。また、受信したredirectorプロセスの動作しているproxyサーバ計算機と選択されたproxyサーバ計算機がともに99である場合、読出要求の中継はネットワーク上に送出されずproxyサーバ計算機99内部で直接行なわれる(93)。そして、proxyプロセス(86または89)が広域ネットワーク61を介してサーバ計算機11に対してアクセス要求を送出する(62または63)。

【0287】同様に、クライアント計算機24aから読出要求があった場合に、その読出要求は、proxyサーバ計算機98上のredirectorプロセス96で受信される。その後、redirectorプロセスは、proxyサーバ計算機の選択を行ない、その選択されたproxyサーバ計算機が99であれば、proxyサーバ計算機99のproxyプロセス89に対してアクセス要求を中継する(95)。

【0288】また、受信したredirectorプロセスの動作しているproxyサーバ計算機と選択されたproxyサーバ計算機がともに98である場合、読出要求の中継はネットワーク上に送出されずproxyサーバ計算機98内部で直接行なわれる(96)。そして、proxyプロセス(86または89)が広域ネットワーク61を介してサーバ計算機11に対してアクセ

ス要求を送出する(62または63)。

【0289】サーバ計算機11から取得したファイルオブジェクトをキャッシュファイル(87または90)に格納する処理は、上述した処理と同様であるので詳細な説明は繰返さない。

【0290】また、このredirectorプロセスの機能を、proxyサーバ計算機上でproxyサーバとしての機能を実現しているプログラムに直接組込むことも可能である。

【0291】したがって、redirector計算機とproxyサーバ計算機の通信の一部がproxyサーバ計算機内部で処理されるため、ローカルネットワーク上での通信量が削減できる。また、独立したredirector計算機を用意する必要がなく、proxyサーバ計算機を利用してredirector計算機の機能を実現できるためコストを削減することが可能になる。

【0292】〔実施の形態7〕実施の形態4のredirector計算機72～77はアクセス要求の中継のみを行なっていた。しかし、図23で説明したようなツリー状の構成が有効であるような大規模なローカルネットワークでは、1次キャッシュproxyサーバ計算機503～508の持つキャッシュ機能をredirector計算機に持たせることによって、ネットワーク上の通信量およびproxyサーバ計算機の負荷を軽減することが可能となる。図14は、そのシステム構成を示す図である。redirectorサーバ計算機40～45は、図11に示すredirector計算機80と図16に示すproxyサーバ計算機13を組合せることにより可能である。すなわち、図16のリード要求17を図11のredirectorプロセス81が受信することによって実現可能であるので、ここでの詳細な説明は繰返さない。また、実施の形態5におけるクライアント計算機内部でキャッシュを行なうことによって、ローカルネットワーク上の通信量の削減とproxyサーバ計算機の負荷を軽減することが可能となる。

【0293】上述したように、redirector計算機においてキャッシュすることにより、ローカルネットワーク上での通信量およびproxyサーバ計算機の負荷を軽減することができる。

【0294】〔実施の形態8〕上述した実施の形態4から7は、排他的に使用されなければいけないものではない。すなわち、実施の形態8においてはredirector機能を実現するすべての計算機(proxyサーバ計算機、クライアント計算機、redirector計算機)において、redirector機能の使用するハッシュ関数が同一であり、すべてのアクセス要求がそれぞれ同一のproxyサーバ計算機等のキャッシュ機能を有する計算機で中継およびキャッシュされれば、実施の形態4～7に記載された各種の計算機は、単一の

ネットワーク内に混在して使用されても全く問題はない。したがって、それぞれの会社等の組織におけるさまざまなネットワークや計算機の運用事情に応じ、非常に柔軟なネットワーク運用を行なうことが可能となる。

【0295】ある程度以上の大規模なローカルネットワークでは、広域ネットワークとの接続が複数の経路(図8の62~64)からなる場合もある。そのような場合、proxyサーバ計算機ごとに異なる経路を使用すると、クライアント計算機からのアクセス要求を複数ある経路に分散して中継することになる。したがって、複数ある広域ネットワークとの通信経路を有効利用し、広域ネットワークとローカルネットワークとの通信のスループット向上が図れる。

【0296】実際の運用においては、すべての計算機の設定を一度に変えることは現実的ではない。移行には暫く時間がかかることが普通である。より効率のよい運用を図るためには、すべてのredirector機能を有する計算機内でハッシュ関数を一致させる必要があるため、すべてのクライアント計算機がredirector計算機を利用するようシステムを構築するか、クライアント計算機がredirector機能を持つ必要がある。しかし、そのようなシステムを構築しなくても

redirector機能を有する計算機を使用することは可能である。すなわち、本発明の各方式では、従来のproxyサーバ計算機やクライアント計算機を全く手を付けない状態で、システムの一部にredirector機能を持たせて、システム全体はそのまま使用することも可能である。

【0297】同様に、システムの再構築の際、複数のredirector機能におけるハッシュ関数が一致しない場合もあり得る。このような場合でも、システム全体の動作自体には影響はないので、通常と同様に利用が可能である。上述したいずれの場合も利用は可能であるが、proxyサーバ計算機その他の有効利用の観点からは望ましい状況とはいえない。したがって、できるだけ早い時期にすべてのredirector機能を有する計算機での設定を統一させることが望ましい。しかし移行期に新旧の設定を混在していても不必要な混乱を起こさず、スムーズに移行することが可能である。

【0298】なお、httpプロトコルを含めた参考文献リストを以下に示す。

【0299】

【数26】

- ・[Ref. 1]横塚実、『インターネットとはなにか』
http://www.rwcp.or.jp/people/yokotuka/internet_report.html
- ・[Ref. 2]page170, "キャッシュ利用でトラフィックを低減するプロキシサーバ"
『企業ユーザのためのインターネットハンドブック』日経コミュニケーション、
ISBN4-822-1365-X, 日経BP社
- ・[Ref. 3]WWWに関する参考文献
<http://www.w3.org/hypertext/WWW/Protocols/Overview.html>
- ・[Ref. 4]HTTPの仕様に関するインターネット上の文献
<http://www.w3.org/hypertext/WWW/Protocols/HTTP/HTTP2.html>
- ・[Ref. 4の日本語資料]日本語のHTTPプロトコル仕様書としては、木内貴弘
(kiuchi@epistat.m.u-tokyo.ac.jp)氏翻訳の
<http://www.epistat.m.u-tokyo.ac.jp/internet/draft-fielding-http-spec-01-jp.txt>、オリジナルファイル名: 著者: T.Berners-Lee, R.T.Fielding, H.Frystyk
Nielsenがある。
- ・[Ref. 5]電子技術総合研究所の佐藤氏の作成されたdelegateについては
<http://www.etl.go.jp:8080/etl/People/ysato@etl.go.jp/DeleGate/> から情報が
入手できる。
- ・[Ref. 6]松下電器産業株式会社、特開平6-290090, "遠隔ファイルアクセス
システム
- ・[Ref. 7]WWWシステム全般の情報は、World Wide Webコンソーシアム
<http://www.w3.org/hypertext/WWW/TheProject/>

【0300】

【数27】

- ・[Ref. 8] CERN httpdに関しては
<http://www.w3.org/hypertext/WWW/Daemon/>
- ・[Ref. 9] 大家隆弘(徳島大学), WWW 多段proxy計画(案)
<http://www.orions.ad.jp/project/multiproxy/multiproxy-jp.html>
- ・[Ref. 10] CERNhttpdでのキャッシュ制御解説
<http://info.cern.ch/hypertext/WWW/Daemon/User/Config/Caching.html>
- ・[Ref. 11] 東田学, 対外接続のトラフィックを軽減するためのキャッシュ・サーバ導入について
<http://www.osaka-u.ac.jp/odins/document/proxy-jp.html>
- ・[Ref. 12] 佐藤豊, “多目的プロキシ・サーバD eleGateの機能詳細” インターフェイス
1995年9月p130-p146
- ・[Ref. 13] インターナショナル・ビジネス・マシーンズ・コーポレーション, 特開平6-309264,
“キャッシュ・サーバ・ノードを有するコンピュータ・ネットワークにおける資源を探索する方法
及び装置”
- ・[Ref. 14] “Siva——A hierarchical caching HTTP proxy”
<http://www.cs.cornell.edu/Info/People/mdw/siva/siva.html>
- ・[Ref. 15] “A Central Caching Proxy Server for WWW users at
the University of Melbourne”
<http://www.scu.edu.au/ausweb95/papers/indexing/ocallaghan/>
- ・[Ref. 16] Lynx ブラウザについて
http://kufacts.cc.ukans.edu/about_lynx/about_lynx.html

【図面の簡単な説明】

【図1】本発明におけるゲートウェイ装置の一例である proxyサーバ計算機の動作概念図である。

【図2】WWWサーバ内の情報ページをクライアント計算機に読出したときの表示画面を示す図である。

【図3】実施の形態1のゲートウェイ装置である proxyサーバ計算機で行なわれるキャッシュ更新プロセスの処理手順を示すフローチャートである。

【図4】proxyプロセスにおけるキャッシュファイル管理の処理手順を示すフローチャートである。

【図5】実施の形態2のゲートウェイ装置である proxyサーバ装置で行なわれるキャッシュ更新プロセスの処理手順を示すフローチャートである。

【図6】実施の形態3のゲートウェイ装置である proxyサーバ装置で行なわれるキャッシュ更新プロセスの処理手順を示すフローチャートである。

【図7】本発明を実施した場合のサーバ計算機とクライアント計算機の間で転送されるデータのタイムチャートである。

【図8】実施の形態4における分散ファイルシステムの全体構成を示す図である。

【図9】実施の形態4における redirector 計算機の処理手順を示すフローチャートである。

【図10】図2に示す同一ページのファイルオブジェクトに対するアクセス要求が2台のサーバ計算機に分散されるようすを示す図である。

【図11】実施の形態4における redirector 計算機の内部構成を示す図である。

【図12】実施の形態5におけるクライアント計算機の内部構成を示す図である。

【図13】実施の形態6における proxyサーバ計算機の内部構成を示す図である。

【図14】実施の形態7における分散ファイルシステムの全体構成を示す図である。

【図15】従来のゲートウェイ計算機の構成を示す図である。

【図16】proxyサーバ装置の概念図である。

【図17】proxyサーバ装置の回路構成の第1の例を示す図である。

【図18】proxyサーバ装置の回路構成の第2の例を示す図である。

【図19】proxyサーバ装置の回路構成の第3の例であって、本願発明の実施の形態で proxyサーバ装置としても使用される proxyサーバ装置の回路構成を示す図である。

【図20】キャッシュヒット率と平均アクセス速度との関係を示すグラフである。

【図21】キャッシュファイルサイズとキャッシュヒット率との関係を示すグラフである。

【図22】キャッシュ有効期限とキャッシュファイルの大きさとの関係を示すグラフである。

【図23】従来の分散ファイルシステムの全体構成の第

1の例を示す図である。

【図24】従来の分散ファイルシステムの全体構成の第2の例を示す図である。

【図25】従来の分散ファイルシステムの全体構成の第3の例を示す図である。

【図26】従来の分散ファイルシステムにおけるサーバ計算機内のデータの更新とクライアント計算機が読出したデータとの関係を示すタイムチャートである。

【符号の説明】

11 サーバ計算機

14 第1のproxyプロセス

15 アクセスログ

16 キャッシュファイル

30 パターンファイル

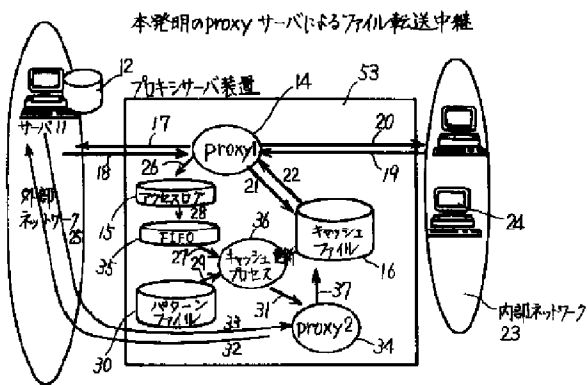
34 第2のproxyプロセス

35 FIFOバッファ

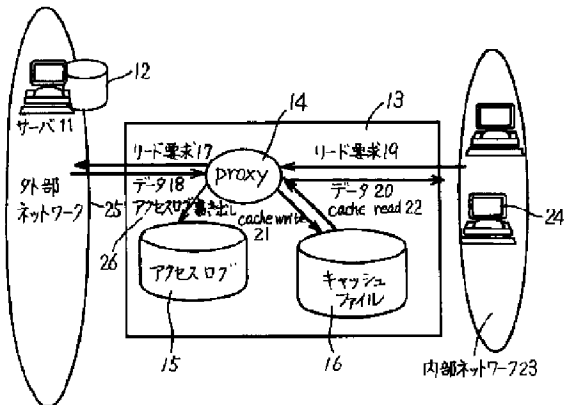
36 キャッシュ更新プロセス

72～77 redirector計算機

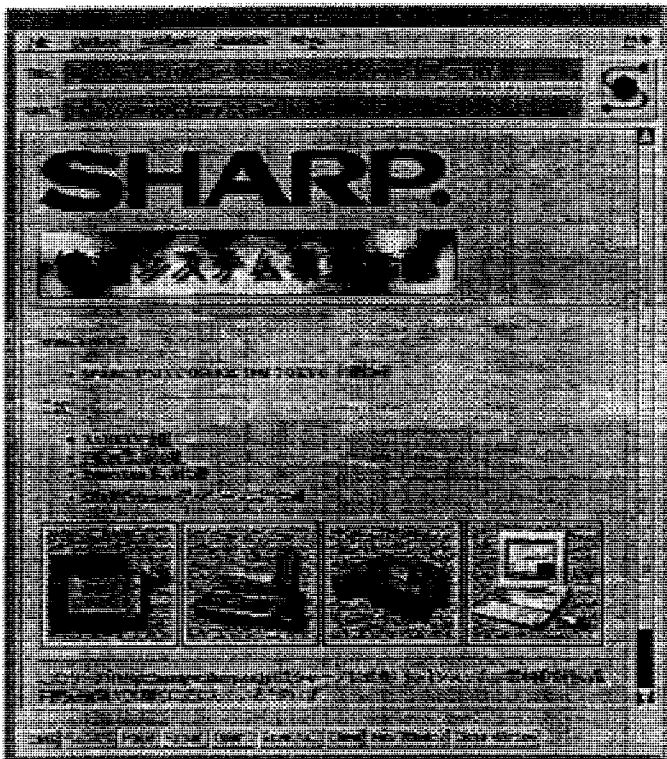
【図1】



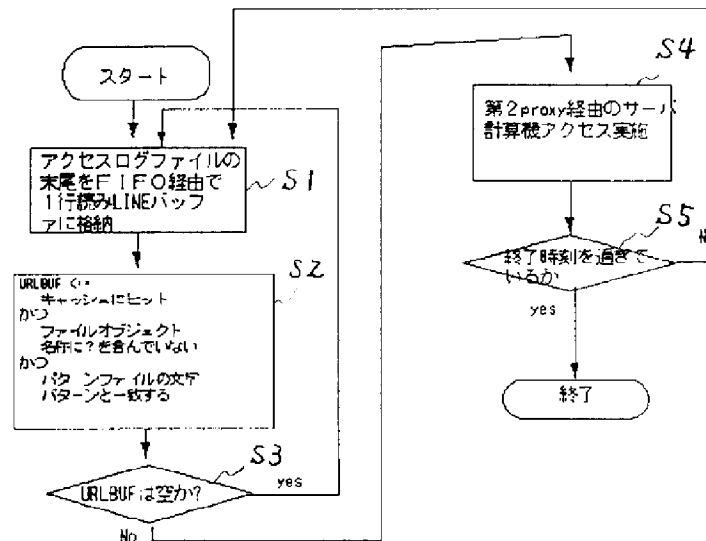
【図16】



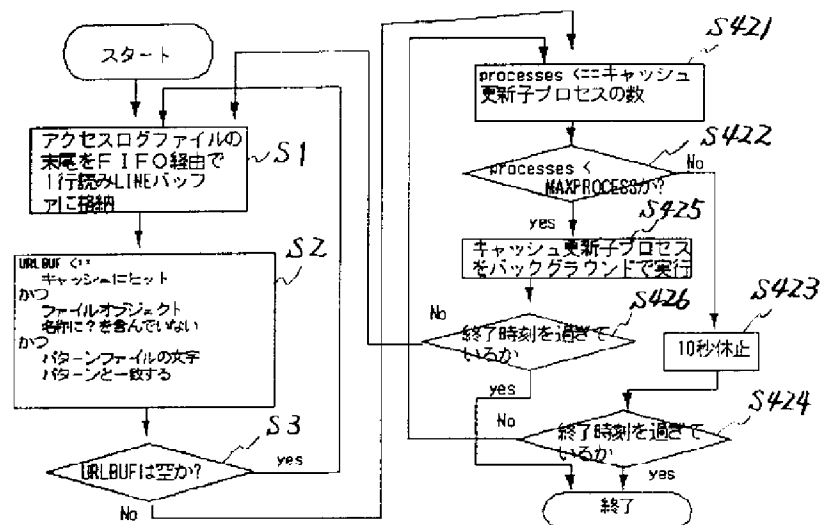
【図2】



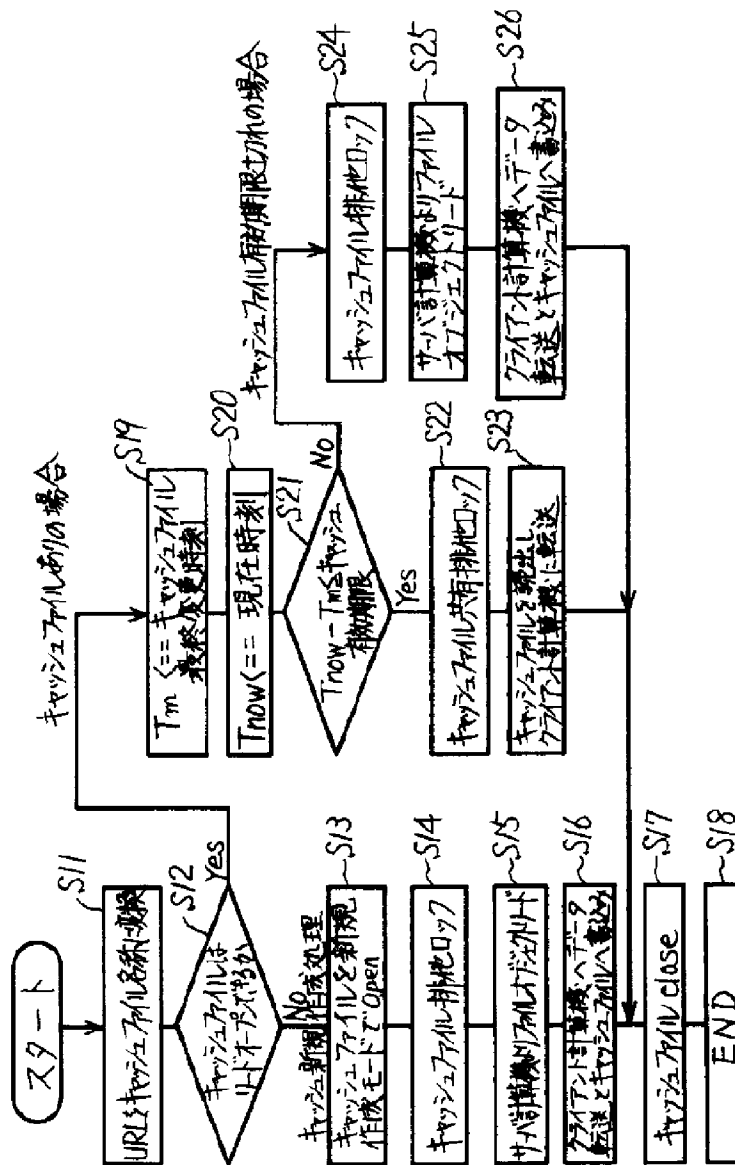
【図3】



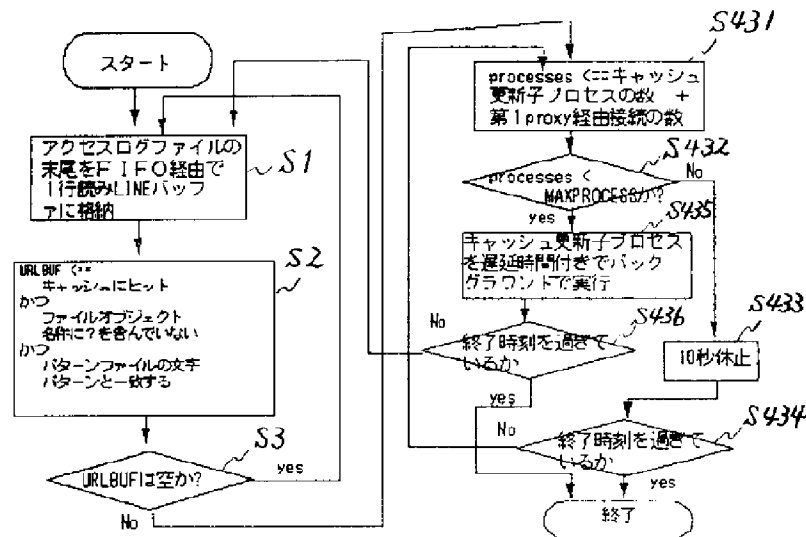
【図5】



【図4】

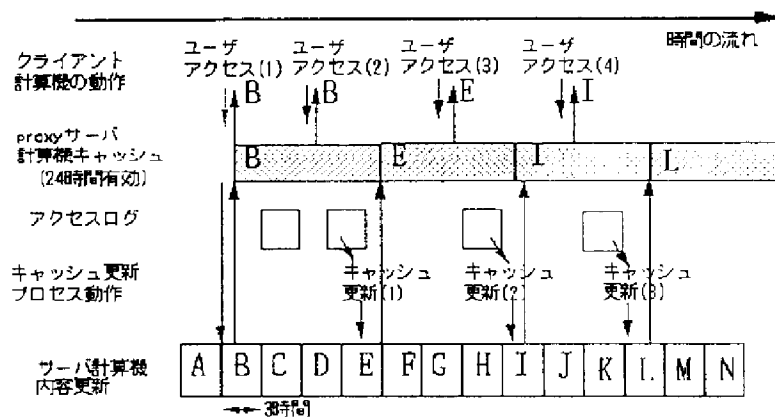


【図6】

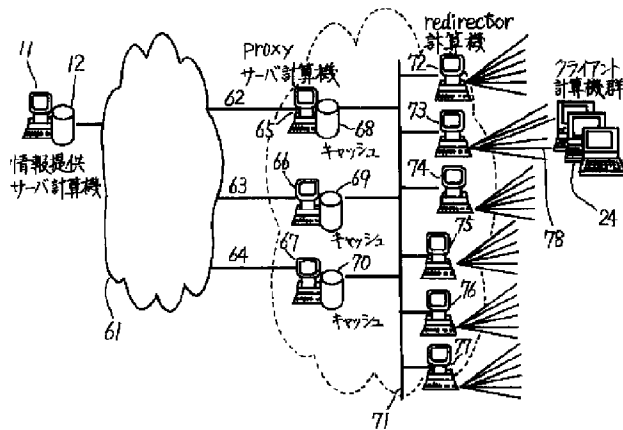


【図7】

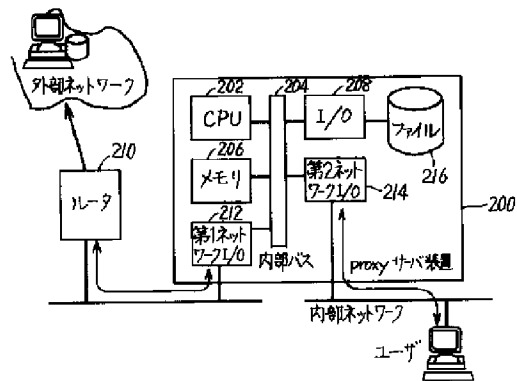
変更間隔3時間のサーバデータアクセスのタイムチャート (キャッシュ更新プロセスの動作時)



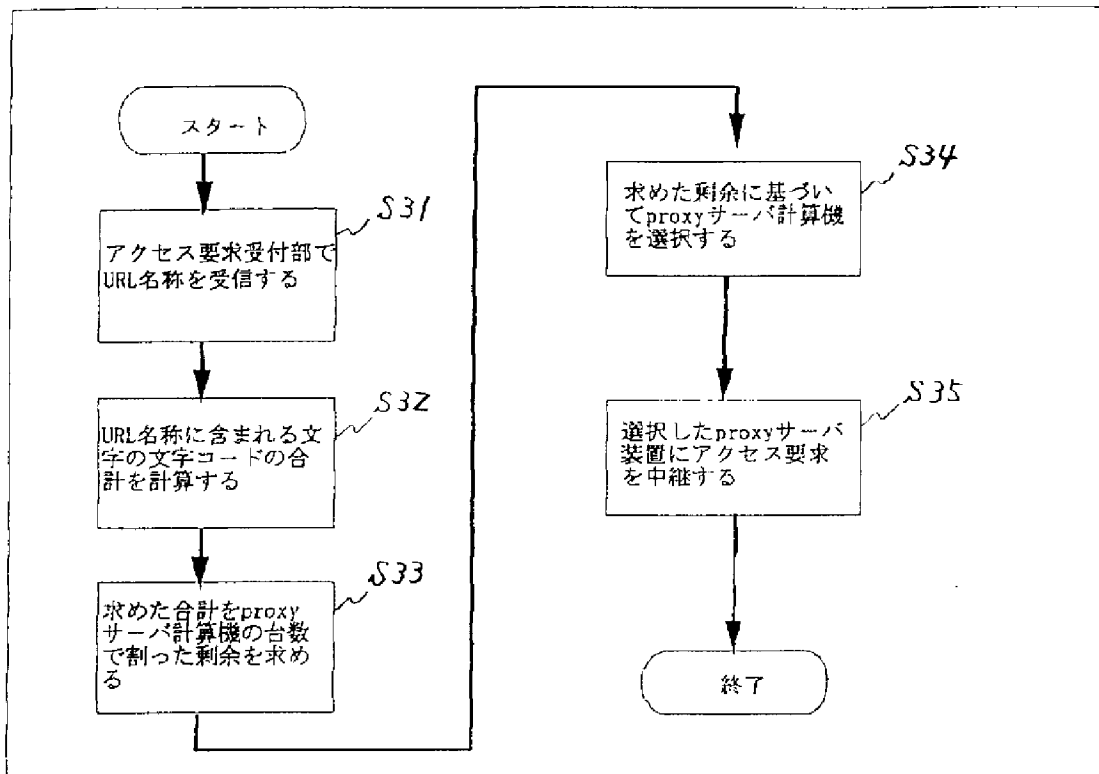
【図8】



【図17】



【図9】

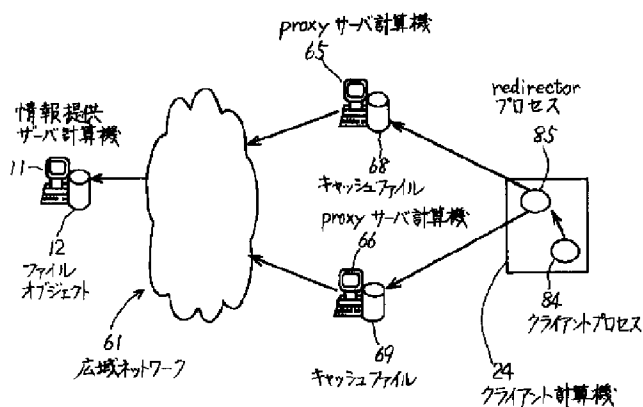


【図10】

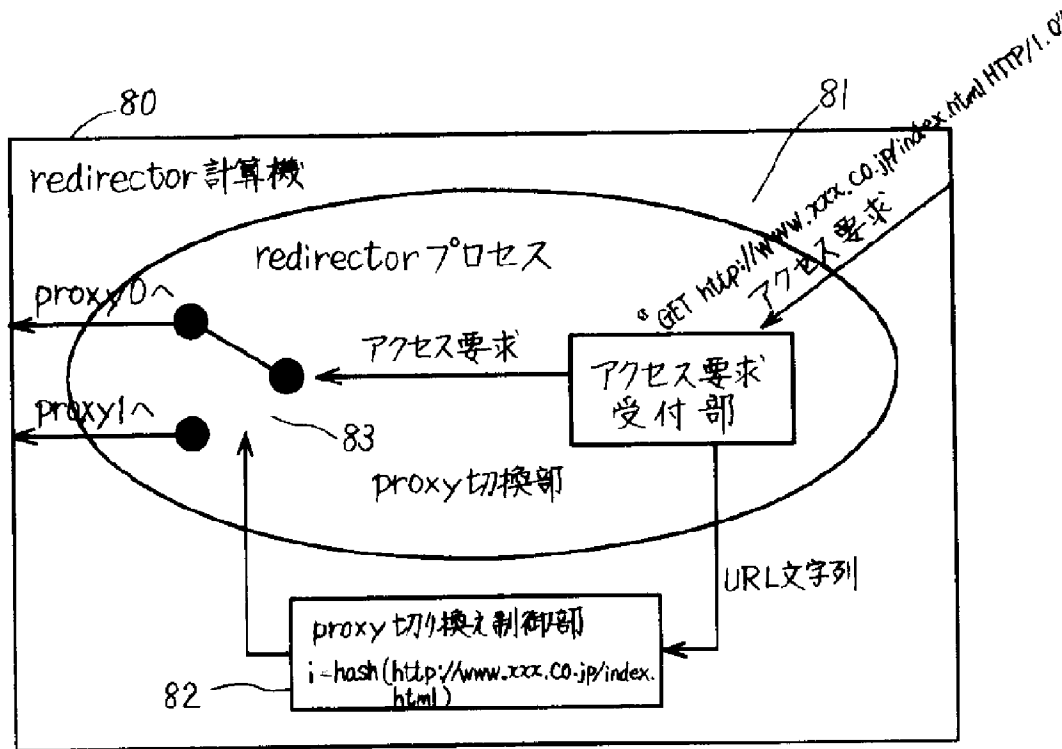
URL	check sum	checksumの 2の剰余	選択された proxyserver
http://naragw.sharp.co.jp/	2395	1	proxyserver1
http://naragw.sharp.co.jp/sharpcolor.gif	3836	0	proxyserver0
http://naragw.sharp.co.jp/isg.gif	3974	0	proxyserver0
http://naragw.sharp.co.jp/Zaurus.gif	3401	1	proxyserver1
http://naragw.sharp.co.jp/Shoin.gif	3264	0	proxyserver0
http://naragw.sharp.co.jp/Prostation.gif	3826	0	proxyserver0
http://naragw.sharp.co.jp/S2.gif	2884	0	proxyserver0

変換後のURL
http://proxyserver1:10080/_-http://naragw.sharp.co.jp/
http://proxyserver0:10080/_-http://naragw.sharp.co.jp/sharpcolor.gif
http://proxyserver0:10080/_-http://naragw.sharp.co.jp/isg.gif
http://proxyserver1:10080/_-http://naragw.sharp.co.jp/Zaurus.gif
http://proxyserver0:10080/_-http://naragw.sharp.co.jp/Shoin.gif
http://proxyserver0:10080/_-http://naragw.sharp.co.jp/Prostation.gif
http://proxyserver0:10080/_-http://naragw.sharp.co.jp/S2.gif

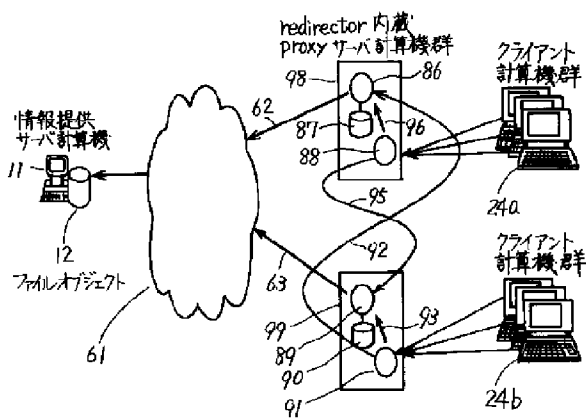
【図12】



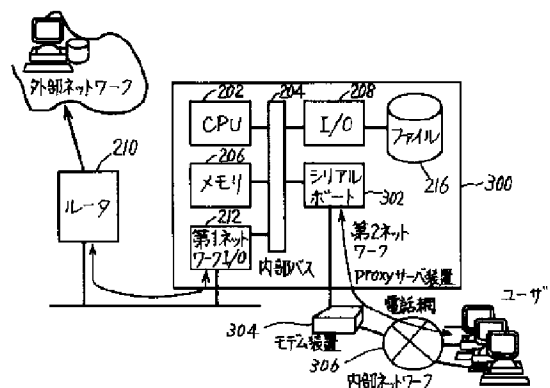
【図11】



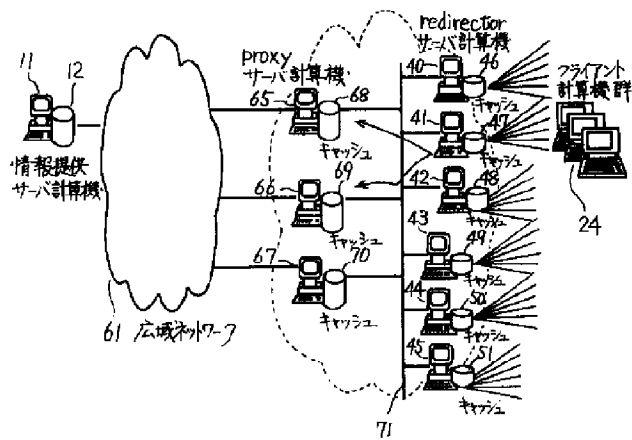
【図13】



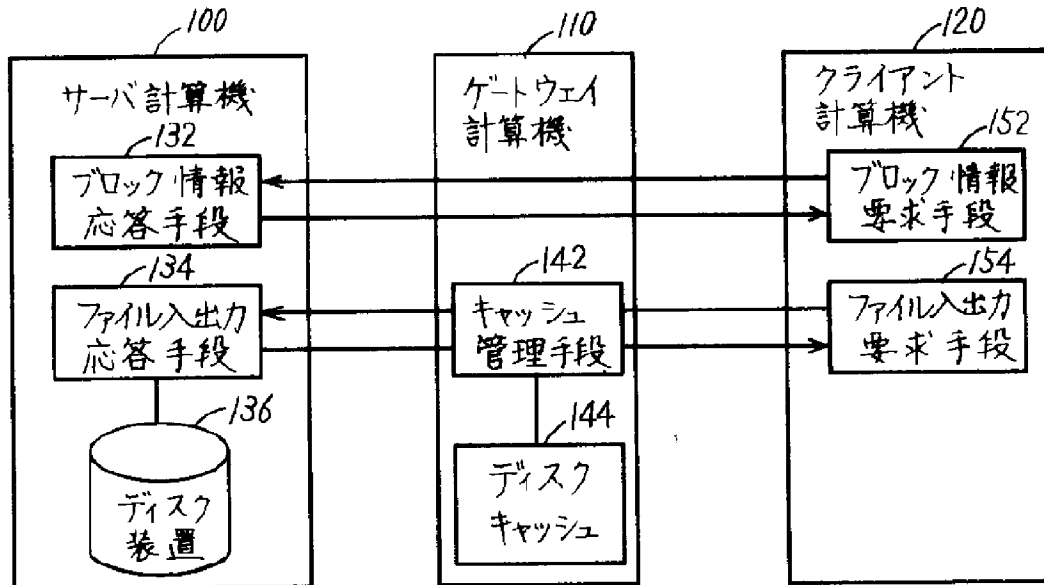
【図18】



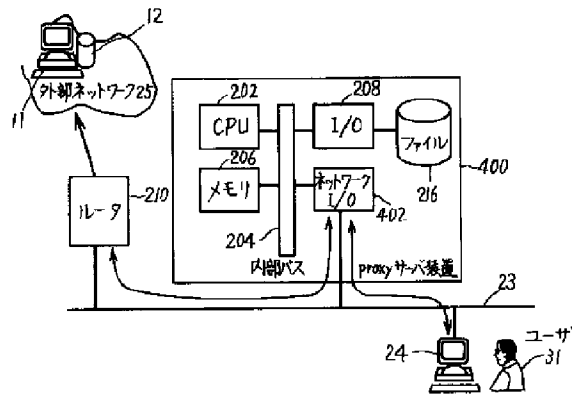
【図14】



【図15】



【図19】

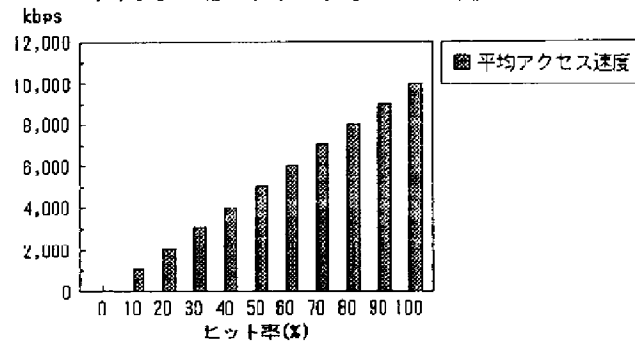


【図20】

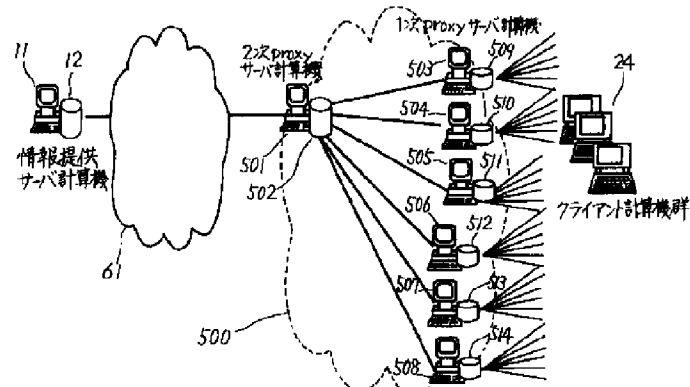
キャッシュヒット率向上による 平均アクセス速度向上

- 平均アクセス速度 = $64\text{ kbps} \times \text{ミスヒット率} + 10\text{ Mbps} \times \text{ヒット率}$

キャッシュヒット率と平均アクセス速度の関係

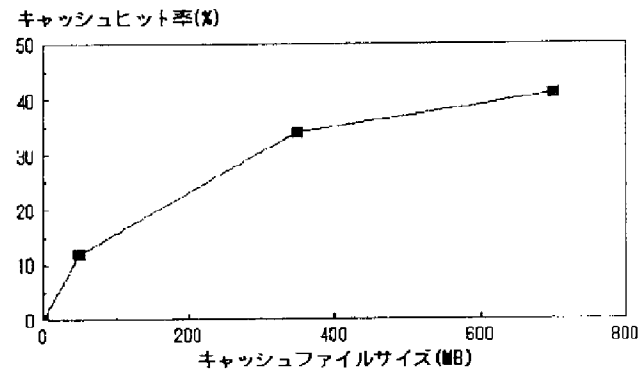


【図23】



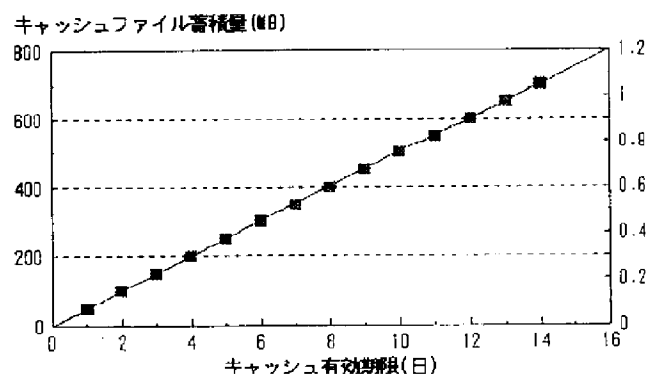
【図21】

キャッシュファイルサイズと キャッシュヒット率の関係

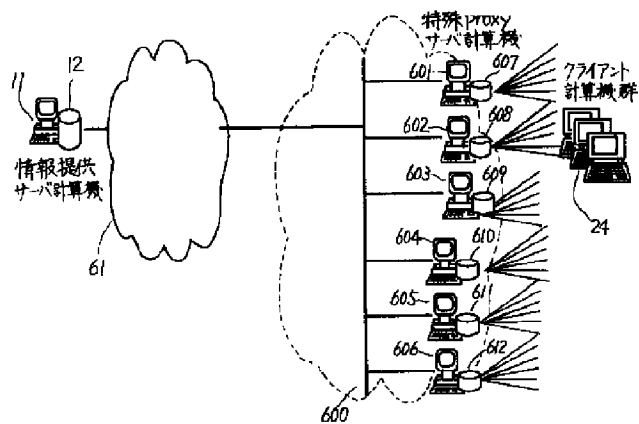


【図22】

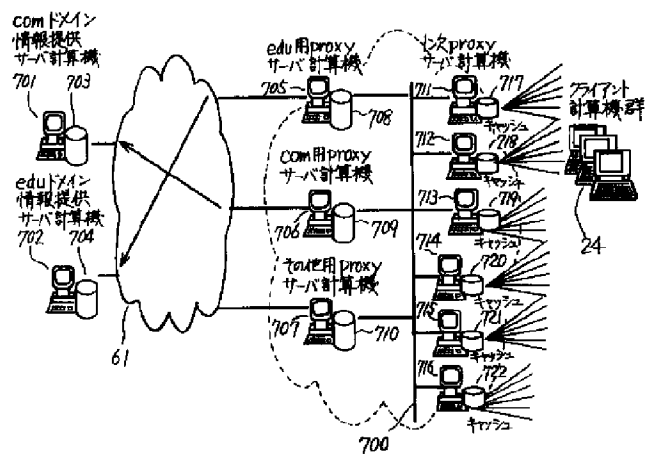
キャッシュ有効期限とキャッシュ ファイルの大きさの関係



【図24】



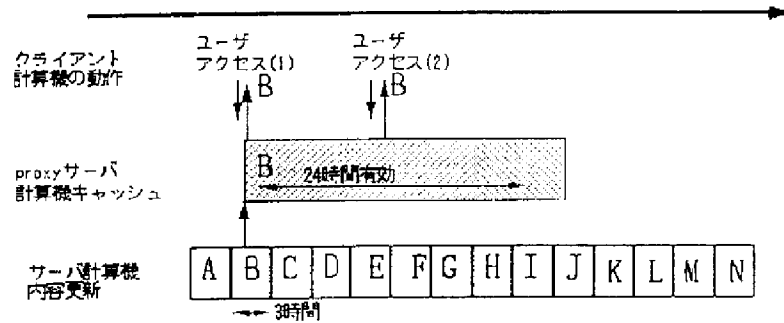
【図25】



【図26】

変更間隔3時間のサーバデータアクセスのタイムチャート

- proxyサーバ計算機でのキャッシュは有効期限24時間
- サーバ計算機のファイルオブジェクトは3時間間隔で更新



(19)



Europäisches Patentamt

European Patent Office

Office européen des brevets



(11)

EP 0 813 133 A2

(12)

EUROPEAN PATENT APPLICATION

(43) Date of publication:
17.12.1997 Bulletin 1997/51

(51) Int. Cl.⁶: G06F 1/00

(21) Application number: 97303445.7

(22) Date of filing: 20.05.1997

(84) Designated Contracting States:
DE FR GB

(30) Priority: 11.06.1996 US 661687

(71) Applicant:
International Business Machines
Corporation
Armonk, N.Y. 10504 (US)

(72) Inventors:
• Anand, Rangachari
Palisades Park, New Jersey 07650 (US)

• Islam, Nayeem
Yorktown Heights, New York 10598 (US)
• Rao, Josyula Ramachandra
Briarcliff Manor, New York 10510 (US)

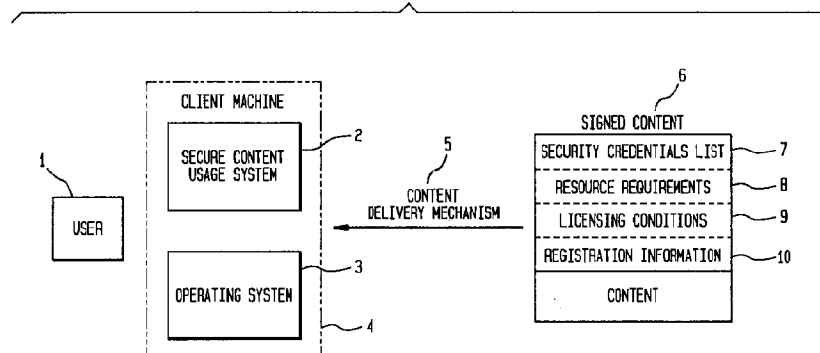
(74) Representative: Waldner, Philip
IBM United Kingdom Limited,
Intellectual Property Department,
Hursley Park
Winchester, Hampshire SO21 2JN (GB)

(54) A uniform mechanism for using signed content

(57) A scheme for downloading signed content onto a machine by a delivery mechanism (such as floppy diskettes and the Internet) is described. While there are no restrictions on the nature of the contents, the signature on the content describes the security credentials of the creator, the resource requirements, and licensing information (such as the time period of its validity). Once the content has been downloaded, it can be used on the client machine in various ways. It can be installed on the client machine and there after users can execute it. Any use of the content of the

content requires access to computing resources on the client machine. This access is mediated by means of a security manager that uses the information in the content's signature to manufacture capabilities that grant and regulate access to different subsets of the computing resources. In this sense, our scheme unifies work done in the previously disparate fields of network security and operating systems and results in data structures and algorithms that combine and manipulate elements of both.

FIG. 1



EP 0 813 133 A2

Description

The present invention relates to a uniform mechanism for using signed content in particular in computer security mechanisms for safely executing software which has been obtained over a network or by other means from an untrusted source.

In order to increase the utility of networked computers, methods have been sought to allow them to execute programs obtained from servers. The primary advantage of such a system from the user's view point is that it decreases the amount of software that must be stored on the user computer. From the software developers viewpoint, the system has a number of advantages, the main one being that the application provider has greater control over the distribution of the programs. The use of embedded Java applets (i.e. programs) in World Wide Web documents is a popular example of such a system.

A significant concern with this approach, is that the software which obtained from the server may be malicious and damage the user's computer or steal data. The downloaded software must therefore be executed in a controlled environment in which they are given only the system resources that they need and no more. The main problem with the current Java applet security mechanism is that it is not flexible enough. All Java applets are considered hostile and are not allowed to access most resources on the user machine's operating system.

There are variety of standard techniques for public cryptography and authentication. RSA is an example of a widely used public key cryptography algorithm. Examples, of implementations include RSAref and PGP.

Mechanisms also exist to create digital signatures for messages. These link persons with the contents of messages. They can also be used to create digital signatures for messages such that the creator of the message can not disavow the message. The MD5 algorithm in conjunction with RSA is an example of a signature system.

A number of computer operating system use capabilities to control access to system resources. A capability is a permission held by a process to perform some action on another object. Notable operating systems that use capabilities for enforcing security are Amoeba and Mach.

In accordance with a first aspect of the present invention, a secure content usage system and method for use in a computing system is provided. The system includes a content importation mechanism; an extractor, operatively connected to receive signed content imported by way of the importation mechanism, for extracting portions of a signature from the signed content, the portions including a security credentials associated with the content; resource requirements for using the content; an analysis module for verifying authenticity and integrity of the signed content using at least the security credentials supplied by the extractor and taking

remedial actions when any of the authenticity and the integrity are in doubt; and, an enforcement module for ensuring that use of the signed content conforms to the resource requirements and security credentials.

In accordance with a second aspect of the present invention, there is provided a computer readable memory having signed content instantiated thereon. The signed content includes a computer readable signature and computer readable content, the computer readable signature including a plurality of fields comprising a security credentials field including cryptographic identifies of at least an originator and intermediaries involved in a chain of distribution of the computer readable content and a resource requirements field identifying computing resources required for using the computer readable content.

In accordance with a third aspect of the present invention, a content usage system and method for enforcing licensing terms in a computing system is provided. The system includes a content importation mechanism; an extractor, operatively connected to receive signed content imported by way of the importation mechanism, for extracting portions of a signature from the signed content, the portions including a computer readable licensing terms associated with the content; and, an enforcement module for controlling operation of the computing system so as to ensure that use of the signed content conforms to the licensing terms.

Figure 1 is an abstract view of a content delivery mechanism according to the principles of the present invention;

Figure 2 shows sources and intermediaries in a content delivery system;

Figure 3 shows how the manufacturer/author and intermediaries add their signatures to the content being delivered in accordance with an embodiment of the present invention;

Figure 4 shows the modules involved in processing the signed content in the user machine according to an embodiment of the present invention;

Figure 5 shows the Access Information table used by the Enforcer module of Figure 4;

Figure 6 depicts the relationship between the capabilities of the various entities in the secure content usage system of Figure 4;

Figure 7 shows the relationship between the privileges given to different users for the signed content in the system of Figure 4;

Figure 8 shows an embodiment of the present invention when the signed content is a Java applet; Figure 9 shows the actions taken by secure content usage system of Figure 4 upon receipt of signed content; and,

Figure 10. shows the method by which security is enforced by the Enforcement module of Figure 4.

Detailed Description of the Preferred Embodiment

The embodiments of this invention will now be described in detail with reference to the drawings.

Figure 1 gives an abstract description of the invention. A user 1, uses a client machine 4 and uses a content delivery mechanism 5 to transfer signed content 6 to his/her machine. Examples of delivery mechanisms include floppy diskettes, CD-ROMS and the Internet. Examples of executable content include Java applets, OLE components and SOM components. The content has a signature. Other types of content could include text, audio and video. The signature includes four fields. The first field 7 is a list of the security credentials of the software. This is described in further detail in Figure 3. Examples of security credentials include the identities of the author and the manufacturer. The credentials guarantee that the content was created and distributed by the principals whose credentials occur on the list. In addition, they provide a means to check that the content has not been altered after it was signed. Further, they provide a means of ensuring that the author cannot disavow the content that he/she has created. The second field 8 describes the computing resources 3 that the content needs on the client machine. These resources are needed for the content to achieve its purpose on the client machine. Examples of purpose include installing and executing the signed content. Examples of computing resources include disk space, file space, file access, RAM, CPU, networking capabilities and the user display.

Once the signed content is downloaded to the user's machine, the user can use the content in various ways. Examples of use of the content include installing it, viewing it and executing it. The content is used in a carefully controlled environment 2 on the client machine. This use of the signed content may require access to computing resources on the client machine. The resources required for using the signed content 8 are part of the signature of the content. Access to such resources is mediated by means of a secure content usage system 2.

The third field (this is optional) provides licensing information 9. Examples of licensing information include terms and conditions for use such as the number of machines and the time period for which the content can be used. The fourth field (this is optional) is the registration information 10. This information is used to automatically register the content with the provider.

Figure 2 depicts an example of the content delivery mechanism. The content originates on a manufacturer's or an author's machine 15, 16, 17 and makes its way via a number of intermediary machines 12, 13, 14 before being downloaded to the client machine 11.

Figure 3 depicts the accumulation of credentials in the signed content as it is distributed from the manufacturer's machine 22 to the user's machine 20. The manufacturer adds its security credentials to the signed content 25 before sending it by some means 27 to an

intermediary 21. The intermediary, in turn, adds its security credentials to the signed content 24, before sending it to the next intermediary in the distribution chain. In this manner, when the signed content finally arrives at the user, it contains a list of the security credentials of all the intermediaries and the manufacturer 23.

Figure 4 depicts the downloading of signed content from a content provider 32 and subsequent processing in a secure content usage system 31. The secure content usage system 31 can be embodied as part of a general purpose computing system (not shown), such as an IBM PC personal computer, an IBM RS/6000 workstation or any other workstation suitable for use as a client system. The signed content is downloaded by a content importation system 33. An extractor 34 parses the fields of the signature and passes this information to the analyzer module 35. The analyzer verifies the integrity of the content. It then examines the list of security credentials to determine the level of access and trust that the content should be used with on the machine. Next, it looks at the resource requirements of the content and determines, possibly with user input, if these requirements can be met. This information is then passed to the content interpreter 36 and the enforcement module 37.

The content importation mechanism 33 can be embodied, for example, as a network interface (e.g. which can couple a user to the Internet), a diskette subsystem, a CD ROM subsystem or a cartridge memory subsystem. The extractor 34, the analysis module 35, the content interpreter 36 and the enforcement module 37 can be embodied as program code executable by the workstation on which the secure content will be executed. The enforcement module is preferably coupled to the workstation's operating system (such as OS/2, UNIX or Windows NT). The content interpreter 36 can be embodied as a module within the operating system or it can be distinct from the operating system (such as a Java interpreter).

The flow chart corresponding to the operation of the system of Figure 4 is shown in Figure 9. The content interpreter is the mechanism to use the content. Examples of the content interpreter include internet browsers and the Java virtual machine. The enforcement module uses the level of trust determined by the analyzer to create entries in an access information table. This table is described in Figure 5.

Use of signed content typically requires access to operating system resources. Figure 5 depicts a table 40 that the enforcement module uses to keep track of the resources that have been requested and consumed by signed contents that are being used on its machine. The enforcement module uses the security credentials 41 on the signed content to determine the limit on the resources 42 that the signed content should be given on the client machine. This determination can be done in a variety of ways including preconfiguration by means of tables and requesting explicit user input to determine

the access that the content should get. Effectively, the enforcement module manufactures a capability for the signed content that reflects "who gets how much access of what". Typically, the resources that a signed content gets are a subset of the resources that the user has access to on the client machine. The security manager keeps track of the resources that have been consumed 43 by the content. This is achieved by ensuring that all accesses to the system resources by the signed content pass through the security manager. The table also contains an entry for the resources that have been requested 43 by the signed content. If at any time, the resources consumed 43 exceed either the resource limit 42 or the resources requested 44, the security manager can take remedial action. Examples of remedial action include terminating the use of the signed content and asking the user for guidance on how to proceed.

Figure 6 depicts the relationship between the capabilities of the various entities. The user's privileges 51 are a subset of the privileges of the operating system 50. The signed content is executed in an environment whose privileges 52 are a subset of the user's privileges. In turn, the privileges of the signed content 53 are a subset of its execution environment. The use of the signed content may cause the use of other content on the client machine. For instance, executing a Java applet may cause another executable to be instantiated into a process on the client machine. The privileges of such spawned content 54 are a subset of the privileges accorded to the signed content. Note that incorporating the resource requirements in the signature of a signed content provides the security manager with an effective mechanism for implementing these restrictions. The spawned content can be allowed to execute as long as the resources it consumes are a subset of the resource limits placed on the signed content. All this information can be tracked in the security manager table shown in Figure 5.

Once the signed content is downloaded onto the user's machine, the user gets a capability to use the content. This capability is associated with the user who initiated the transfer. The user can permit other users to use the signed content on his/her machine. Figure 7 shows the relationship between the privileges of other users such as 61, 62 and 63 and the privileges of the installing user 27. For instance, if the signed content was a Lotus document, then a user's privileges would reflect whether he could read, write or change the document.

Figure 8 shows an embodiment where the signed content is a signed Java applet 80. The list of security credentials 79 on the applet are those of its author, the manufacturer and the retailer. The applet resides on a server machine 77 and is managed by a server process 78. Note that the server machine and server process are merely a distribution mechanism and they need not have any relation with the author. The content delivery mechanism is the Internet 76.

A client agent 72 acting on behalf of a user 71 and residing on a client machine 70 downloads the applet by contacting the server process. The client agent sends its security credentials, such as the identity of the user (such as its public key or certificate) and the identity of the client machine (such as its IP address) to the server process. This information is used by the server process to authenticate the users and keep track of the usage of the applet. In response, the server process sends the signed applet, the identity of the server machine and the server process' public key (or certificate) back to the client. The server must encrypt its response with the public key of the user to ensure that the applet is securely conveyed to the client machine.

The client agent verifies the integrity of the content and the associated signature. Once this is done, the client agent determines the security credentials and the resource requirements of the signed content. It decrypts the server's response using its private key and extracts the security information in the response, that is, the identities of the creator (such as public key or certificate), the server process (such as public key or certificate) and the identity of the server machine (such as IP address). This information along with the name of the applet, the resource requirements stated in the signature and the identities of the user and the client machine are passed to the security enforcer 74. The security credentials of the signed applet are stored as a capability which consists of a triple consisting of the name of the signed content, the security credentials and the stated resource requirements are given to the security manager.

The security enforcer is akin to the security manager in the Java runtime environment. It is a trusted system service that cannot be changed. It uses security credentials of the signed content to compute the capabilities with which the applet can be executed on the client machine. When the signed content is set in execution, all calls to system resources are mediated through the security manager. The security manager uses the capability associated with the applet to determine if the resources requested by the applet should be granted (Figure 10). The manager can be used to program a range of security policies to determine the type of access that the signed applet has to the system resources. This can range from simple policies such as no access, complete access, access configured in advance by the user and access can be explicitly granted by promoting the user by means of dialog boxes.

The user who downloads the applet determines who else is allowed to access it. For each user a special capability is manufactured. When the content is it does so with a subset of the access rights of the invoker. At any time the security manager may revoke capabilities given to users of the applet.

Now that the invention has been described by way of the preferred embodiment, various modifications and improvements will occur to those of skill in the art. Thus,

it should be understood that the preferred embodiment has been provided as an example and not as a limitation. The scope of the invention is defined by the appended claims.

In summary there is described a scheme for downloading signed content onto a machine by a delivery mechanism (such as floppy diskettes and the Internet) is described. While there are no restrictions on the nature of the contents, the signature on the content describes the security credentials of the creator, the resource requirements, and licensing information (such as the time period of its validity). Once the content has been downloaded, it can be used on the client machine in various ways. It can be installed on the client machine and there after users can execute it. Any use of the content of the content requires access to computing resources on the client machine. This access is mediated by means of a security manager that uses the information in the content's signature to manufacture capabilities that grant and regulate access to different subsets of the computing resources. In this sense, our scheme unifies work done in the previously disparate fields of network security and operating systems and results in data structures and algorithms that combine and manipulate elements of both.

Claims

1. A secure content usage system for use in a computing system, comprising:

a content importation mechanism;

an extractor, operatively connected to receive signed content imported by way of the importation mechanism, for extracting portions of a signature from the signed content, the portions including a security credentials associated with the content;

resource requirements for using the content;

an analysis module for verifying authenticity and integrity of the signed content using at least the security credentials supplied by the extractor and taking remedial actions when any of the authenticity and the integrity are in doubt; and, an enforcement module for ensuring that use of the signed content conforms to the resource requirements and security credentials.

2. The system of Claim 1 wherein the extractor further includes one or both of:

i) means for extracting registration information from the signature and further comprising means for registering the signed content with a provider without further user intervention; and

ii) means for extracting licensing terms from the signature and wherein the enforcement module includes means for interacting with the operating system to ensure that the use conforms with the licensing terms.

3. The system of Claim 1 further including a data structure stored in a memory of the computing system, the data structure including a table of correspondence between users, the security credentials and features of the signed content, wherein the enforcement mechanisms is connected to read the table of correspondence from the data structure and includes means for enforcing use of the signed content by the users in accordance with the correspondence.
4. The system of Claim 1 wherein the enforcement module includes means for tracking processes spawned from the signed content and for ensuring operation of the processes conform to the resource requirements and security credentials.
5. The system of Claim 1 wherein the importation mechanism is one of:

i) a communication channel coupled to a communication network;

ii) rotating storage; or

iii) a removable memory card.

6. The system of any of Claims 1 to 5 further including a data structure stored in a memory of the computing system, the data structure including a table of correspondence between the signed content, the resource requirements, actual resources consumed by the signed content and any resource limits imposed on the signed content by the computing system.
7. The system of Claim 6 wherein the table further includes usage restrictions imposed on the signed content by the licensing terms.
8. A computer readable memory having signed content instantiated thereon, the signed content including a computer readable signature and computer readable content, the computer readable signature including a plurality of fields comprising a security credentials field including cryptographic identifies of at least an originator and intermediaries involved in a chain of distribution of the computer readable content and a resource requirements field identifying computing resources required for using the computer readable content.
9. A content usage system for use in a computing sys-

tem, comprising:

a content importation mechanism;

an extractor, operatively connected to receive
signed content imported by way of the importa-
tion mechanism, for extracting portions of a signa-
ture from the signed content, the portions
including a computer readable licensing terms
associated with the content; and

an enforcement module for controlling opera-
tion of the computing system so as to ensure
that use of the signed content conforms to the
licensing terms.

10. A method for ensuring secure usage of signed con-
tent in a computing system, comprising:

importing the signed content into the comput-
ing system;

extracting portions of a signature from the
signed content, the portions including a secu-
rity credentials associated with the content;

resource requirements for using the content;

verifying authenticity and integrity of the signed
content using at least the security credentials
and taking remedial actions when any of the
authenticity and the integrity are in doubt; and

controlling an operating system of the comput-
ing system so as ensure that use of the signed
content does not exceed the resource require-
ments and security credentials.

11. The method Claim 10 comprising one or both fur-
ther steps of:

extracting registration information from the sig-
nature and registering the signed content with
a provider by way of a communication channel
without further user intervention; and

extracting licensing terms from the signature
and controlling the operating system to ensure
that the use of the signed content conforms
with the licensing terms.

12. The method of Claim 10 or 11 comprising the fur-
ther steps of forming a data structure in a memory
of the computing system, the data structure includ-
ing a table of correspondence between users, the
security credentials and features of the signed con-
tent, and enforcing use of the signed content by the
users in accordance with the correspondence.

13. The method of Claim 10, 11 or 12 comprising the
further steps of tracking processes spawned from
the signed content and constraining operation of
the processes so as to conform to the resource
requirements and security credentials.

14. The method of any of Claims 10 to 13 wherein the
signed content includes at least one of an applica-
tion program and a document.

15. A method of controlling content usage in a comput-
ing system, comprising the steps of:

importing contents into the computing system,
the contents including computer readable
licensing terms;

extracting the computer readable licensing
terms from the imported contents; and

controlling operation of the computing system
so as to ensure that use of the signed content
conforms to the licensing terms.

16. The method of Claim 15 comprising the further
steps of extracting registration information from the
signature and automatically registering the signed
content with a provider by way of a communication
channel without further user intervention.

FIG. 1

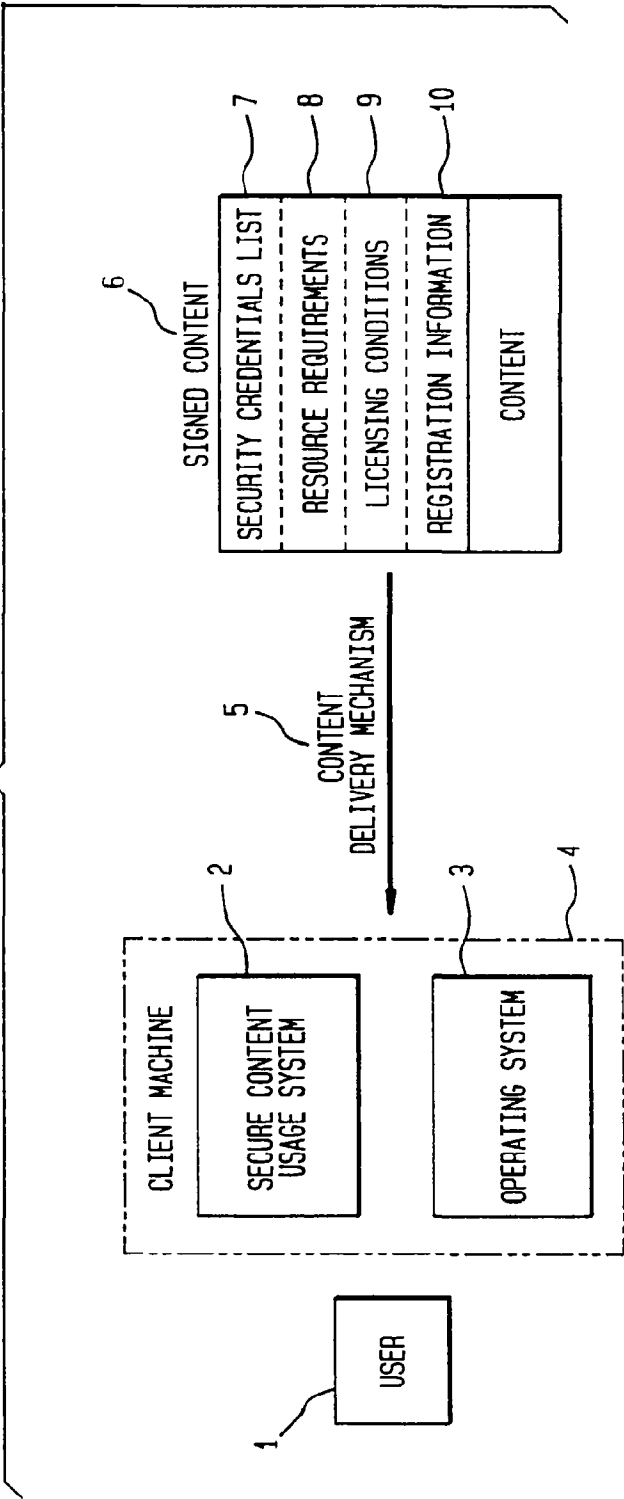


FIG. 2

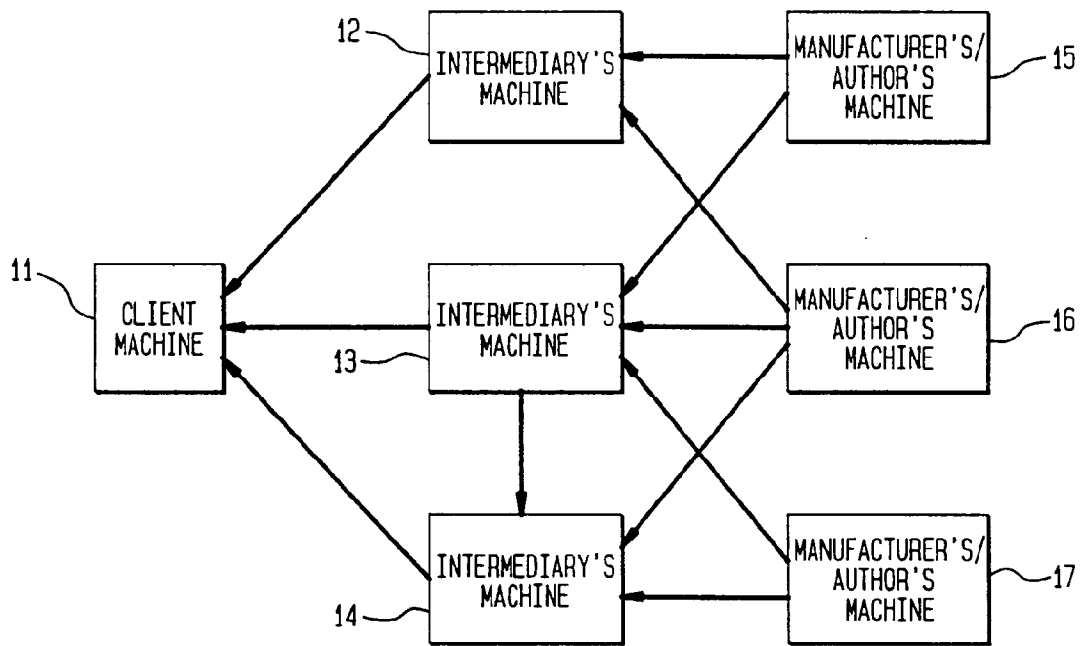


FIG. 3

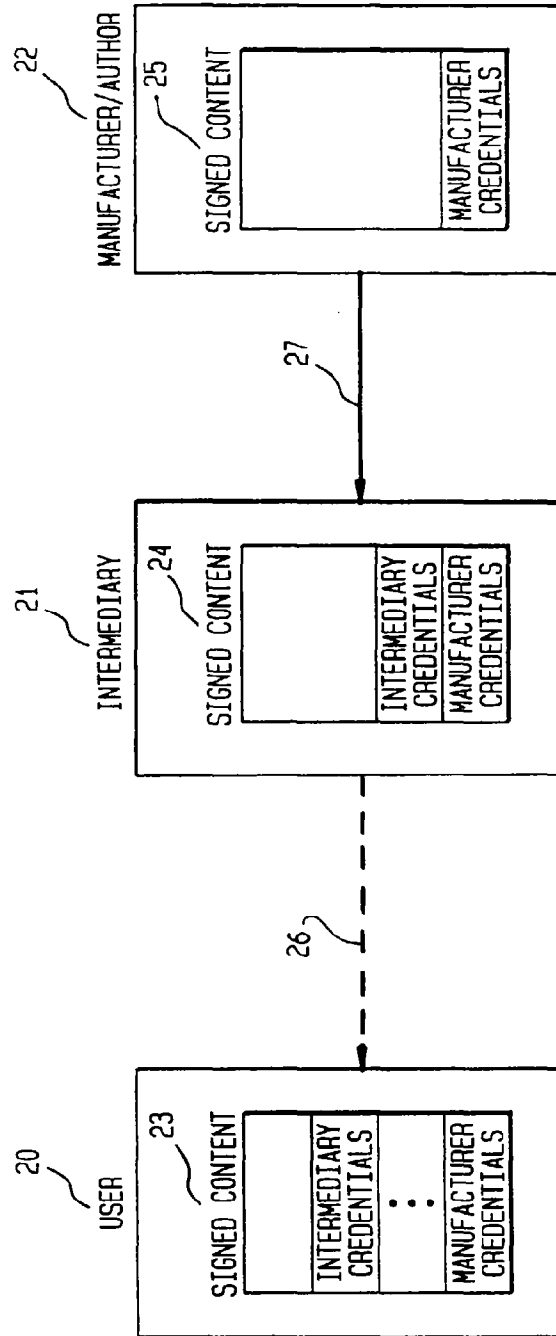


FIG. 4

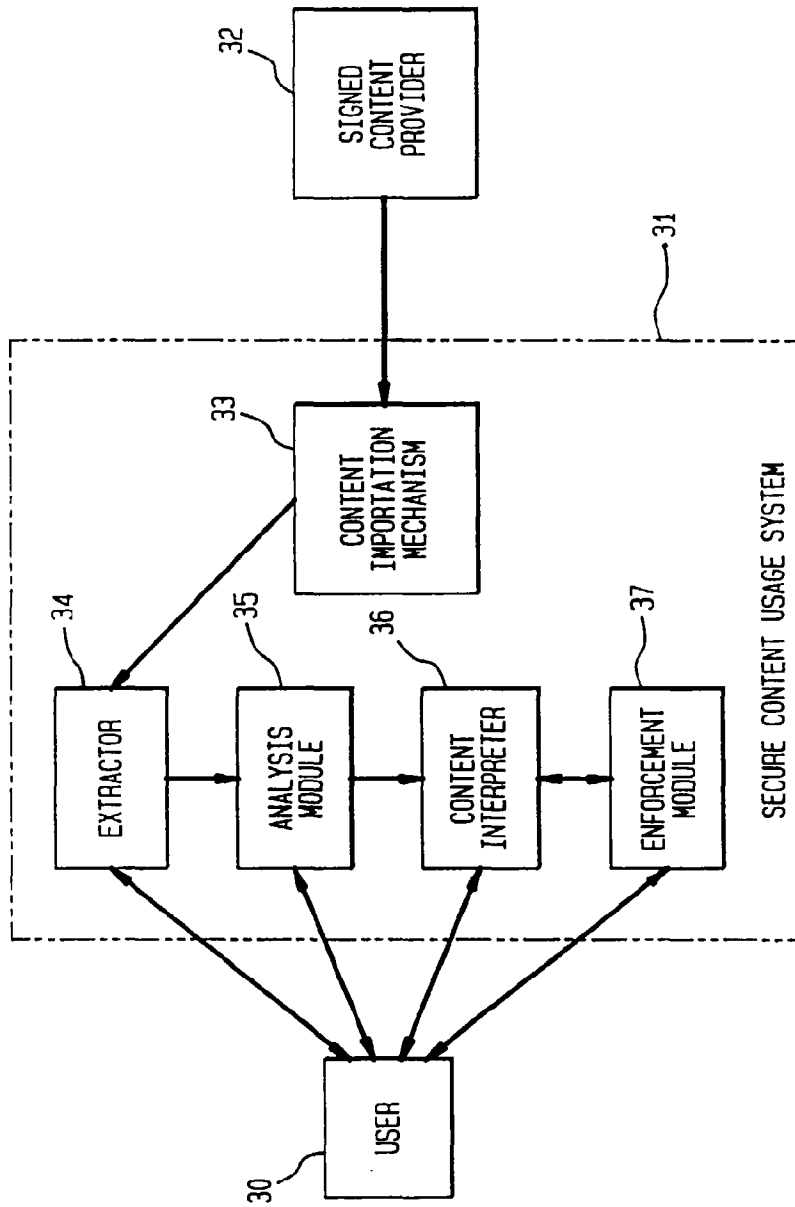


FIG. 5

ACCESS INFORMATION TABLE			
41	SECURITY CREDENTIALS	42	44
	RESOURCES LIMIT	RESOURCES CONSUMED	RESOURCES REQUESTED

FIG. 6

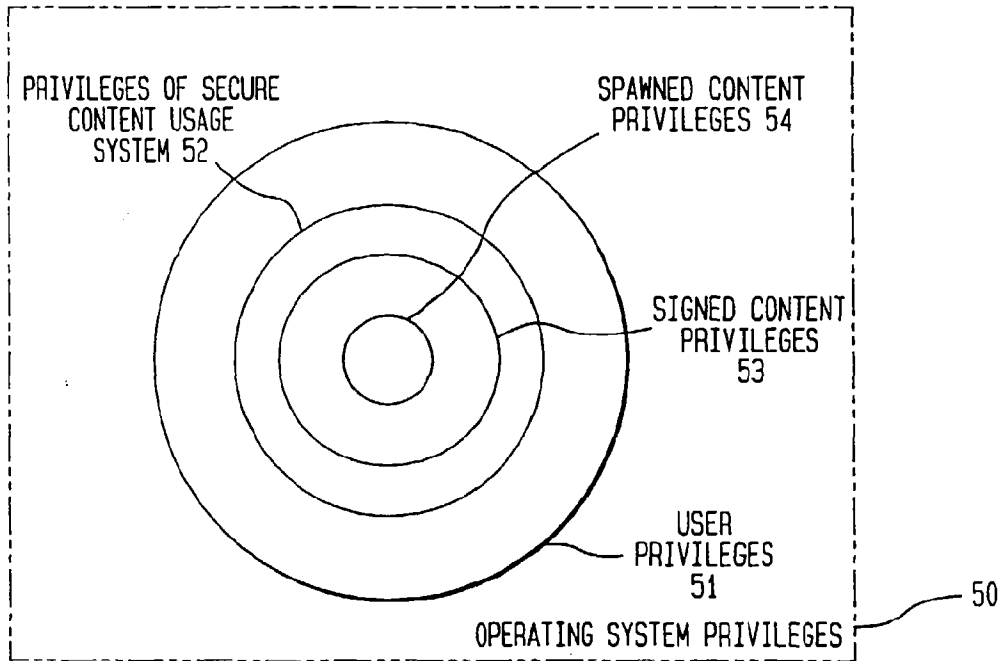


FIG. 7

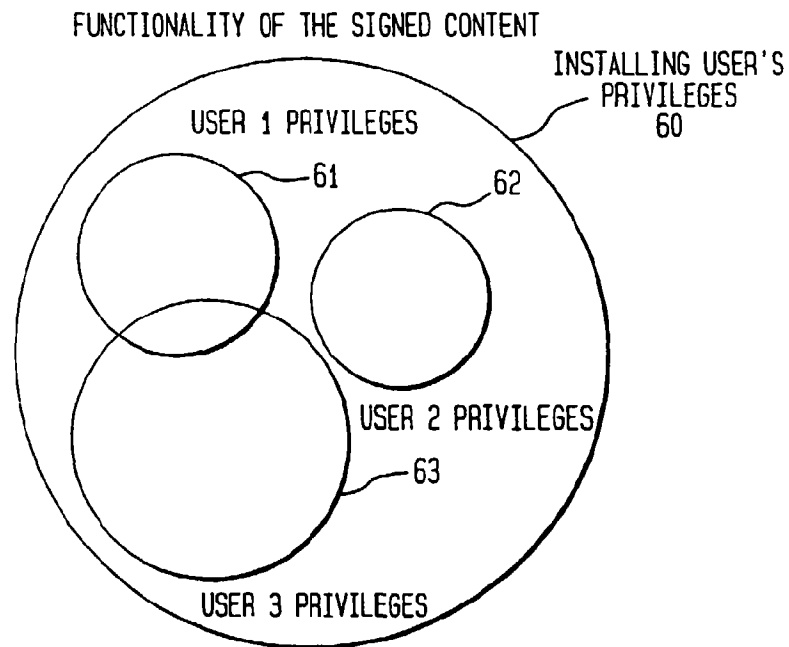


FIG. 8

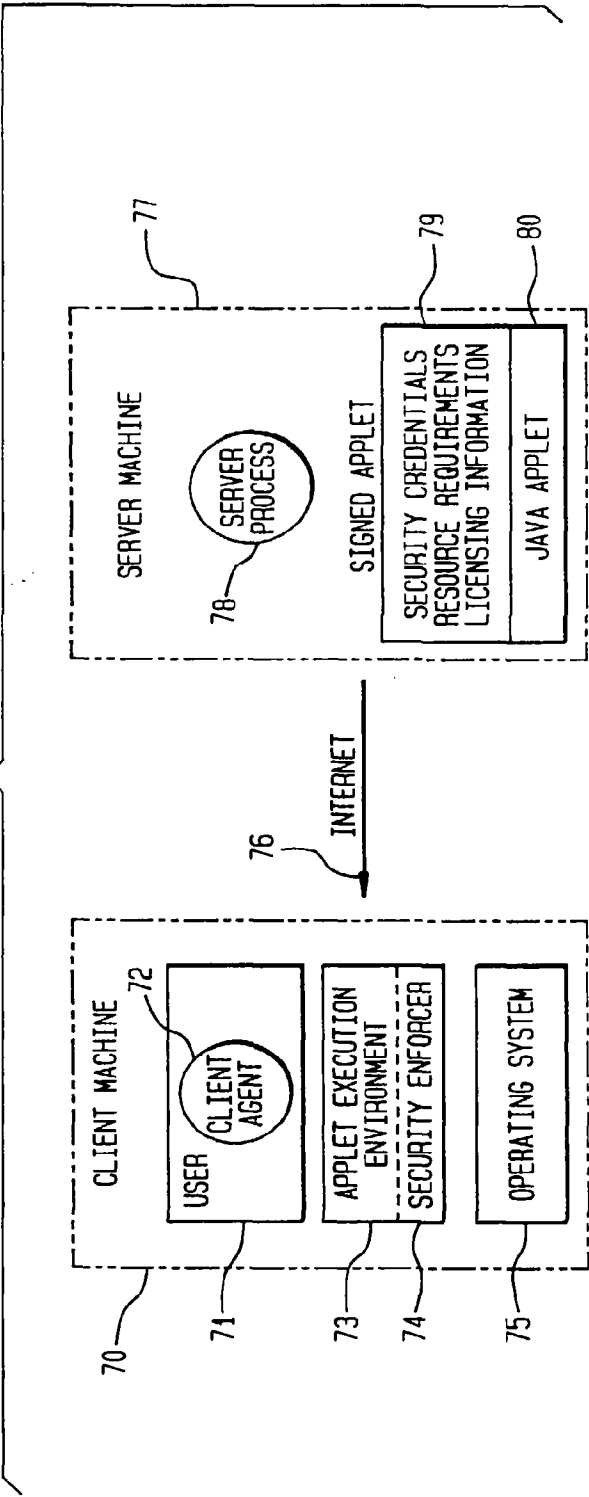


FIG. 9

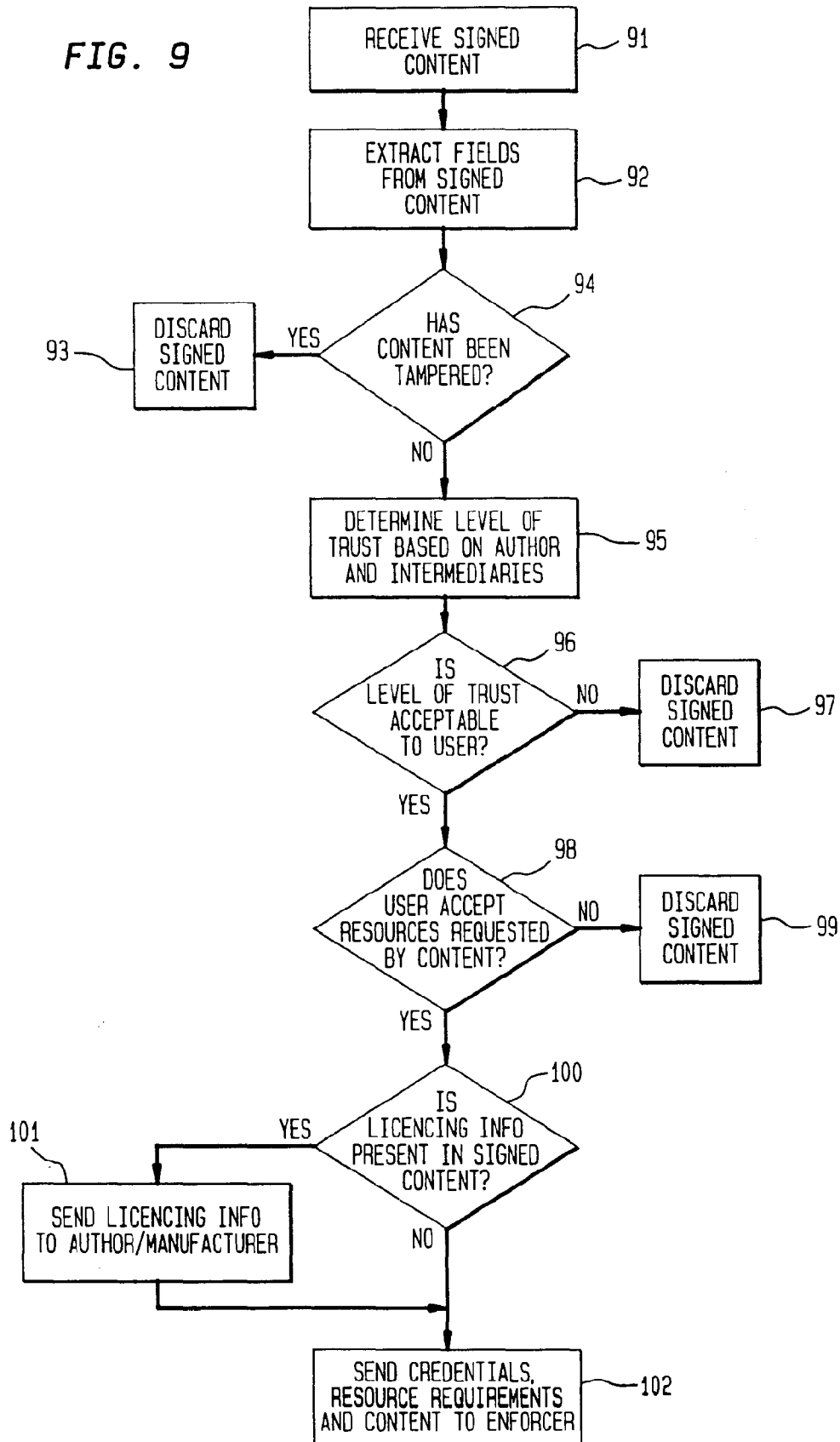
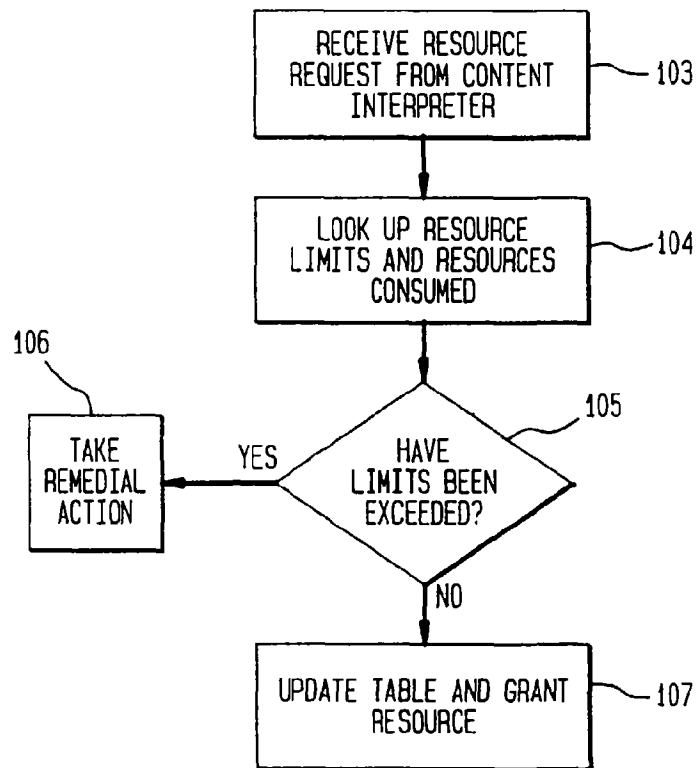


FIG. 10





INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶: B01D 53/04, 53/14, 53/48, 53/52, 53/54, 53/58	A1	(11) International Publication Number: WO 97/31698 (43) International Publication Date: 4 September 1997 (04.09.97)
(21) International Application Number: PCT/IB97/00162 (22) International Filing Date: 25 February 1997 (25.02.97) (30) Priority Data: 516/96 28 February 1996 (28.02.96) CH (71) Applicant (for all designated States except US): TRANSDIF-FUSIA S.A. [CH/CH]; 7, avenue Krieg, CH-1208 Geneva (CH). (72) Inventors; and (75) Inventors/Applicants (for US only): LAMARTINE, Roger [FR/FR]; 166, avenue Roger-Salengro, F-69100 Villeurbanne (FR). CHOQUARD, Philippe [CH/CH]; 23, chemin de la Gradelle, CH-1223 Cologny (CH). GORELOFF, Philippe [FR/FR]; 16, avenue Pasteur, F-01700 Saint-Maurice-de-Beynost (FR). MARTELIN, Yvon [FR/FR]; 8, chemin des Bottes, F-01700 Saint-Maurice-de-Beynost (FR). (74) Agent: MOHNHAUPT, Dietrich; IPTO S.A., 8, route de la Fonderie, CH-1705 Fribourg (CH).		(81) Designated States: AL, AM, AT, AU, AZ, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG). Published <i>With international search report.</i>

(54) Title: PROCESS FOR THE RECOVERY OF VOLATILE LOW MOLECULAR COMPOUNDS

(57) Abstract

A process for recovering volatile, generally bad smelling low molecular, low polar substances called VLC, contained in a gaseous environment, typically gas streams, by capturing same in contacting said gaseous environment with a calixarene or a resorcinol calixarene present in the form of particles, granules as a coating or impregnation on or in a solid support, or as an organic solution. Further, a capturing agent for these VLCs is disclosed, comprising at least one of the cyclic phenolic oligomers mentioned above and corresponding to formulae (I), (II), or (III) defined in the specification and claims.

FOR THE PURPOSES OF INFORMATION ONLY

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

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

- 1 -

Process for the recovery of volatile low molecular
compounds

BACKGROUND OF THE INVENTION

1. Field of the Invention

5 The present invention belongs to the technical field of gas phase absorption. More specifically, the invention refers to a process for the removal or recovery of gaseous, volatile organic or inorganic low molecular polar compounds, called "VLC" or "volatile light compounds" in this
10 technical field, that are contained in gaseous environments such as ambient air or gas streams.

 In the modern society, the environment suffers from various VLC sources. Industrial gaseous emissions greatly contribute to the pollution of the atmospheric environment.
15 Human activities linked to that pollution are the following: energy production, wood and paper industries, paints and polymers, agro-alimentary industries such as gist and sugar manufacture, etc. The olfactive nuisances generated by wastes (industry of animal by-products, animal excre-
20 ments, household garbage) will add to this pollution. Furthermore, sewage purification plants are sources of bad smells.

 There is a still growing interest to treat the nuisances generated by the industrial world, and especially
25 the elimination of bad odors has considerably increased during the last years in following the general requirements of a better quality of life.

- 2 -

There are already technical solutions available and corresponding necessary regulations set out regarding the partial or complete removal of odors. Thus, in France, Section 17 of the decree No. 77 of September 12, 1972, states that "the regulations consider in particular, on one hand, the effectiveness of available techniques and their economy and, on the other hand, the quality of the vocation and the use of environmental mediums". The regulation emerging from the laws and the decrees thus comprises the necessary tools for an adaptation to the evolution of anti-pollution techniques and their application.

2. Description of the Prior Art

Different deodorizing techniques are already known: odor masking, thermal oxidation with and without catalysts, active carbon adsorption, biological purification, gas-solid absorption combined with chemical washing, etc. Therefore, the treatment of odors is generally effected by washing and oxidation, by adsorption (on charcoal, natural adsorbents, synthetic adsorbents), or by biopurification.

It has already been suggested to use the so-called calixarenes (which will be described below) as adsorbents or absorbents for certain low molecular compounds. The document PCT/AU89/00079 for example discloses the removal of organic compounds, namely polar hydrocarbons such as trihalomethanes from fluids, in particular from drinking water. Furthermore, Japanese Patent Publication No. JP-3,249,941 (application No. JP-90-0'046'450) discloses a deodorant composition for garbage and toilets; this composition is the reaction product of p-tert-butylcalix[8]arene, cupric chloride and ascorbic acid. The calixarenes used are thus not the active deodorant matter but monitor or enhance the action of the acidic combination of cupric chloride and ascorbic acid.

- 3 -

The ability of calixarenes for absorbing non-polar, relatively low molecular substances has already been known. Thus, the following documents disclose this finding: U.K. Patent No. 2,292,156 (separation of alkanes from a hydrocarbon mixture); and WO 95 11,208 (separation of alkenes and aromatics from a hydrocarbon stream).

Taking into account the sometimes very high volatility of the odorous VLC and the very low olfactive perception level of most of these compounds, no method and technique whatsoever affords a satisfactory solution of the problem. The absorption of gaseous, bad smelling sulfur or nitrogen containing compounds from gaseous environments in using pure calixarenes is neither disclosed or suggested in the prior art.

15

SUMMARY OF THE INVENTION

The first and major objective of the present invention is to overcome the drawbacks of the solutions suggested until now and to provide a new and useful purification method, more precisely a process for the recovery and removal of VLC contained in a gaseous environment, typically in a gas stream, which uses a new class of solid capturing materials.

A further object of the present invention is to develop a method for the purification of gas streams, for example such emitted by water purification plants, which are contaminated with bad smelling light-weight gaseous components containing nitrogen or sulfur such as ammonia, hydrogen sulfides, amines, thiols and disulfides.

Still another object of this invention is to provide a new adsorbing or, more precisely, capturing material in

- 4 -

which the adsorption process is based on another principle than that of the processed used until now.

A further objective of this invention is to provide such an absorbing or capturing material which can easily
5 regenerated in a simple, low-cost manner.

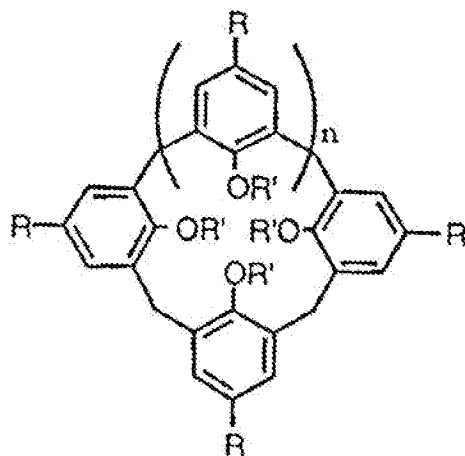
The process of the invention comprises contacting a gaseous mixture that contains VLC, with cyclic phenolic oligomers. The adsorbent of the invention contains, as an active constituent, cyclic phenolic oligomers.

10 The following listing gives examples of bad smelling organic and inorganic compounds of polar nature and having generally a low molecular weight that are preferably intended to be removed from a gaseous environment by the process and the adsorbent of the present invention: ammo-
15 nia, lower alkyl amines such as mono, di and trimethyl amine, mono, di and triethyl amine; and sulfur containing compounds such as hydrogen sulfide, lower organic thiols (mercaptans) such as methanethiol, ethanethiol, propane-
thiol, and sulfides such as dimethylsulfide and di-
20 ethylsulfide. This listing is not exhaustive.

All the compounds listed above have dipole polarizabilities (in units of 10^{-24} cm³) of from about 2.25 (ammonia) to 4.7 (methylamine) whereas the polarizabilities of the trihalomethanes described in the above cited PCT appli-
25 cation No. PCT/AU89/00079 range from 4.75 (chloroform) to 18.0 (triiodomethane).

The cyclic phenolic oligomers mentioned above that are used in the present invention are called "calixarenes"; they form a class of chemical compounds corresponding to
30 the following structural formula I:

- 5 -



(I)

wherein n is an integer from 1 to 7,

R is hydrogen, an organic radical, hydroxy, amino, carboxyl ($-\text{COOH}$), sulfo ($-\text{SO}_3\text{H}$) or a halogen, a halogen containing radical, or a nitrogen, sulfur or phosphorus containing radical, and

R' has one of the meanings defined for R in being identical to R or different from R , and R' may further be a radical $-\text{Si}(\text{CH}_3)_3$.

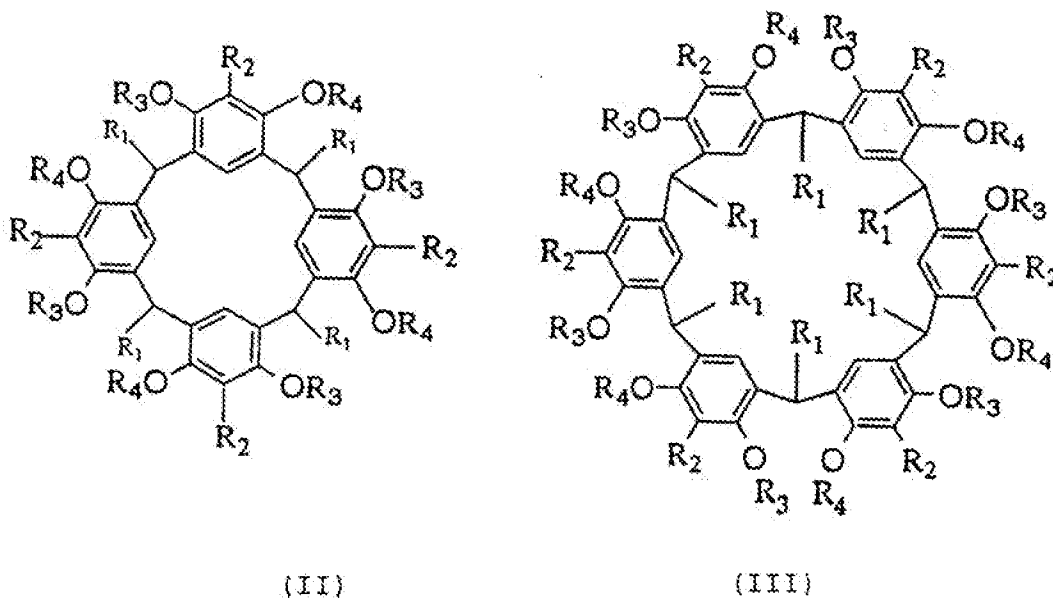
Such compounds of formula (I) are preferred wherein R is hydrogen, alkyl, substituted alkyl, allyl, aryl, substituted aryl, hydroxy, amino, carboxy, sulfo; R' is hydrogen, alkyl, substituted alkyl, allyl, aryl, substituted aryl, acyl such as acetyl, or $-\text{Si}(\text{CH}_3)_3$, and n is an integer from 1 to 5 included.

Still further preferred are calixarenes that correspond to formula (I) above wherein $n = 1, 3$ or 5 and R' is hydrogen, and particularly preferred are *p*-tert-butyl calixarenes.

- 6 -

The above mentioned calixarenes are already known compounds and may be obtained by simple reactions, generally one-pot reactions, starting from a corresponding p-substituted phenol wherein the two ortho positions of the phenolic hydroxy group are free, and formaldehyde. The reactions are generally effected in the presence of a suitable catalyst, known per se. If, in formula (I), R and/or R' are hydrogen, the produced calixarenes should be submitted to a dealkylation or, respectively, dearylation reaction.

The following resorcinol calixarenes of formulae (II) and (III) can also be used instead of or together with the calixarenes defined above:



wherein R₁, R₂, R₃ and R₄ are independently hydrogen or an organic radical.

It is preferred to use resorcinol calixarenes of formula II wherein R₁ is alkyl or aryl, or substituted alkyl or aryl, R₂, R₃ and R₄ are hydrogen, and n is 1. These resorcinol calixarenes are particularly useful for capturing

- 7 -

nitrogen containing compounds such as ammonia and amines. In the practice of the present invention, C-benzyl-resorcinolcalixarenes are preferably used.

5 These macrocyclic compounds which are solid at normal temperature, can be obtained by one-shot, low cost reactions, for example by a condensation reaction starting from an alcoholic resorcinol solution and an equivalent of an aldehyde R_1 -CHO in an acidic medium. Regarding the details of the preparation and isolation of calixarenes and resor-
10 cinol calixarenes, see, e.g., Gutsche et al., Org. Syn. 1989, 68, 234; Niederl and Vogel, Ber. 1940, 62, 2512; and Hogberg, J. Amer. Chem. Soc. 1980, 102, 6046.

15 In the practice of the present invention, "odd" calixarenes (odd number of rings in the molecule) may also be used as well as the normally available "even" ones. As to resorcinol calixarenes, C-benzylresorcinol calixarenes are preferred.

BRIEF DESCRIPTION OF THE DRAWING

In the attached drawing,

20 FIGURE 1 is a schematic view of an apparatus for determining the effectiveness of the process

FIGURE 2 represents in diagram form the results of a test according to Example 1, and

25 FIGURE 3 represents in diagram form the results of a test according to Example 2.

DETAILED DESCRIPTION OF THE INVENTION

- 8 -

It has now been found in a surprising manner that the above described macromolecules are capable of selectively capture odorous VLC's such as ammonia, light weight amines, sulfur containing VLC's, etc., such as listed above, from a gaseous medium that contains these VLC's when the latter is contacted or treated with the calixarenes and/or resorcinol calixarenes described above. As a general rule, a gas stream which contains VLC's is passed through a bed containing the mentioned macrocyclic material or materials. It appears that the retention of the VLC's does not preferentially occur by physical adsorption but by a capturing complexation mechanism. For example, when a low molecular amine or a thiol (mercaptan), thus compounds which are VLC's, are dissolved together with a calixarene or resorcinol calixarene of the above formulae, well defined supra-molecular species are obtained which are sometimes even crystallized and present the following molar ratios between VLC and calixarene:

VLC	4	6	8	4H	CER	CBR
MA	1	2.4	3.3	1	0.97	0.8
DM	1	2.2	2.6	1.7	1.1 to 2.4	2.3 to 4.3
TM					1.7	0.4
EA	1	3.2	6.3	1	2.4	1.1
DE	2.6	2.2	4.8	1.2	3.6	6.1
TE					2.1	2.4 to 4.3

In this Table, the following abbreviations have been used:

Head: 4: p-tert-butylcalix[4]arene
 6: p-tert-butylcalix[6]arene
 8: p-tert-butylcalix[8]arene

- 9 -

4H: calix[4]arene
CER: C-ethyl resorcinolcalixarene (n=1)
CBR: C-benzyl calixarene (n=1);

5 Left-hand column: MA: methylamine
 DM: dimethylamine
 TM: trimethylamine
 EA: ethylamine
 DE: diethylamine
 TE: triethylamine.

10 The behavior of the mentioned calixarenes, when brought into contact with a gaseous substrate, may be considered as a reference for any use in the practice of the present invention, namely in a process for the deodorizing of gaseous media by complexing capture.

15 In practice, it is preferred to use the calixarenes in the form of a deposit on the surface of solid substrates that are more or less finely divided or in the form of granules as it is the current use in gas absorption techniques. As a substrate or support, for example silica,
20 alumina, active coal or charcoal (where there is often a synergistic effect by combined physical adsorption and capture to be observed), sand, gravel, cellulose, synthetic resins such as polystyrene, etc. The one skilled in the art will immediately know and find the support the best
25 adapted to a certain task. It is possible to prepare a solution of the calixarenes and resorcinol calixarenes to be used in an organic solvent and to impregnate the support with this solution, and the material which is finally obtained by drying is ready for use.

30 The calixarene compounds to be used in the process of the present invention can generally be employed in the form they come from synthesis; it is normally not necessary to

- 10 -

purify them in any way. Of course, if these compounds contain impurities stemming from their preparation, their activity as capturing agents will be less than the 100 % pure compounds; this is a question of economy whether they will
5 be purified and to which extent.

In order to purify a gas stream containing the VLC's, it will be sufficient to pass the stream through a bed having appropriate dimensions, of the calixarene particles or the particles containing same, as it has been indicated
10 above. Typically, the gas stream is air contaminated by one or more VLC's. It is not required to treat the gas stream in any manner whatsoever; in particular, the gas stream need not be dried before. When the use is intended for deodorizing the gases coming from a water purification
15 plant, these exhaust gases are pushed by a fan through a column that contains a layer or a bed of a particulate support impregnated with calixarenes. However, it will also be possible to wash the gaseous medium, typically a gas stream, which contains the VLC's in contacting this gas
20 mixture with a homogeneous or heterogeneous liquid such as a solution or a dispersion, of the calixarenes or the resorcinol calixarenes concerned. The absorption technique of a gaseous component from a gas mixture by washing is in principle known; typically, a vertical column is used in
25 which the gas stream flows from the bottom to the top, and a liquid current in finely divided form flows countercurrently down from the top to the bottom. Of course, any washing installation whatsoever may be used such as bubble tray columns, sieve tray columns, packed columns, etc.

30 Typically, the process of the present invention, namely the capture of VLC's from a gaseous medium, is carried out at ambient temperature and pressure. However, if other gas parameters are pre-established, such as elevated pressure and/or temperature, these conditions do normally

- 11 -

not require any adaptation, i.e. their reduction to ambient conditions.

Reconditioning

5 It has surprisingly been found that, in spite of their polar nature, the captured VLC's can easily be separated from the capturing material. This fact allows to regenerate the absorbing material for repeated uses and has only be reported in the prior art for the case of non-polar captured compounds.

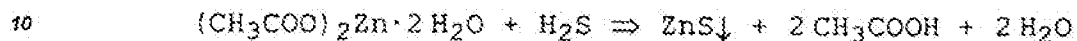
10 The interactions between the VLC's, called "substrate" in the following explanation paragraphs, and the calixarenes or resorcinol calixarenes occur in heterogeneous media, most frequently at a solid/gas (or sometimes liquid/gas) interface in the described cases. The solid phase having
15 captured and fixed the substrate may easily be recovered. Different methods may be used to separate the substrate from the solids loaded with the substrate and to recycle the recovered solids. All these methods are based upon the principle to offer to the substrates stronger bindings than
20 those effected by the calixarenes and the resorcinol calixarenes.

Regarding the nitrogen containing substrates, the treatment of the solids having captured the said substrates in a heterogeneous mixture of water and calixarenes allows
25 to extract the amines into the water; in order to speed up the dissociation of the amines from the calixarenes, the water temperature may be increased, acids may be added to the water, and/or ultrasonics may be applied.

Sulfur containing substrates such as hydrogen sulfide
30 and thiols or disulfides are only slightly soluble in water and highly volatile. It is therefore suggested to trans-

- 12 -

form the sulfur containing substrates in non volatile metallic sulfides. In the case of hydrogen sulfide, the solids containing this substrate may be treated with a metal salt solution, for example zinc acetate solution. The hydrogen sulfide is thus transformed into zinc sulfide which is insoluble and weighty. After filtration and drying the precipitate, the weight of the ZnS may be determined. This recycling method thus allows at the same time to quantitatively analyze the captured H₂S:



Finally, regarding the thiols or disulfides, the treatment of the solids having captured the said substrates with an aqueous solution of a heavy metal salt, for example copper acetate, is able to isolate the thiols or disulfides in the form of insoluble metal sulfides. However, electrochemical methods using silver electrodes may also be used, or the thiols or disulfides may be transformed to metal sulfides by means of an aqueous silver nitrate solution, or they may be oxidized into disulfides, e.g. by potassium ferricyanide.

The following Examples are given for illustrating purposes only and will not limit the invention in any way.

Example 1

A air stream that contains diethylamine is treated in the installation shown in FIG. 1 which schematically represents a system for the treatment of a gas flow. A peristaltic pump 10, driven by an adjustable electric motor 11 and connected through a conduit 13 to an injection device 12 having a skirt seal 15 for inserting an injection needle. The injection device 12 is connected by a conduit 17 to a tubular reactor 14 containing the calixarene or resor-

- 13 -

cinol calixarene, respectively, to be tested, enclosed between two frits 16 and 18. The circuit is closed by another conduit 19 leading from the reactor 14 to the peristaltic pump 10.

5 In the reactor 14, 159 mg of C-benzyl-resorcinol calixarene are placed. The pump 10 is put on, and $9.66 \cdot 10^{-5}$ moles of diethylamine are injected through the seal 15 into the system which totals a volume of 160 ml so that the air within the system contains about 1450 ppm of
10 diethylamine. The pump 10 is adjusted to a flow rate of 300 ml/min. Samples are taken in predetermined time intervals and quantitatively analyzed by gas chromatography according to the method of an inner n-pentane standard. This measuring method permits to follow the evolution of the gas
15 flow regarding the concentration in function of time. Blind tests are also carried out for testing the validity of the method.

FIG. 2 shows the results that have been obtained in diagram form. The amine content, given in arbitrary units,
20 is plotted on the abscissa against the test time in minutes on the ordinate. After 85 minutes, the amine concentration in the blind test (curve A) has dropped from 48 to 36 units (decrease by 25 %) whereas, when calixarene was used in the reactor (curve B), 50 % of the amine were already captured
25 and removed from the circuit after 20 minutes (from 1450 to 700 ppm). The degree of capture is 62 % after 85 minutes. The diagram further shows that there is no liberation of amine during the test from the capture system.

Example 2

30 An air stream that contains methylamine is treated in the installation of FIG. 1.

- 14 -

47 mg of C-benzylresorcinol calixarene are placed in the reactor 14. The pump 10 is put on, and $5.81 \cdot 10^{-5}$ moles of methylamine are injected into the device 12. The system contains thus about 8700 ppm of methylamine. The pump 10 is adjusted and controlled to a flow rate of 300 ml/min. Samples are taken in regular time intervals and analyzed by vapor phase chromatography against an internal standard. Blind tests are conducted in order to test the reliability of the method.

FIG. 3 shows the results that have been obtained. After 5 minutes, the concentration of the amine in the blind test is about 40 units whereas, when the calixarene is used, 90 % of the amine have been captured and removed from the circuit in passing from 8700 to 830 ppm. The degree of capture is 100 % after 10 minutes. The diagram shows that there is no subsequent liberation of amine in the capture system.

Example 3

An air stream that saturated with hydrogen sulfide is treated in the installation shown in FIG. 1.

140 mg of p-tert-Butylcalix[4]arene are placed in the reactor, and the air of the reactor is then saturated with H_2S . After having conducted the circulation as described above in Example 1, the atmosphere of the reactor is washed with some air free from H_2S .

The calixarene loaded with H_2S is collected and transferred into a solution of zinc acetate and mixed thoroughly. A white precipitate of ZnS is formed. The reaction medium is filtered, and the residue is treated with an appropriate organic solvent for dissolving and extracting the calixarene. The ZnS residue is then filtered and

- 15 -

dried, giving 19.6 mg of ZnS which corresponds to the capturing of 1 molecule of H_2S per molecule of calixarene.

If other calixarenes are used, the results are different. For example, when the p-tert-butylcalix[4]arene is substituted by p-isopropylcalix[4]arene, $\frac{1}{2}$ molecule of H_2S per molecule of calixarene is found, and p-allylcalix[4]arene shows a capturing capacity of 2 molecules of H_2S per molecule of calixarene.

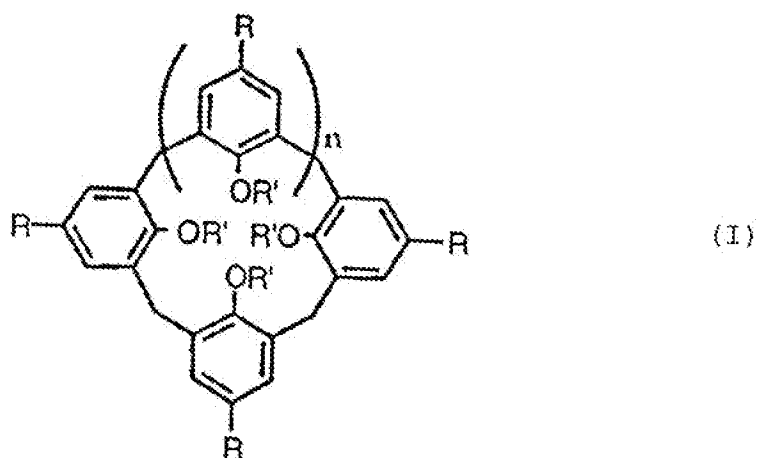
- - - - -

- 16 -

Claims

What is claimed is:

1. A process for the capture of volatile light compounds contained in a gaseous environment wherein the gaseous environment is contacted with a calixarene, the process comprising contacting a gaseous environment containing at least one bad smelling low molecular, low polar gaseous component (VLC) with at least one cyclic oligomer of a phenol, namely a calixarene, of the general formula I:



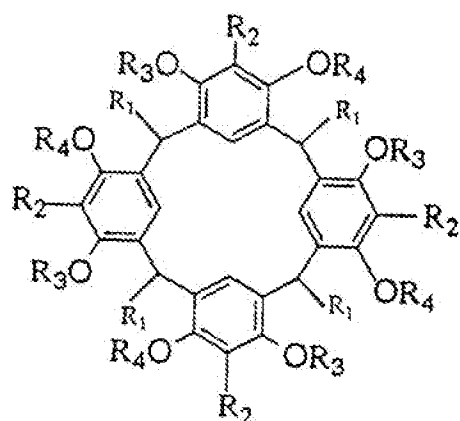
wherein n is an integer from 1 to 7,

R is hydrogen, an organic radical, hydroxy, amino, carboxy (-COOH), sulfo (-SO₃H) or halogen, a halogen containing radical, or a nitrogen, sulfur or phosphorus containing radical, and

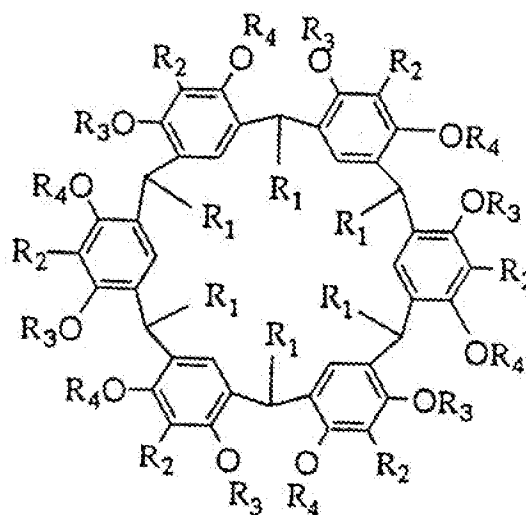
15

R' has one of the meanings defined for R in being identical to R or different from R, and R' may further be a radical -Si(CH₃)₃, or/and at least one resorcinol calixarene of formulae I or II:

- 17 -



(II)



(III)

wherein R_1 , R_2 , R_3 and R_4 are independently hydrogen or an organic radical.

2. The process of claim 1, wherein at least one calixarene of formula I is used wherein R is hydrogen, alkyl, substituted alkyl, allyl, aryl, substituted aryl, hydroxy, amino, carboxy, sulfo; R' is hydrogen, alkyl, substituted alkyl, allyl, aryl, substituted aryl, acyl such as acetyl, or $-\text{Si}(\text{CH}_3)_3$, and n is an integer from 1 to 5 included, or of formulae II or III wherein R_1 is substituted aryl and R_2 , R_3 and R_4 are hydrogen.

3. The process of claim 1, wherein a gas stream containing the said volatile substances is made to flow through a bed of solid bodies which are impregnated or coated with at least one compound of formulae I, II and/or III.

4. The process of claim 1, wherein a gas stream containing the volatile substances is washed with a liquid containing at least one of the compounds of formulae I, II and/or III in dissolved or dispersed state.

- 18 -

5. The process of claim 1, wherein the capture of the VLC's is effected at ambient pressure and temperature.

6. The process of claim 1, wherein said gaseous environment is air.

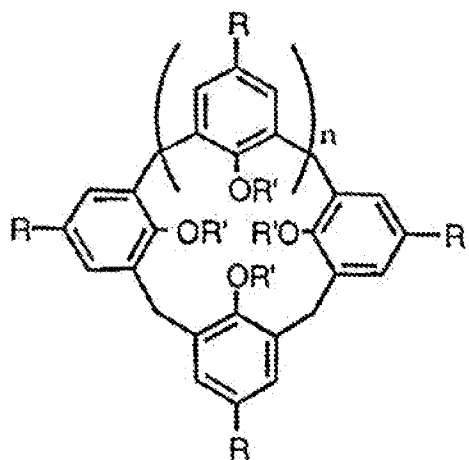
7. The process of claim 1, wherein said volatile substances comprise volatile amines, thiols, disulfides, hydrogen sulfide and ammonia.

8. The process of claim 1, wherein a gaseous industrial exhaust is deodorised.

9. The process of claim 8, wherein said industrial exhaust is an air stream issued by a waste water purification plant.

10. The process of any one of the preceding claims, wherein the calixarene compound(s) having captured the VLC's are regenerated for reuse.

11. A capturing agent for volatile organic compounds contained in a gaseous environment, containing or comprising at least one cyclic oligomer of a phenol, namely a calixarene, of the general formula I:



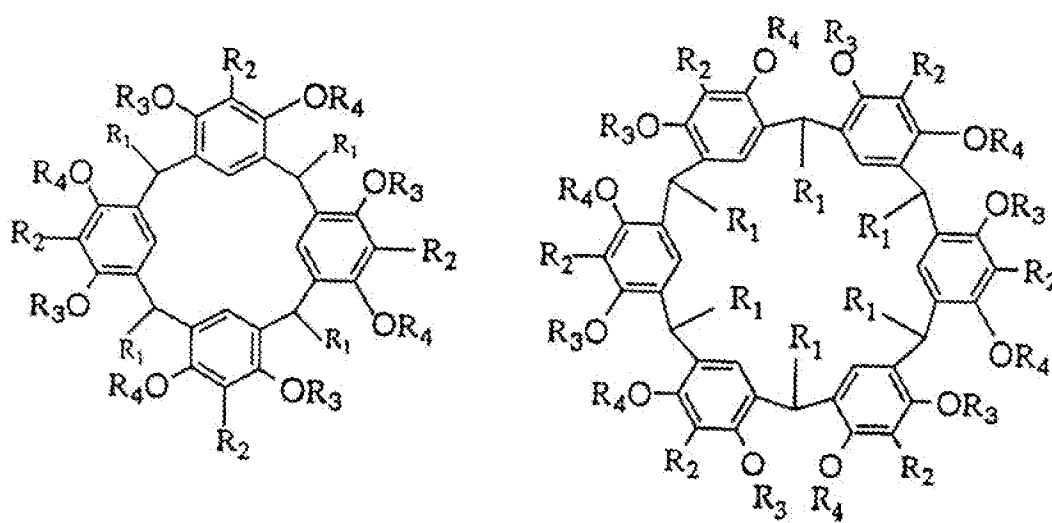
(I)

- 19 -

wherein n is an integer from 1 to 7,

R is hydrogen, an organic radical, hydroxy, amino, carboxy ($-\text{COOH}$), sulfo ($-\text{SO}_3\text{H}$) or a halogen, a halogen containing radical, or a nitrogen, sulfur or phosphorus containing radical, and

R' has one of the meanings defined for R in being identical to R or different from R , and R' may further be a radical $-\text{Si}(\text{CH}_3)_3$, or/and at least one resorcinol calixarene of formulae I or II:



(II)

(III)

wherein R_1 , R_2 , R_3 and R_4 are independently hydrogen or an organic radical.

12. The capturing agent of claim 11, containing or comprising at least one calixarene of formula I wherein R is hydrogen, alkyl, substituted alkyl, allyl, aryl, substituted aryl, hydroxy, amino, carboxy, sulfo; R' is hydrogen, alkyl, substituted alkyl, allyl, aryl, substituted aryl, acyl such as acetyl, or $-\text{Si}(\text{CH}_3)_3$, and n is an integer from 1 to 5 included, or of formulae II or III wherein R_1 is substituted aryl and R_2 , R_3 and R_4 are hydrogen.

- 20 -

13. The capturing agent of claim 11, constituting a bed of particulate solid bodies impregnated or coated with at least one compound of formulae I, II and/or III.

14. The capturing agent of claim 11, constituting a washing liquid containing at least one compound of formulae I, II and/or III.

- - - - -

1/2

FIG. 1

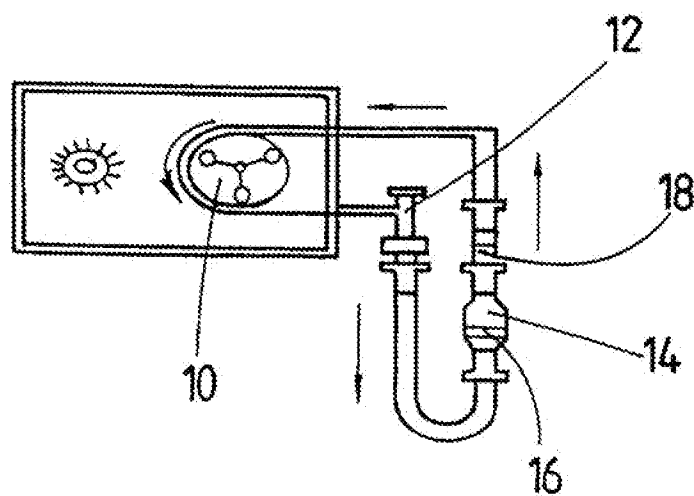
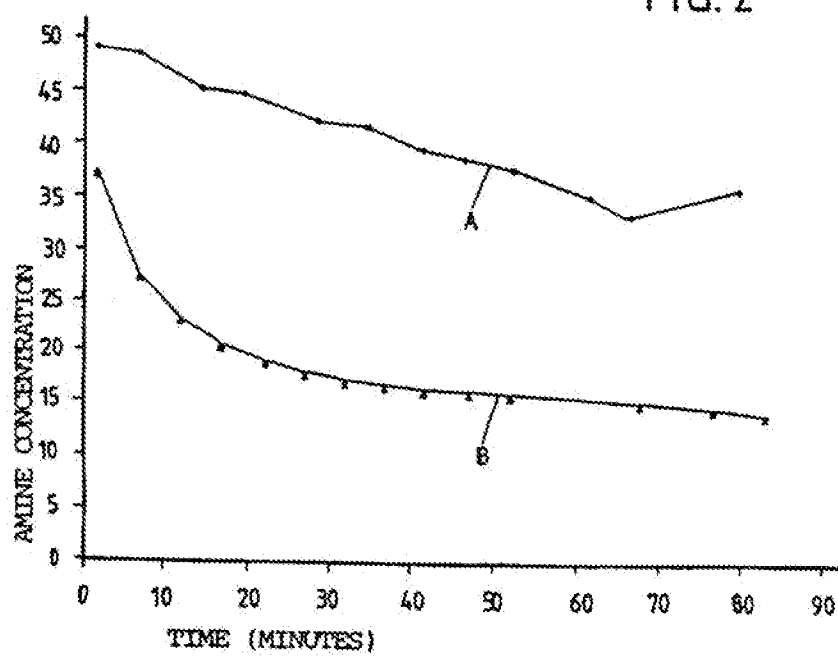


FIG. 2



2/2

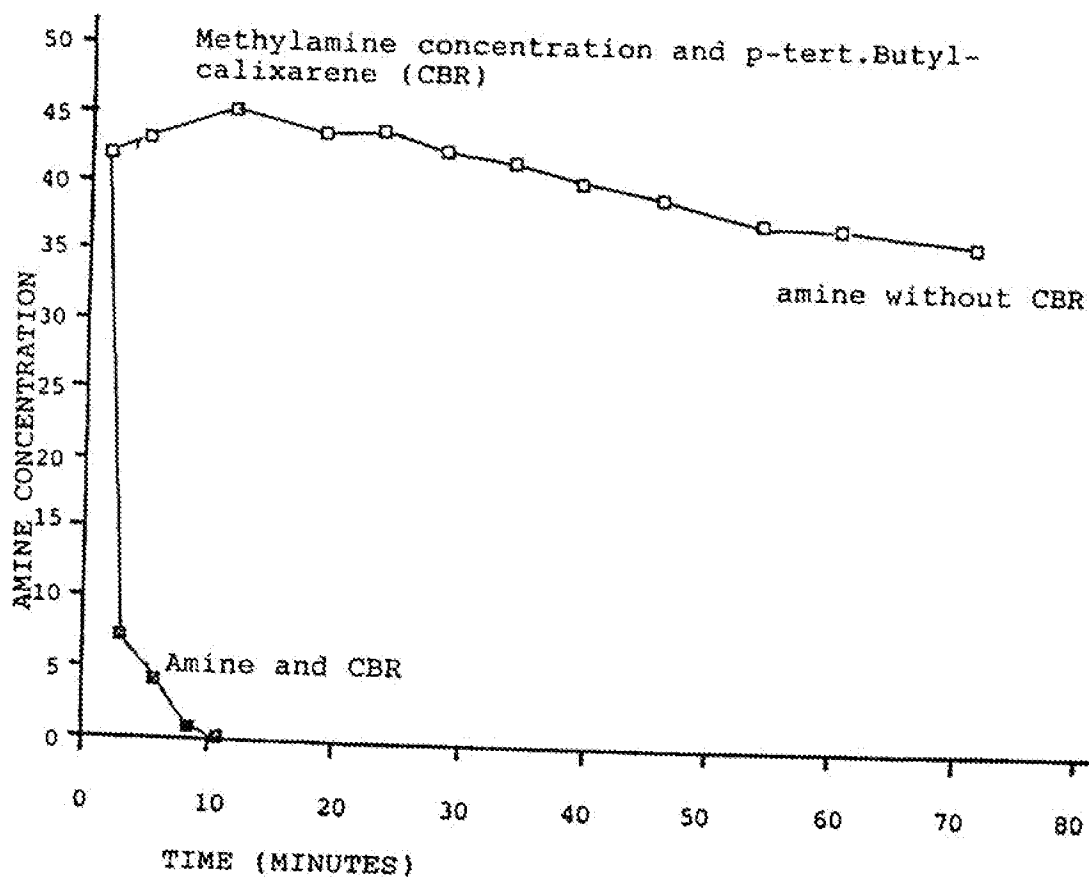


FIG. 3

INTERNATIONAL SEARCH REPORT

Int. Appl. No.
PCT/IB 97/00162

A. CLASSIFICATION OF SUBJECT MATTER

IPC 6 B01D53/04 B01D53/14 B01D53/48 B01D53/52 B01D53/54
B01D53/58

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 6 B01D

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

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	DATABASE WPI Section Ch, Week 9151 Derwent Publications Ltd., London, GB; Class D22, AN 91-372690 XP002016915 & JP 03 249 941 A (TOYO INK MFG KK) , 7 November 1991 see abstract	1-3,5-8, 11-13
X	WO 89 08092 A (UNIV SOUTHERN AUSTRALIA) 8 September 1989 see page 3, line 19 - page 4, line 30; claims 1-27; example 3	1-3,5,8, 9,11-13
X	GB 2 292 156 A (BRITISH PETROLEUM CO PLC) 14 February 1996 see page 2, line 9 - line 20; claims 1-5	1,2,4,5, 11,12
	-/-	

☒ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents:

- "A" document defining the general state of the art which is not considered to be of particular relevance
- "E" earlier document but published on or after the international filing date
- "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- "O" document referring to an oral disclosure, use, exhibition or other means
- "P" document published prior to the international filing date but later than the priority date claimed

- "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
- "&" document member of the same patent family

Date of the actual completion of the international search

24 April 1997

Date of mailing of the international search report

14.05.97

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl,
Fax (+ 31-70) 340-3016

Authorized officer

Eijkenboom, A

INTERNATIONAL SEARCH REPORT

International Application No.
PCT/18 97/00162

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 95 11449 A (SECR DEFENCE BRIT ;BEER PAUL (GB); SHADE MATTHEW (GB); CHEN ZHENG) 27 April 1995 see page 4, paragraph 4; claims 1-29; figures 2-13 ----	1,2,4-6, 11,12,14
X	WO 95 11208 A (BRITISH PETROLEUM CO PLC ;TENNISON STEPHEN ROBERT (GB); WEATHERHEA) 27 April 1995 see page 2, line 26 - page 3, line 3; claims 1-9 ----	1-3,5, 11-13
X	DATABASE WPI Section Ch, Week 9025 Derwent Publications Ltd., London, GB; Class A41, AN 90-189711 XP002016916 & JP 02 124 850 A (WAKAYAMA PREFECTURE) , 14 May 1990 see abstract ----	1,2,11, 12
A	WO 94 04483 A (HARRIS STEPHEN JOHN ;DIAMOND DERMOT (IE); MCKERVEY MICHAEL ANTHONY) 3 March 1994 see page 25, line 1 - page 28, line 2; claims 1-13; example 9 -----	1-3,5,7, 11

INTERNATIONAL SEARCH REPORT

information on patent family members

International Application No

PCT/IB 97/00162

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 8908092 A	08-09-89	NONE	
GB 2292156 A	14-02-96	NONE	
WO 9511449 A	27-04-95	AU 7942594 A	08-05-95
		CA 2174521 A	27-04-95
		CN 1136845 A	27-11-96
		EP 0724721 A	07-08-96
		GB 2299173 A	25-09-96
WO 9511208 A	27-04-95	NONE	
WO 9404483 A	03-03-94	AU 4581893 A	15-03-94
		US 5599913 A	04-02-97



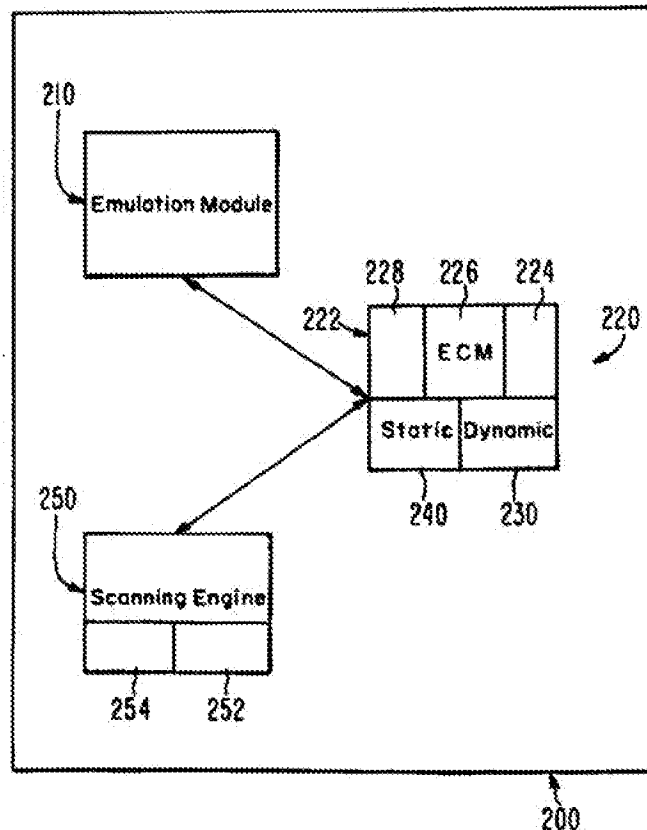
INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : G06F 11/00	A1	(11) International Publication Number: WO 97/12322
		(43) International Publication Date: 3 April 1997 (03.04.97)
(21) International Application Number: PCT/US96/15512 (22) International Filing Date: 27 September 1996 (27.09.96) (30) Priority Data: 08/535,340 28 September 1995 (28.09.95) US (71) Applicant: SYMANTEC CORPORATION [US/US]; 10201 Torre Avenue, Cupertino, CA 95014 (US). (72) Inventor: NACHENBERG, Carey; 19533 Citronia Street, Northridge, CA 91324 (US). (74) Agents: RADLO, Edward, J. et al.; Fenwick & West L.L.P., Suite 700, Two Palo Alto Square, Palo Alto, CA 94306 (US).		(81) Designated States: AM, AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, JP, KE, KG, KP, KR, KZ, LK, LR, LT, LU, LV, MD, MG, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SI, SK, TJ, TT, UA, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BI, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG). Published <i>With international search report. Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i>

(54) Title: POLYMORPHIC VIRUS DETECTION MODULE

(57) Abstract

A Polymorphic Anti-virus Module (PAM) (200) comprises a CPU emulator (210) for emulating the target program, a virus signature scanning module (250) for scanning decrypted virus code, and an emulation control module (220), including a static exclusion module (230), a dynamic exclusion module (240), instruction/interrupt usage profiles (224) for the mutation engines (162) of the known polymorphic viruses (150), size and target file types (226) for these viruses, and a table (228) having an entry for each known polymorphic virus (150). Prior to emulation, the static exclusion module (230) examines the gross characteristics of the target file for attributes that are inconsistent with the size/type data (226), and excludes polymorphic viruses (150) from the list (228) accordingly. During emulation, the dynamic exclusion module (240) compares fetched instructions with the instruction/interrupt usage profiles (224) to determine when emulation has proceeded to a point where at least some code from the decrypted static virus body (160) may be scanned for virus signatures.



FOR THE PURPOSES OF INFORMATION ONLY

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

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

POLYMORPHIC VIRUS DETECTION MODULE

Background of the Invention

Technical Field This invention relates to the field of computer viruses, and in particular to methods and systems for detecting polymorphic viruses.

Background Art Polymorphic viruses are a type of computer virus designed to evade detection by infecting each new file with a mutated version of the virus. By providing each newly infected file with viral code having a different appearance, polymorphic viruses frustrate most standard virus-detection schemes, which rely on some type of string scanning to identify computer viruses in a file.

Polymorphic viruses comprise a static virus body and a mutation engine. In the most common polymorphic viruses, the virus does not mutate. Rather, the mutation engine generates a virus decryption routine (polymorphic decryption loop) and uses the dual of this routine to encrypt the static virus body and the mutation engine. The new decryption routine and the newly encrypted virus body are then inserted into the host file. Common mutation strategies employed by the mutation engine include reordering of instructions, substituting equivalent instructions or equivalent sequences of instructions, inserting random "garbage" instructions (which have no effect on the virus functionality), interchanging function calls, in-line code, JMP instructions, and the like, and using equivalent registers interchangeably.

Thus far, the most successful technique for detecting polymorphic viruses has been cue-directed program emulation (CDPE). CDPE methods assumes that the polymorphic code contains at least one section of machine code, the static viral body, that is consistent from generation to generation. CDPE methods also assume that when executed the decryption routine of the polymorphic virus deterministically decrypts the encrypted static virus body and transfers control to the static virus body when decryption is complete. The strategy employed by CDPE methods is to emulate the polymorphic virus until it has decrypted itself and then analyze the decrypted virus body using standard scanning techniques.

CDPE virus detection systems comprise a scanner module, a CPU emulator (80x86), a set of virus signatures, and an emulation control module. The scanner module locates a file's entry point and the CPU emulator performs a limited emulation of the file's machine code under control of the emulation control module. Emulation proceeds until the emulation control module believes either that the virus is fully decrypted or that the file is not infected with a virus, at which point string scanning for virus signatures commences.

The CDPE emulation control module examines each emulated instruction with the aid of certain heuristics to determine whether the instructions being emulated are likely to be part of a polymorphic decryption loop or a normal program. For example, certain sequences of instructions are frequently found in polymorphic decryption loops. These instruction sequences are referred to as "boosters" since they indicate to the emulation control module that it is seeing a potential decryption loop and should continue emulating instructions. Other sequences of

instructions are rarely found in decryption loops. These instruction sequences are referred to as "stoppers" since they indicate to the emulation control module that the instructions are probably not from a virus decryption loop. Stoppers may be present if the host file is not infected or if the emulation has fully decrypted the static virus body. In the latter case, the static virus body, like
5 any other program, may use any instructions supported by the processor architecture. In addition, stoppers may be present if a virus designer has included them in a decryption loop to foil CDPE detection methods.

CDPE based methods employ additional heuristics to determine what the detection of various stoppers and boosters indicates about the code being emulated. For example, if a number
10 of stoppers have been found prior to the detection of any boosters, the emulation control module will likely decide that the host file is uninfected. On the other hand, if one or more stoppers are detected following detection of a number of boosters, the emulation control module will likely decide that the polymorphic loop has been fully decrypted to reveal the static virus body. In this case, virus scanning will proceed.

The selection of boosters and stoppers included in the emulation control module can have a substantial impact on the speed and accuracy with which the CDPE system detects viruses. Ideally, stoppers and boosters are selected to work accurately for all known polymorphic viruses. However, it may not be possible to find a set of such heuristics that does not significantly slow virus scanning. Stoppers and boosters useful for detecting several polymorphic viruses may
20 actually prevent the detection of other polymorphic viruses, as for example, where a virus writer includes a standard stopper in polymorphic loop code to confuse CDPE modules. In general, any change in the stoppers or boosters used must be accompanied by extensive regression testing to insure that previously detected viruses are not missed using the new heuristics. Since new polymorphic viruses are continually being developed, the time consuming and awkward selection and regression testing of new combinations of stoppers and boosters can not be avoided.
25

Thus, there is a need for polymorphic virus detection systems that can be readily expanded to cover newly discovered viruses, without need for extensive regression testing and modification of the heuristics of the emulation control module. In addition, the system should be able to provide accurate results without emulating unnecessarily large numbers of instructions.
30

Summary of the Invention

The present invention is a polymorphic anti-virus module or PAM (200) for detecting polymorphic viruses (150) using mutation-engine specific information for each known polymorphic virus rather than heuristic stopper and booster code sequences. The PAM system
35 (200) comprises a CPU emulator (210) for emulating the target program, a virus signature scanning module (250) for scanning decrypted virus code, and an emulation control module (220), including a static exclusion module (230) and a dynamic exclusion module (240), for determining how long each target file is emulated before it is scanned. The emulation control module (220) also includes data (222) specific to each known polymorphic virus (150) and

organized in a format that facilitates comparison with target files being tested for infection. This data (222) includes instruction/interrupt usage profiles (224) for the mutation engines (162) of the known polymorphic viruses (150), as well as size and target file types (226) for these viruses. The emulation control module (220) also includes a table (228) having an entry for each known polymorphic virus (150) which can be flagged when characteristics inconsistent with the polymorphic virus are detected.

In accordance with the present invention, the static exclusion module (230) examines the gross characteristics of the target file for attributes that are inconsistent with the mutation engine specific data for known polymorphic viruses (150). These characteristics are the type of target file, the size of the target file's load image, the presence of certain instructions at the file entry point, and the distance between the file entry point and the end of the load image. The last characteristic is useful because most viruses append themselves to the files they infect. In some cases, the static exclusion module (230) allows certain target files to be identified as infected without any emulation.

The dynamic exclusion module (240) examines the instruction/interrupt usage profiles (224) of each known polymorphic virus (150) as each instruction is fetched for emulation. The instruction/interrupt usage profiles (224) indicate which polymorphic viruses (150) employ mutation engines that do not use the fetched instruction in decryption loops they generate, and the emulation control module (220) flags these viruses. The emulation control module (220) continues until all mutation engines have been flagged or until a threshold number of instructions have been emulated. The flagging technique implemented by the dynamic exclusion module (240) determines when emulation has proceeded to a point where at least some code from the decrypted static virus body (160) may be scanned and substantially reduces the number of instructions emulated prior to scanning the remaining target files without resort to booster or stopper heuristics.

It is not always necessary to fully decrypt the static virus body (160) to identify the underlying virus. In the preferred embodiment of the invention, the emulation control module (220) tracks those parts of virtual memory modified during emulation and periodically interrupts the emulation process to call the scanning module (250). The scanning module (250) tries to identify the virus type from the portion of decrypted static virus code (160). In order to speed up the process, the scanning module (250) implements a coarse scan of tagged memory locations to identify data bytes most likely to be associated with decrypted static virus code (virus signatures). It implements a more detailed binary search process only when selected bytes are encountered during the coarse scan. This approach greatly speeds up scanning without decreasing the accuracy of the scanning module (250). When code matching one of the viral signatures is identified, the PAM system (200) signals to the host computer that an infected file has been located.

Brief Description of the Drawings

Figs. 1A-1C are schematic representations of the load images of an uninfected computer file, a file infected by a virus, and a file infected by a polymorphic virus, respectively.

Fig. 2 is a block diagram of a Polymorphic Anti-virus Module (PAM) in accordance with
5 the present invention.

Fig. 3 is an example of an instruction/interrupt usage profile employed in the emulation control module (220) of the present invention.

Fig. 4A is a flowchart of the emulation process implemented by an emulation control module (220) in accordance with the present invention.

Fig. 4B is a flowchart of the scanning process implemented by a scanning module (250) in
10 accordance with the present invention.

Detailed Description of the Preferred Embodiments

Computer viruses infect a variety of files in a number of different ways. In the DOS
15 environment, computer viruses have been used to infect three different types of executable files: COM files, SYS files, and EXE files. A common feature of these files is that at some point after loading, control of the computer is passed to the program code stored in the file. Computer viruses infect these executable files by attaching themselves to the file and modifying the machine language at the file entry point to transfer control to the virus rather than to the
20 executable file. In order to camouflage their presence, computer viruses typically return control to the infected executable file once they have run.

Referring to Figs. 1A and 1B, there are shown executable images 100, 100' of an EXE file before and after infection, respectively, by a virus 130. Infection modes for EXE, COM, and SYS files are discussed in greater detail in Nachenberg, *A New Technique For Detecting*
25 *Polymorphic Computer Viruses*, Master Thesis, University of California at Los Angeles (1995), which is hereby incorporated by reference. Executable image 100 comprises a header 110 and program code 120. Header 110 includes a signature (MZ), size field, a code segment field CS, an instruction pointer field IP, a stack segment field SS, and a stack pointer field SP. MZ indicates the file type (EXE in this case) and the following field specifies the size of executable image 100. CS and IP specify an entry point 122 of program code 120, and SS and SP point to the end of
30 program code 120, where a stack (not shown) may be generated.

Upon infection by computer virus 130, header 110' of executable image 100' is modified so that size field equals the size of executable image 100 incremented by the size of computer virus 130. In addition, computer virus 130 has replaced CS, IP of image 100 with CS', IP' in
35 image 100'. CS', IP' point to an entry point 132 of virus 130 rather than entry point 122 of program code 120. Similarly, computer virus 130 has replaced SS, SP of image 100 with SS', SP' in image 100', which point to the end of virus 130. In order to return control of the computer to program code 120 following execution of virus 130, CS, IP, SS, and SP of uninfected image 100 are retained by virus 130.

Computer viruses 130 which are added to EXE, COM, or SYS files in the manner of Fig. 1B are relatively easy to detect. A virus detection program need only scan executable image 100' for code segments associated with known viruses 130. These code segments, known as virus signatures, are code segments unique to different viruses, and their presence in an executable image 100' is taken as a clear indication that the corresponding file has been infected. A number of methods are available for scanning executable images 100' for virus signatures. Different viruses 130 may implement a number of strategies to hide their presence in an executable image 100'.

One of the most successful strategies is that implemented by polymorphic viruses, which include a mutation engine that encrypts a static virus body according to a different (mutated) encryption key with each new infection. The encrypted virus is appended to the image with a mutated decryption routine, which decrypts the encrypted virus to reveal the static virus body only when the file is executed. The new appearance presented by such polymorphic viruses on each infection frustrate those detection methods which would simply scan images 100' for viral signatures.

Referring now to Fig. 1C, there is shown an executable image 100" infected by a polymorphic virus 150. Polymorphic virus 150 comprises a static virus body 160 including a mutation engine 162, both of which are shown hatched in the Fig. 1C to indicate their encrypted state. On infection, mutation engine 162 generates a variable encryption routine (not shown) that encrypts static virus body 160 (including mutation engine 162) to prevent detection of polymorphic virus 150 by conventional scanning techniques. A decryption routine 164, which is the dual of the encryption routine, is prepended to encrypted static virus body 160. When image 100" is executed, a decryption routine 164 decrypts and passes control to static virus body 160, which employs the CPU of the host computer to attach itself to other files and implement whatever mischief its designer intends.

One anti-virus detection scheme designed specifically for polymorphic viruses 150 is Cue Directed Program Emulation (CDPE). CDPE operates by emulating a target image 100" or file for a sufficient number of instructions to allow a polymorphic virus 150 present in image 100" to decrypt itself and reveal its static virus body 160. Static virus body 160 is then scanned to identify the type of virus present and steps are taken to deactivate it. As noted above, CDPE methods rely on the detection of prescribed stopper and booster code segments as a target file is emulated in order to determine whether an encrypted static virus body 160' is present, and if so, to determine whether it has been at least partially decrypted. The problem with this approach is that stopper and booster segments must be selected carefully and thoroughly tested in order to detect viruses accurately and completely. Stoppers and boosters added to detect new viruses require thorough regression testing to insure that they do not interfere with the detection of other polymorphic viruses. In short, stoppers and boosters that will work effectively with all polymorphic viruses must be identified, even as virus designers use these same stopper and booster heuristics to better camouflage their creations.

In addition to the problems posed by any changes in the stopper, booster heuristics, CDPE emulation is done by virtual machines in order to isolate potentially infected files from the actual CPU and memory of the host computer. These virtual machines tend to operate slowly relative to the actual CPUs, and since each file must be checked, virus checking can be a very
5 time consuming process. The speed of these programs is slowed further as more complicated heuristics are developed to detect polymorphic viruses.

Referring now to Fig. 2, there is shown a block diagram of a polymorphic anti-virus module (PAM) 200 in accordance with the present invention. PAM 200 comprises an emulation module 210, an emulation control module 220, and a scanning module 250. As in CDPE
10 systems, emulation module 210 allows PAM 200 to emulate a target file without allowing the target file to interact with either the actual CPU or memory of the host computer. Scanning module 250 includes virus signatures 252 for identifying polymorphic viruses 150 and a scanning engine 254 for efficiently searching decrypted virus code for these signatures. The scanning engine 254 is discussed in greater detail below in conjunction with Fig. 4B.

Emulation control module 220 comprises virus profile data 222, a static exclusion module 230, and a dynamic exclusion module 240, which combine to substantially reduce the number of file instructions that must be emulated in order to determine whether a target file is infected by a virus. Virus profile data 222 comprises an instruction/interrupt usage profile 224
15 for each known polymorphic virus 150 as well as data on the sizes of known polymorphic viruses 150 and type of target files infected by each (size/type data 226). Size/Type data 226 is accessed by static exclusion module 230 prior to emulation to eliminate certain polymorphic viruses 150 from consideration, and instruction/interrupt usage profiles 224 are accessed by dynamic
20 exclusion module 240 during emulation to determine whether the emulated code may be part of a virus decryption loop. Emulation control module 220 also includes a table 228 of all known
25 polymorphic viruses 150 which is initialized with all viruses 150 flagged. As each virus 150 is eliminated from consideration by static or dynamic exclusion modules 230, 240, respectively, the corresponding flags are reset to preclude further consideration of the virus.

For example, gross features of executable image 100 that are inconsistent with various polymorphic viruses 150 allow the static exclusion module 230 to rule out infection of a target
30 file 100 by these polymorphic viruses before any emulation is done. If features inconsistent with a polymorphic virus 150 are detected in target file 100, the associated flag in table 228 is reset and it is excluded from further consideration during the subsequent emulation phase. If the gross features of target file 100 are inconsistent with infection by all known polymorphic virus 150, no emulation is required to determine that target file 100 is uninfected, and the next target file may
35 be considered. More often, analysis by static exclusion module 230 allows only some of polymorphic viruses 150 to be excluded from further consideration.

Data on polymorphic viruses 150 considered by static exclusion module 230 are: (1) the type of target file each known polymorphic virus 150 is designed to attack; (2) the minimum size of the load image of each polymorphic virus 150; (3) whether a polymorphic virus 150 uses a

JMP instruction as the first instruction of in target COM file; and (4) the maximum size of the load image of each polymorphic virus 150. In order to take advantage of this data, static exclusion module 230 determines: the target file type and load image size for a target file being analyzed. In addition, if a COM target file is being analyzed, static exclusion module 230
5 determines its first instruction, and if an EXE target file is being analyzed, static exclusion module 230 determines the distance between the entry point and end of the load image. These gross characteristics and their relationship to features of known polymorphic viruses 150 are considered below.

Type of Executable File Targeted Different viruses infect different executable file
10 formats. Some infect only COM files, some infect only EXE files, and some infect both COM and EXE files. Very few viruses infect SYS files but some of these can infect EXE or COM files as well. Consequently, if target file is an EXE file, all polymorphic viruses 150 that attack only COM files or SYS files may be excluded from further consideration in the analysis of target file 100. In this case, flags are reset in table 228 for each of polymorphic viruses 150 excluded by the
15 type of target file 100, and subsequent analysis of target file 100 considers only unexcluded polymorphic viruses 150.

Minimum Size of Polymorphic Virus Depending on the encryption routine employed, polymorphic viruses 150 may generate executable images having a range of sizes. However, a minimum size for the executable image of each polymorphic virus 150 is provided by
20 unencrypted static virus body 160, including mutation engine 162. Consequently, each polymorphic virus 150 having an executable image that is larger than the executable image 100 of the target file being analyzed may be excluded from further consideration in that analysis.

JMP Instruction Usage Many polymorphic viruses 150 that infect COM files do so by appending themselves to the COM file and inserting a JMP instruction at the entry point of the
25 COM file. Consequently, when static exclusion module 230 examines a COM file and determines that the first instruction is not a JMP instruction, each polymorphic virus 150 that employs such an instruction at the entry point of infected COM files may be excluded from further consideration.

Entry Point Distance in EXE Files Polymorphic viruses 150 that infect EXE files have a
30 maximum load image size. Since these viruses infect EXE files by appending themselves to the EXE file load image, the distance between entry point 132 and the end of the load image must be less than this maximum value. Any polymorphic viruses 150 having maximum sizes less than the distance calculated for an EXE file under analysis may be excluded from further consideration.

35 This list of features examined during the static exclusion phase is not intended to be exhaustive. Additional features of polymorphic viruses 150 may also be suitable for use by static exclusion module 230 to exclude various polymorphic viruses 150 from further consideration.

Typically, only a subset of known polymorphic viruses will be ruled out by static exclusion module 230 and some emulation will be required. In these cases, dynamic exclusion

module 240 is implemented to initiate and control emulation module 210. During emulation, emulation control module 220 instructs emulation module 210 to fetch an instruction from load image 100. As each instruction is fetched, emulation control module 220 compares the fetched instructions with an instruction/interrupt usage profile 224 for each known polymorphic virus 150. For each polymorphic virus 150 that does not implement the fetched instruction as indicated by its instruction/interrupt usage profile 224, the corresponding flag in table 228 is reset to exclude the polymorphic virus from further consideration. This process continues with each succeeding instruction until all polymorphic viruses 150 have been excluded. Alternatively, the emulation phase for a target file may be stopped or suspended when one of two other conditions occurs. These conditions are discussed in greater detail below, in conjunction with Fig. 4A.

Referring now to Fig. 3, there is shown an example of an instruction/interrupt usage profile 224 employed by emulation control module 220 to detect a corresponding polymorphic virus 150. Instruction/interrupt usage profiles 224 are made possible by the fact that mutation engines 162 of known polymorphic viruses 150 do not use the entire instruction set available for various processor architectures. For example, the 80x86 instruction set allows for variable length instructions. However, in most cases the first byte of each instruction determines its basic functionality, providing 256 possible basic instruction types. Mutation engines 162 typically employ substantially fewer than 256 instruction types of the 80x86 instruction set.

Referring still to Fig. 3, each bit of instruction/interrupt usage profile 224 corresponds to a different possible instruction type supported by the 80x86 architecture. A 1 indicates that mutation engine 152 of polymorphic virus 150 uses the corresponding instruction in its decryptors, and a 0 indicates that the instruction is not used by mutation engine 152. For example, the mutation engine employed by the Everfire polymorphic virus uses eight different instruction types in its decryptors, while the DSCE mutation engines uses 190 different instruction types in its decryptors. Consequently, instruction usage profile 224 for the Everfire polymorphic virus includes only eight ones in the bits corresponding to these instructions. On the other hand, 190 of 256 bits of instruction usage profile 224 for the DSCE polymorphic virus are ones, making this virus more difficult to detect.

The instructions/interrupts used by the mutation engine of a polymorphic virus may be determined by infecting a large number of files with the polymorphic virus and analyzing the decryption loops generated in the infected files. Virus infection may be done automatically under software control, and the resulting polymorphic decryption loops may likewise be analyzed automatically to generate instruction/interrupt usage profile 224 appropriate for the virus' mutation engine. Further, since each polymorphic virus 150 is tested with data specific to its mutation engine 162, there is no need for regression testing when a new instruction/interrupt usage profile is added to emulation control module 220.

Polymorphic viruses 150 typically have no reason to use interrupts in their decryption loops, and consequently, these were used as "stoppers" by CDPE anti-virus program developers. Not surprisingly, virus developers began to include interrupts in their decryption loops

specifically because conventional CDPE programs would interpret their presence as an indication that the associated code was not part of a decryption loop. In the present invention, emulation control module 220 treats detection of an interrupt in a manner similar to detection of any instruction, to eliminate from further consideration each polymorphic virus 150 that does not
5 employ such an interrupt in its decryption loop.

Emulation control module 220 compares the instructions/interrupts fetched by emulator module 210 with the corresponding entry in instruction/interrupt usage profile 224 of each polymorphic virus 150 still under active consideration. When emulator 210 fetches an instruction (or an interrupts) that is not employed by one of polymorphic viruses 150 still being
10 considered, the corresponding flag in table 228 is reset indicating that the virus should no longer be considered in the emulation phase of PAM 200.

Typically, Emulation module 210 eventually retrieves an instruction that is not implemented by the last actively considered polymorphic virus 150. This indicates either that (1) target file 100 is not infected with any of polymorphic viruses 150 or (2) one of polymorphic
15 viruses 150 has been decrypted to reveal static virus body 160, which like any program can employ any of the instructions supported by the CPU architecture. Most mutation engines 162 can be eliminated from consideration on one pass through the decryption loop of polymorphic virus 150. However, in order to eliminate the possibility of getting trapped in an infinite loop or spending too much time in the emulation phase, an upper limit may be set for the number of
20 instructions to be emulated. In the preferred embodiment of the invention, emulation control module 220 terminates the emulation phase when either all polymorphic viruses 150 have been excluded or 1.5 million instructions have been executed.

Once the emulation phase has been terminated, scanning can begin on decrypted static virus body 160 or at least those parts decrypted by the first 1.5 million instructions. In order to
25 facilitate scanning of static virus body 160, emulation control module 220 keeps track of which locations of virtual memory are most likely to be infected. In particular, emulation control module 220 tags the page or pages of virtual memory containing an instruction executed by emulator 210. In addition, every time an emulated instruction writes to memory, the altered pages are tagged as containing modified data. Tagged pages in virtual memory are scanned for
30 signatures of static virus body 160 as follows:

- (1) if a page contains executed code, it and the following page are scanned
- (2) if a page has been written to during emulation and more than a threshold number of memory writes occurred anywhere in memory during emulation, the modified page and the following page are scanned.

35 In the preferred embodiment of the invention, a threshold number of memory writes equal to 32 bytes has been found to speed up scanning without sacrificing accuracy. However, the 32 byte value is a heuristic and may be adjusted up or down as desired. All existing polymorphic viruses 150 modify far more than 32 bytes when they decrypt themselves. On the other hand, the number of data bytes modified by an uninfected program is typically fewer than the number

modified by a polymorphic virus 150. If fewer than 32 bytes have been modified by the time the emulation phase of PAM has concluded, emulation control module 220 assumes that nothing was decrypted and does not scan modified pages.

At the completion of emulation phase, PAM 200 proceeds to a scanning phase in which all pages from which instructions were retrieved during emulation are scanned, and all pages modified during emulation are scanned provided at least 32 bytes of virtual memory have been modified during the emulation phase. During the scanning phase, virus signatures for each known polymorphic virus 150 are compared with executed and modified code in the virtual memory locations identified above. PAM 200 may employ a scanning module 250 that implements any of a number of scanning methods to search the tagged memory locations. However, in the preferred embodiment of PAM 200, scanning module 250 implements a scanning process that allows each target buffer to be scanned for signatures of all known polymorphic viruses 150 in a single pass. As used here, a target buffer corresponds to a tagged page of virtual memory.

The preferred scanning process requires that all virus signatures comply with two conditions:

- (1) Virus signatures are chosen to begin with one of a selected number of special bytes. In the preferred embodiment, sixteen special bytes are selected.
- (2) Virus signatures are chosen such that emulation of corresponding polymorphic virus 150 will not modify bytes of the signature.

The first rule limits the number of valid virus signatures, but it speeds up the scanning rate of scanning module 250, which simply skips any data not beginning with the selected bytes. Scanning module 250 performs a more elaborate scan when data beginning with one of the selected bytes is encountered. In the preferred embodiment, the selected bytes are:

2E, 3D, 50, 74, 8B, 8C, B8, B9, BA, BB, BE, BF, CD, E8, E9,

These bytes are selected based on their frequent use in 80x86 machine code. The set of selected bytes is not fixed in either number or content and may be adjusted as necessary. However, use of additional selected bytes will slow the scanning phase.

For each selected byte, scanning module 250 includes a pre-computed bit array which determines whether the selected bit is part of a virus signature. When scanning module 250 encounters one of the selected bytes, the top thirteen bits of the next word provide an index into the bit array associated with the selected byte. The associated bit arrays are designed so that the indexed bit is a zero when scanning engine 250 includes no virus signature that matches the first three bytes of the target data and a one when scanning engine includes a virus signature that matches at least these three bytes. In the first case, scanning engine 250 proceeds to the next data having one of the selected bytes.

When the indexed bit is a one, scanning module 250 contains a signature matching at least the first three bytes of the data under consideration and proceeds with a binary search to determine whether the target bytes match one of its virus signatures. Thus, a full binary search is

performed on only a small subset of potential target bytes. When the binary search matches these target bytes to a virus signature, scanning engine 250 terminates scanning of the target file, and PAM 200 indicates the identity of the detected polymorphic virus to a program designed to deactivate the virus. Deactivation programs are well known and are not discussed in greater detail here.

Referring now to Fig. 4A, there is shown a flow chart of static and dynamic exclusion phases of a polymorphic virus detection method 400 implemented by PAM 200. Method 400 begins by initializing 410 a virtual machine for emulating files in isolation from the host computer and loading 414 a target file for analysis. Analysis of the target file begins with a static exclusion phase, in which the target file is examined 418 for features that exclude certain known viruses from further consideration (static exclusions). If all known viruses have been excluded 420 at step 418, the target file is deemed infection-free 440 and the next target file is loaded 414 for emulation. If all known viruses are not excluded 420, method 400 enters the emulation phase.

During the emulation phase, instructions are fetched 424 from the target file and compared 428 with instruction/interrupt usage profiles 224 of known polymorphic viruses 150. For each polymorphic virus 150 that does not implement the fetched instruction, a corresponding flag in table 228 is reset. The fetched instruction is then emulated 430 and any pages in virtual memory affected by the instruction are tagged 434. Affected pages include pages of virtual memory from which the instruction is fetched 424 as well as any pages in virtual memory that are modified when the instruction is emulated 430. Following emulation 430 and tagging 434, method 400 determines 438 whether emulation should proceed or branch to the scanning phase.

In the disclosed embodiment of the invention, the scanning phase is entered if flags have been reset for all polymorphic viruses 150 in table 228 or if a preselected number of instructions have been emulated. In addition, the scanning phase may be entered periodically, prior to the occurrence of either of these conditions, to attempt to identify a virus that has been partially decrypted. This periodic scanning approach is discussed below in conjunction with the scanning phase.

If method 400 determines 438 that none of the above conditions for entering the scanning phase have occurred, the next instruction of the target file is fetched 424 for analysis. Eventually, either all polymorphic viruses 150 have been flagged 428 or the preselected number of instructions have been emulated 430, and method 400 enters the scanning phase 450. In the preferred embodiment of the invention, the preselected number of instruction is 1.5 million. This allows emulation of sufficient instructions to decrypt a substantial part of static virus body 160, even if it is not fully decrypted.

Referring now to Fig. 4B, there is shown a flow chart of the scanning phase of method 400. Page locations tagged 434 during the emulation phase are accessed 454 and scanned 458 byte at a time to determine 460 whether each scanned byte is a selected byte. If not, the next byte is scanned 458 and tested 460 until it is determined 462 that no more bytes remain on the current

tagged page. When the last byte of the tagged page is reached 462, the next tagged page is accessed 454, and the process is repeated with each byte of the new tagged page.

When a selected byte is identified 460, the corresponding bit array is retrieved 464 to determine 468 whether the selected byte and two following bytes of the word are consistent with any of the viral signatures. If no match is possible, scanning process 450 branches to step 462. If a match is possible 468, process 450 begins a binary scan 470 to determine whether the full word actually matches 474 a viral signature. If a match is detected 474, process 450 indicates that the file is infected and proceeds to load 488 the next file for analysis. At this point, another module may be activated to eliminate the detected virus from the infected file.

If no match is detected 474, scanning continues with the first byte of the next tagged page, unless it is determined 478 that the scanned byte is the last byte the last tagged page. When scanning of all tagged pages is completed without a match, the next step depends on which condition at step 438 of the emulation phase initiated scanning process 450. If the just completed scan was one initiated periodically 480 during decryption in the emulation phase, process 450 returns control 482 to the emulation process 400 in order to continue decrypting the target file. On the other hand, if the emulation phase was terminated 438 by either exclusion of all known polymorphic viruses 150 or emulation of 1.5 million instructions, no further emulation of the current target file is necessary. The target file is deemed uninfected 484 and the next target file is loaded 488 for analysis.

In order to test PAM system 200, instruction/interrupt profiles 224 were generated for the six mutation engines 162 (implemented in the indicated viruses) and twelve polymorphic viruses 150 listed in Table 1. Each mutation engine 162/polymorphic virus 150 was used to infect 200 files, and 100 of these infected files were executed, generating a sample of 100 different decryption loops for each mutation engine 162/polymorphic virus 150. The sample of decryption loops were then scanned, and the instructions/interrupts used by each mutation engine 162/polymorphic virus 150 were tallied to provide a corresponding instruction/interrupt profile 224 (Fig. 3). Instruction/interrupt profiles 224 generated by the 100 infected files were then employed by PAM 200 in scanning the remaining 100 infected files. PAM system 200 detected each mutation engine 162/polymorphic virus 150 in each of the 100 infected files.

<u>Mutation Engine (& Virus(es) Used)</u>	<u>Polymorphic Virus</u>
Dark Angel Mutation Engine (DAME) Virus used: Lame	Bishop Virus
Dark Slayer Confusion Engine (DSCE) Virus used: Connie	Commander Bomber Virus
Dark Slayer Mutation Engine (DSME) Virus used: Connie	Everfire Virus
The Mutation Engine (MtE) Viruses used: Coffeshop, CryptLab, Fear, Groove, Insuf, Pogue, Questo, Sara, Shocker SMEG	IAMV Virus
Virus used: Queeg	K-Hate Virus
Trident Polymorphic Engine (TPE) Viruses used: Bosnia, Giraffe, YB	Mutagen Virus
	Natas Virus
	One-Half Virus
	Satan Bug Virus
	Tremor Virus
	Uruguay (10) Virus
	V2P6Z Virus

Referring now to Table 2, there is shown a comparison of the detection statistics for PAM system 200 and four other shareware and commercial virus detection products. Each anti-virus program was run on the same 100 infected files examined by PAM system 200. A 'yes' entry for a virus indicates that the corresponding anti-virus program detected each of the instances of infection for the corresponding virus. Only an anti-virus scanner that detects all infections of a given virus can guarantee that a cleaned system will not become reinfected. As can be seen from the table, only the PAM system 200 detected every instance of every virus.

Engine/Virus	McAfee SCAN 2.20 (CDPE-Based)	KAMI AVP 2.1 (CDPE-Based)	Command F-Prot 2.17	Thunderbyte TBAV 6.34 (CDPE-Based)	PAM
DAME	no	yes	no	yes	yes
DSME	no	yes	yes	yes	yes
MtE	no	yes	yes	yes	yes
Mutagen	no	yes	yes	yes	yes
SMEG	no	yes	yes	yes	yes
TPE 1.4	no	yes	no	yes	yes
Cmdr. Bomber	no	yes	no	no	yes
Everfire	no	yes	yes	yes	yes
IAMV	yes	yes	yes	yes	yes
K-Hate	no	yes	no	no	yes
Natas	yes	yes	yes	yes	yes
One Half	no	yes	no	yes	yes
Bishop	no	no	yes	yes	yes
Satan Bug	yes	yes	yes	yes	yes
Tremor	yes	yes	yes	yes	yes
Uruguay (10)	no	no	no	no	yes
DSCE	not tested	not tested	not tested	not tested	yes
VsP6Z	yes	yes	yes	yes	yes

Thus, a system and method have been provided for detecting polymorphic viruses without recourse to the booster and stopper heuristics employed by available CDPE-based systems.

- 5 Systems and methods in accordance with the present invention employ a static phase in which the gross features of a file are first examined to determine whether infection by some polymorphic viruses can be excluded at the outset. A subsequent emulation phase looks for a polymorphic detection loop in the file by comparing each instruction generated during file emulation with instruction/interrupt usage profiles for known polymorphic virus/mutation engine. Virtual memory pages executed or modified during emulation are tested during a scanning phase for virus signatures.
- 10

Claims

1. A computer implemented method for detecting polymorphic viruses in a computer file, the method comprising the steps of:
 - fetching an instruction from the computer file;
 - 5 determining whether the instruction may be used in decryption loops generated by known polymorphic viruses identified in a list;
 - when the instruction is used in decryption loops generated by known polymorphic viruses, tagging a memory location associated with the instruction and repeating the fetching and determining steps; and
 - 10 scanning the tagged locations for virus signatures when the emulated instruction is not used in decryption loops generated by known polymorphic viruses.
2. The computer implemented method of claim 1, wherein the determining step comprises the substeps of:
 - 15 comparing the instruction with instruction/interrupt usage profiles for known polymorphic viruses identified on the list;
 - excluding from the list known polymorphic viruses having instruction/interrupt profiles that do not include the instruction; and
 - identifying the instruction as used if it appears in the instruction/interrupt usage profile of non-excluded polymorphic virus.
- 20 3. The computer implemented method of claim 2, wherein the identifying step comprises identifying the instruction as used if it appears in the instruction/interrupt usage profile of a non-excluded polymorphic virus or if a threshold number of instructions have been fetched.
4. The computer implemented method of claim 1, wherein the tagging step comprises the substeps of:
 - 25 emulating the instruction;
 - tagging a page in memory from which the emulated instruction is fetched; and
 - tagging a page in memory including data modified by the emulated instruction.
5. The computer implemented method of claim 1, comprising the additional steps of:
 - 30 prior to the fetching step, determining size and type features of the computer file;
 - and
 - excluding from the known polymorphic viruses identified on the list, those known polymorphic viruses having characteristics inconsistent with the size and type features of the computer file.
6. The computer implemented method of claim 5, wherein the step of determining size and type features of the computer file comprises the substeps of:
 - 35 determining the size of a load image of the computer file;
 - determining the distance between an entry point of the computer file and an end of the load image of the computer file; and
 - determining the type of the computer file.

7. A computer implemented method for detecting a polymorphic virus in a computer file, the method comprising the steps of:

- initializing a list of known polymorphic viruses
- generating an instruction/interrupt usage profile for each polymorphic virus in the
5 list;
- fetching an instruction from the computer file;
- comparing the fetched instruction with instruction/interrupt usage profiles for each listed polymorphic virus to identify those polymorphic viruses that can generate the fetched instruction;
- 10 delisting each listed polymorphic virus that can not generated the fetched instruction;
- when at least one polymorphic virus remains listed, tagging a memory location associated with the fetched instruction and repeating the fetching, comparing, flagging, and tagging steps; and
- 15 scanning the tagged locations for virus signatures when all listed polymorphic viruses have been removed from the list.

8. A computer implemented method for determining whether an instruction fetched from a computer file may be part of a decryption loop generated by a polymorphic virus infecting the computer file, the method comprising the steps:

- 20 initializing a list of polymorphic viruses;
- for each listed polymorphic computer virus, generating a list of instructions and interrupts employed by the polymorphic virus in producing decryption loops; and
- fetching instructions from the computer file and for each fetched instruction:
 - comparing the fetched instruction with the list of instructions and
25 interrupts used by each polymorphic virus on the list;
 - removing from the list of polymorphic viruses each polymorphic virus that does not employ the fetched instruction; and
 - repeating the comparing and removing steps until all polymorphic viruses have been removed from the list.

30 9. A computer readable storage medium on which is stored data for analyzing computer files for the presence of polymorphic viruses, the data being suitable for implementation by a computer to perform the steps of:

- fetching an instruction from a computer file;
- determining whether the instruction may be used in decryption loops generated by
35 known polymorphic viruses identified in a list;
- when the instruction is used in decryption loops generated by know polymorphic viruses, tagging a memory location associated with the instruction and repeating the fetching and determining steps; and

scanning the tagged locations for virus signatures when the emulated instruction is not used in decryption loops generated by known polymorphic viruses.

10. A system for testing a computer file for polymorphic viruses, the system comprising:

an emulation module for emulating instructions from the computer file to be tested;

a scanning module including viral signatures to identify each polymorphic virus to be tested for; and

an emulation control module coupled to the scanning module and the emulation module, the emulation control module including:

an instruction/interrupt usage profile for each polymorphic virus for identifying decryption loop instructions generated by the polymorphic virus; and

a dynamic emulation module for comparing computer file instructions processed by the emulation module with the instruction/interrupt usage profiles to determine whether the instruction may be from a polymorphic virus decryption loop and triggering the scanning module to scan the file for viral signatures when the comparison indicates a scan condition has occurred.

11. An emulation control module for directing emulation of a computer file to be tested for a polymorphic virus by an emulation module, the emulation control module comprising:

a table of known polymorphic viruses;

instruction/interrupt usage profiles for the known polymorphic viruses;

a dynamic exclusion module coupled to the instruction/interrupt usage profiles, the table, and the emulation module for comparing instructions fetched by the emulation module with the instruction/interrupt usage profiles of the known polymorphic viruses in the table and excluding from the table polymorphic viruses having instruction/interrupt usage profiles that are inconsistent with the fetched instruction, the dynamic exclusion module being effective to determine whether the instruction may be from a decryption loop in the computer file;

size/type data for known polymorphic viruses; and

a static control module coupled to the size/type data, the table, and the emulation module for excluding from the table polymorphic viruses having size/data properties inconsistent with the size/type data in the table.

12. A computer implemented method for detecting polymorphic viruses in a computer file, the method comprising the steps of;

fetching instructions from the computer file;

comparing the fetched instructions with instruction/interrupt usage profiles associated with listed polymorphic viruses;

tagging memory locations associated with the fetched instructions that are consistent with the instruction/interrupt usage profiles associated with the listed polymorphic viruses;

delisting polymorphic viruses having associated instruction/interrupt usage profiles that are inconsistent with the fetched instructions; and
scanning the tagged locations for virus signatures when all polymorphic viruses have been delisted.

- 5 13. The method of claim 12, including the additional step of scanning the tagged memory location for virus signatures at preselected intervals of fetched instructions.
14. The method of claim 12, comprising the additional step of scanning the tagged locations for virus signatures when a preselected total number of the fetched instructions reaches a preselected number.

10

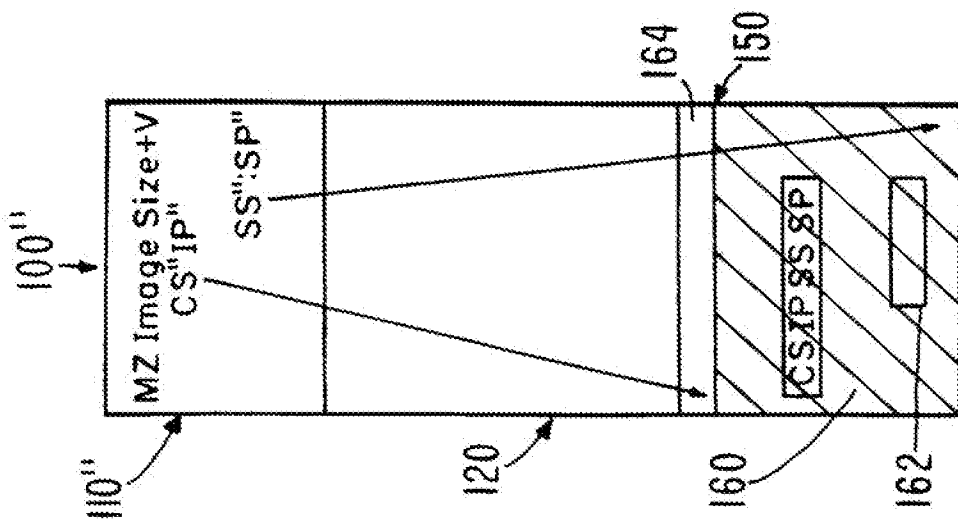


FIG. 1A

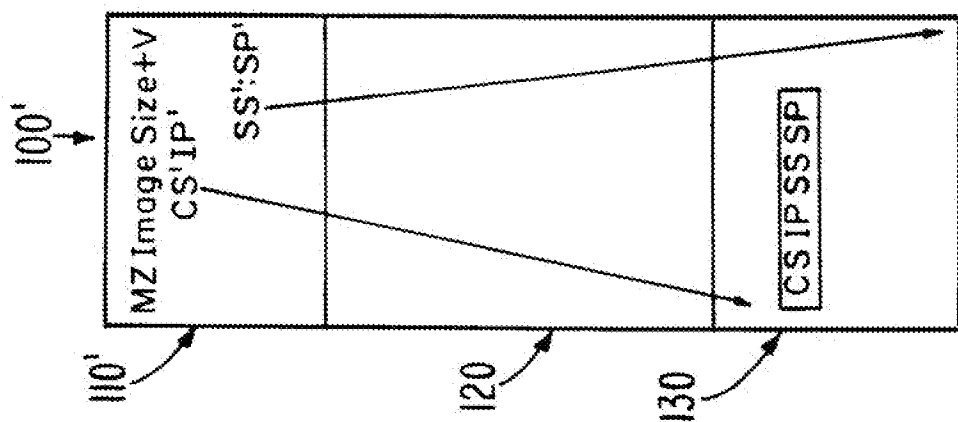


FIG. 1B

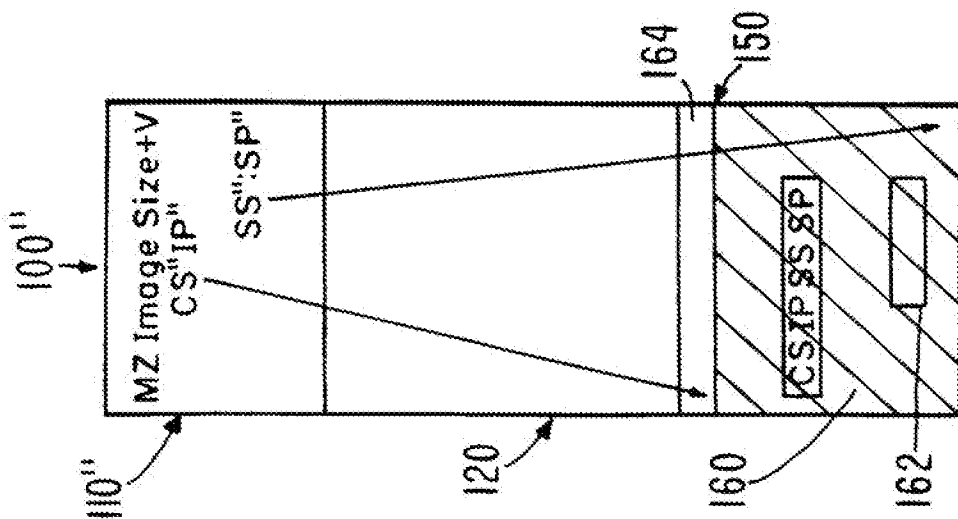


FIG. 1C

2/5

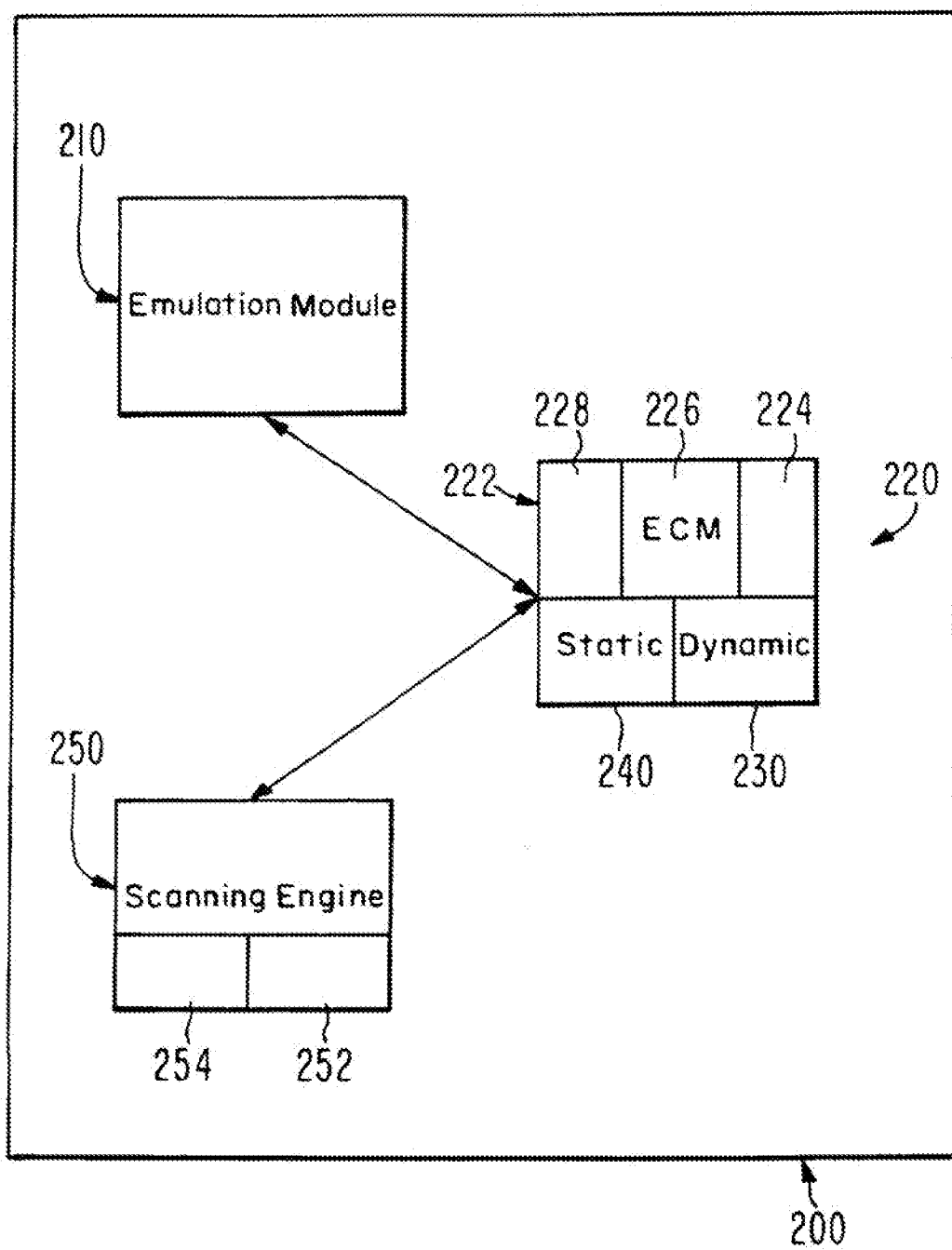


FIG.2

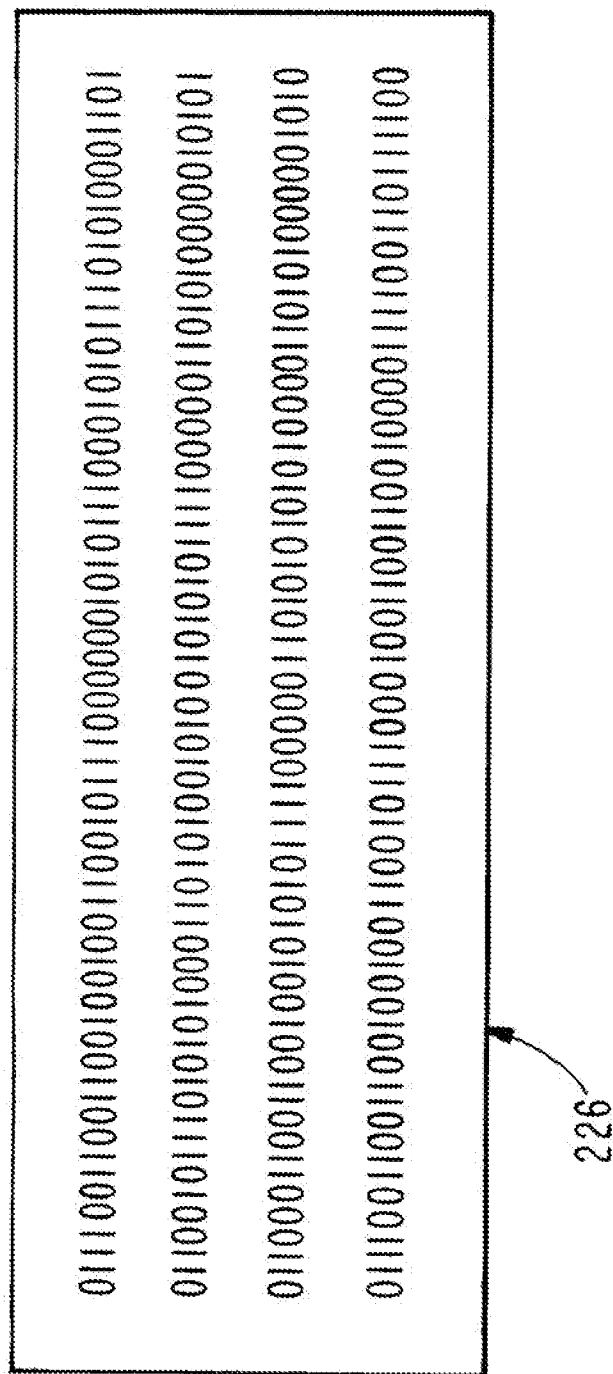


FIG.3

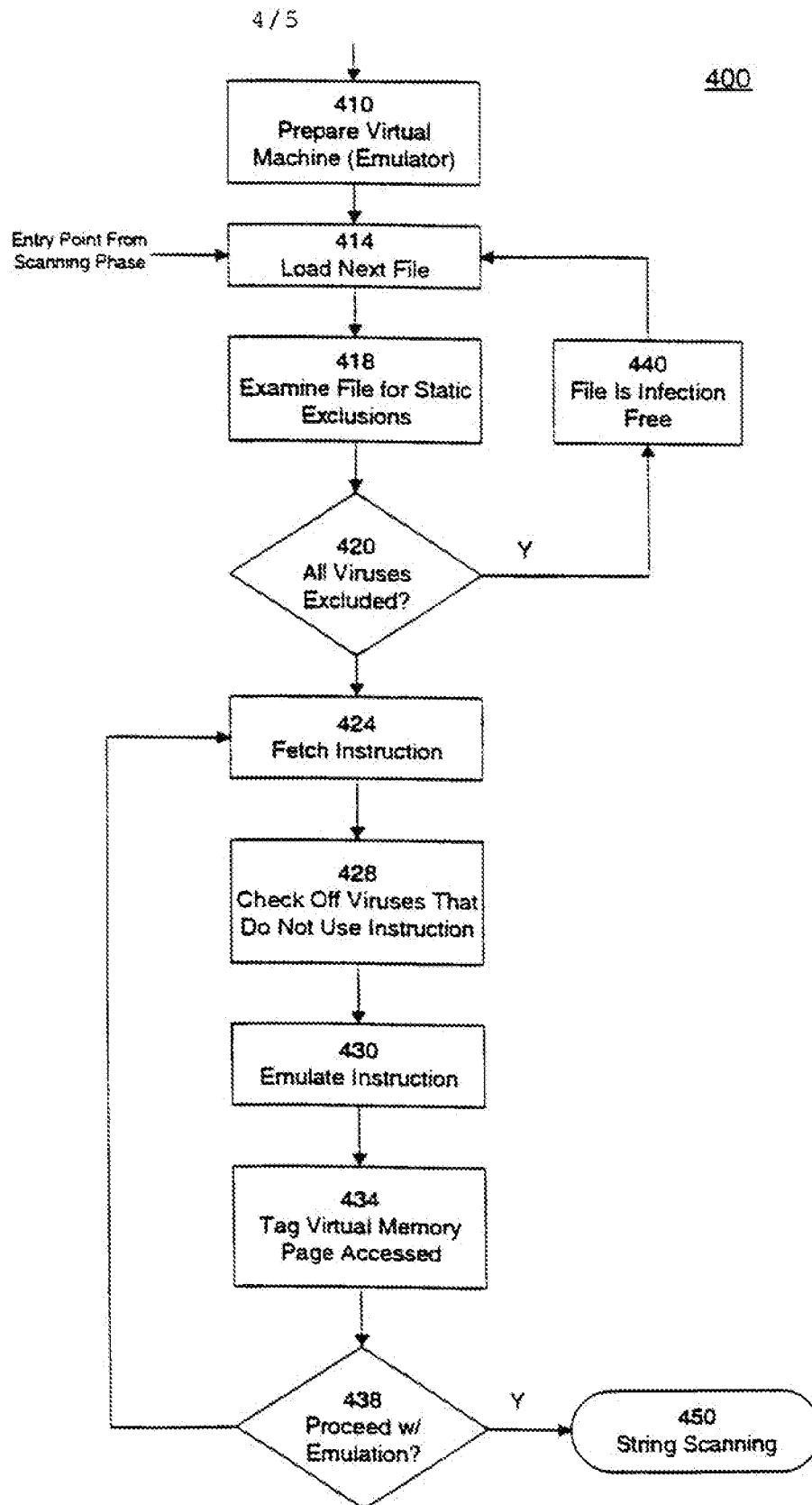


Fig. 4A

5/5

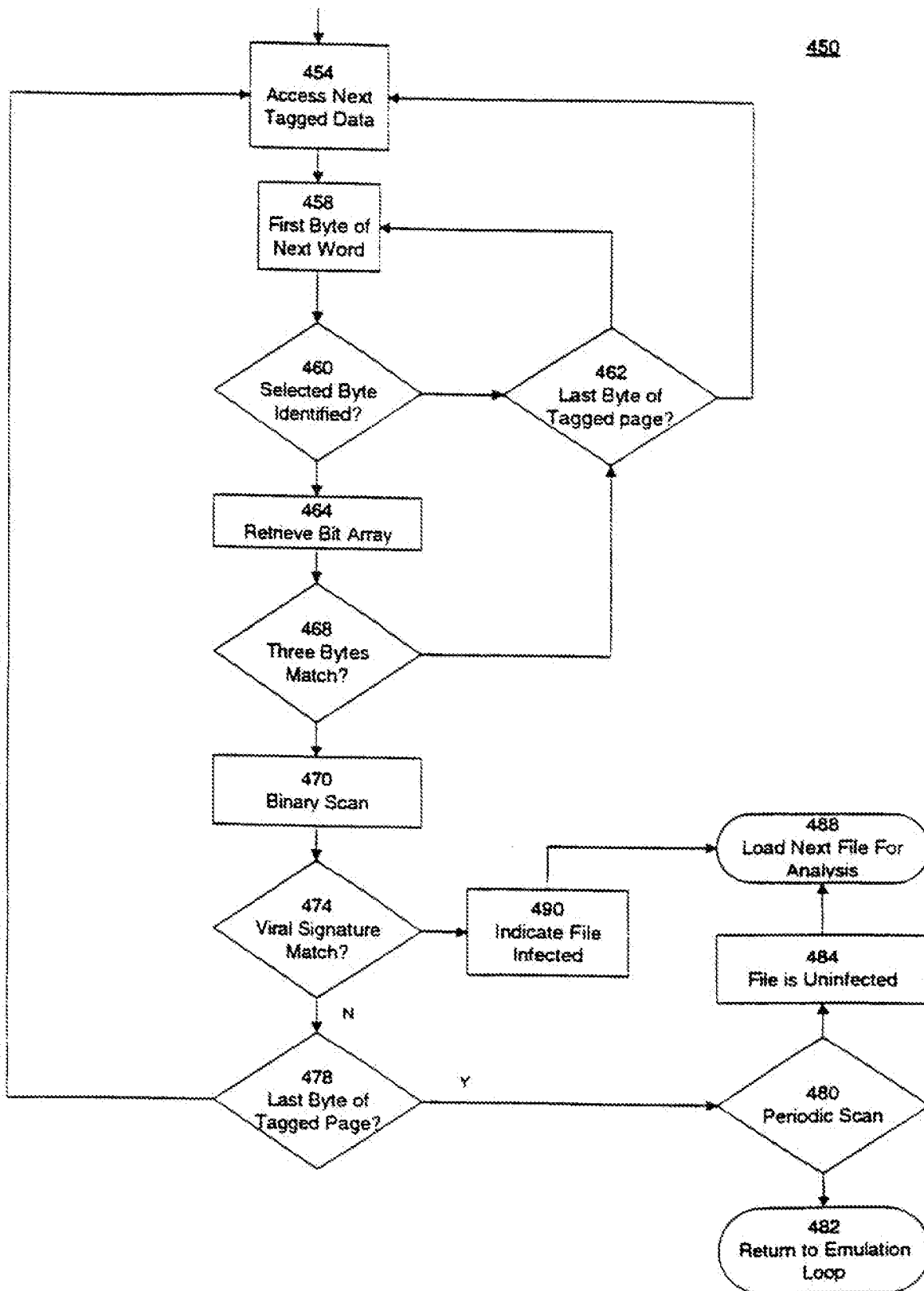


Fig. 48

INTERNATIONAL SEARCH REPORT

Intern al Application No
PCT/US 96/15512

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F11/00

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

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

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	IBM TECHNICAL DISCLOSURE BULLETIN, vol. 34, no. 2, 1 July 1991, pages 415-416, XP000211158 "AUTOMATED PROGRAM ANALYSIS FOR COMPUTER VIRUS DETECTION" see the whole document	1,7-12
A	IBM TECHNICAL DISCLOSURE BULLETIN, vol. 34, no. 2, 1 July 1991, pages 150-154, XP000211065 "ARTIFICIAL IMMUNITY FOR PERSONAL COMPUTERS" see the whole document - / - -	1,7-12

☒ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

- *A* document defining the general state of the art which is not considered to be of particular relevance
- *E* earlier document but published on or after the international filing date
- *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- *O* document referring to an oral disclosure, use, exhibition or other means
- *P* document published prior to the international filing date but later than the priority date claimed

- *T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- *A* document member of the same patent family

Date of the actual completion of the international search

21 January 1997

Date of mailing of the international search report

03.02.97

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl,
Fax (+ 31-70) 340-3016

Authorized officer

Herreman, G

INTERNATIONAL SEARCH REPORT

International Application No.

PCT/US 96/15512

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	LAN MAGAZINE, EMAP BUSINESS PUBLISHING, vol. 3, no. 6, June 1995, UK, pages 55-56, 58, 61, 63-64, 67, XP000613971 G. MARSHALL: "Pest Control" see page 56, right-hand column, line 22 - page 58, left-hand column, line 8 ---	1,7-12
A	EP 0 636 977 A (CHAMBERS DAVID ALAN) 1 February 1995 see the whole document -----	1,7-12

INTERNATIONAL SEARCH REPORT

information on patient family members

Internal Application No

PCT/US 96/15512

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP-A-0636977	01-02-95	US-A- 5398196	14-03-95

PATENT ABSTRACTS OF JAPAN

(11)Publication number : **08-263447**

(43)Date of publication of application : **11.10.1996**

(51)Int.Cl.

G06F 15/16

G06F 9/445

G06F 13/00

(21)Application number : **07-349164**

(71)Applicant : **SUN MICROSYST INC**

(22)Date of filing : **20.12.1995**

(72)Inventor : **GOSLING JAMES A**

(30)Priority

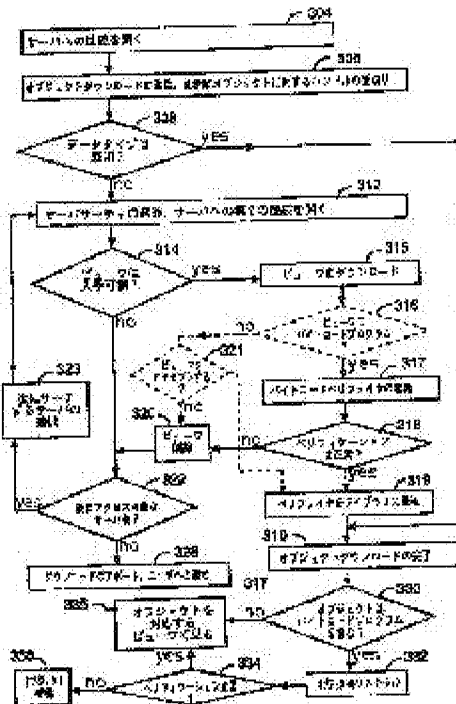
Priority number : **94 359884** Priority date : **20.12.1994** Priority country : **US**

(54) **DISTRIBUTED COMPUTER SYSTEM AND OPERATION METHOD THEREFOR**

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a distributed computer system provided with a computer for automatically clown-loading the viewer of an object to be referred to and verifying the maintainability of a loaded program and the operation method.

SOLUTION: At the time of loading data (object) stored in another server or the like through a network and referring to them, the viewer corresponding to the object is automatically searched, and in the case that the appropriate viewer is found in the other server or the like, verification is performed so as to confirm the maintainability before activating the viewer. Especially, importance is placed in the verification relating to the use of a stack and a data type to the program written in a byte code language.

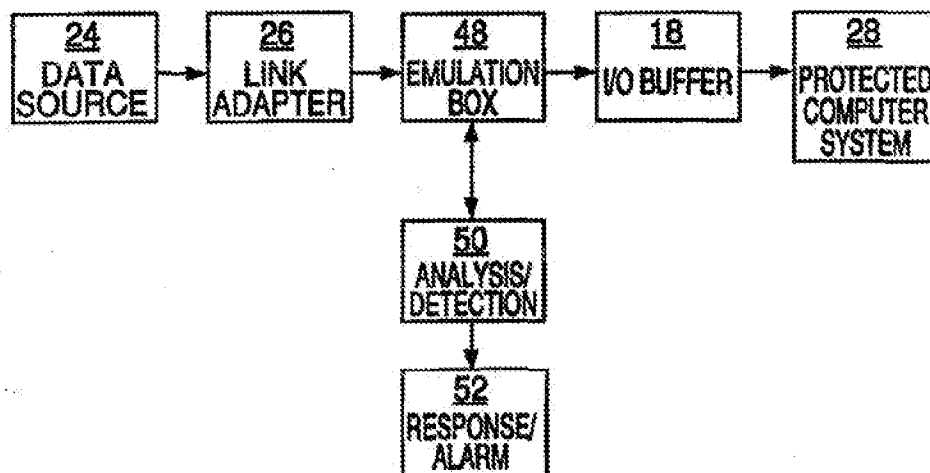


PCTWORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : G06F 11/00, 17/00		(11) International Publication Number: WO 95/33237
A1		(43) International Publication Date: 7 December 1995 (07.12.95)
(21) International Application Number: PCT/US95/06659		(81) Designated States: CA, CN, DE, GB, JP, MX, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).
(22) International Filing Date: 30 May 1995 (30.05.95)		
(30) Priority Data: 08/252,622 1 June 1994 (01.06.94) US		
(71) Applicant: QUANTUM LEAP INNOVATIONS INC. [US/US]: 410 Briarcliff Road, Briarcliff Manor, NY 10510 (US).		
(72) Inventors: SCHNURER, John; P.O. Box 446, Yellow Springs, OH 45387 (US). KLEMMER, Timothy, J.; 410 Briarcliff Road, Briarcliff Manor, NY 10510 (US).		
(74) Agent: AMARAL, Anthony, Jr.; Reid & Priest L.L.P., 40 West 57th Street, New York, NY 10019 (US).		Published With international search report.

(54) Title: COMPUTER VIRUS TRAP



(57) Abstract

A computer virus trapping device (10) is described that detects and eliminates computer viruses before they can enter a computer system and wreck havoc on its files, peripherals, etc. The trapping device (10) creates a virtual world that simulates the host computer system (28) intended by the virus to infect. The environment is made as friendly as possible to fool a computer virus into thinking it is present on the host (28), its intended target system. Within this virtual world, the virus is encouraged to perform its intended activity. The invention is able to detect any disruptive behaviour occurring within this simulated host computer system. It is further able to remove (52) the virus from the data stream before it is delivered to the host (28) and/or take any action previously instructed by a user (38).

DEF-FIN00007699

FOR THE PURPOSES OF INFORMATION ONLY

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

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

DEF-FIN00007700

COMPUTER VIRUS TRAP

BACKGROUND OF THE INVENTION

The computer virus problem that exists today had its beginnings sometime in the late 1980s. At that time computer viruses were a novelty and plagued mainly DOS and Macintosh computers. Today, almost every Fortune 500 company has experienced computer viruses with the current rate being about one virus incident every 2 to 3 months.

The term computer virus is applied in common and legal usage to software, code, code blocks, code elements and code segments which perform certain functions in the digital computer environment. Code is intended to mean the digital instructions which the computer responds to. Non damaging or legitimate software, code, code blocks, code segments and code elements that serve a useful purpose would not be considered a virus.

Computer viruses have been known to cause physical harm to computer hardware in addition to erasing and destroying data. While rare, there have been cases of viruses that have made calls to disk drive heads actually scoring the media; still others have been discovered that ramped up the scan rate on a monitor causing failure. Most viruses do not, however, intentionally cause explicit physical harm and they are discovered before they are triggered to cause damage to data and files. However, it is after discovery that the real cost of viruses becomes apparent in connection with their detection and removal. In an average computer site this might entail searching 1000 PCs and 35,000 diskettes. If the software engineer misses even one instance of the virus, other computers will be re-infected and the clean up search must be repeated all over again.

A common misconception is that there are good viruses and bad viruses. Some viruses are claimed to be benign because they do not have a malicious

trigger event and cannot do intentional harm. However, this misses the point that the problems computer viruses cause are mainly due to the trigger events. It is a fact that computer viruses replicate. This by itself is harmful because it necessitates a search to clean up all instances of the viruses in a computer installation.

The damage caused by viruses, not so much due to erased files or data, but in the cost of detection, removal and also the accompanying lowered worker productivity can be very high. It has been calculated that the average computer site will spend on the order of about \$250,000 on a computer virus cleanup. It has been estimated that computer viruses will cost U.S. computer users over a billion dollars in 1994 alone.

The problem will grow exponentially due to the advent of the Information Super Highway. The increased connectivity among individuals, companies and government will allow a computer virus to create havoc. Currently disjoint computer systems that perform various functions that we take for granted today, such as, banking, telecommunications, radio, information databases, libraries and credit might meld together in the future. Thus, computer viruses, unchecked, could have a crippling effect on our society.

A virus can only cause trouble when it enters a system and finds a location on which to act. In a general sense, the virus must perform an intended function or a function the user or operator did not intend, expect, compensate for or otherwise protect against. Some examples of malicious virus activity are: changing names of files making it difficult for the user to access the files, moving a file to a new location, deleting files, interfering with working programs (i.e. causing all the words on a screen to fall to the bottom of the screen in a heap), replicating themselves and clogging up the system making it nonfunctional or waiting for a predetermined time period or after a certain number of toggle operations such as boot, access, cursor movements, mouse clicks, etc. before acting.

More felonious type viruses are those that have been released to cause ruin or impairment of a system for the purposes of sabotage, espionage, financial gain or to impair a competing business. Some examples include: creating a trap door which allows access to an unauthorized user for any purpose such as espionage, dumping files or erasure, navigation programs which find routes into systems, password cracking programs, modifying the executable segment of legitimate programs and attaching themselves to a code block and travel to another site.

In addition to traditional PCs and networks being vulnerable to virus infections, embedded control systems often used in industrial process control settings are also vulnerable. These systems control machinery, motors, industrial robots and process data from sensors. Because embedded systems are vulnerable to viruses just as PCs are, the results are potentially quite damaging. The smooth flow of a factory or assembly line could be devastated by a virus' uncontrolled behavior.

There are many possible ways for a virus to act on a computer system. All computers go through a boot procedure in which the Basic Input Output System (BIOS) and/or other resident system tools perform a variety of startup tasks such as, finding drives, testing memory and the system, initiating system files, loading DOS or other Operating System (OS) and bringing up an initial startup program. The system performs certain housekeeping tasks such as establishing various links among other functions. A computer system of any utility is complex enough that someone writing a virus has a myriad of opportunities and possibilities in which to cause trouble and interfere with the proper operation of the system.

The most common solution to the virus problem is to employ anti-virus software that scans, detects and eliminates viruses from computer systems. These programs work by searching a storage medium such as a hard disk drive or floppy diskette for known patterns of various viruses. However, there are

problems associated with this method of virus elimination. The software can only scan for known viruses which have an identifiable pattern that can be detected using repetitive string searches. To protect against new viruses frequent upgrades must be distributed. In addition, for the program to detect a virus it must already have infected that computer. The virus might have done some damage or even replicated itself and spread before it is detected. Also, the program must be run often to provide effective protection against viruses especially on systems where programs and data are transferred frequently between computers via diskettes.

In addition further liabilities, pitfalls and limitations to the current breed of anti-virus software solutions exist. This software breaks down into 3 categories: scanners, monitors, CRC's. Scanners as previously mentioned work off of databases of known strings. These databases are in constant need of updates. Monitors are memory resident programs monitoring the computer for questionable behavior. Monitors suffer from high rates of false positives, and they occupy and take a large portion of the limited conventional memory of a PC. CRC's are error checking programs that generate a unique "signature" in the form of a 2-byte number for each and every file to be protected. CRC programs either place the "signature" in the file itself or in a separate file. CRC programs suffer from the fact that they are easy to identify and thus easily tricked into recreating a "signature" for an infected file. Further, Scanners & Monitors & CRC programs must be run on the PC in question. Often this is a time consuming chore. These programs usually must have full control of the PC to operate further inconveniencing the user because he must wait for the scanner to finish before he can begin his normal work. The other critical concept is that the anti-virus software is run on the PC in question. It is subject to the limitations and liabilities of the operating system and may already be running on an infected PC without knowing it. The invention takes a unique approach by performing

its logic outside of the PC, not inconveniencing the user and is more effective because the invention's hardware guarantees a clean uninfected start.

Another possible solution is to increase computer security to the point where viruses cannot enter the system. Login/password control and encryption do not effect computer viruses. With encryption, detection and elimination is made more difficult because the virus along with good data is encrypted, only becoming decrypted when it attempts to replicate. Clearly, this is quite burdensome and expensive to implement.

Another possible solution is to avoid computer bulletin boards, both the commercial type such as, Compuserve, Prodigy, the Internet and Usenet, and the private, local, small type. However, this will not prevent viruses from spreading because most viruses do not result from software or data downloaded from information databases or computer bulletin boards. The operators of both commercial on-line services and private bulletin boards are very careful to keep viruses off their systems. They are constantly searching and scanning anything that is uploaded to their systems before making it available to their subscribers. In addition, most computer viruses of the boot track type do not spread through download data or software. The majority of viruses are spread through diskettes. There are known instances of commercial software being distributed after being infected by a virus. There are known instances of viruses being distributed unwittingly by diskette manufacturers on blank diskettes. There are no rules for which diskettes are more likely to be free from viruses.

Thus, there is a long felt need for a device that can search for, detect and eliminate viruses before they ever enter into a computer system that is transparent to a user and effective against all viruses in existence today and those not yet created.

SUMMARY OF THE INVENTION

One characteristic of almost all viruses is that on their own they are not capable of crossing from one computer OS to another. This is because different

computer systems in use today have different internal instructions or command sets. The language perfectly acceptable and intelligible to one OS does not have any correlation to another. An analogy to humans would be two people speaking different languages not being able to communicate. Although there might exist identical words present in both languages it is statistically very unlikely for a misinterpreted or cross over string of words or set of computer instructions (i.e. a virus) to convey a significant amount of information or be able to effectively execute a series of instructions. It is even more unlikely for this misinterpreted or cross over string of words or series of instructions to migrate from one language or system to another language or system and still be able to convey any useful information or execute a series of commands.

The present invention utilizes this characteristic of viruses to create an impenetrable barrier through which a virus cannot escape. The use of a foreign operating system guarantees the invention a high degree of safety and impenetrability. While the inventors recognize that such invention can be built without the use of a foreign operating system, such a version of the invention would lack any creditable degree of security. In addition, without the use of a foreign operating system the invention itself risks contamination. A foreign operating system different from the one being protected is introduced into the data stream before the data arrives at the computer system to be protected. To illustrate: if a program written for DOS will not run as intended on a Macintosh neither will a virus. A foreign operating system in order to complete its operation must provide an emulation of the target computer operating system (disk drives, memory configuration, ports, etc.). The virus is therefor fooled into thinking it is resident on the target computer system it was intending to infect. It is here, while the virus is resident within the emulated target operating system, that the virus is encouraged to infect files, destroy data and wreak havoc. It is here that the invention diverges from all other strategies in virus-detection and prevention. All other strategies are defensive in nature: they mark

files to detect unwarranted changes, they scan for unintended behavior in an attempt to prevent the virus from performing its damage. The present invention takes an offensive strategy by encouraging the virus to infect and destroy files.

The most critical behavior of a virus that computer users to prevent is the virus ability to replicate. Once a virus has erased a file, made a hard drive inoperable, it is detected. Once the virus has done anything considered malicious, it usually is detected. At this point anti-virus software and hardware must be brought in and run to detect and clean files. Prior to its performing this malicious act, a virus must replicate. If it does not replicate, it cannot grow and stay alive. If it has the ability to replicate, it can travel from PC to floppy to PC to network, etc. It is this behavior of viruses to replicate that the present invention preys on. The virus is encouraged to act within this cross platform generated emulation so that it can be detected. It is this use of cross platform technology and offensive strategy that allows a virus to be detected at any level before any damage occurs to the protected system. It is in the emulation that the invention can detect the virus and in the use of transplatform logic/environment that it can safely contain the virus. Where the virus can get around DOS or MAC scanners or Operating System or BIOS, it cannot infiltrate and contaminate the foreign operating system.

A foreign operating system is chosen based on its ability to monitor and watch any emulations, and for being able to manipulate elements within the emulation (files, falsifying BIOS information, creating sham peripherals), and for the sheer speed and computational horsepower.

The inventors recognize that it can be done without a transplatform, but it will be slow and absolutely unsafe. The use of a foreign operating system can be likened to the use of lead walls and glass walls and mechanical arms used by people manipulating radioactive materials in labs. While it is certainly possible to pick up radioactivity with one's bare hands, it is not highly recommended or is

it safe. While the invention can be had without the use of a foreign operating system, it is not highly recommended nor is it safe.

A primary object of the present invention is to provide a virus detection system to detect and eliminate viruses at their most basic level by simulating the host's environment by creating a virtual world to fool the virus into thinking it is resident on the host so as to allow disruptive behavior to be detected and the virus destroyed without harm to the host.

Another object of the present invention is to provide a virus detection system able to detect and trap viruses at any level using in a way other than performing string searches through memory or files to detect viruses.

Yet another object of the present invention is to provide a virus detection system able to detect as of yet unknown viruses thereby obviating the need for software updates to keep the detection device current.

Still another object of the present invention is to minimize the down time of the host computer system in the event a virus is detected.

Still another object of the invention is to record at the user's discretion the virus to another media for transferal to virus analysis groups. The object is to feed the virus to an internal analysis to compare against a know, previously acquired attempt, such as a trapdoor or file change, or industrial espionage or sabotage code, etc.

Still another object is to record from which incoming source the virus came, i.e., modem, which digiboard channel, internet, Compuserve, LAN station/Usenet, WAN line, etc.

Another object is to alert system administration of the attack.

BRIEF DESCRIPTION OF THE DRAWINGS

Serving to illustrate exemplary embodiments of the invention are the drawings of which:

Fig 1 is a high level functional block diagram of the preferred embodiment of the present invention.

Fig. 2 is a functional block diagram of the preferred embodiment of the present invention;

Fig. 3 is a functional block diagram showing the application of the present invention in a local area networking environment;

Fig. 4 is a functional block diagram showing the application of the present invention in a telecommunications networking environment;

Fig. 5 is a high level software logic diagram showing the operating steps of the present invention;

Figs. 6A to 6C together comprise a high level flow chart of the operating steps of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In order to afford a complete understanding of the invention and an appreciation of its advantages, a description of a preferred embodiment of the present invention in a typical operating environment is presented below.

Operating on the principle that a virus cannot cross operating systems, the present invention creates a virtual world for a potential virus. An OS that emulates the system to be protected provides a friendly familiar environment for the virus. The virus is encouraged to act in this virtual world created for it. The results of the virus' disruptive behavior can be detected and consequently the virus can be flagged and eliminated or stored and further analyzed. This scheme is based on the assumptions that almost all viruses are executable in nature, no user would try to purposely communicate a destructive virus to another and that it is possible to identify executable instructions in an environment where the instruction cannot possibly operate.

Shown in Figures 1 and 2 are functional block diagrams of the virus trapping device 10. The Central Processing Unit (CPU) 12 can be any computing device (i.e. Intel, Motorola, Paramid, National Semiconductor or Texas Instruments microprocessor, multiple chip set CPUs, board level CPUs, etc.). The Transputer is particularly well suited because almost all PCs in use

today employ CPUs other than the Transputer. A guide to the application and programming of the Transputer can be found in The Transputer Handbook, by Mark Hopkins, copyright 1989 INMOS Ltd. and The Transputer Databook, by Mark Hopkins, 3rd Edition copyright 1992 INMOS Ltd. Italy. As a typical microprocessor circuit design, EPROM 14 holds the operating software for the CPU 12. RAM 16 provides a temporary storage facility for the CPU 12 to execute the virus detection software. Link adapters 20 provide physical connections to interface the virus trapping device 10 to the outside world. The trap device 10 is not limited to two link adapters, any number could be implemented to handle a multitude of input data streams. The device 10 reads an incoming data stream from one or more outside sources. An example of a communication link 24 are a Local Area Network (LAN) (i.e. Novell), Wide Area Network (WAN) (i.e. networked LANs), the telephone network (i.e. Modems), radio frequency (RF) type cellular network or some type of data storage device (i.e. floppy diskette, hard disk, tape, CD-ROM, magneto-optical, etc.). The communication link 24 provides an incoming data stream for the device 10 to operate on. Diskettes are commonly used to transfer data and programs from one computer to another, thus making it a common entry point into the system for viruses. An input/output (I/O) interface 18 provides a means for the virus trapping device 10 to communicate with the computer system being protected 28.

The application of the virus trapping device 10 in a typical operating environment is shown in Figure 3. The file server 42 is the computer system to be protected. The virus trapping device 10 is placed in the data stream that connects the file server 42 to other workstations 38. The hubs 40 serve to connect the workstations 38 into a LAN and the modems 36 serve to connect remote workstations 38 to the file server 42. In this scenario, all traffic to and from the file server 42 is monitored for viruses by the trap 10.

Another application of the trapping device 10 is shown in Figure 4. In this scenario, data traffic passing through the telecommunications network 34 is protected from viruses. A user might have a mainframe file server 30 at a remote site connected to the telephone network 34. Nodes 32 located in the telephone company's central offices perform access and cross connect functions for customers' data traffic. To prevent the spread of a virus through the network, the trapping device 10 is placed in front of each node 32. Data traffic between workstations 38 connected to the telephone network 34 via modems 36 and the mainframe file server 30 is constantly checked for viruses because the traffic must pass through the virus trapping device 10.

Operation of the virus trapping device 10 is as follows. The trapping device 10 monitors the data stream that enters from the outside world, such as from the communications link 24. All data is treated as data whether it is actually data (i.e. data files) or instructions (i.e. executables) as it passes over the link 24. At this point the actual instructions have not been executed but rather they are in the process of being transmitted for execution. While in this state of transmission, emulation means 48, controlled by the CPU 12, provide a friendly environment for a potential virus. The data is put into the emulation chamber 48 where the virus is fooled into acting as if it were really present on the host system. It is desired that any disruptive behavior the virus is capable of displaying take place in emulation chamber 48 such as replicating, attacking another program or destroying data. In this virtual world the virus has complete access to its environment. It is at this point that analysis and detection means 50 controlled by the CPU 12 catches the virus in the act of self replication and prevents it from infecting the host system. The virus cannot escape the emulation box 48 because the box exists in a foreign operating environment with no access to critical files, keyboard, screen, etc. Access to the real world is completely blocked.

Upon startup of the trapping device 10, the emulation software is read from EPROM 14 and executed. When a user turns on his workstation 38, a connection is established between the workstation 38 and the file server 30 (or 42). A connection session is created in the RAM 16 of the CPU 12. In like fashion, a session is created for each user.

As the user at a workstation 38 runs commands and moves file about, data is ultimately written to and read from the file server 30. The trapping device 10 splits the data into two paths. One path connects directly to the protected computer system 28 without modification. Data over the other path is written into the emulation box or virtual world created for each user. The write is performed in this box just as it would have been performed on the file server 30, protected computer 28 or workstation 38. Changes in data and time are simulated to trigger time sensitive viruses, fooling them as to the actual data and time. If the environment changes, it is checked to determine whether simply data was written or whether executable code was written.

Once the executable is inside the emulation box, a Cyclic Redundancy Check (CRC) is made of the Interrupt Request table (IRQ). Also, CRCs are generated on all files that are placed in the emulation box. The CRC is an error detection and correction code widely used in the computer and engineering fields. Other aspects of the environment, such as available memory, are saved too. All information saved is stored outside of the emulation box where it cannot be altered by a virus. The executable is forced to run.

If absolutely nothing happens, a self replicating virus does not exist. If anything within the environment changes (i.e. size of files, sudden attempts to write to other executables in the emulation box, etc.) it is determined that a virus does exist and is attempting to self replicate itself.

The first step is to determine whether the IRQ table was modified. The second step is to determine if another program was written to. Many programs attach themselves to IRQs (i.e. network shell programs, mouse drivers, some

print drivers, communication and fax drivers). However, none of these programs will try to write code to other executables. No legitimate program will attempt direct changes to the File Allocation Table (FAT) or other internal OS disk area. They typically pass their changes (or writes) through standard well behaved DOS interrupts (INTs) (i.e. INT 21). Or, for example, in the case of file repair programs (i.e. Norton Utilities) which do at times write directly to the FAT, they will also not grab IRQs. It is the combination of grabbing one or more IRQs and attempting changes to either the FAT or executables that allows virus activity to be detected.

In the architecture of the IBM PC, for example, IRQs are prioritized and have different dedicated purposes. IRQ 0 is the system clock, IRQ 1 is the keyboard, etc. Almost no program needs to grab IRQ 0 having the highest priority, however a virus must. A virus must grab the highest priority IRQ because if it had a lower IRQ, then a conventional anti-virus program can get in at a higher priority and make the virus more vulnerable to detection. Many viruses grab several IRQs, allowing a virus to be detected by its 'signature'. In addition, most programs except viruses return to DOS about 95% or more of the memory they used for execution upon exiting or receiving an unload instruction. Therefore the following activities, monitored in the virtual environment created in the emulation box, can be used to detect viruses: attachment to IRQs, which IRQs have been attached, whether multiple IRQs have been attached, changes to the FAT, changes to executables, changes to the environment, changes to memory and any Terminate and Stay Resident (TSR) activity after the unload command has been issued and the program should have terminated.

In addition a further series of checks can then be initiated: check the "hard drive" and look for additional sectors or blocks being marked "bad" which were good before or vice versa. Has the program attached itself to the internal clock and is it incrementing its own internal clock? Have any of the error-checking algorithm results changed?

Upon detection of a virus by the analysis and detection means 50, response/alarm means 52 can execute any number of user definable optional commands such as messaging or beeping a system administrator, notifying the sender and receiver of the file or program, deleting the file, writing to a specially prepared floppy drive, calling a pager with a virus message or shutting down a network segment. A logic flow diagram showing the operating steps the trap device 10 performs is shown in Figures 6A to 6C.

A high level logic diagram of the software is shown in Figure 5. The input data stream is generated by communication links 24. Link adapters 20 convert the data input stream from a hardware and software protocol specific to the particular communication link (i.e. X.25, Novell IPX/SPX, Microsoft NetBEUI, etc.) to a common protocol understandable by the CPU 12. After protocol conversion, the data packets are disassembled into a data stream having a common data format the CPU 12 is able to understand. The data is then processed and analyzed for the presence of virus activity. Following processing, data packets are re-assembled and converted to its original hardware and software protocol by the I/O Interface 18 before being output to the protected computer system 28.

The trap device 10 passes data directly through to the host system in addition to simultaneously processing it. This is to reduce the processing delays associated with sending large data files to the host system and having the trap device 10 process this data before the host receives it. The entire contents of a large file except for the final write command or the file close command is transmitted to the host. If no virus is detected, the write or close command is issued. If a virus is detected, the write or close is never issued and the response/alarm means 52 takes appropriate action.

It is clear that the above description of the preferred embodiment in no way limits the scope of the present invention which is defined by the following claims.

What is claimed is:

1. A computer virus trapping device comprising:

link adapter means connected to a source of data input for converting external protocols into a data format understood by said trapping device;

emulation means connected to said link adapter means for accepting said data stream from said link adapter means; said emulation means providing an environment isolated from a protected computer system that simulates the architecture of said protected computer system whereby a computer virus is coaxed into performing its intended activity; and

detection means for monitoring said emulation means and determining when said computer virus either has performed or is performing its said intended activity.

2. The device of claim 1, whereby said emulation means comprises processing means suitably programmed to create a virtual world for said computer virus that simulates said protected computer system.

3. The device of claim 2, whereby said processing means comprises a microcomputer circuit, temporary and permanent data storage and an I/O interface.

4. A computer virus trapping device comprising:

link adapter means connected to a source of data input for converting external protocols into a data format understood by said trapping device;

emulation means connected to said link adapter means for accepting said data stream from said link adapter means; said emulation means providing an environment isolated from a protected computer system that simulates the architecture of said protected computer system whereby a computer virus is coaxed into performing its

intended activity;

detection means for monitoring said emulation means and determining when said computer virus either has performed or is performing its said intended activity; and

response means responsive to said detection means to take action according to preset user instructions upon said detection means determining said computer virus exists.

5. A computer virus trapping device comprising:

link adapter means connected to a source of data input for bidirectionally converting external protocols into a converted data format understood by said trapping device;

emulation means connected to said link adapter means for accepting said data stream from said link adapter means; said emulation means providing an environment isolated from a protected computer system and simulating the architecture of said protected computer system so as to coax a computer virus into performing its intended activity;

detection means for monitoring said emulation means and determining when said computer virus either has performed or is performing its said intended activity;

response means responsive to said detection means to take action according to preset user instructions upon said detection means determining said computer virus exists; and

I/O buffer means for reassembling said converted data back into said external data stream protocol and delivering said data stream to said protected computer system.

6. The device of claim 5, whereby said emulation means comprises

microprocessor means programmed to simulate the environment of said protected computer system.

1/8

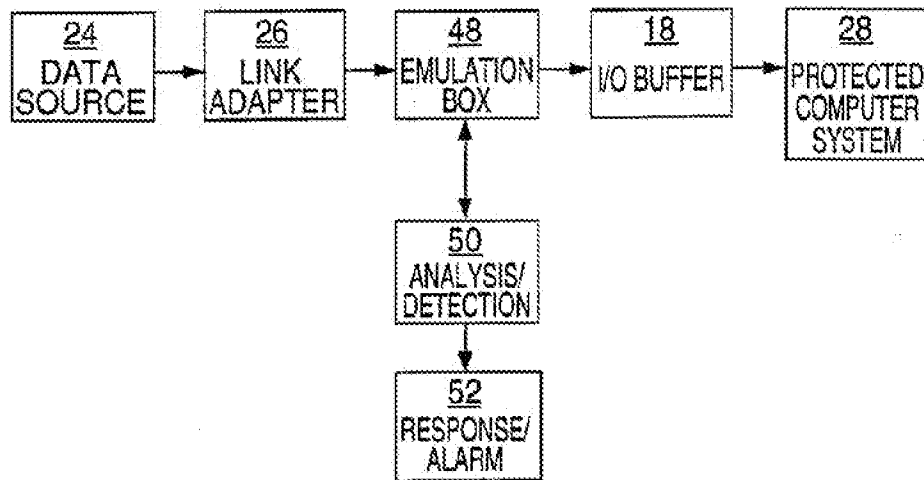


FIG. 1

SUBSTITUTE SHEET (RULE 26)

DEF-FIN00007717

2/8

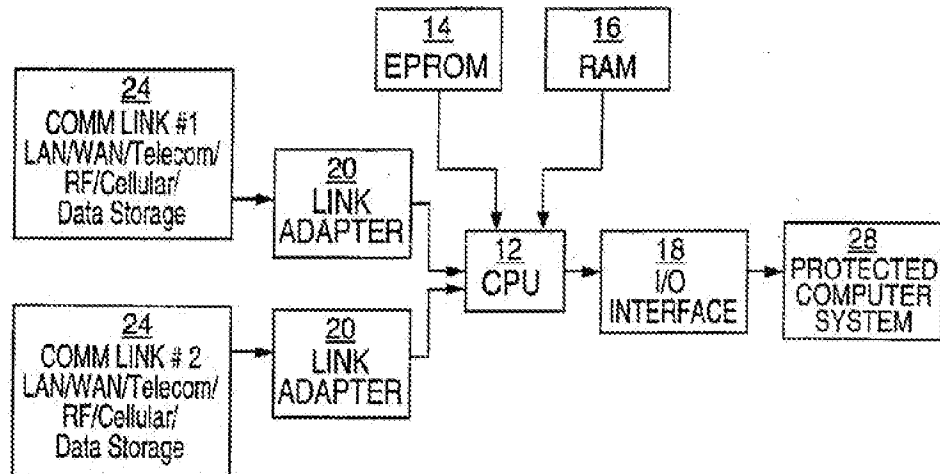


FIG. 2

SUBSTITUTE SHEET (RULE 25)

DEF-FIN00007718

3/8

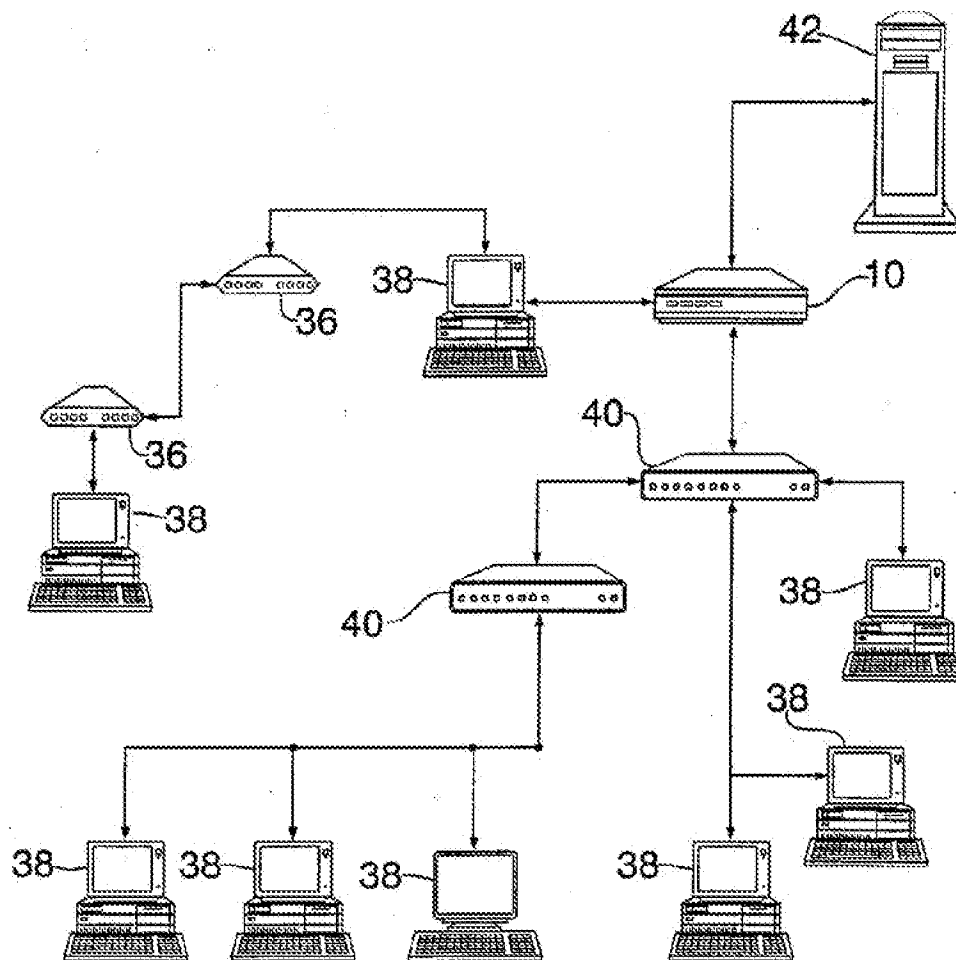


FIG. 3

SUBSTITUTE SHEET (RULE 26)

DEF-FIN00007719

4/8

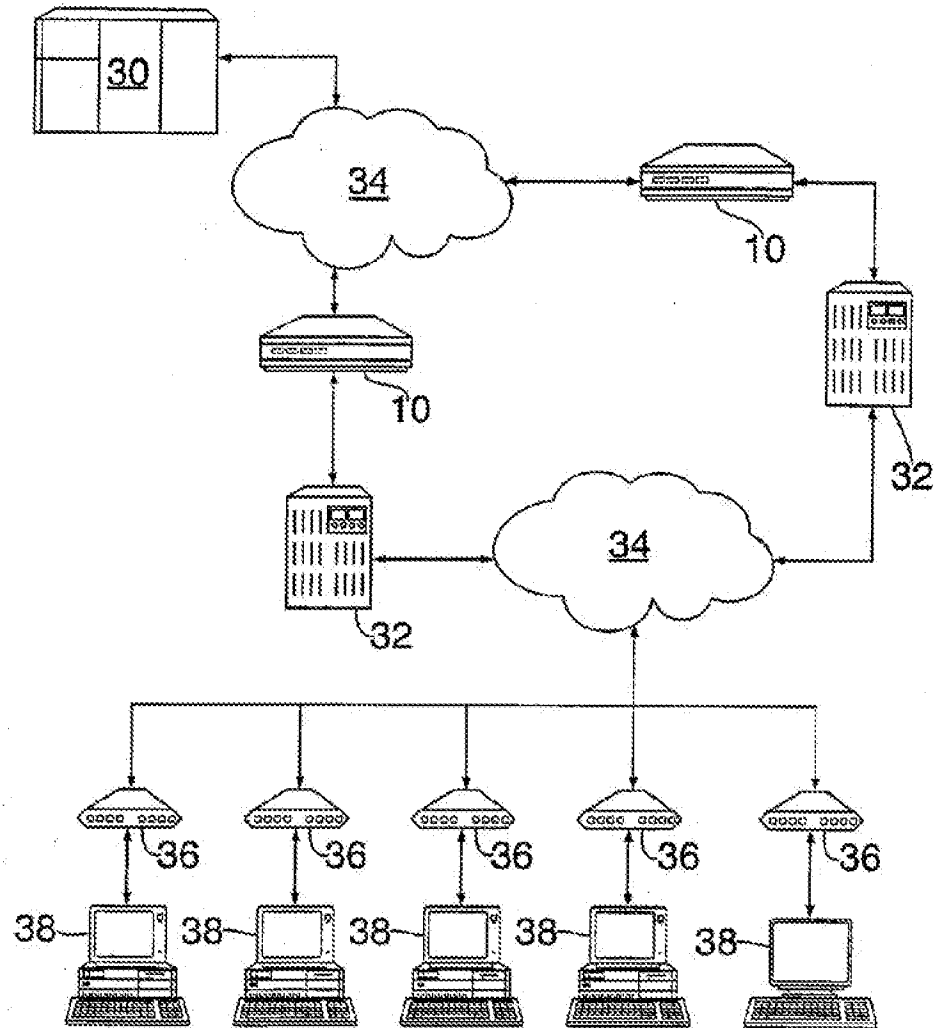


FIG. 4

SUBSTITUTE SHEET (RULE 26)

DEF-FIN00007720

5/8

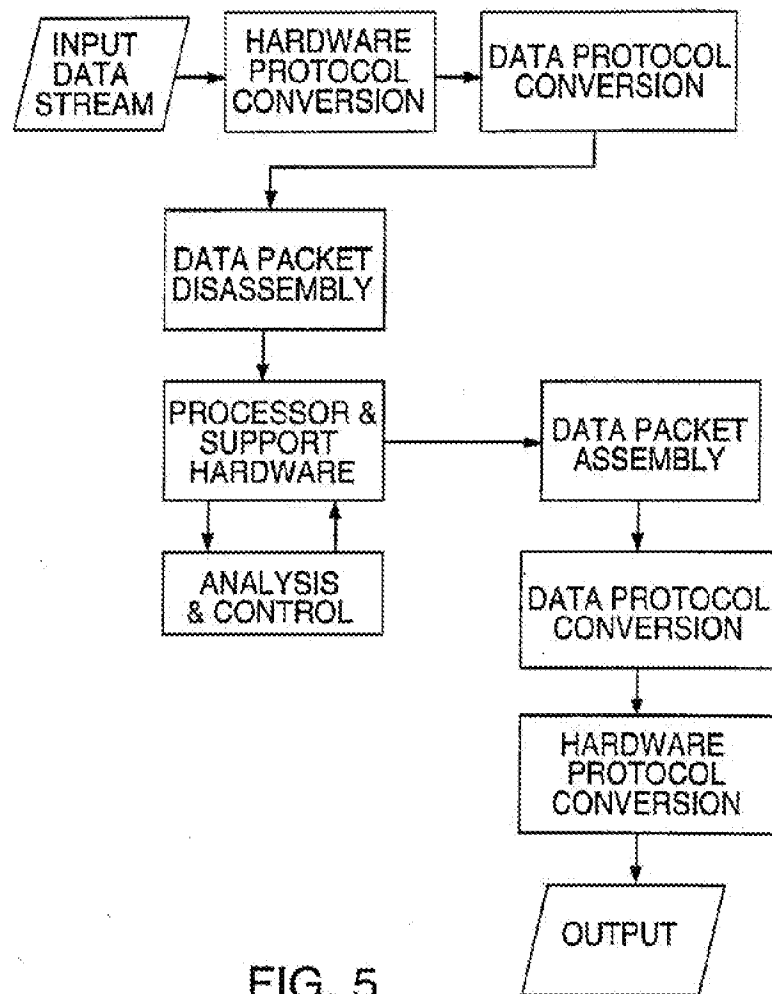


FIG. 5

6/8

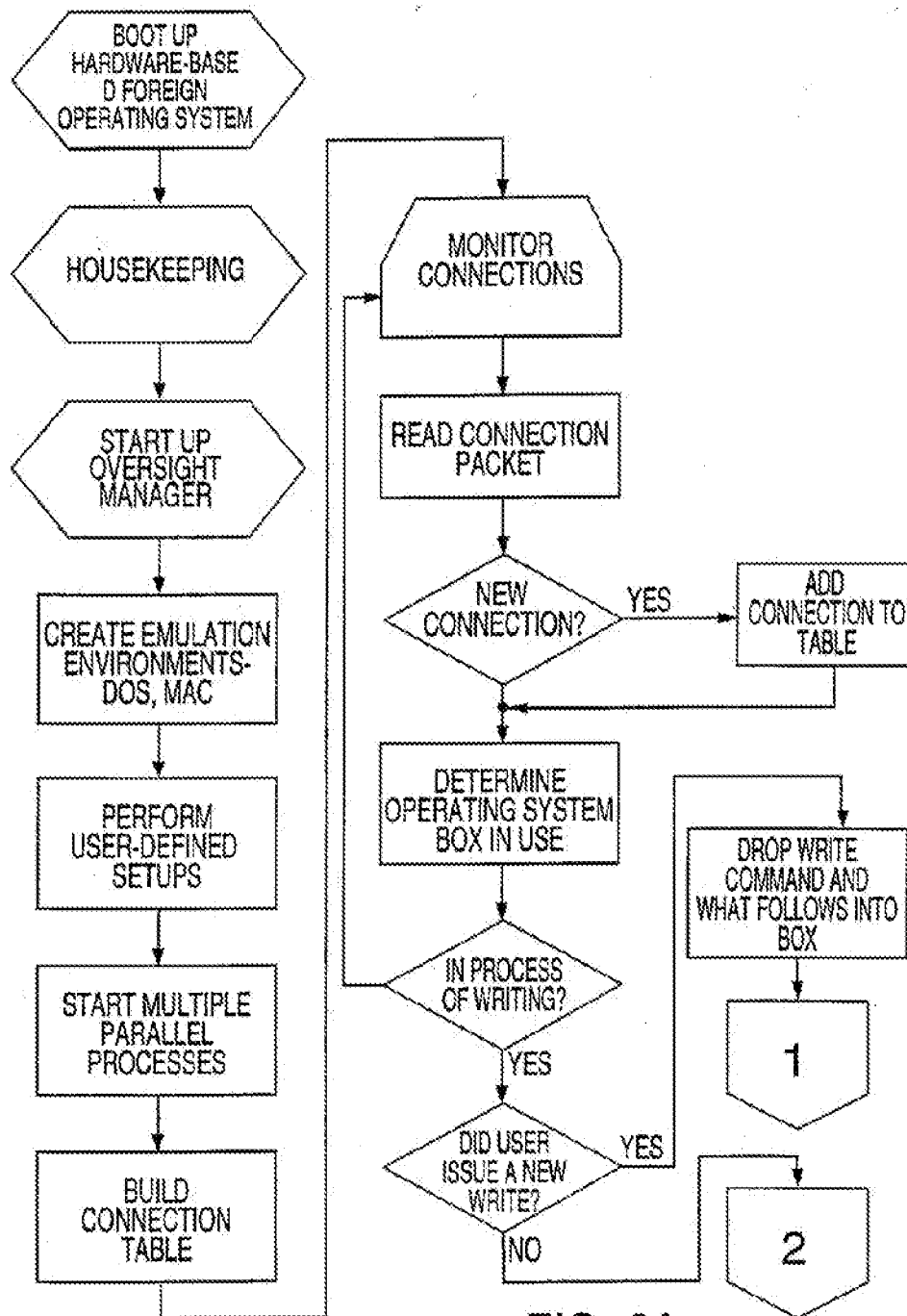


FIG. 6A

SUBSTITUTE SHEET (RULE 26)

DEF-FIN00007722

7/8

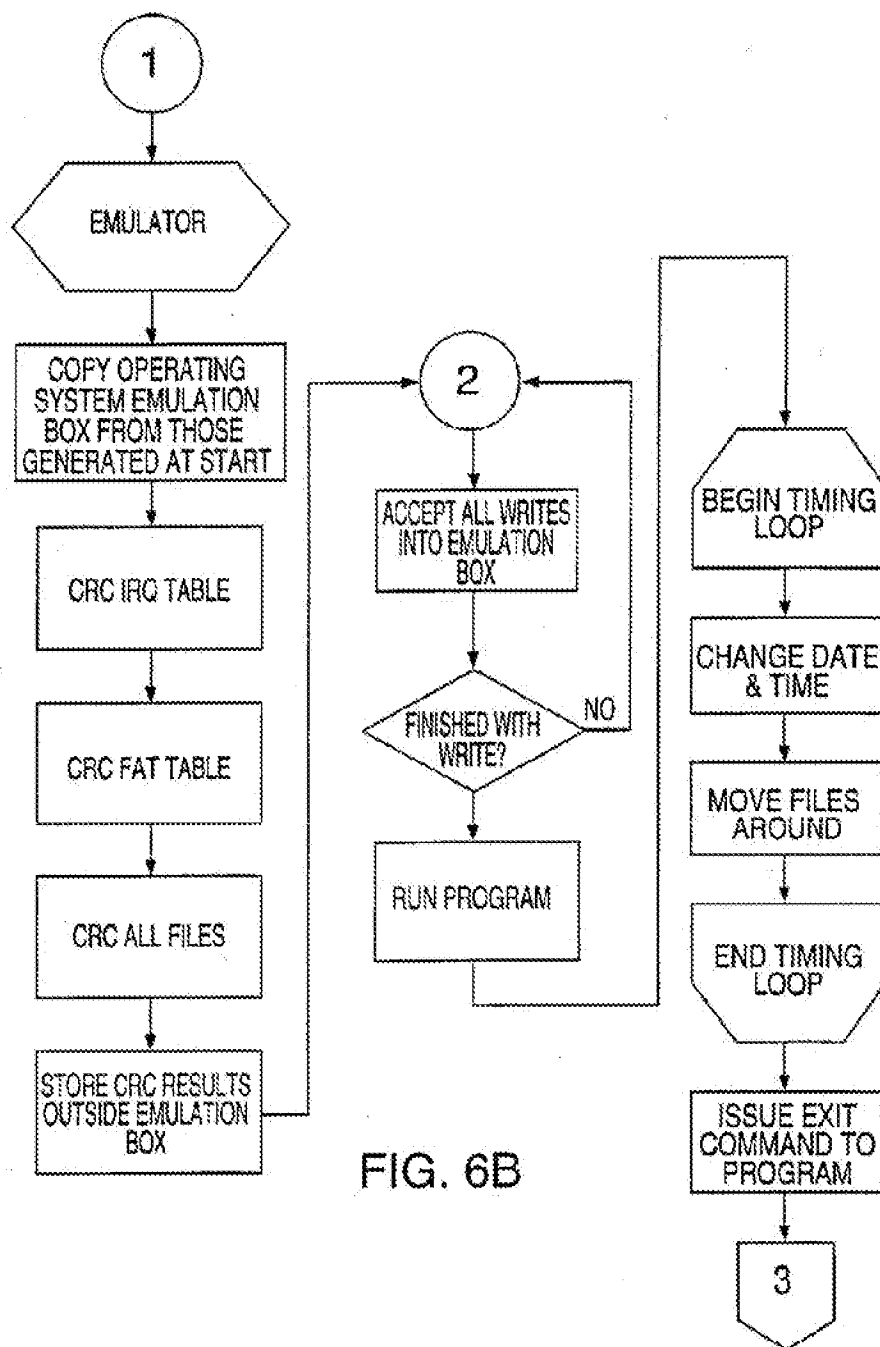


FIG. 6B

SUBSTITUTE SHEET (RULE 26)

DEF-FIN00007723

8/8

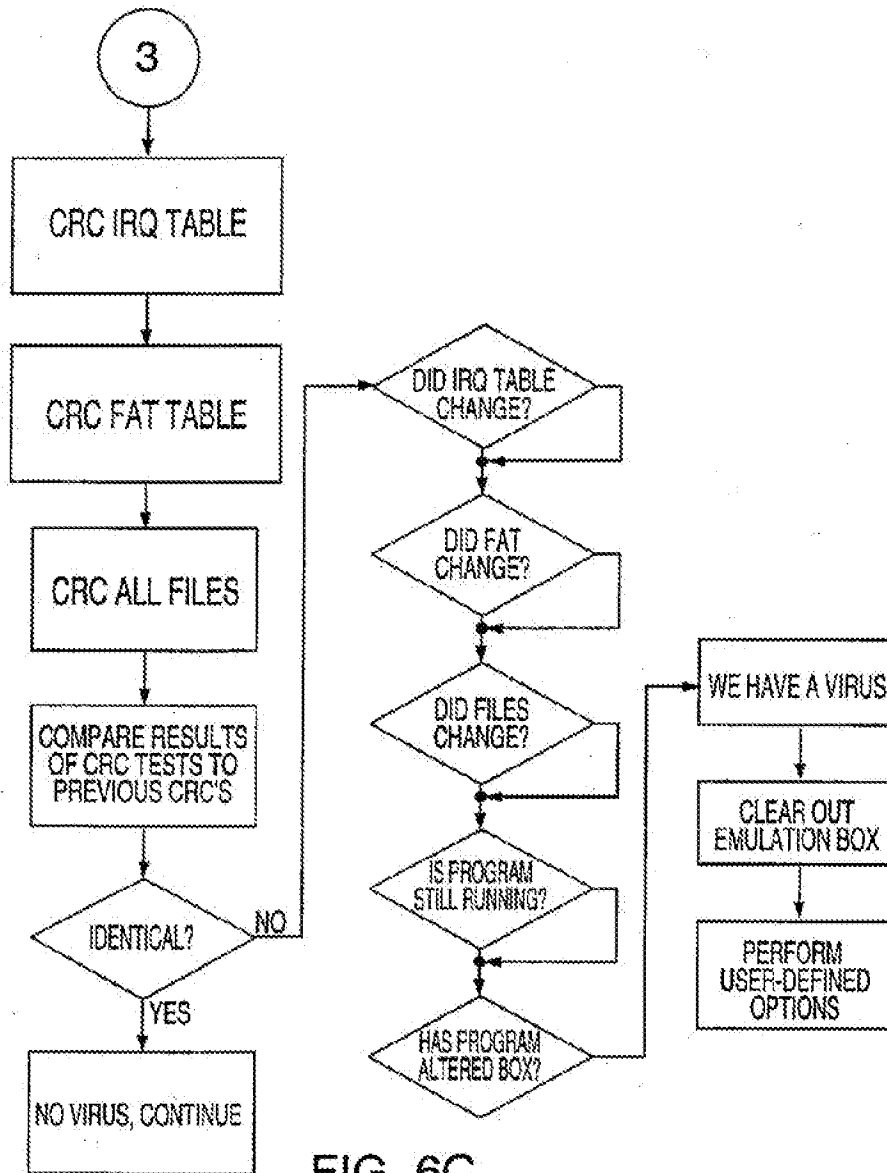


FIG. 6C

SUBSTITUTE SHEET (RULE 26)

DEF-FIN00007724

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US95/06659

A. CLASSIFICATION OF SUBJECT MATTER IPC(6) : G06F 11/00, 17/00 US CL : 395/500 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. : 395/500, 575; 371/16.2, 23 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) APS, DIALOG,		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,P	US, A, 5,398,196 (CHAMBERS) 14 March 1995, col. 3, lines 38-63, col. 4, lines 1-14, 58-et seq. and the claims.	1-6
A, E	US, A, 5,440,723 (ARNOLD ET AL) 08 August 1995, see the entire document.	1-6
A	US, A, 5,274,815 (TRISSEL ET AL) 28 December 1993, see the entire document.	1-6
A,P	US, A, 5,359,659 (ROSENTHAL) 25 October 1994, see the entire document.	1-6
☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.		
* Special categories of cited documents: "A" documents defining the general state of the art which is not considered to be part of particular relevance "D" earlier documents published on or after the international filing date "L" documents 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" documents 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 documents 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 "Z" document member of the same patent family		
Date of the actual completion of the international search 08 AUGUST 1995		Date of mailing of the international search report 13 SEP 1995
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230		Authorized officer AYNI MOHAMED <i>B. X. Mohamed for</i> Telephone No. (703) 305-9684

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

DEF-FIN00007725



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : G06F 11/00	A1	(11) International Publication Number: WO 95/27249
		(43) International Publication Date: 12 October 1995 (12.10.95)

(21) International Application Number: PCT/US95/04109

(22) International Filing Date: 3 April 1995 (03.04.95)

(30) Priority Data:

08/223,221

5 April 1994 (05.04.94)

US

(71) Applicant: INTEL CORPORATION [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

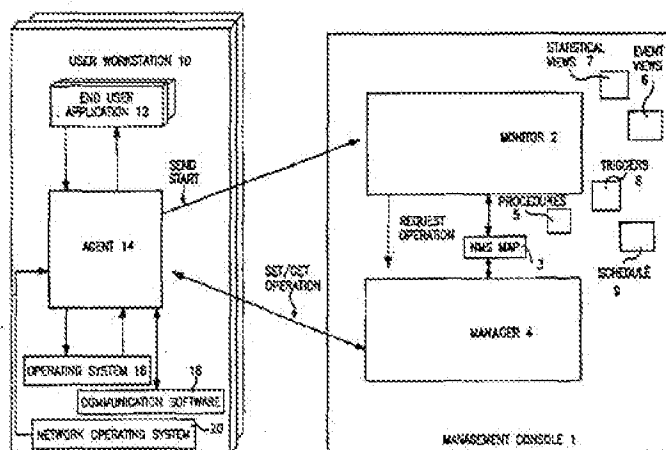
(72) Inventor: TOUBOUL, Shlomo; 42495 Kefar Haim (IL).

(74) Agents: SALTER, James, H. et al.; Blakely, Sokoloff, Taylor & Zafman, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025-1026 (US).

(81) Designated States: AM, AT, AT (Utility model), AU, BR, BG, BR, BY, CA, CH, CN, CZ, CZ (Utility model), DE, DE (Utility model), DK, DK (Utility model), EE, EE (Utility model), ES, FI, FI (Utility model), GB, GE, HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LT, LU, LV, MD, MG, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (Utility model), TJ, TM, TT, UA, UG, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), ARIPO patent (KE, MW, SD, SZ, UG).

Published*With international search report.**Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.*

(54) Title: METHOD AND APPARATUS FOR MONITORING AND CONTROLLING PROGRAMS IN A NETWORK



(57) Abstract

A system for monitoring and controlling at least one program capable of being executed on any one of at least two workstations in a network. The network includes at least one agent module (14) resident on each of the at least two workstations (10) and a management console (1) connected to each of the at least two workstations (10). The system includes modules for identifying an event occurring with respect to a program executing on one of the at least two workstations (10), modules for sending an alert to the management console (1) which identifies the event, memory for storing a plurality of triggers (8), each of the triggers (8) adapted to cause an action to be taken within the network, memory for storing at least one procedure (5) comprising at least one of the plurality of triggers (8), and modules for sending at least one of the procedures (5) from the management console (1) to the agent module (14) resident on the one of the at least two workstations (10) in response to receipt of the alert. A method is also provided for monitoring and controlling programs capable of being executed on any of at least two workstations (10) in a network.

FOR THE PURPOSES OF INFORMATION ONLY

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

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

METHOD AND APPARATUS FOR MONITORING AND CONTROLLING PROGRAMS IN A NETWORK

BACKGROUND

The present invention is directed to a method and apparatus
5 for controlling programs in a network. In particular, the present invention is
directed to a method and apparatus which automatically detects and corrects
error conditions occurring in programs running on network workstations.

Today's networks are expanding in size and complexity. A
network administrator is typically in charge of planning, organizing and
10 maintaining the network. His responsibilities include troubleshooting not
only network hardware and software problems, but hardware and software
problems on each of the workstations in the network. As much as eighty
percent of his time can be spent on troubleshooting problems on the
workstations, including problems specific to each program that the users may
15 be running. Until the network administrator can fix the problem for a user,
the workstation may be down. Such downtime can be costly for any
organization whose operations depend upon proper functioning of the
network and its workstations. Further, because the network administrator
must be able to diagnose and fix any problem that can occur with all the
20 programs that are running on the network, he must be a highly skilled
individual with at least a working knowledge of all network programs.

The present invention relates to a system for assisting the
network administrator in solving the problems encountered in the network.
A number of earlier versions of the program according to the present

invention have been available in the marketplace for more than one year which will detect network problems and report them. The newest of these versions, released November 1992, is AlertVIEW™, Version 2.0, available from Shany, Inc., Mountain View, California. These earlier versions can inform the network administrator that a problem exists with a particular application program running on one of the network workstations. However, the prior versions have only a limited capability in that they can only send a single command, or trigger, to the workstations in response to the detection of the problem, that is, upon receipt of an alert at a management console. In particular, the management console sends a trigger causing one of the following actions to occur: freeze, unfreeze, or reboot a workstation, start and stop a program running in the foreground, send a message, or send any single command in the form of a custom trigger, that the user indicates should be performed in response to specific alerts.

SUMMARY

It is accordingly an object of the present invention to improve upon the earlier versions of the above-noted program in a manner which offers increased flexibility in the handling of problems that occur at workstations.

It is an object of the present invention to provide a network maintenance system which can identify failures of programs running on

network workstations and take the appropriate corrective action to correct the problems that led to those failures.

It is another object of the present invention to provide a system which can correct problems occurring on workstations within the network by sending procedures to agents active on the workstations, each procedure consisting of one or more actions to be taken.

It is another object of the present invention to provide a network maintenance system which allows integrated remote access and control of the network workstations by the network administrator.

It is another object of the present invention to provide a system which allows the network administrator to schedule the automatic performance of network administration and maintenance tasks.

It is another object of the present invention to provide a system which allows the network administrator to automatically send keystroke jobs to the workstations in the network.

It is another object of the present invention to provide a system which allows automatic discovery of agents.

It is another object of the present invention to provide a system which provides specific agents which are developed so as to be tailored to specific applications.

According to one embodiment, a system is provided for controlling at least one program capable of being executed on any of at least two workstations in a network. The network includes at least one agent

4

module resident on each of the workstations and a management console connected to each of the workstations. The system comprises means for identifying an event occurring with respect to a program executing on one of the workstations, means for sending an alert to the management console which identifies the event, and means for storing a plurality of triggers. Each of the triggers is adapted to cause an action to be taken within the network. The system further comprises means for storing at least one procedure, the procedure comprising at least one of the triggers, and means for sending at least one of the procedures from the management console to the agent module resident on the one of the workstations in response to receipt of the alert.

According to another embodiment, a system for monitoring and controlling at least one program in a network is provided. The network comprises at least two workstations and a management console connected to each of the workstations. The system comprises at least one generic agent means resident on each of the workstations for transmitting alerts indicating occurrence of an event with a program executing on the workstation, and monitor means resident on the management console for logging alerts transmitted by any of the agent means. The system further includes means for storing a plurality of triggers to be sent from the monitor means to the agent means, the triggers comprising commands which cause actions to be taken by the agent means in response to the event, means for defining at least one procedure, the procedure including at least one of the stored

5

triggers, and manager means for monitoring and controlling operations of the agent means, the manager means comprising means for sending the procedure to the agent means in response to an alert.

5 According to another embodiment, a method is provided for monitoring and controlling at least one program capable of being executed on any of at least two workstations in a network. The network comprises at least one agent module resident on each of the workstations and a management console connected to each of the workstations. The method comprises the steps of storing a plurality of triggers, each of the triggers
10 adapted to cause an action to be taken within the network, storing at least one procedure, the procedure comprising at least one of the plurality of triggers, identifying an event occurring on one of the workstations, sending an alert to the management console which identifies the event, and sending at least one of the procedures from the management console to the agent
15 module resident on the workstation in response to receipt of the alert.

Still other objects, features and attendant advantages of the present invention will become apparent to those skilled in the art from a reading of the following detailed description of the embodiments constructed in accordance therewith, taken in conjunction with the accompanying
20 drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention of the present application will now be described in more detail with reference to the preferred embodiments of the system, given only by way of example, and with reference to the accompanying drawings, in which:

Figure 1 is a block diagram of the network system in accordance with an embodiment of the present invention;

Figure 2 is a block diagram of an exemplary embodiment of a computer system in accordance with the present invention;

Figure 3 is a block diagram of the generic agent used in the system of Figure 1 in accordance with an embodiment of the present invention;

Figure 4 is a block diagram of the monitor used in the system of Figure 1 in accordance with an embodiment of the present invention;

Figure 5 is a block diagram of the manager used in the system of Figure 1 in accordance with an embodiment of the present invention;

Figure 6 is a flow diagram of a routine for the initialization phase of the generic agent used in the system of Figure 1 in accordance with an embodiment of the present invention;

Figure 7 is a flow diagram of a routine for the operation phase of the generic agent used in the system of Figure 1 in accordance with an embodiment of the present invention;

Figure 8 is a flow diagram of a routine for the fault management phase of the generic agent used in the system of Figure 1 in accordance with an embodiment of the present invention;

5 Figure 9 is a flow diagram of a routine for the controlling and management phase of the generic agent used in the system of Figure 1 in accordance with an embodiment of the present invention;

Figure 10 is a flow diagram of a routine for the operation phase of the monitor used in the system of Figure 1 in accordance with an embodiment of the present invention;

10 Figure 11 is a flow diagram of a routine for the operation phase of the manager used in the system of Figure 1 in accordance with an embodiment of the present invention;

Figure 12 is a block diagram of the network system for a specific agent in accordance with an embodiment of the present invention;

15 Figure 13 is a flow diagram of a routine for the operation of the specific agent of Figure 12 loaded as a non-TSR application in accordance with an embodiment of the present invention;

Figure 14 is a flow diagram of a routine for the initialization phase of the operation of the specific agent of Figure 12 loaded as a TSR application in accordance with an embodiment of the present invention; and

20

Figure 15 is a flow diagram of a routine for the operation phase of the specific agent of Figure 13 loaded as a TSR application in accordance with an embodiment of the present invention.

GENERAL DESCRIPTION

The present invention relates to a network management tool that provides real-time solutions to network programs' problems by monitoring and controlling programs executing on the workstations in the network. The reference herein to programs includes communication software, network operating systems programs, operating system programs and end user applications programs.

The system according to the present invention detects, reports, corrects and prevents end-user program errors, which represent the most common, difficult, and time consuming class of problems. The system according to the present invention can significantly reduce users' downtime, provide real-time control and support of all network programs, free the network administrator from frustrating hours of troubleshooting, and allow the administrator to manage the network with optimum efficiency. The system transforms troublesome user support tasks into satisfying results oriented experiences, by focusing on the programs.

The system according to the present invention allows the administrator to schedule procedures which automatically initiate housekeeping tasks required to ensure that programs continuously run smoothly. The system retains information for preemptive analysis of the network programs problems, which can be used to prevent problem reoccurrences. This information can also be used to maintain the network by designating scheduling procedures which perform necessary actions on the

workstations to ensure its safe operation and by designating correction procedures which react to the appearance of smaller problems to prevent the occurrence of bigger problems. The system according to the present invention also manages its own scheduled procedures, guaranteeing successful completion for the maintenance and backup tasks.

The system according to the present invention includes an agent resident on the network workstations which actively monitors interaction between the users, applications, and system software, providing the network administrator with real-time detection of problems. Important event details are continually captured, including information that is unavailable or incomprehensible to the users. For purposes of this description, an event is the occurrence of a problem with an interrupt or program on a workstation. The system according to the present invention provides real-time problem solving and provides the administrator with invaluable data that identifies and informs the administrator of the problem, its location, the program during which it occurred, and the recommendation for correcting it. The elimination of countless, frustrating hours of guesswork time is achieved because the system according to the present invention detects the source of the security violations, error messages, frozen stations, and other time consuming problems.

Via the network, the agent according to the present invention transmits an alert containing comprehensive, detailed information about each event as it occurs, allowing the network administrator to focus and address

printing errors, security violations, and other program errors critical to the smooth operation of the network. The system guarantees accurate, complete communication of all reported details. Nothing is lost to limited or inadequate communication with the system's memory capabilities. The system actively reports critical program problems directly to the network administrator, regardless of where he/she is, via electronic mail, modem or network, allowing the administrator to customize the reporting procedures. Complete reporting of information needed to achieve a clear understanding of the problem, including, for example, detailed reporting on the application file servers, print queues, etc. which were involved in the event which caused the alert, allows the network administrator to quickly solve the network's problems.

In a typical network, forty percent of the problems encountered by users of the workstations repeat themselves each day. The system according to the present invention will automatically correct these problems, allowing user support time to be converted into valuable productive time and freeing the administrator from the timely hours of support calls. The system analyzes the existing alerts to determine the appropriate corrective procedure for each event, providing the network administrator immediate solutions without the guess work. The present invention differs from the prior versions of AlertVIEW™ described above in a number of ways, including but not limited to, the ability to send procedures consisting of more than one command, or trigger, to the agent to

correct a problem on the workstation. This provides an advantage in that the programs according to the present invention have increased flexibility, versatility and power.

5 The system according to the present invention allows initiation of corrective action in the background, without having to disrupt the user's programs. Interactive analysis and correction are supported through the system's application control panel and remote access feature, giving the administrator complete control of the user's machine through a simple procedure. The application control panel provides a user interface enabling
10 the network administrator through the management console to control and display various aspects of the end user's machine, including displaying running programs, starting and stopping programs, and redirecting system standard input and output.

For example, users often call network administrators with the
15 complaint that they cannot print. The system according to the present invention automatically solves printing problems, virtually eliminating the single most common source of network and workstation support calls. The agent detects user attempts to use an unavailable printer and reports the condition by sending an alert over the network. Transparent to the user, the
20 system's manager then proactively corrects the problem. Using the agent to redirect the printer, the system automatically connects the user to the network printer, asking the user to retry the print. The system provides a

customized view of printing alerts that allows the administrator to optimize printing setups and prevent further problems.

Another example occurs when a user receives a system message "Too many open files". When a program, such as a complex database query, opens more files than were allocated in system setup, the agent detects the condition even before it is displayed to the user. The system reports a "Too many open files" alert to the system manager, which in turn corrects the situation by automatically triggering the agent to make appropriate changes to the configuration files and reboot the workstation. Since these changes can be made permanent, the system automatically prevents future occurrences.

Another example occurs when access is denied to a prospective user by the network operating system. The system's agent according to the present invention constantly monitors file activity on each workstation, detecting security and access violations as they occur. It reports each event with specific information about the accessed file and the nature of the violation. The system corrects and prevents continued serious security violations by enabling the network administrator to freeze or disconnect the offending workstation prior to the user gaining access to the file and before actually violating his privileges. It further prevents ongoing access problems by providing detailed information about every unsuccessful access attempt to the network administrator.

DETAILED DESCRIPTION

The present invention relates to a method and apparatus for managing the operation of network programs. According to a preferred embodiment, the present invention is implemented using a plurality of software modules to perform the monitoring, managing, and logging tasks.

Figure 1 is a block diagram of the system according to one embodiment of the present invention. The network administrator has access to a computer, preferably not the network server terminal, on which the management console 1 is installed. The management console 1 consists of two main modules, the monitor 2 and the manager 4. The network includes a plurality of user workstations 10. Each workstation 10 includes one or more end user applications 12, an operating system 16, communication software 18, and a network operating system 20.

Figure 2 is a block diagram of a computer system for use in accordance with the present invention. The computer system of Figure 2 is an exemplary embodiment of the workstation and management console computers. The particular configuration of the management console and workstation consoles to be used can be modified by one of ordinary skill in the art once in possession of the present disclosure. Computer 400 includes a main section 402 (shown in dashed lines) and a peripheral section 404. The main section 402 includes the central processing unit (CPU) 406, a hard disk 408, read only memory (ROM) 410, random access memory (RAM) 412 and an input/output (I/O) controller 414. The I/O controller 414

connects the main section 402 to the peripheral section 404. The peripheral section 404 includes, according to one embodiment, a printer 416, a cursor control 418, a keyboard 420, a video monitor 422, a CD-ROM reader 424, and a tape drive 426. The peripherals can be changed according to the needs of the user, as can the components of the main section. According to one embodiment, the various routines and files which implement the invention are stored on the hard disk 408 and the routines are executed by the CPU 406. The cursor control 418 can be a mouse, pen, or any other comparable device.

Returning to Figure 1, the network management system (NMS) map module 3 is connected to the monitor 2 and the manager 4. The module 3 provides mapping integration for NOVELL™'s network management system map. In particular, the routine modifies the icon color on the map to show that the agent is running on the workstation. This module can be run at anytime by the network administrator to update the NMS map. The administrator then has access to the functions provided by the system according to the present invention through the NMS map. When the modified icon is accessed, icons are displayed for each of the administrator functions available through the present invention, such as the monitor and manager user interface.

The system according to the present invention includes an agent 14 which communicates with each of the components of the workstation 10 shown in Figure 1, as well as with the management console

15

1. The agent 14 may be a generic agent, capable of handling errors from many different programs, or a specific agent, designed to address the errors occurring in one specific program. One embodiment of a specific agent is described below.

5 It is also possible to have more than one generic or specific agent executing on a workstation, each generic agent designed for a particular operating systems, such as DOS™, WINDOWS™, or OS/2™, and each specific agent designed to handle a different program. That is, generally, only one generic agent can be active on a particular workstation. However, an exception to this occurs when multiple operating systems are
10 being used. If the workstation runs both DOS™ and WINDOWS™ operating systems, both DOS™ and WINDOWS™ agents will be active on the same workstation. For workstations running OS/2™, DOS™ and WINDOWS™, it is possible to have three generic agents simultaneously for these three
15 operating systems. However, in the latter situation, the OS/2™ agent and the DOS™ agent would not use the same communication protocol.

 The agent 14 described in the present disclosure refers to a DOS™ agent as an exemplary embodiment. However, as discussed above, at least three different generic agents, DOS™, WINDOWS™, and/or OS/2™ may
20 be used. While explaining all interrupts and services, the present disclosure refers to DOS™, since only in that environment do all these things exist. However, since WINDOWS™ operating systems heavily depend on DOS™ for most of their services, the same DOS™ agent can detect, for example,

"ACCESS DENIED" or "FILE NOT FOUND" problems within
WINDOWS™ applications as well. The WINDOWS™ agent complements the
DOS™ agent for those functions which the DOS™ agent cannot perform.
These functions include: determining WINDOWS™ applications' names and
5 supplying them to the DOS™ agent, performing starts and stops of
WINDOWS™ applications, monitoring critical WINDOWS™ resource
utilization, and monitoring general protection traps of WINDOWS™
applications. A WINDOWS™ agent supplies all this information for the
DOS™ agent. The OS/2™ agent differs from the DOS™ and WINDOWS™
10 agents in that it intercepts system or communication dynamic link libraries
(DLLs) instead of intercepting vital interrupts. Generally, however, the
configuration of the three generic agents is the same. According to one
embodiment, the different generic agents can use a predefined interrupt, e.g.
the F2h interrupt, for internal communication between themselves.

15 The agent 14 monitors the applications and the operating
system and when an interrupt is generated, hooks or traps that interrupt and
determines if there is an error condition. If so, the error is recorded as an
alert which is reported to the monitor 2. The alert is sent by the agent to the
monitor 2 and includes identification of the type of problem, the workstation
20 on which it occurred, the name of the program which caused the error, and a
recommended corrective action. This recommended action can be modified
by the administrator before it is executed.

A list of interrupts which are trapped according to one embodiment of the present invention is shown in Table 1.

TABLE 1

INTERRUPT	ACTION
Int 2Fh	multiplexor for DOS™ and WINDOWS™
Int 13h	hard disk handler
Int 21h	DOS™ and WINDOWS™ operating system service calls
Int NOVELL™ 21h	NOVELL™ DOS™ shell services
Int 24h	critical error handler
Int 5Ch	NetBIOS™ communication services
Int 17h	printer interface
Int 7Bh	NOVELL™ Btrieve services
Int 08h	timer handler
Int 28h	idle handler
Int 09h	keyboard
Int 16h	keyboard buffer interface
Any IPX™/SPX™ call	IPX interrupt from NOVELL™
NOS and Communication DLL of OS/2™	

Generally, the monitor 2 logs and reports all alerts coming from the agents 14 and the manager 4 monitors and controls the operation of the agents 14 on each of the workstations 10. Further, the monitor 2 may communicate a request for an operation to the manager 4 when specific alerts are reported by any of the workstations 10. Monitor 2 may also

communicate a request to the manager 4 for an action to be taken on the agent 14 when the schedule 9 indicates that a specific time has arrived for performing a procedure. The manager 4 communicates with the controlled (or managed) workstation 10, and particularly, with the agent 14 via set or
5 get operations. Set operations involve commands that are sent from the manager 4 to the agent 14 to perform some operation on the workstation and get operations involve commands or procedures that are sent from the manager 4 to the agent 14 to request that information be sent back to the manager. These will be discussed more fully below.

10 According to one embodiment, the network administrator has the ability to set up the management console 1 to view subsets of the available information at a given time via a user interface 52 (shown in Figure 4). In particular, the administrator can set up one or more event
views 6 which display different alerts from certain or all workstations in the network. For example, one possible view may show all alerts occurring in
15 any WORDPERFECT™ application running on any of the workstations. In addition, the administrator can set up one or more statistical views 7 which display information on network statistics. In particular, statistical views 7 display information about program fault or alert statistics, such as how many
20 times a program was started, how many different type events happened in the program (for example, how many time "Too many files open" occurred during particular programs or on particular workstations), etc.

The administrator can also attach one or more triggers 8 to any of the views in the form of correction procedures. In particular, triggers are specific commands sent to the manager 4 from the monitor 2, and then to the agents 14, to cause actions to be taken, generally by the agents. For example, if a problem is detected with a WORDPERFECT™ application, the system accesses the appropriate view and sends a freeze workstation trigger to the workstation running the failed application.

The triggers 8 represent stored triggers which, according to one embodiment, are stored in a module called a trigger library (which is in effect a WINDOWS™ DLL). One trigger library may contain more than one trigger. Several trigger libraries may be provided with the system for different purposes. One trigger is the function (or command) which is stored in a trigger library and can be called by the monitor 2 to be executed automatically as part of a procedure. The triggers can be called automatically in two cases, either in a correction procedure executed in response to an alert or in a scheduled procedure executed at a desired time as set up by the scheduling module 900.

Table 2 is a list of predefined triggers which can be included in procedures according to one embodiment of the present invention. The numbers in the first column of Table 2 are for purposes of this description only and have no programming significance.

The predefined triggers can be stored in a trigger library module as described above, which is stored in memory, for example, on the

20

hard disk 408 or ROM 410. The network administrator can also set up and design custom triggers by choosing a setup trigger option from the monitor menu user interface 52. These custom triggers can be commands to execute available programs, such as NORTON™ Utilities. Alternatively, the administrator can use a software developer's kit to create a new custom trigger by writing a new program to execute the desired functions.

TABLE 2

LIST OF TRIGGERS TO BE SENT TO AGENT	
1	change files in CONFIG.SYS
2	add line to AUTOEXEC.BAT
3	install driver
4	check disk
5	send message
6	send keystroke job
7	start program
8	stop program
9	send SNMP trap
10	send alert via modem/cc:mail/pager
11	freeze workstation
12	reboot workstation
13	run program on local workstation
14	unfreeze workstation
15	generate nms alarm
16	copy files
17	set last drive

LIST OF TRIGGERS TO BE SENT TO AGENT	
18	set disk buffers

Table 3 is a list of additional triggers available according to another embodiment of the present invention. Again, the numbers in the first column in Table 3 are for purposes of the description only and have no programming significance.

TABLE 3

LIST OF TRIGGERS TO BE SENT TO AGENT	
19	stop program by name
20	pause
21	stop failed program
22	set number of NCB
23	copy alert to database, text or printer file

The lists in Tables 2 and 3, which will be discussed together for simplicity, are lists of possible triggers that can be included in the correction procedures or in scheduled procedures. These lists are not meant to be exhaustive but are exemplary of the types of operations that could be performed. The implementation of other triggers is within the skill of the ordinary artisan once in possession of the present disclosure.

Trigger 1 causes the manager 4 to notify the agent 14 to change the specified number of files in the CONFIG.SYS file. This trigger

would be implemented if the agent generated an alert that indicated that the program required more files open than previously specified, for example, in response to detection of an error code from Interrupt 21h. Trigger 2 causes the manager 4 to notify the agent 14 to modify the AUTOEXEC.BAT file as
5 necessary to correct the problem that led to the generation of the alert.

Trigger 3 causes the manager 4 to notify the agent 14 to install a driver into the startup procedure at the workstation at which the alert was generated.

For example, if a printer or a mouse driver is missing and is needed by a particular program, a trigger would be executed to install the missing driver.

10 Trigger 4 causes the manager 4 to notify the agent 14 to check the disk for bad sectors or spots, either at scheduled times or when a serious disk error is reported.

Trigger 5 allows the manager 4 to send messages to any of the workstations in the network. Trigger 6 causes the manager 4 to send a
15 keystroke job, that is, a sequence of keystrokes for execution, to the agent 14 on a particular workstation. Triggers 7 and 8 cause the manager 4 to notify the agent 14 to start a program or stop the program currently executing in the foreground.

Trigger 19 causes the manager 4 to notify the agent 14 to stop
20 a particular program by name. This trigger is useful for stopping programs executing in the background operations of the workstation. Trigger 21 causes the manager 4 to notify the agent 14 to stop the program that has failed, that is, the program that was executing when the alert was generated.

Trigger 9 allows the manager to send a simple network management protocol (SNMP) trap. That is, if the network has a SNMP management console, a trap or alert is sent to that console notifying it of the error. Trigger 10 allows the manager to forward an alert via another electronic medium, providing problem notification to locations not connected to the network. Trigger 11 freezes the workstation to prevent further problems, until the problem which caused the alert is fixed. Trigger 14 would then be used to unfreeze the workstation when the problem had been corrected.

Trigger 12 causes the manager 4 to notify the agent 14 to reboot the workstation as necessary. For example, if the AUTOEXEC.BAT or CONFIG.SYS files had been altered by triggers 1 and 2, the reboot trigger would be executed to allow the changes to take effect. Trigger 13 causes the manager 4 to notify the agent 14 to run a particular program on the workstation, supporting such actions as custom notification via audio or visual alarms. Trigger 15 allows the manager 4 to generate a NOVELL™ management system (NMS) alarm, to notify the NOVELL™ management system that an alert has occurred on a workstation.

Trigger 16 causes the manager 4 to notify the agent 14 to copy files to the workstation which generated an alert as necessary to correct the problem, for example, to transfer device drives, program modules or data files to the workstations. Trigger 17 causes the manager 4 to notify the agent 14 to set the last drive in CONFIG.SYS. Trigger 18 causes the

manager 4 to notify the agent 14 to set the number of disk buffers in
CONFIG.SYS.

Trigger 20 causes the manager 4 to pause a procedure for a
period of time. This could be used to wait until the workstation has
5 completed a portion of a procedure. Trigger 22 causes the manager 4 to
notify the agent 14 to set the number of network control blocks (NCB) in
CONFIG.SYS. This would be used in response to a problem with
NetBIOS™, and particularly, interrupt 5Ch. Trigger 23 allows the alert to be
copied to a database, text or printer file for review or storage.

10 Returning to Figure 1, automatic procedures 5 may be set up
by the network administrator which consist of a set of triggers, with
specified parameters, chosen from one or more trigger libraries. The
procedures each consist of one or more triggers which triggers are executed
in sequence in response to an event or at scheduled times. Accordingly,
15 such procedures can be attached to either event views 6 or to the schedule 9.
In the first case, the procedures are referred to as correction procedures. In
the second case, the procedures are referred to as scheduled procedures. Of
course, the same procedure may be used both as a correction procedure and
a scheduled procedure.

20 The triggers in the procedures may either be one of the
triggers listed in Tables 2 and 3 with their associated parameters, or they
may be custom triggers. For example, "change FILES in CONFIG.SYS"
supplied with parameter 50 becomes a command to set FILES=50 in the

CONFIG.SYS file. In this way, Tables 2 and 3 list commands for building procedures.

In any case, the monitor 2 automatically and sequentially calls the triggers according to the procedure in which they are contained.

5 According to one embodiment, the procedures execute in the background on the workstations so that they are not visible to the user of the workstation. The monitor 2 calls all the triggers from a correction procedure at the time when a new alert arrives at the event view with which the correction procedure is associated. The monitor 2 call triggers from a scheduled
10 procedure when the time interval specified in the scheduled procedure has passed.

A schedule 9 can be set up by the administrator to schedule procedures to be run on any of the workstations at desired times. For example, the check disk procedure 4, backups and database maintenance
15 functions can be run at scheduled times during the week or month, and/or automatic workstation logoff after a set amount of time can be provided.

Figure 3 is a block diagram of a generic agent 14 according to one embodiment of the present invention. The program 21 module includes all functional blocks in the workstation, except the agents; in particular, the
20 end user application 12, the operation system 16, the communication software 18 and the network operation system 18 shown in Figure 1. There are two ways that the program 21 may communicate with generic agent 14: via the application programmer interface (API) or via the probe modules 32.

26

The program 21 may communicate with the agent 14 via a module 22 which sends an alert to a dispatcher 24 via the API when an error condition occurs. The program 21 is also connected to each of the probe modules $32_1, 32_{i+1} \dots 32_n$, the function of which will be described below.

5 The dispatcher 24 is a decision making procedure that decides which function to execute in the agent based on the communication received from API module 22, receiver module 34, or agent "control" module 26. The dispatcher 24 sends a command to the agent control module 26 once the decision is made. The agent control module 26 controls the operations of
10 the agent through the agent engine module 28. In particular, the generic agent 14 may be instructed by the triggers to perform actions on the operating system or the workstation, such as start program, change file, freeze keyboard, display message, etc. The agent control module 26 performs these functions on the workstation. The probe modules $32_1, 32_{i+1}$
15 ... 32_n are used to hook the interrupts. There is a probe module 32_i for every interrupt handled by the system. The agent engine module 28 performs the processing of events received from the probe modules 32_i and directs actual alerts via the agent control module 26 to the monitor 2 through the network 100.

20 When program 21 accesses the agent 14 via the "send alert API" module 22, the program 21 actually creates and sends a specific application alert. In particular, the generic agent 14 detects generic errors which can happen to any program. The agent also provides an API (send

alert API) which allows any program to create specific alerts and send them via the agent 14 to the monitor 2. For example, the WORDPERFECT™ application could use this API to send an alert concerning a "Paragraph formatting error". The agent 14 transmits this alert to the network, and further to monitor 2 for processing by the system. This case is described above. However, it is also possible for the user programs to access the generic agent 14 indirectly via the probes 32_a. When the program calls the operating system or any other system services, the probes 32_a of the agent 14 get control. This processing is discussed below with respect to Figure 7.

The receiver module 34 and the transmitter module 36 are used by the agent 14 for communicating with the network 100. The receiver module 34 and transmitter module 36 interface with the communication software of the network card. According to one embodiment, the receiver module and transmitter module can communicate using any one of four network communication protocols: IPX™, NetBIOS™, DLC™ 802.2, and TCP/IP™. It is within the skill of the ordinary artisan to provide the capability for communication using other protocols, either protocols known today or developed in the future.

An agent automatic discovery (autodiscovery) module 38 communicates with the transmitter 36 and the receiver 34 to facilitate the automatic discovery of newly activated agents 14 by the manager 4, eliminating the need for the network administrator to perform the tedious process of manually defining each user. The autodiscovery module 38 sends

to the manager 4 an identification packet of the newly activated agent to be discovered by the manager 4. This transmission occurs at the time the new agent is activated and at predetermined intervals thereafter until the autodiscovery module 38 receives confirmation from the manager 4 that the newly activated agent is discovered. According to one embodiment, the predetermined interval is about 60 seconds.

Figure 4 is a block diagram of the monitor 2 according to one embodiment of the present invention. The monitor 2 communicates with the generic and specific agents on the individual workstations through the network 100. Information is received from the network 100 through a receiver 40. According to one embodiment, the receiver 40 has the capability noted above with respect to receiver 34. In particular, the receiver allows communication using any one of four of network communication protocols. Of course, the receiver 40 could alternatively be provided to only allow communication using one of the protocols. The receiver 40 transmits the alerts received from the network 100 to an event log manager module 42. All of the alerts received from the monitor 2 are stored in the event log database 44 by the event log manager module 42. The event log manager module 42 can export data into DBASE™ format or several other database formats.

According to one embodiment of the present invention, the event log database 44 consists of two files, an event details file and an event index file, shown in Tables 4 and 6, respectively. Table 5 shows the alert

details structure. The event details file contains variable length records,
while the event index file contains fixed length records.

TABLE 4

SIZE	FIELD NAME
word	alert_ord
evnidx	alert_idx
byte	alert_det[maxdetail]

TABLE 5

2b	detail length	2b
detail length + 4	alert SNA subvectors	detail length + 4

TABLE 6

SIZE	FIELD NAME
byte	ri_type
byte	ri_reserv
long integer	ri_det_off
word	ri_det_len
word	ri_idx
word	ri_idxid
byte	ri_gtype
byte	ri_time[3]
byte	ri_date[3]
byte	ri_adapter[macaddr_size]
word	ri_segment
byte	ri_prname[prname_size]
byte	ri_resname[resname_size]

SIZE	FIELD NAME
byte	ri_restype

In Table 4, the alert_ord field, consisting of one byte, stores an alert ordinal number which is unique for each alert. The alert_idx field contains the alert key fields of Table 6 having a size evnidx equal to the size of the event index record. The alert_det field, having a field size of maxdetail, contains the alert details, in the layout shown in Table 5. In particular, the first and last 2 bytes contain the detail length plus four bytes, and the middle "detail length" bytes consist of the alert SNA subvectors. The SNA subvectors refer to the SNA network management vector transport (NMVT) system used by IBM for alerts within networks.

In Table 6, the ri_type field is the record type field for identifying the record type. In particular, according to one embodiment of the present invention, most of the records in this database are events that are reported by the agents. Some of the records are generated by the monitor. An example of this latter type of event is congestion in the network. Which of these types of records is identified by the record type field. The ri_reserve field is reserved for future use. The ri_det_off field is the detail record offset, that is, the location within the event details file. The ri_det_len field is the detail record length.

The ri_idx is a hexadecimal number assigned to each alert representing the textual description which is used to identify the alert. All of

the ri_idx values for each possible alert are stored in a separate database which database is indexed by the ri_idxid value. The ri_idxid value is thus the index number which refers to the particular alert entry in the index database which is part of the user database 74 (in Figure 5).

5 The ri_gtype field contains the type of event, which is used for filtering the reported alerts. The ri_time and ri_date fields indicate the time and date at which the event causing the alert occurred. The ri_adapter field contains the adapter number representing the actual address of the workstation from which the alert was generated. The ri_segment field is the
10 segment number indicating the segment of the network in which the workstation is located. The ri_proname field indicates the product name of the program which caused the alert. The ri_resname field is the user name of the workstation corresponding to the adaptor number. The ri_restype field indicates the name of the resource being used when the alert was
15 generated, for example, the printer or other system component.

 A message log database 45 is provided which is maintained by the procedures manager 54, described below. The message log database 45 includes the outcome of triggers, that is counters of events, triggers, and the like to show the size of the databases and support the operation of the
20 modules. The contents of the message log database 45 is displayed to the administrator through the user interface 52.

 The statistical views module 700 is accessed by the end user 50 through the user interface 52 to create, modify, delete and access the

desired statistical views 7. The event views module 600 is accessed through the user interface 52 to create, modify, delete and access the desired event views 6. Additionally, the scheduling module 900 is accessed through the user interface 52 to create, modify, delete, and access the scheduled
5 procedures to be sent to the agent by the manager 4 to be executed on the workstation at desired times.

The event log manager module 42 communicates each alert to an event list manager module 46. This module 46 reads each new event and displays it on the display at the computer on which the management console
10 1 resides within one of the views set up by the administrator, after the module 46 filters out those events or alerts which are not requested in the selected statistical view 7 or event view 6.

The triggers manager module 56 allows the administrator to manage, including create, modify, and delete, custom triggers. The
15 correction procedures module 58 allows the administrator to create the procedures, including, as described above, one or more of the triggers, either predefined or custom triggers.

The procedures manager module 54 takes the output from the event views module 600 and the scheduling module 900 and chooses one or
20 more of the procedures to be sent to the manager 4 that are associated with a schedule or selected event views in response to an alert. In particular, the procedures manager module 54 picks up a correction procedure assigned to the view assigned to the alert that was reported and sends that correction

procedure to the manager 4 for execution. Of course, it possible that one alert is assigned to more than one view and more than one correction procedure is assigned to any view. In addition, the procedures manager 54 picks up a scheduled procedure at the scheduled times and sends it to the manager 4 for execution.

Figure 5 is a block diagram of the manager 4 according to one embodiment of the present invention. The network 100 communicates with the manager 4 through a receiver 62 and a transmitter 64 of the type described above with respect to receiver 34 and transmitter 36 shown in Figure 3 for use with the agents 14. The receiver 62 and transmitter 64 communicate with the dispatcher 60. The dispatcher 60 is of the same type as described above with respect to the dispatcher 24 shown in Figure 3. The monitor 2 and the manager kernel 67 communicate with the dispatcher 60 through an API interface 66. The manager kernel 67 performs the triggers set up in the correction procedures upon the occurrence of a triggering event and performs the scheduled procedures at the desired times. This will be described below with respect to Figure 11.

The end user 50 communicates with the manager 4 through the user interface 52. The dispatcher 60 communicates with the menu system 68. The user interface 52 also communicates with the menu system 68 through which the administrator selects the functions to be performed. The menu system 68 allows access to the event tables manager and editor

70, the information manager 72, the agent monitoring module 80, the remote access module 82, and the setup management module 84.

The event tables manager and editor 70 allows the administrator to edit and/or manage the event tables. Event tables are specifications of events to be reported. These tables specify various control functions, most importantly the masking specification to determine which events are to be reported over the network. The information manager 72 allows the administrator to have access to any of a number of databases 74, 76, and 78. The users database 74 stores information regarding the agents, that is, the setup of the agents on the workstations. In particular, the users database 74 stores the login names, and addresses, masks and agent control parameters. The user database also stores the index definitions which is indexed by an index number for each alert as described above. The inventory configuration database 76 stores information relating to the configuration of each workstation. In particular, the information consists of things such as the type of machine, the processor type, the memory size, etc. The setup database 78 stores the customized setup configuration for the manager 4. In particular, the setup database 78 stores a collection of common parameters for the manager 4 such as locations of database files, the set of event tables with which to work, fonts used for text, etc.

Also connected to the menu system 68 is the application control module (ACP) 57. The ACP 57 allows the administrator to view all the programs, device drivers and TSRs currently loaded in a remote

workstation. Further, the ACP 57 starts and stops programs in the background operation of the user application executing in the remote workstation. In particular, if a program is executing on a remote workstation, the ACP 57 can launch and execute capture of the workstation in the background without stopping or pausing execution of the program executing in a foreground operation of the workstation. Using the ACP 57, the administrator can view or update any configuration file on the remote workstation without disturbing the end user by taking control of his workstation, by performing the update/view functions in the background.

This is accomplished slightly differently in DOS™, WINDOWS™ or OS/2™. In DOS™, because it is not a multi-task operating system, the ACP 57 uses the undocumented backdoor feature. In particular, the ACP 57 gains access to DOS™ through a handle known as the backdoor, launches the background application for making the changes, and closes the backdoor and returns control to the foreground application. In WINDOWS™, a second task is started in the background and the focus of the new task is set to allow it to coexist with the foreground application. In OS/2™, the procedure is the same, but the task is referred to as a process.

The agent monitoring module 80 performs the monitoring of the behavior of the agents on the workstations in the network. The agent monitoring module 80 works in conjunction with the autodiscovery module 38 to detect users signing onto the network and maintains information about active agents for the manager 4 and also identifies undefined active agents

(autodiscovery). The remote access module 82 allows the management console to have remote access to each of the workstations, that is, the management console becomes the "master" over the individual workstations. In this way, the administrator can take direct control of the operations on each workstation. In particular, the remote access module 82 takes direct control over the screen, keyboard, and mouse of each of the workstations in a known manner. The setup management module 84 controls the setup of the manager and is thus connected to the setup database 78.

Figure 6 is a flow diagram of a routine performing the initial phases of the generic agent program according to one embodiment of the present invention. This routine is executed whenever the agent is started on the workstation. First the generic agent is loaded as a TSR program (Load Agent).

At step 102, the initialization is begun. The command line is parsed and the routine checks whether the agent is loaded. In addition, the routine checks the existence of the network and the network protocol and the network and OS/NOS resources are allocated as necessary.

The programs according to the present invention, on a NOVELL™ network, are automatically installed on the file server through the file server login script. Accordingly, for NOVELL™ networks, there is no need to install the programs in the hard drive on every workstation. When the user logs in to his/her workstation, the programs according to the present invention are automatically downloaded on the workstation.

Conventionally, the NOVELL™ login script allows a batch file to be executed on the workstation when the user logs in, but a terminate-and-stay-resident (TSR) program cannot be executed because NOVELL™ login automatically clears memory upon termination. The installation program according to the present invention includes a "load tsr" module which is executed in the login script which puts a stamp of which TSR should be loaded in a protected area of memory which is not touched by the NOVELL™ terminate login. Then, when the login script terminates, the installation program takes control from DOS, accesses the protected area and takes the TSR and its parameters and loads them in the clean memory.

If the initialization at step 102 is successful, the connection to the manager is started at step 104. A connection request is sent. The agent waits for a response from the manager and then loads the event table setup and agent parameters. The event table setup and agent parameters are downloaded from the users database 74 to the memory resident agent database 30. The event table setup includes the specifications of filters that control reporting of events over the network. The agent parameters include status and control information about the agent other than filter setup, e.g., frozen/unfrozen status of the workstation.

If this operation is successful, the agent is installed at set 106. In particular, the interrupts described in Table 1 are set up and the agent replaces the interrupts listed in Table 1 for the interrupts normally present on the workstation. In this way, whenever a program calls an interrupt

corresponding to one on the list in Table 1, the agent takes control, in other words, hooks or traps the interrupt, and executes the substituted instructions. The return code is verified and, if the interrupt operation is not successful, an alert is sent to the monitor 2 with a report of the failure. At step 106, the agent also obtains the workstation environment information, installs extended memory system (EMS) support, and installs the API interface.

If the installation of the agent is successful at step 106, a successful start has occurred at step 108. All procedures are started and hooks to software interrupts are enabled. Finally, a return is made to the OS command interpreter.

If any one of the operations at steps 102, 104 and 106 fails, a failure message is printed at step 110. In particular, if step 102 fails, a "fail to initialize" message is sent. If step 104 fails, a "fail to start" message is sent. Finally, if step 106 fails, a "fail to install" message is sent.

Figure 7 is a flow diagram of a routine for the operation phase of the generic agent program according to one embodiment of the present invention. At step 112, if it is detected that the end user program calls OS, NOS, communications, a device driver or any other software interrupt, the operation of the agent begins. At step 114, an OS call to the software interrupt handler of DOS™ and/or OS/2™ DLL/OS service or WINDOWS™ protected mode is executed.

Entry points 116, 117, 118, 120, and 122 illustrate possible options for the events that may occur which would cause the agent to

determine whether an error or unmasked event has occurred at step 126.

Entry point 116 represent the system monitoring and polling functions performed by the generic agent. Entry point 117 represents the interrupt from the communications and network operating systems (NOS) services.

5 Entry point 118 represents the interrupt handler for interrupt 21h, the DOS™ and WINDOWS™ operating system service calls. Entry points 120 represent the interrupt handler for interrupt 2Fh, the multiplexor for DOS™ and WINDOWS™. A plurality of entry points 120 are provided to schematically illustrate that the execution of any one of the interrupts listed in Table 1 may
10 trigger the generic agent to determine whether an error or unmasked event has occurred. Entry points 122 represent all OS/2™ DLL's, and OS kernel services.

 If there is no error with any of these operations or if the event is a masked event, the operating SYSTEM SERVICE is performed and
15 control is returned to the caller at step 128. If an error occurs during execution of one of the programs or functions identified in 116, 117, 118, 120, and 122, and if that error is an unmasked event at step 126, control passes to PRAV001 at step 130 (shown in Figure 8).

 The purpose of the test at step 126 is explained as follows.
20 When the agent detects an error, that is, an event, the agent can decide whether to send a report and alert on the basis of whether the event is masked or filtered. In particular, certain events can be masked at the agent level to prohibit reporting of the event to the monitor. If the event is not

masked at the agent level, the alert is sent to the monitor. In particular, the network administrator can set up filters in the manager to cause reporting of only certain events under certain conditions. This involves the creation of user database 74 discussed above with respect to Figure 5.

5 Figure 8 is a flow diagram of a routine for the fault management phase of the generic agent according to one embodiment of the present invention. Step 130 is the PRAV001 routine entered from Figure 7. First, at step 132, a determination is made whether the event alert is required to be stored. If the event alert is not to be stored, the routine
10 returns to the caller. A determination is then made whether the event is saturated. In particular, when the generic agent generates alerts that are constantly repeating, to avoid overflow of the network, the agent will stop reporting an alert according to parameters defined in the event table. This saturation is communicated to the monitor so that it knows that the event is
15 saturated and will not be reported further.

 Finally, a determination is made whether the event is to be filtered. If the event is to be filtered, the event is not reported. In particular, it is possible to set up filters such that, for example, a particular event is reported only from specific programs. In this case, if the event has
20 occurred from a non-specified program, the event would not be reported. A basic filtering for masked and unmasked events is done in step 126 in Figure 7 as described above. At this step (step 132), the program or file name that generated the alert may optionally be compared to a names table set up in

41

the user interface 52, stored in the users database 74, and downloaded to the agent database 30. If the program is not found in the names table, no alert is sent.

At step 134, it is determined from the results of the test at step 132, whether it is necessary to send an application alert to the monitor. If not, control is returned to the caller at step 136. If a determination is made at step 134 that an alert should be sent, the application alert is created and sent at step 138. Then the agent prepares for the next operation at step 142 by resetting itself to an initial condition able to accept a new interrupt. Control is then returned to the caller at step 148.

Figure 9 is a flow diagram of a routine for the controlling and management phase of the generic agent. This phase is entered at step 150 when a management request is obtained from the manager 4. At step 152, it is determined whether the type of the request is an action (set) or a request (get). If it is a set request, it is passed to the dispatcher 154 which determines which of the actions to execute. Boot, freeze or unfreeze set commands for the workstation are executed at step 156. Messages, keystroke jobs, start or stop program orders, or orders to take or release control of a workstation are executed at step 158. The manager 4 may also send a set request to the generic agent to download a new event table at step 160. Finally, the manager 4 can send an file modification request which is performed at step 162. This module executes the modification orders such as changing the CONFIG.SYS or AUTOEXEC.BAT files. After the actions

42

in steps 156, 158, 160 and 162 are performed, confirmation is sent back to the manager 4 at step 164.

If a get request is detected at step 152, the get command is sent to the dispatcher 165 which performs one of the functions at steps 166, 168, 170 and 172. If the manager 4 has requested the program list, it is sent at step 166. This program list is the list of programs running on the workstation for the receiving agent at the time of receipt of the request. If basic configuration information is requested, this is sent at step 168. If information is desired for remote access, such as video driver, mouse, or operating system information, the information is gathered at step 170. Using information from this get order, the manager 4 can take control of the agent's workstation through a set request processed at step 158. If the manager requests event table information, it is retrieved at step 172. The requested result from steps 166, 168, 170, and 172 are sent to the manager 4 at step 174.

Figure 10 is a flow diagram of a routine for the operation phase of the monitor 2 according to one embodiment of the present invention. At step 176, an alert is received from the network and sent to the event log database 44 at step 178. The event is then written to the event log database 44 containing a detailed descriptions of the event, such as the nature of the error, the name of the program, the location of the error, and recommendations for corrective action. The format of the event log database 44 can be as is described above. According to one embodiment of the

present invention, every event that generates an alert is written to the event log database 44. A manage event list module 180 is provided which manages the list of events using the filters that have been set up by the network administrator. An event filtering engine 182 is provided in real time to filter the views. In other words, the manage event list module 180 is a database manager that places events into and retrieves events from the event log database 44. Real time event filtering engine 182 selects events from the event log database 44 for passage to the execute view function 184. The events are filtered to be displayed in various views. In particular, for each view that is set up by the network administrator, only certain of the alerts are to be displayed. The view function is executed at step 184.

For each event, it is determined at step 186 whether a correction procedure has been defined for that event. In other words, as described above, the network administrator, or the system by default, sets up triggers which are specific commands sent to the manager by the monitor to cause actions to be taken, generally by the agents. As described above, a correction procedure is a set of triggers that are executed in sequence in response to an event. For example, detection of a "Too many files" error would cause a correction procedure consisting of two triggers to be sent to the workstation, the first would be a trigger to modify the CONFIG.SYS file to increase the number of files. This would be followed by a reboot trigger to reboot the workstation for the change to take effect.

If a correction procedure has been defined to be executed in response to the alert received at step 176, it is determined at step 188 whether that correction procedure is active. In particular, the correction procedure, or set of triggers, are defined to be "active" or "inactive" by the user of the monitor. A particular procedure may be set inactive because the network administrator wishes to temporarily suspend the automatic correction feature for various reasons while continuing to log and view the events. If the procedure is active, the manager is called to execute the procedure at step 190. If the answer at steps 186 and 188 is negative or after the manager executes the trigger procedure at step 190, maintenance of the message log database 45 (containing the outcome of triggers, the statistics, the counters, etc.) is performed at step 192. In particular, counters of events, triggers, and the like are maintained to show the size of the databases and support the operation of the system. Additionally, the administrator is given an opportunity to define a correction procedure using the procedures manager 54 so that the next time the particular event occurs, a correction procedure will be executed.

Figure 11 is a flow diagram of a routine for the operation phase of the manager 4 according to one embodiment of the present invention. The manager 4 either gets a communication from the agent 14 at step 194 or a communication from the monitor 2 at step 218. In either case, the communication is sent to a respective dispatcher 196 and 220.

If the communication is from the agent, the communication can be instructions to perform one or more of the modules 198 through 208. Module 198 is a heartbeat procedure which updates the status of workstation presence at predetermined intervals. According to one embodiment, the manager 4 is provided with a heartbeat at a predetermined frequency signifying that the agent 14 is active. According to one embodiment, a configurable parameter is provided which defines a heartbeat frequency in the range of from 1 to 80 seconds, with a default at 40 seconds. Module 199 is a configuration communication which updates the software and/or hardware information from the agent 14 concerning the agent's workstation. Module 200 is the ACP communication which updates the program list for the agent. Module 201 is the filter procedure which sets up the event table according to the determined filter. Module 202 is the control procedure which performs control according to the procedures listed in Tables 2 and 3.

Module 203 is a setup procedure which performs the setup menu. The setup menu includes setup procedures for the manager database, including, according to one embodiment, accessing the user list (i.e., managed workstations), monitoring undefined users (i.e., workstations with active agents which are not yet defined in the database), set default event tables for defined workstations, and set an index number for each agent type. The index numbers are numeric references to events and characteristics of events which are predefined and modifiable by the administrator. Module 204 forwards the keystrokes sent by the manage keystroke jobs trigger to the

agents for execution. Module 206 provides access to the file menu.

According to one embodiment, the file menu includes the following

functions: show current agent parameters from the database (i.e., name,

type, etc.), start monitor, and exit manager. Module 207 is the discovery

5 procedure which is used when a new user or workstation is added to the

network. The discovery module is executed upon receiving the identification

packet concerning a newly active agent from the autodiscovery module 38

shown in Figure 3. Finally, the module 208 manages the user's database 74.

If the communication is from the monitor 2 through the

10 dispatcher 220, it can be one of the commands or triggers listed in Tables 2

and 3 or one or more custom triggers. According to the illustrated

embodiment, the list shown in block 222 includes only those triggers listed

in Table 2 plus custom trigger(s). It is within the skill of the ordinary

artisan to add the triggers listed in Table 3 to this block once in possession

15 of the present disclosure. The procedures are sent to the dispatcher 196 so

that the various modules 198 through 207 can be executed to perform their

respective functions.

In addition, the procedures in block 222 are sent via the

perform set module 214 to be executed. The modules 200, 201, 202, 203,

20 and 204 are also connected to the perform set module 214. The output of

the perform get module 212 is sent to the modules 198, 199, or 207 as

appropriate.

Menu 68 provides access by the network administrator to any of the modules 197-207 as well as the perform get module 212, perform set module 214, and get user menu command module 216. Module 212 performs the get operation to get information from a workstation as
5 described above. Module 214 performs a set operation to set information or perform a trigger within a workstation as described above. Module 216 performs the procedure to get a user selection from the manager menu choices.

The following paragraphs describe two examples of how the
10 system according to the present invention operates.

If an error occurs in execution of interrupt Int 17h (Table 1), indicating that the requested printer is unavailable, an alert is reported to the monitor 2. The alert is logged, and if not filtered or saturated, is displayed in the appropriate view preferably, in this case, a customized view of
15 printing alerts. If a correction procedure for handling this alert has been defined, it will be sent to the manager 4 and then to the agent 14 which generated the alert. The correction procedure could include, for example, a trigger (6) to start capture application to redirect the printer to the network printer and a trigger (5) to send a message to the user to retry the print.

20 If an error occurs during execution of a NOS interrupt indicating an access is denied to a prospective user by the network operating system, an alert is reported to the monitor 2. As in the above example, the alert is logged, and if not filtered or saturated, is displayed in the appropriate

view. If a correction procedure is defined, it will be sent to the manager 4 and then to the agent 14. The correction procedure could include a trigger (5) to send a message indicating an unauthorized access attempt followed by a trigger (11) to freeze the workstation.

5 The above description of Figures 1-11 illustrates one embodiment of the system according to the present invention comprising a generic agent and a management console provided to communicate with the generic agent. In addition, according to another embodiment of the present invention, a specific agent may be provided to handle errors or problems that
10 may occur during execution of a specific program. For example, a specific agent may be provided to handle any errors occurring in the LOTUS™ cc:MAIL™ application. LOTUS™ cc:MAIL™ is an electronic mail program available from Lotus Development Corporation, in Cambridge, Massachusetts. An example of an embodiment of such a specific agent is
15 illustrated in Figures 12-15. As described, the specific agent must have the generic agent available on the workstation and defined in the management console to run. However, it is within the skill of the ordinary artisan to provide a combined generic and special agent to handle one or more specific programs as well as generic errors that may occur within workstations on the
20 network once in possession of the present disclosure.

Figure 12 is a block diagram of the system view of an exemplary special agent, in particular, the special agent for the cc:MAIL™ router. The cc:MAIL™ router is a program that directs electronic mail

messages between electronic post offices. The specific agent for cc:MAIL™ routers monitors the operation of the cc:MAIL™ router and reports its status as well as the results of its communication, successful or unsuccessful. These reports can be used to initiate correction procedures in the same way as occurs with a generic agent. The various elements of the special agent are connected to the generic system and particularly to the agent 14 and monitor 2. In Figure 12, like elements as those shown in Figure 2 are labelled with like reference numerals.

First, a setup and definition file (.INI) is provided at block 300. The setup and definition file 300 contains the specification for handling the LOTUS™ cc:MAIL™ router error log file 312 and forwarding of events to the agents 14. This communicates with the special agent for cc:MAIL™ router which is provided in block 310. An error log file 312 is provided for tracking the events that occur during execution of the LOTUS™ cc:MAIL™ router. The specific agent 310 communicates with the agent 14 of the generic system via a standard send alert API as shown in Figure 3. In addition, new indices in the form of a new index database is provided for the cc:MAIL™ router agent at block 314. The specific agent requires that additional, special purpose indices be added to this database. These indices are part of the user database (module 74 in Figure 5). For each particular event there should be one index which describes this alert in text format in this database. Thus, when any specific agent is added to the system, this

database contains an index definition for each of the alerts that this new specific agent can generate.

The specific agent may be loaded as a TSR program or as a non-TSR program. Embodiments for both installations are described.

5 Figure 13 illustrates a flow diagram of a routine for the cc:MAIL™ router specific agent which is loaded as a non-TSR application. First, the agent is loaded at step 316, followed by the initialization phase in which the command line is parsed, and a determination is made whether the generic agent is loaded. Next, at step 318, the LOTUS™ cc:MAIL™ router
10 is started from within the specific agent. The specific agent acts as a protective shell to the router. That is, the specific agent is a program that, in effect, surrounds the router and is able to observe and/or control all external communication by the router. At step 320, the connection to the cc:MAIL™ router error log is verified and at step 322, the log is polled and
15 a determination is made whether a new router event has occurred.

 If at step 316, 318 or step 320, an error occurs, a failure message is printed at step 336. In particular, if the generic agent is not loaded at step 316, an error message "fail to initialize" is printed at step 336. If step 318 fails, a "fail to load" message is printed and if step 320
20 fails, a "fail to connect" message is printed.

 At step 322, the router error log is polled and a determination is made whether a new router event is detected. If a router event is determined to be an error in step 323, a router malfunction alert is created at

step 324. If no problem is detected, a determination is made whether a successful operation or session has occurred at step 326. If not, a wait state is entered for a predetermined number of seconds at step 328. According to one embodiment, this wait state is entered for 3 seconds, although this value can be changed. After the predetermined number of seconds has elapsed, control passes again to step 322 to determine whether a new router event has occurred. If a successful operation or session is detected at step 326, the router operation is reported at step 330. At step 332, the generic agent 14 is accessed to report the alert or router operations from steps 324 and 330, respectively. Control then passes to the generic agent 14 at step 334.

Figure 14 is a flow diagram of a routine for the initialization phase for the specific agent cc:MAIL™ router which is loaded as a DOS™ TSR program. First, the command line is parsed and a determination is made whether the generic agent is loaded at step 336. If the generic agent is not loaded, a "fail to initialize" error message is printed at step 337. If so, the interrupts listed in Table 1 are set up at step 338. In particular, interrupts 08h, 28h, 09h, and 16h are set up. Step 340 is then executed which allows the cc:MAIL™ router specific agent to stay resident in memory.

Figure 15 is a flow diagram of a routine for the operation phase of the cc:MAIL™ router specific agent loaded as a DOS™ TSR. When a system call to the interrupt handler is detected at step 342, the router agent interrupt handlers are accessed at step 344. Step 346 performs a jump to the

old interrupt handler. Step 348 causes the TSR to wake up. That is, the jump in step 346 to the old interrupt handler returns control to the LOTUS™ cc:MAIL™ router program so that it can complete its normal function. The wake up function in step 348 stimulates the specific agent TSR to begin its operation of analyzing and reporting the event. In particular, according to one embodiment, a timer interrupt handler for cc:MAIL™ router specific agent is provided which gets control from the system 18 times per second through block 342 in Figure 15. Then, in the block 344, the cc:MAIL™ router specific agent decides if a specified time from the last wake up (or beginning of the work) has passed. If so, it invokes block 348 to wake up the specific agent. The specified time is determined by the wait state parameter described above. If the specified time has not passed, block 344 invokes block 348 to perform a jump to the old timer interrupt handler (i.e., the system timer).

After the TSR is woken up at step 348, the cc:MAIL™ router specific agent process is performed in the same way as described above with respect to Figure 13. Because the operations are the same, they will not be described again but are noted in the figure with the same reference numbers as used in Figure 13.

The software modules described above may be provided for operation with PC-DOS™, MS-DOS™, WINDOWS™, OS/2™, or other comparable operating systems. According to one embodiment, the present invention requires 4MB RAM in the management console, 8MB hard disk

space, MICROSOFT™ WINDOWS™ 3.0 or later, NOVELL™ NETWARE™
2.2 or above, IBM™ LAN Server, MICROSOFT™ LAN manager, or any
other NetBIOS™, IPX™, DLC™, or TCP/IP™ compatible network. The
detailed implementation of the software modules is within the skill of the
ordinarily skilled artisan once in possession of the present disclosure. In
addition, any modifications required to run the software modules on other
systems is also within the skill of the ordinarily skilled artisan once in
possession of the present disclosure.

The foregoing description of the specific embodiments will so
fully reveal the general nature of the invention that others can, by applying
current knowledge, readily modify and/or adapt for various applications such
specific embodiments without departing from the generic concept, and,
therefore, such adaptations and modifications should and are intended to be
comprehended within the meaning and range of equivalents of the disclosed
embodiments. It is to be understood that the phraseology of terminology
employed herein is for the purpose of description and not of limitation.

WHAT IS CLAIMED IS:

1 1. A system for monitoring and controlling at least one
2 program capable of being executed on any of at least two workstations in a
3 network, said network comprising at least one agent module resident on each
4 of said at least two workstations and a management console connected to
5 each of said at least two workstations, said system comprising:

6 means for identifying an event occurring with respect to a
7 program executing on one of said at least two workstations;

8 means for sending an alert to said management console which
9 identifies said event;

10 means for storing a plurality of triggers, each of said triggers
11 adapted to cause an action to be taken within said network;

12 means for storing at least one procedure, said at least one
13 procedure comprising at least one of said plurality of triggers; and

14 means for sending at least one of said procedures from said
15 management console to the agent module resident on said one of said at least
16 two workstations in response to receipt of said alert.

1 2. The system according to claim 1, further comprising:

2 means for scheduling performance of selected ones of said at
3 least one procedure at specific times; and

4 means for sending said selected ones of said at least one
5 procedure from said management console to at least one of said at least one
6 agent modules at said specific times.

1 3. The system according to claim 1, wherein one of said
2 plurality of triggers comprises sending at least one keystroke job from said
3 management console to said at least two workstations.

1 4. The system according to claim 1, further comprising
2 means, in said management console, for providing remote access to said at
3 least two workstations to enable said management console to take and release
4 control of said at least two workstations.

1 5. The system according to claim 1, wherein one of said
2 plurality of triggers comprises starting programs on said at least two
3 workstations and another of said plurality of triggers comprises stopping
4 executing programs on said at least two workstations.

1 6. The system according to claim 1, further comprising
2 means for providing mapping integration with a network management
3 system.

1 7. The system according to claim 1, further comprising
2 means for providing automatic discovery of a newly activated agent module
3 on said at least two workstations for identifying said newly activated agent
4 module to said management console.

1 8. The system according to claim 7, wherein said means
2 for providing automatic discovery is provided in said at least one agent
3 module and sends to said management console an identification packet of the
4 newly activated agent module at the time the newly activated agent module
5 starts operation and at predetermined intervals thereafter until it receives
6 confirmation from said management console that the newly activated agent
7 module is discovered.

1 9. The system according to claim 8, wherein said
2 predetermined intervals are about 60 seconds.

1 10. The system according to claim 1, further comprising an
2 application control means for launching and executing capture of said at least
3 two workstations in a background operation of said at least two workstations
4 wherein said capture is accomplished without affecting any program
5 executing in a foreground operation of said at least two workstations.

1 11. A system for monitoring and controlling at least one
2 program in a network, said network comprising at least two workstations and
3 a management console connected to each of said at least two workstations,
4 said system comprising:

5 at least one generic agent means resident on each of said at
6 least two workstations for transmitting alerts indicating occurrence of an
7 event with a program executing on the workstation;

8 monitor means resident on said management console for
9 logging alerts transmitted by any of said at least one generic agent means;

10 means for storing a plurality of triggers to be sent from said
11 monitor means to said at least one agent means, said triggers comprising
12 commands which cause actions to be taken by said at least one generic agent
13 means in response to the event;

14 means for defining at least one procedure, said procedure
15 including at least one of said stored triggers; and

16 manager means for monitoring and controlling operations of
17 said at least one generic agent means, said manager means comprising means
18 for sending said at least one procedure to said at least one generic agent
19 means in response to an alert.

1 12. The system according to claim 11, wherein said
2 plurality of triggers comprise a plurality of predefined triggers and user
3 defined triggers.

1 13. The system according to claim 11, further comprising
2 means for sending a heartbeat to said manager means from said at least one
3 generic agent means at a predetermined frequency signifying that said at
4 least one generic agent means is active.

1 14. The system according to claim 11, wherein said
2 predetermined frequency is determined by a configurable parameter which
3 ranges from 1 to 80 seconds, with a default of 40 seconds.

1 15. The system according to claim 11, further comprising:
2 means for scheduling performance of selected ones of said at
3 least one procedure at desired times; and
4 means for sending said selected ones of said at least one
5 procedure from said monitor means to at least one of said at least one
6 generic agent means at said desired times.

1 16. The system according to claim 11, wherein one of said
2 plurality of triggers comprises sending at least one keystroke job from said
3 manager means to said at least two workstations.

1 17. The system according to claim 11, further comprising
2 means, in said manager means, for providing remote access to said at least

3 two workstations to enable said manager means take and release control of
4 said at least two workstations.

1 18. The system according to claim 11, wherein one of said
2 plurality of triggers comprises starting programs on said at least two
3 workstations and another of said plurality of triggers comprises stopping
4 executing programs on said at least two workstations.

1 19. The system according to claim 11, further comprising
2 means for providing mapping integration with a network management
3 system.

1 20. The system according to claim 11, further comprising
2 means for providing automatic discovery of a newly activated agent means
3 on said at least two workstations for identifying said newly activated agent
4 means to said management console.

1 21. The system according to claim 20, wherein said means
2 for providing automatic discovery is provided in said at least one generic
3 agent means and sends to said management console an identification packet
4 of the newly activated agent means at the time the newly activated agent
5 means starts operation and at predetermined intervals thereafter until it

6 receives confirmation from said management console that the newly activated
7 agent means is discovered.

1 22. The system according to claim 21, wherein said
2 predetermined intervals are about 60 seconds.

1 23. The system according to claim 11, further comprising
2 receiver and transmitter means for connecting said at least one generic agent
3 means with said network using one of a plurality of network communication
4 protocols.

1 24. The system according to claim 11, wherein said
2 monitor comprises means for maintaining statistical views for displaying
3 information concerning alert statistics.

1 25. The system according to claim 11, further comprising
2 an application control means for launching and executing capture of said at
3 least two workstations in a background operation of said at least two
4 workstations wherein said capture is accomplished without affecting any
5 program executing in a foreground operation of said at least two
6 workstations.

1 26. The system according to claim 11, further comprising
2 at least one specific agent means for handling problems during execution of a
3 specific user program.

1 27. The system according to claim 26, wherein said one
2 specific user program is LOTUS™ cc:MAIL.™

1 28. A method for monitoring and controlling at least one
2 program capable of being executed on any of at least two workstations in a
3 network, said network comprising at least one agent module resident on each
4 of said at least two workstations and a management console connected to
5 each of said at least two workstations, said method comprising the steps of:
6 storing a plurality of triggers, each of said triggers adapted to
7 cause an action to be taken within said network;
8 storing at least one procedure, said at least one procedure
9 comprising at least one of said plurality of triggers;
10 identifying an event occurring on one of said at least two
11 workstations;
12 sending an alert to said management console which identifies
13 said event; and
14 sending at least one of said procedures from said management
15 console to the agent module resident on said one of said at least two
16 workstations in response to receipt of said alert.

1 29. The method according to claim 28, further comprising
2 the steps of:
3 scheduling performance of selected ones of said at least one
4 procedure at desired times; and
5 sending said selected ones of said at least one procedure from
6 said management console to at least one of said at least one agent modules at
7 said desired times.

1 30. The method according to claim 28, wherein one of said
2 plurality of triggers comprises sending at least one keystroke job from said
3 management console to said at least two workstations.

1 31. The method according to claim 28, further comprising
2 the step of providing remote access by said management console to said at
3 least two workstations to enable said management console to take and release
4 control of said at least two workstations.

1 32. The method according to claim 28, wherein one of said
2 plurality of triggers comprises starting programs on said at least two
3 workstations and another of said plurality of triggers comprises stopping
4 executing programs on said at least two workstations.

1 33. The method according to claim 28, further comprising
2 the step of providing mapping integration with a network management
3 system.

1 34. The method according to claim 28, further comprising
2 the step of providing automatic discovery of a newly activated agent module
3 on said at least two workstations for identifying said newly activated agent
4 module to said management console.

1 35. The system according to claim 34, wherein said step of
2 providing automatic discovery comprises sending to said management
3 console an identification packet of the newly activated agent module at the
4 time the newly activated agent module starts operation and at predetermined
5 intervals thereafter until it receives confirmation from said management
6 console that the newly activated agent module is discovered.

1 36. The system according to claim 35, wherein said
2 predetermined intervals are about 60 seconds.

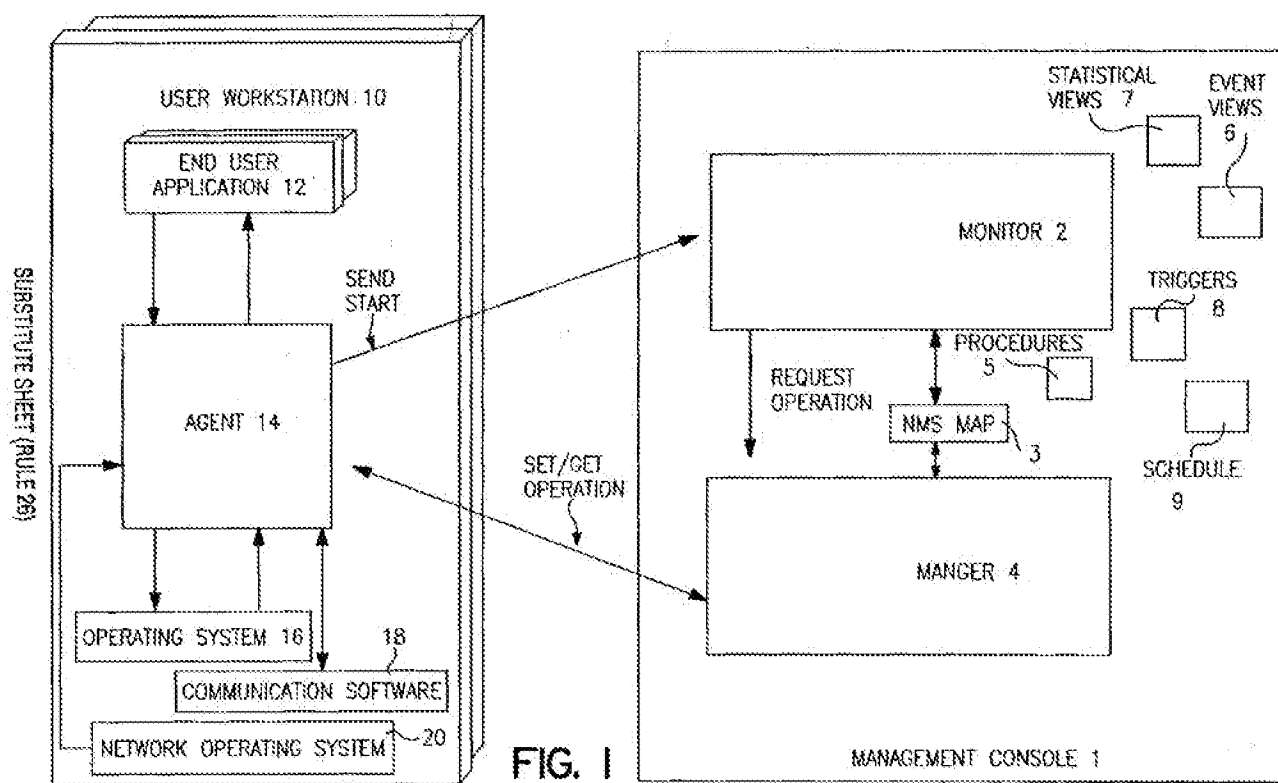


FIG. 1

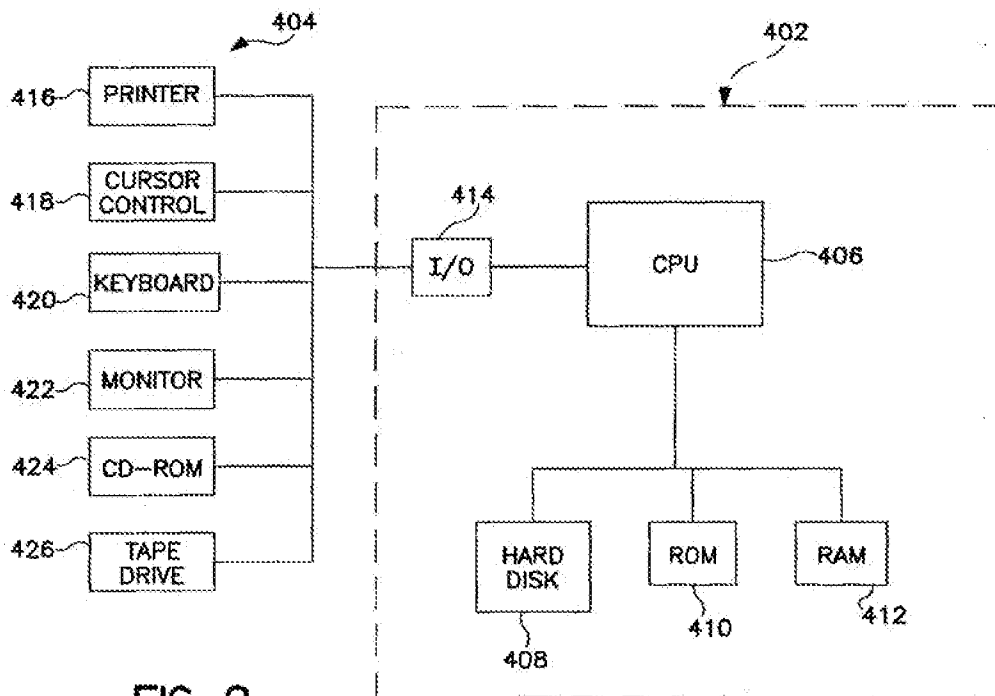


FIG. 2

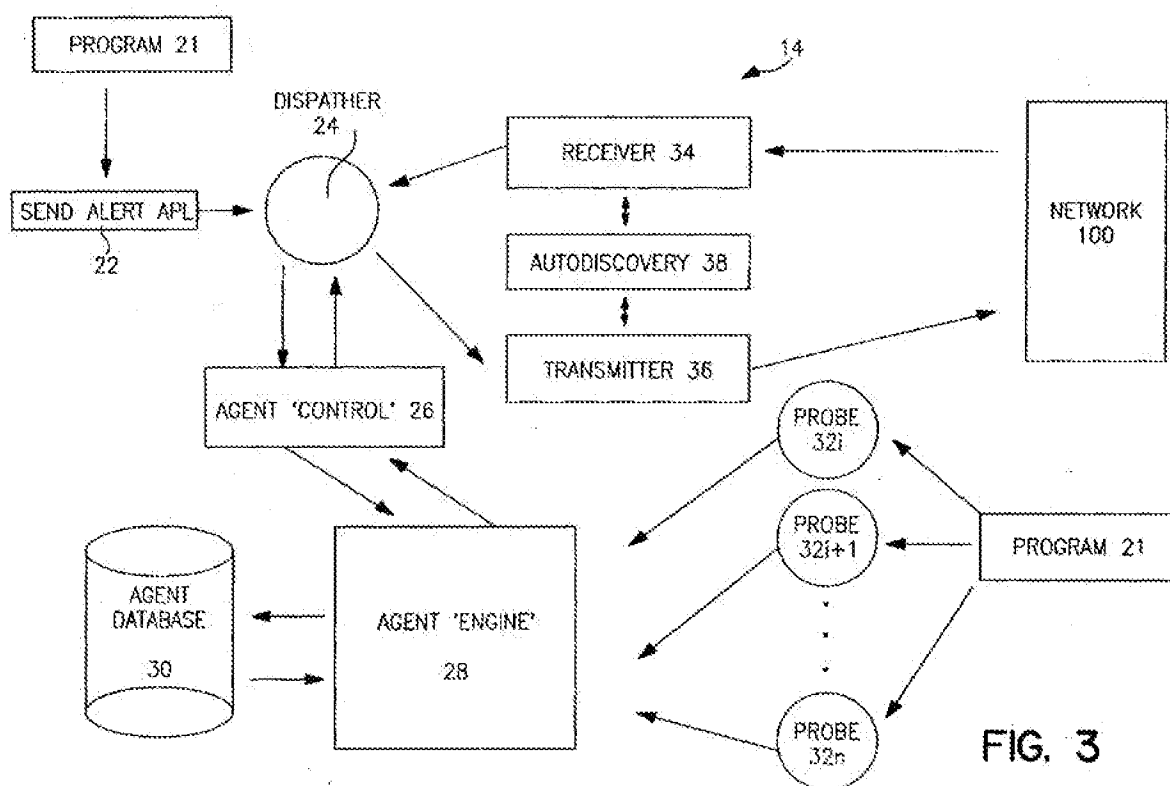
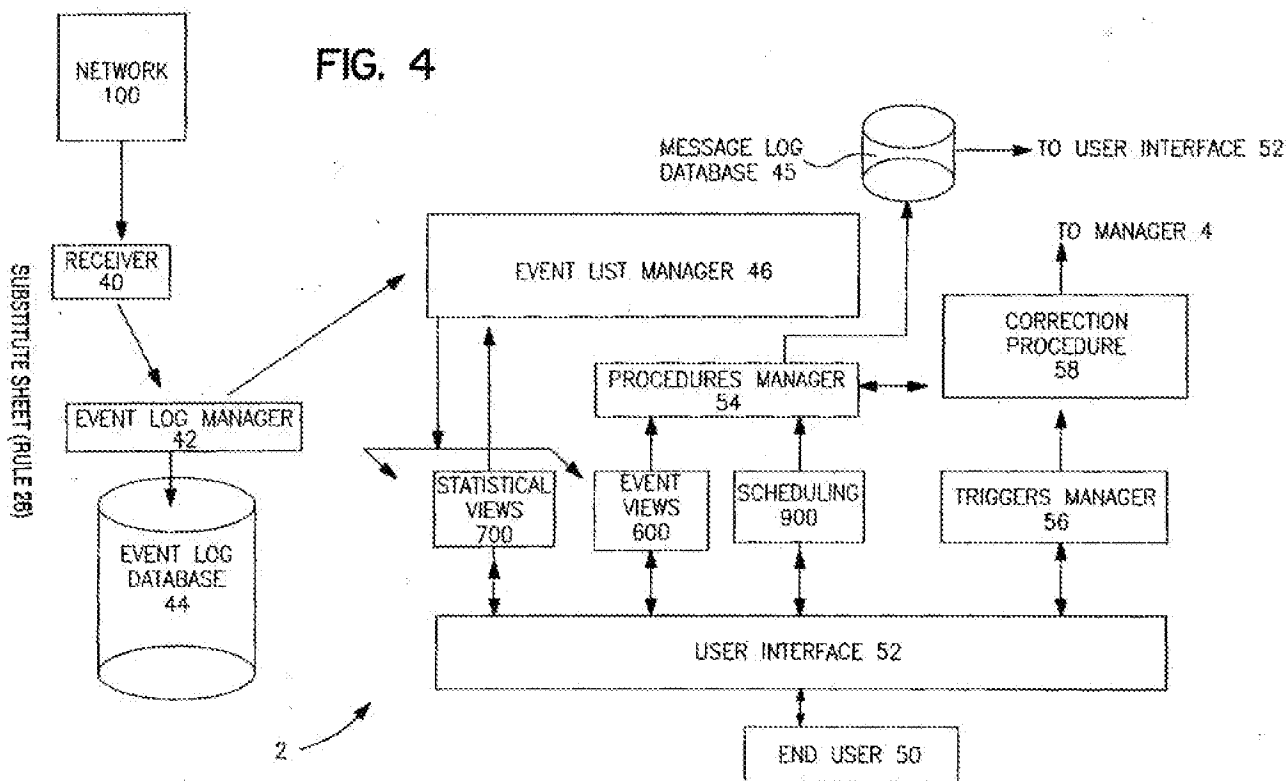


FIG. 3



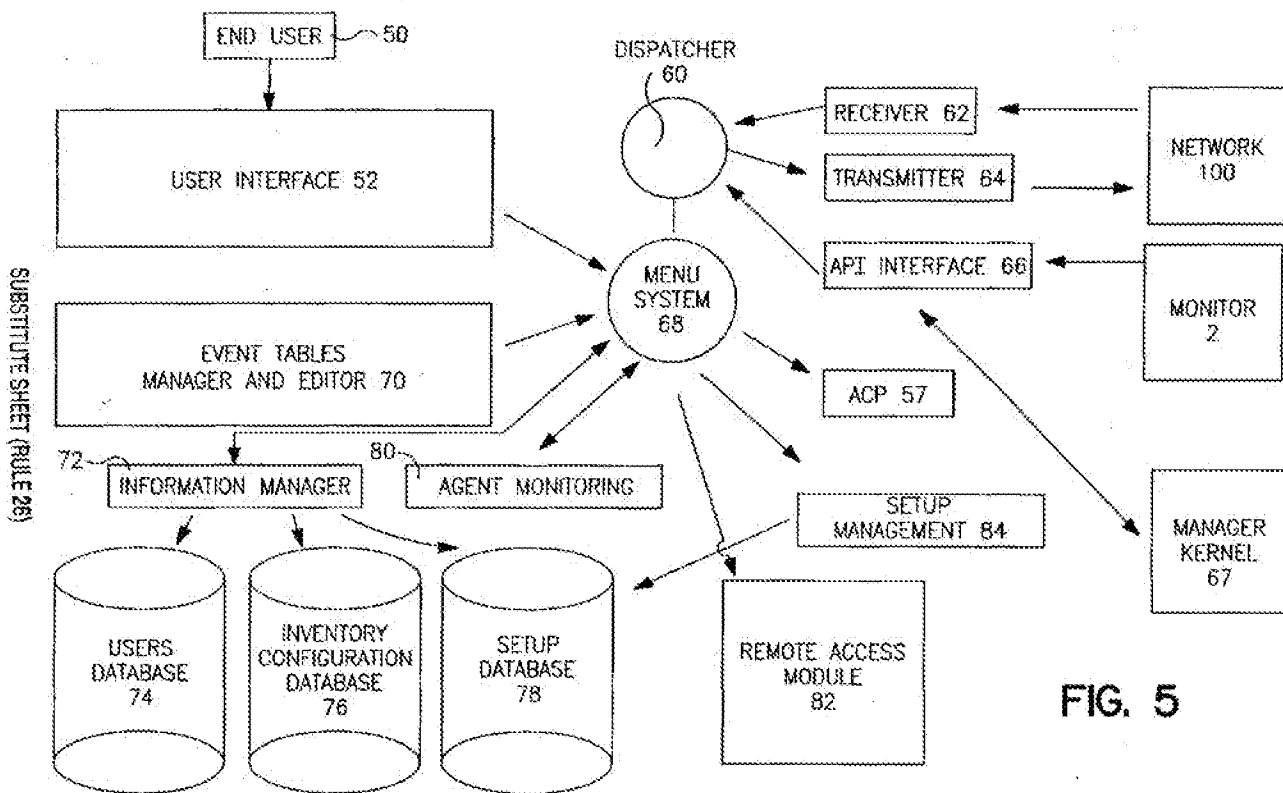


FIG. 6

LOAD AGENT

WO 95/27739

6 / 15

PCT/US95/04109

SUBSTITUTE SHEET (RULE 26)

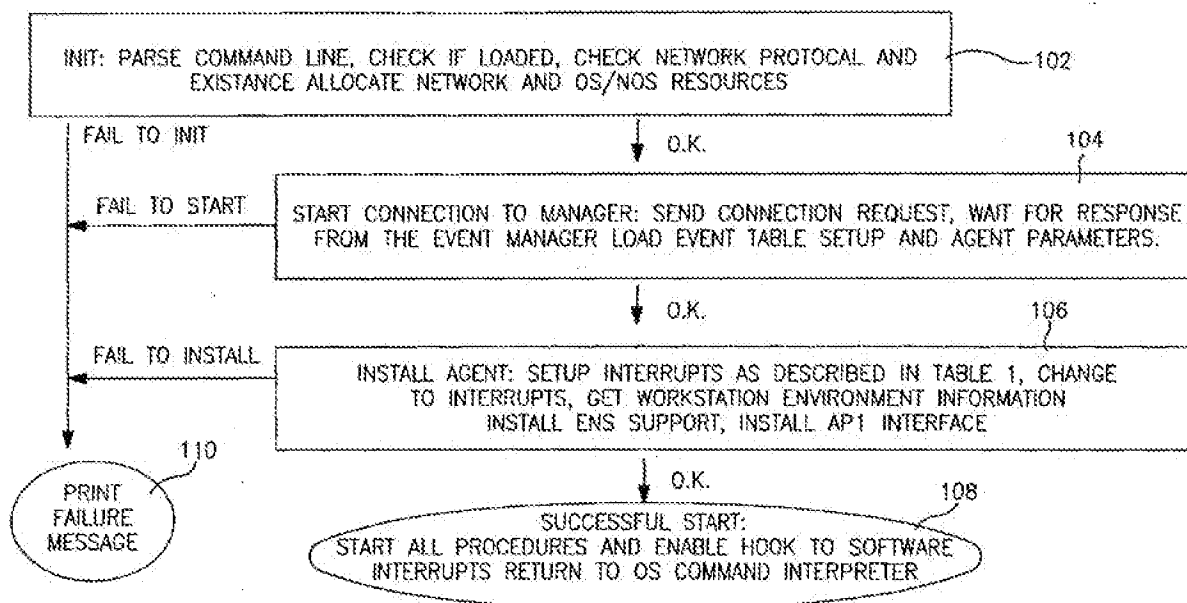
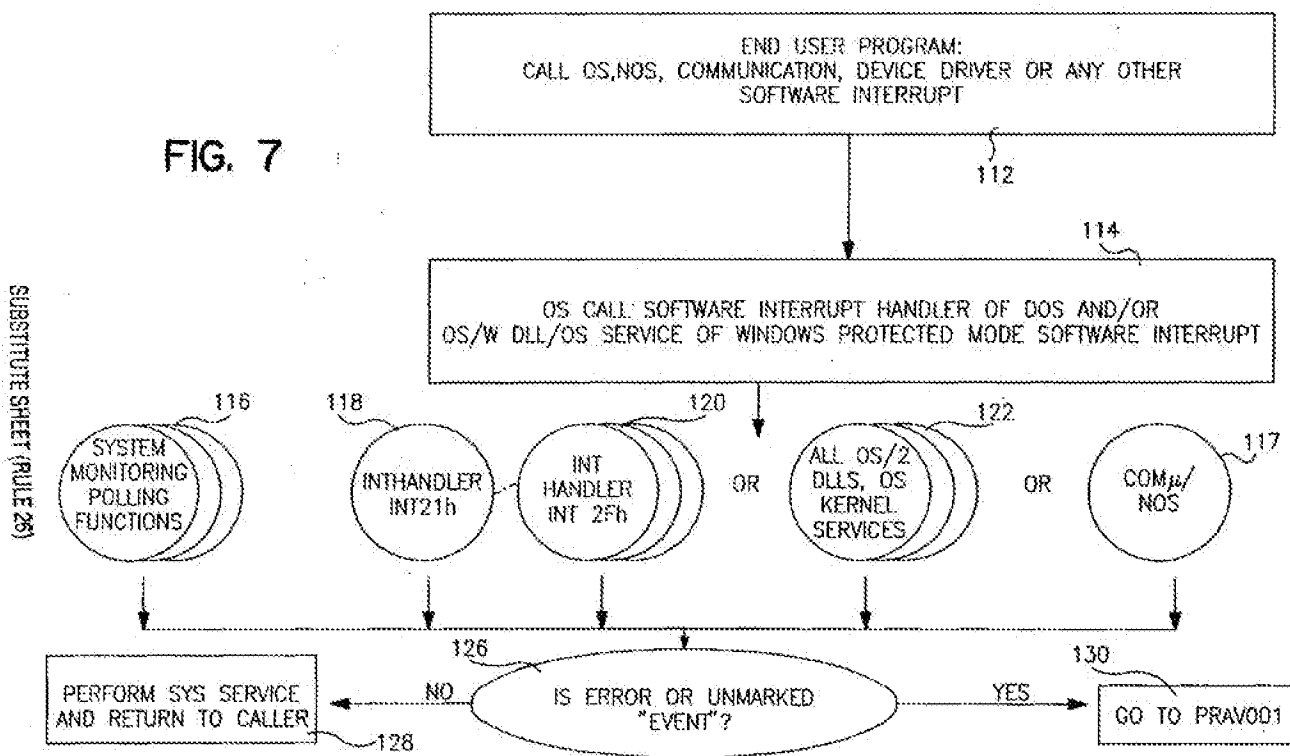


FIG. 7



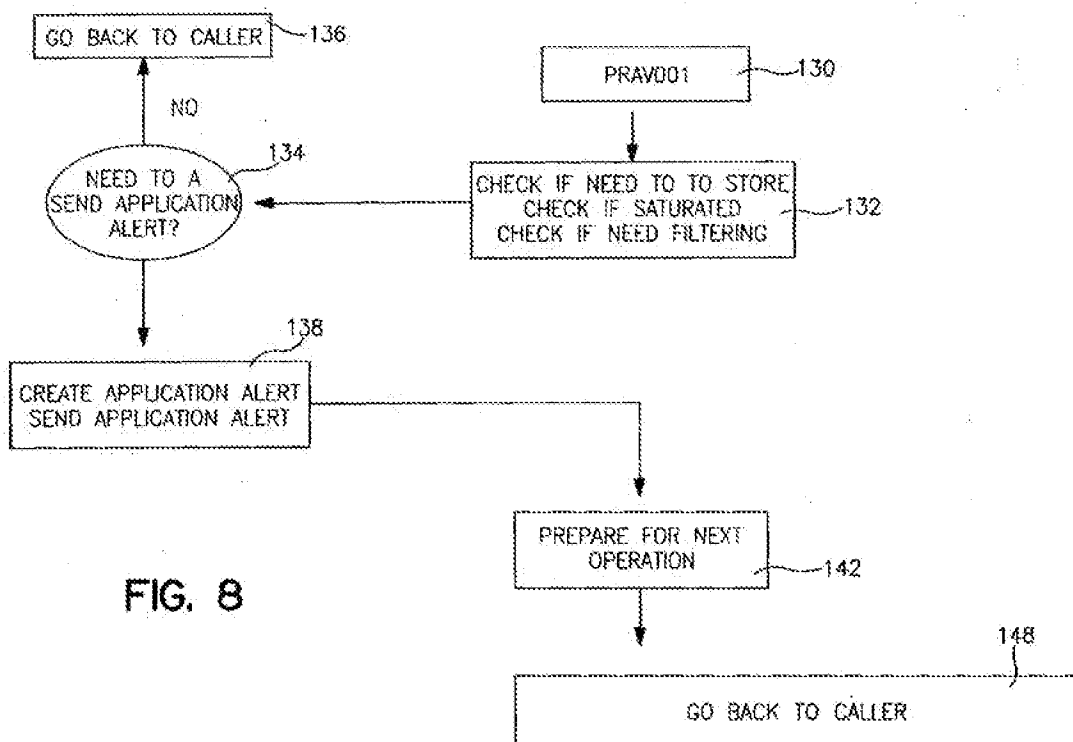
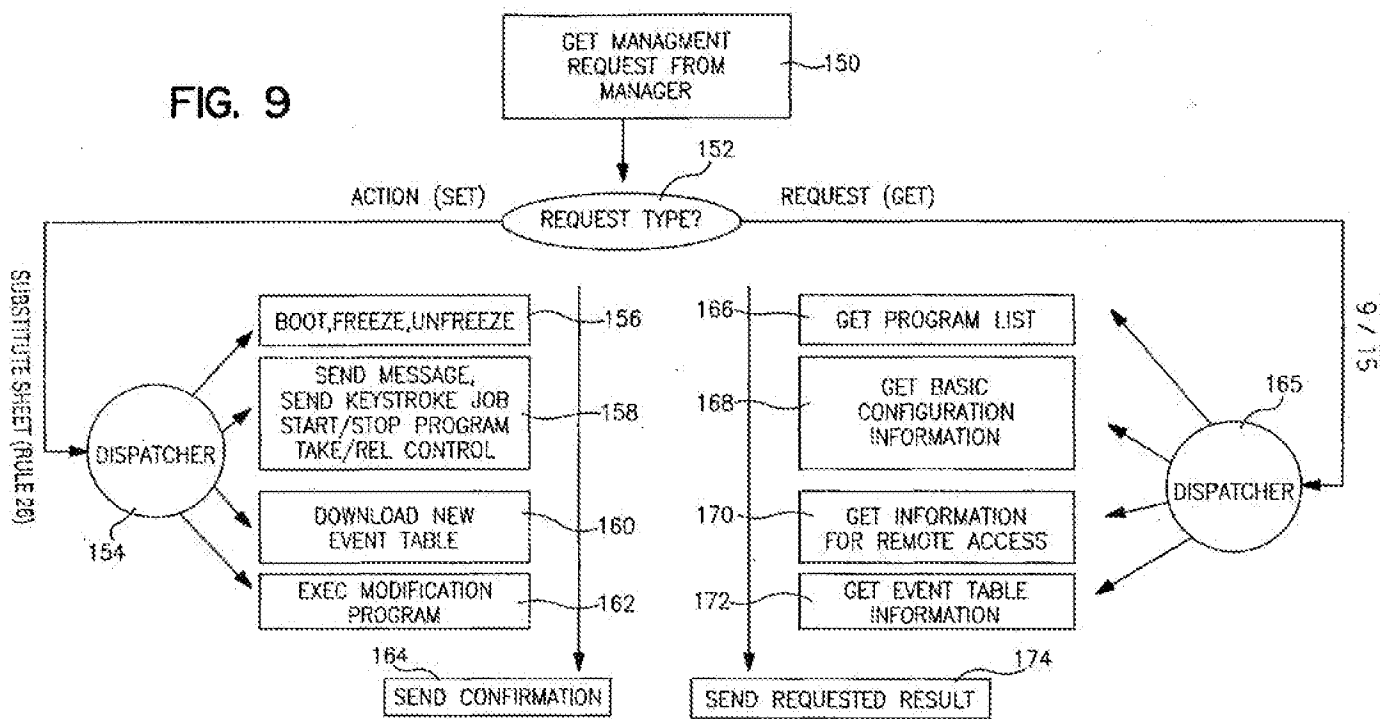


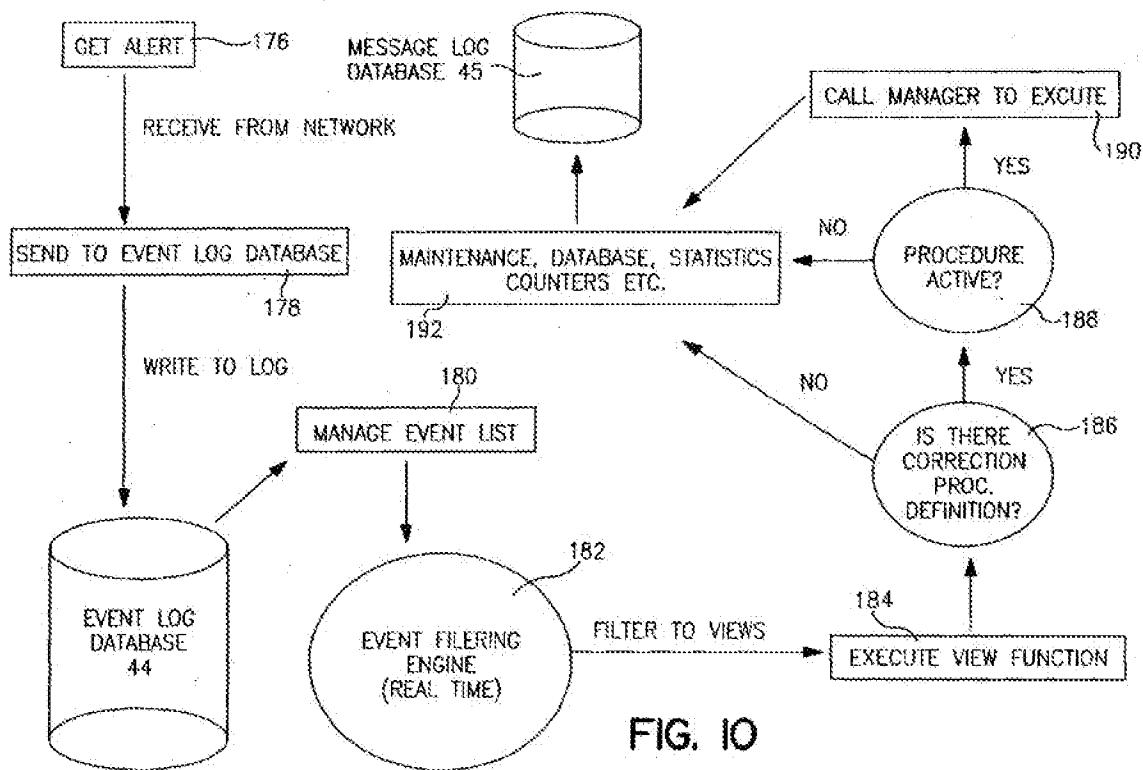
FIG. 8

FIG. 9



SUBSTITUTE SHEET (RULE 26)

9 / 15



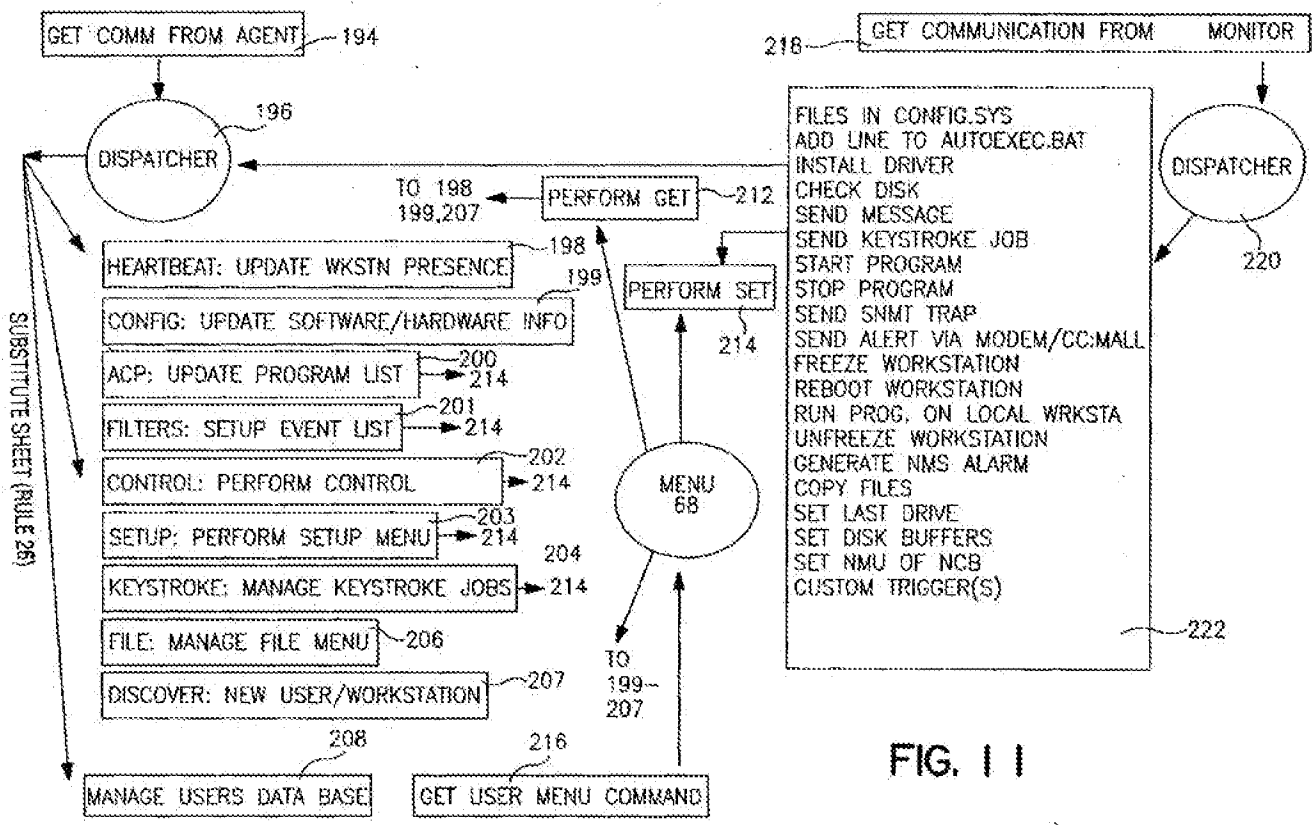
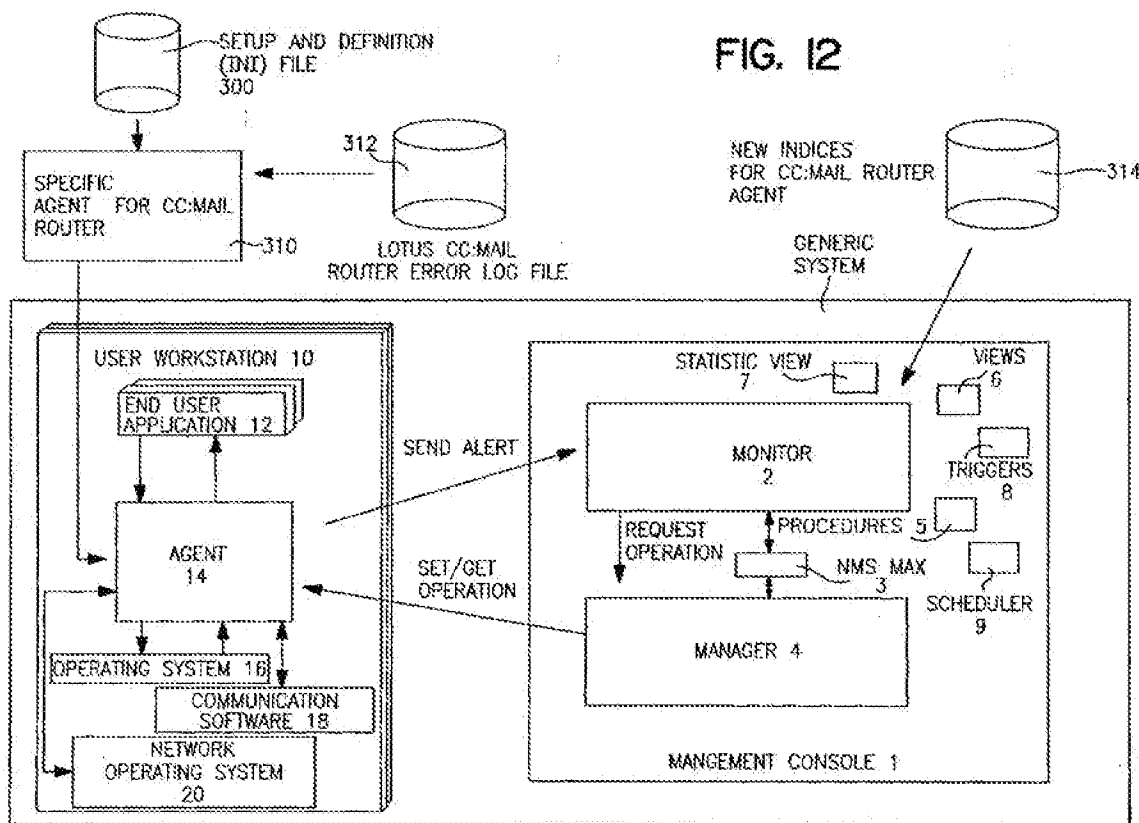
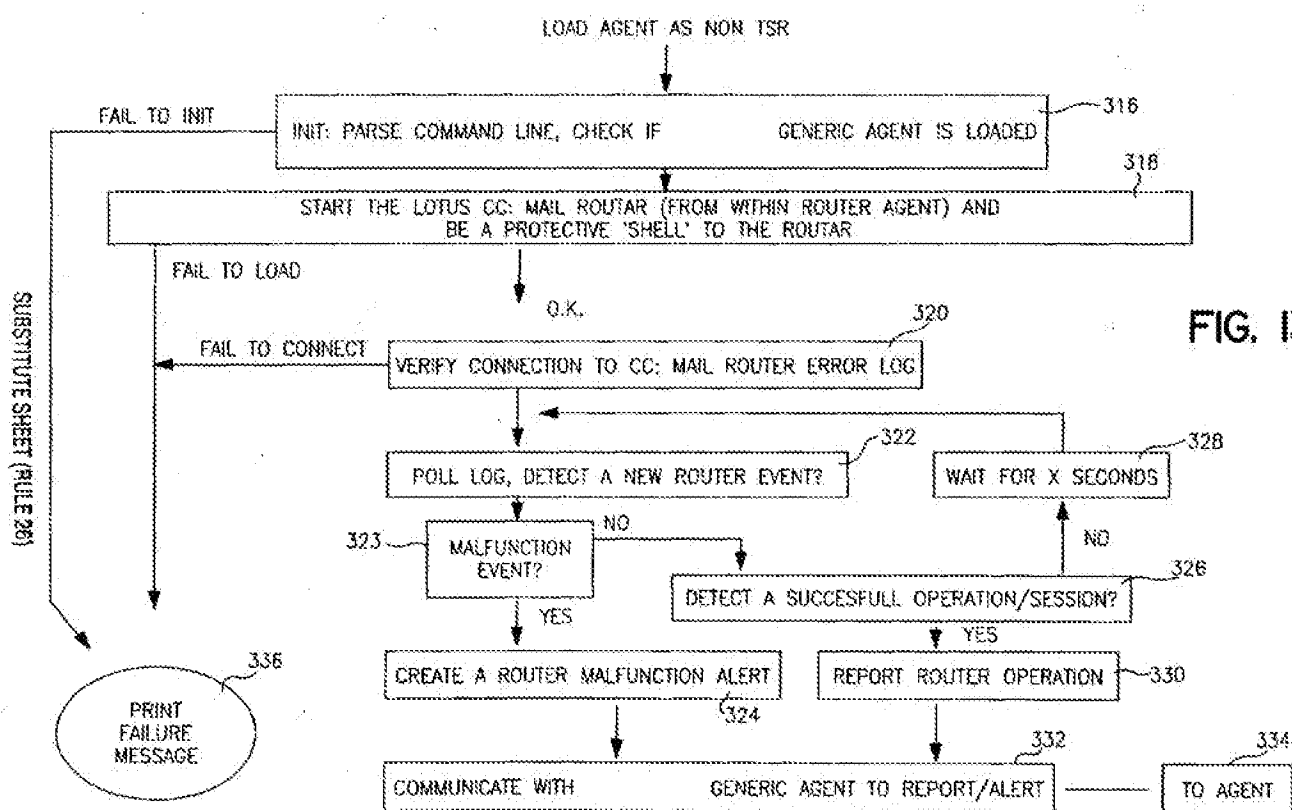


FIG. 11





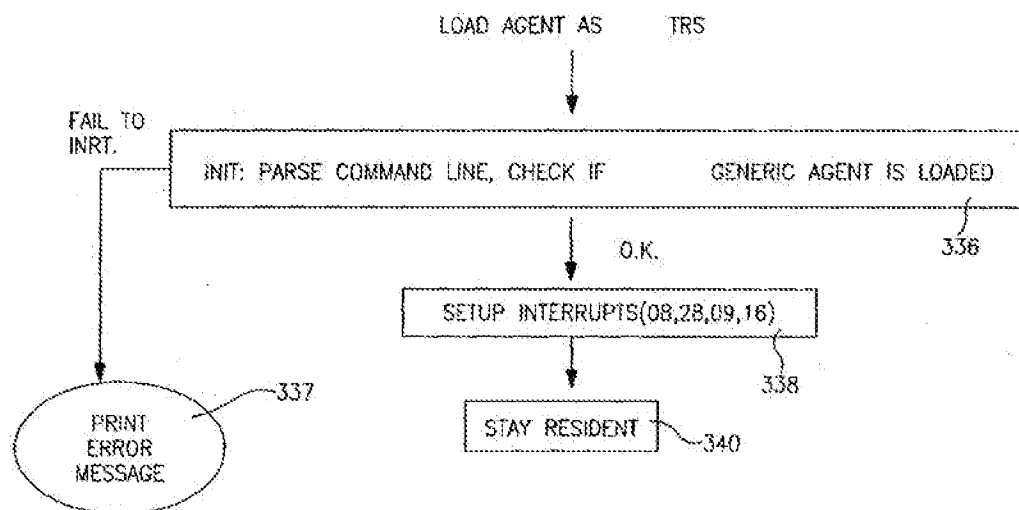
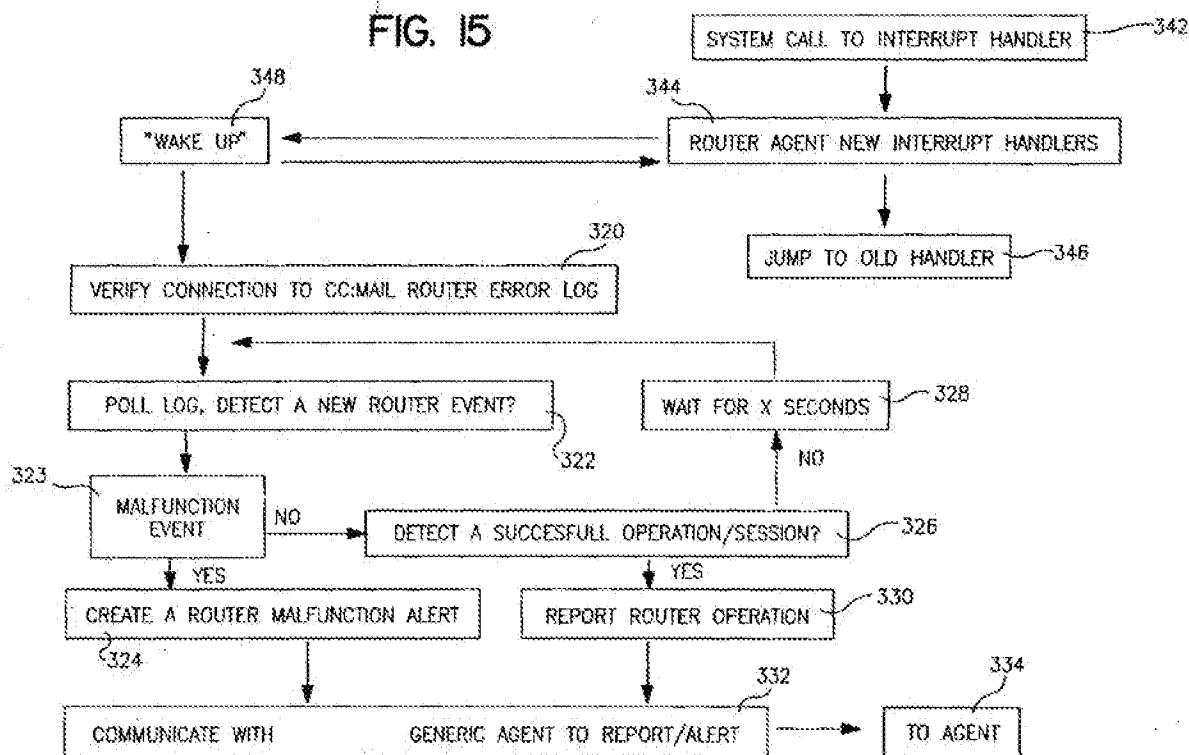


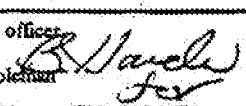
FIG. 14

FIG. 15



INTERNATIONAL SEARCH REPORT

International application No.
PCT/US95/04109

A. CLASSIFICATION OF SUBJECT MATTER IPC(6) : G06F 11/00 US CL : 395/575, 200 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. : 395/575, 200 Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched NONE Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) APS, IEEE PUBLICATIONS																				
C. DOCUMENTS CONSIDERED TO BE RELEVANT																				
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.																		
Y	DataComm Magazine, issued January 1993, Peter Heywood, "Troubleshooting Applications From the Inside", 2 pages.	1-36																		
Y	US, A, 5,206,948 (De Angelis et al.) 27 April 1993, abstract and col. 9.	1-5, 10-12, 15-18, 23-26, 28-32																		
Y	US, A, 5,153,909 (Beckle et al.) 06 October 1992, col. 5.	7-9, 13, 14, 20, 22, 34, 36																		
Y	US, A, 4,965,772 (Daniel et al.) 23 October 1990, col. 8.	6, 19, 33																		
Y	US, A, 5,283,856 (Gross et al.) 01 February 1994, cols. 3 and 10.	27																		
☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.																				
<table border="0"> <tr> <td>* Special categories of cited documents:</td> <td>"T"</td> <td>later document published after the international filing date or priority date and not in conflict with the application but used to understand the principle or theory underlying the invention</td> </tr> <tr> <td>"A" document defining the general state of the art which is not considered to be part of particular relevance</td> <td>"X"</td> <td>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</td> </tr> <tr> <td>"E" earlier documents published on or after the international filing date</td> <td>"Y"</td> <td>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 combinations being obvious to a person skilled in the art</td> </tr> <tr> <td>"L" document which may throw doubt on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)</td> <td>"Z"</td> <td>document member of the same patent family</td> </tr> <tr> <td>"O" document referring to an oral disclosure, use, exhibition or other means</td> <td></td> <td></td> </tr> <tr> <td>"P" documents published prior to the international filing date but later than the priority date claimed</td> <td></td> <td></td> </tr> </table>			* Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but used to understand the principle or theory underlying the invention	"A" document defining the general state of the art which is not considered to be part of particular relevance	"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	"E" earlier documents published on or after the international filing date	"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 combinations being obvious to a person skilled in the art	"L" document which may throw doubt on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Z"	document member of the same patent family	"O" document referring to an oral disclosure, use, exhibition or other means			"P" documents published prior to the international filing date but later than the priority date claimed		
* Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but used to understand the principle or theory underlying the invention																		
"A" document defining the general state of the art which is not considered to be part of particular relevance	"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																		
"E" earlier documents published on or after the international filing date	"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 combinations being obvious to a person skilled in the art																		
"L" document which may throw doubt on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Z"	document member of the same patent family																		
"O" document referring to an oral disclosure, use, exhibition or other means																				
"P" documents published prior to the international filing date but later than the priority date claimed																				
Date of the actual completion of the international search 16 MAY 1995		Date of mailing of the international search report 29 AUG 1995																		
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230		Authorized officer:  Eric Coleman Telephone No. (703) 305-9674																		



Europäisches Patentamt
European Patent Office
Office européen des brevets



Publication number: **0 636 977 A2**

EUROPEAN PATENT APPLICATION

Application number: **94305551.7**

Int. Cl.⁸ **G06F 11/00**

Date of filing: **27.07.94**

Priority: **29.07.93 US 99368**

Date of publication of application:
01.02.95 Bulletin 95/05

Designated Contracting States:
BE DE FR GB IT

Applicant: **Chambers, David Alan**
3655 Eastwood Circle
Santa Clara,
California 95054 (US)

Inventor: **Chambers, David Alan**
3655 Eastwood Circle
Santa Clara,
California 95054 (US)

Representative: **O'Connell, David Christopher**
et al
HASELTINE LAKE & CO.
Hazlitt House
28 Southampton Buildings
Chancery Lane
London WC2A 1AT (GB)

Method and apparatus for detection of computer viruses.

A behavior analyzing antivirus program detects viral infection of a target program by emulating the execution of the target program and analyzing the emulated execution to detect viral behavior. The antivirus monitor program contains both variables corresponding to the CPU's registers and emulation procedures corresponding to the CPU's instructions. The target program is loaded into memory and its execution is emulated by the antivirus monitor program. Intelligent procedures contained in the monitor program are given control between every instruction emulated so as to detect aberrant or dangerous behavior in the target program in which case the danger of a viral presence is flagged and emulation is terminated.

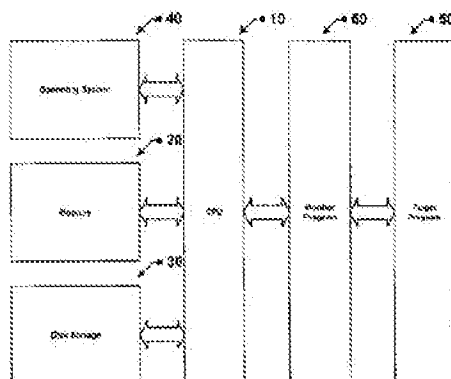


Figure 1B

EP 0 636 977 A2

BACKGROUND OF THE INVENTION

The present invention relates generally to a method and apparatus for emulating the execution of a program on a computer system. In particular, the present invention relates to monitoring program behavior to detect and terminate harmful or dangerous behavior in a program. More particularly, the present invention relates to monitoring program behavior to detect computer viruses.

In recent years, the proliferation of "computer viruses" (generally designed by rogue programmers either maliciously or as "pranks") has become an increasingly significant problem for the owners and users of computer systems. True computer viruses vary, but they share the general characteristic that they comprise executable computer code capable of replicating itself by attachment to and modification of standard computer files. Such files are then considered "infected". On most computer systems, viruses are limited to infecting program applications. When the application is executed, the virus can then replicate and attach copies to further application files. Typically, viruses also engage in other forms of behavior that are considered undesirable, such as re-formatting a hard disk.

Often grouped with true computer viruses are some other types of malevolent computer programs: worms and trojan horses. Worms do not infect other applications but merely replicate, either in memory or in other storage media. The harmful effect of worms is generally to reduce system performance. Worms are of concern for large multi-user computer systems, but are generally not of concern for personal computers. Trojan horses are programs that masquerade as useful programs or utilities; they generally run only once and have a harmful effect (such as destroying or damaging the computer system data storage). Trojan horses do not replicate, and after being run once by a user, the user is usually alerted to the harmful behavior and will not run the trojan horse again.

In response to the proliferation of computer viruses, a variety of "antivirus" methodologies and programs have been developed to detect the presence of infected files. These antivirus programs can be generally categorized into groups: behavior interceptors, signature scanners, and checksum monitors.

BEHAVIOR INTERCEPTORS

The earliest antivirus programs were generally of the behavior interceptor type: they would allow a virus program to execute in memory but would intercept strategic operating system function requests made by the computer virus. Such requests would generally be functions which the virus re-

quired to be performed in order to replicate or to destroy its host, i.e., "Write to a file", "Erase a file", "Format a disk" etc. By intercepting these requests, the computer operator/user could be informed that a potentially dangerous function was about to be performed. Control could be halted or continued as necessary. Some antivirus programs actually modify the instructions of the discovered virus program and make them inoperable so as to "kill" them.

The behavior interceptor method of virus detection has several drawbacks. The first problem is that it relies entirely on user input and decision making when potentially dangerous behavior is detected. This places a great burden on the user, for it is often very difficult to determine whether the flagged behavior is part of the normal operation of the program being executed. For example, disk optimizing programs routinely reformat hard disks to improve the interleaves value. In response to a warning message, a user might suspect that their disk optimizer was infected with a virus (when in fact it was not) and halt program execution. Or, worse yet, if the user knows that such behavior is part of the normal operation of a disk optimizer program, they would likely allow the format to continue uninterrupted, which would be disastrous if the program were actually infected.

A second problem with behavior interceptor antivirus programs is that computer virus technology has advanced to such a state that some computer viruses are able to bypass the interception points used by the antivirus. The virus can then make operating system function requests that are never intercepted by the antivirus, thus avoiding detection.

A third problem with behavior interceptor antivirus programs is that by allowing the virus to execute, the virus has an opportunity to locate and identify the antivirus program in computer memory. Once the antivirus program is located, the virus can modify the antivirus-- rendering it completely ineffective in exactly the same manner that antivirus programs locate and modify virus programs to render them ineffective.

A fourth and very significant problem with behavior interceptor antivirus programs is that there are no low level operating system function requests employed by computer viruses that are not also used by any of thousands of non-virus programs. At an instruction by instruction level, or at a function-call by function-call level, a computer virus performs the same operations as legitimate computer programs. In other words, the closer a computer virus is examined, the less distinguishable it becomes from any other computer program.

SIGNATURE SCANNERS

The next generation of antivirus technology, signature scanners, answered the problem of over-reliance on user interaction as well as the problem of allowing the virus to execute. A signature scanner operates by knowing exactly what a target virus program code looks like ("signature" code) and then scanning for these program codes in any programs requested to be executed or otherwise requested to be scanned. As long as the signature codes were sufficiently long enough so as not to be confused with another program's code, then positive identification was virtually guaranteed and the request to execute could be stopped before execution ever began. The primary problem with this technique is that it requires the antivirus developer to have previously collected and analyzed the target viruses, and included the signature codes in the antivirus program. The antivirus program thus relies on an extensive virus signature library, for there are currently several thousand known IBM PC viruses and several new viruses appear each day. Any new viruses appearing after the antivirus program was developed are not included in the library of program codes for which the antivirus can scan. Signature scanning antivirus programs therefore require frequent updates to keep them current with the increasing number of viruses. If the antivirus developer is lax in providing updates, or the user is lax in obtaining and employing available updates, a signature scanning antivirus program can rapidly lose its effectiveness.

CHECKSUM MONITORS

The last standard technique of virus detection does not look for anything to do with viruses in particular, but concentrates on the host programs which the viruses attack. Every program on a system can be "checksummed" at antivirus installation time. Then, when a virus attaches itself to the unsuspecting host program, the checksum value will (probably) be different and the file infected with the virus can be isolated. The primary problem with this technique is that many programs store varying program information within themselves; this will change the checksum value and thus trigger a false alarm virus detection. Another problem is ensuring the integrity of the checksum information, which is typically attached to the program file itself or stored in a separate file. Both locations are vulnerable to covert virus modification. Once a virus infects a host, it can then update the stored checksum value to correspond to the newly infected file and then execute undetected.

SUMMARY OF THE INVENTION

An improved antivirus program according to a first aspect of the present invention avoids the problems of the prior art and detects viral infection of a target program by emulating the execution of the target program and scanning for viral behavior. By emulating the execution of the target program, viruses are prevented from circumventing the monitor program's protective mechanisms. A second aspect of the present invention recognizes that a key viral behavior is replication: viruses generally operate by passing replication/program-modification code onto uninfected programs. Uninfected programs, on the other hand, do not generally add program-modification code to other programs. According to this aspect of the invention, the emulated target program is tested for replication behavior to determine whether the target program is virus-infected.

A monitor program according to the first aspect of the present invention contains both variables corresponding to the CPU's registers and emulation procedures corresponding to the CPU's instructions. The monitor program includes means for loading a target program into memory and emulating its execution. The monitor program also includes means for analyzing the emulated behavior of the target program and for signalling a warning if the emulated behavior is determined to be aberrant, dangerous or otherwise undesirable.

In one embodiment according to the second aspect of the present invention, the monitor program further includes means, responsive to a file access request by the target program, for providing a dummy program, having known behavior, for modification by the target program. The monitor program also has means for emulating the execution of the modified dummy program after the emulation of the target program is complete. If the modified dummy program is determined to have modified functionality, the original target program is flagged as possessing viral behavior. In one particular embodiment according to this aspect of the invention, a first dummy program is known to not possess the ability to modify another file. If after modification by the target program the first dummy program is emulated and found to modify a second dummy program, then the original target program is flagged as virus infected, for having "infected" the first dummy file with aberrant behavior.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1A is a block diagram illustrating the primary components of a computer system executing a target program in a standard manner.

Fig. 1B is a block diagram illustrating the primary components of a computer system executing a target program according to the present invention.

Figs. 2A and 2B are diagrams illustrating memory maps of the computer systems of Figs. 1A and 1B, respectively.

Fig. 3 is a block diagram illustrating the register set emulated by a particular embodiment of the present invention.

Figs. 4A and 4B are flowcharts illustrating respectively the installation and replication procedures typically employed by computer viruses.

Fig. 5 is a flowchart illustrating the general emulation process performed by a monitor program according to a particular embodiment of the present invention.

Fig. 6 is a flowchart illustrating in further detail the memory access control step of the flowchart of Fig. 5.

Fig. 7 is a flowchart illustrating in further detail the procedure access control step of the flowchart of Fig. 5.

Fig. 8 is a flowchart illustrating in further detail the operating system entry point monitoring step of the flowchart of Fig. 5.

Fig. 9 is a flowchart illustrating the process performed by a particular embodiment of the present invention to check the behavior of an interrupt handler.

Fig. 10 is a flowchart illustrating the process performed by a particular embodiment of the present invention to identify viral replication behavior.

In Fig. 1A a block diagram is shown illustrating the primary components of a computer system executing a target program in a standard manner. The computer system includes a CPU 10, a memory 20, and a disk storage device 30. This is simply an exemplary configuration; the system could of course employ a tape storage device rather than disk storage, and many other variations are possible as well. Operating system 40 typically exists in Read Only Memory, but may also be partially loaded from the disk storage 30 into memory 20. At power up, the CPU begins executing the instructions of operating system 40, which thereafter controls the loading and execution of application programs such as target program 50.

In this standard configuration, if a user selects target program 50 for execution, operating system 40 would load target program 50 from disk storage 30 into memory 20 and then transfer control to target program 50 by loading the start address of target program 50 into the program counter register, or instruction pointer register, of CPU 10. CPU 10 would then begin executing the instructions of target program 50, as pointed to by the instruction

pointer register. Target program 50 will typically include calls to operating system routines, which are identified by a table of pointers, commonly known as interrupt vectors. It is by remapping these interrupt vectors that standard behavior interceptor antivirus programs attempt to maintain control and supervision of target programs. As discussed above, however, many computer viruses are able to circumvent this remapping of the interrupt vectors and are able to use operating system routines without being monitored by the antivirus program.

In order to prevent this circumvention of monitoring code, a particular embodiment of the present invention is invoked by a user to request that an application program be analyzed for viral behavior. This embodiment takes the form of a monitor program that emulates the execution of the application for a period of time, monitoring its behavior. By emulating the execution of the application program, the application program can be maintained in a controlled environment that cannot be circumvented by a virus.

The configuration of the monitor program and target application program is illustrated in Fig. 1B. Monitor program 60 loads target program 50 into memory and emulates the execution of the instructions of target program 50, serving as a protective barrier between the application program and the remainder of the computer system. If the application program has not shown any viral behavior at the end of the monitor period, then it is loaded and executed in the standard manner, such as illustrated in Fig. 1A.

In the secure environment created by the monitor program of Fig. 1B, every aspect of execution can be scrutinized and the operation of the virus can be controlled completely. If the virus were to request a hard disk format operation, a successfully completed status would be returned to it making the virus "believe" that the operation was successful when in fact it was never executed in the first place.

Figs. 2A and 2B respectively show the general layout in memory 70 for an IBM PC type computer system with a target program loaded directly by PC DOS as in Fig. 1A, and for a target program loaded by an embodiment of the present invention as in Fig. 1B. As shown in Fig. 2A, ROM occupies the upper portion of the memory address space with the remainder of memory being filled up from the bottom: first the operating system 40 in lower memory, followed by device drivers and memory resident programs, then user selected programs such as target program 50. Fig. 2B illustrates memory usage as in Fig. 2A, but additionally with monitor program 60 loaded.

Fig. 3 illustrates the various CPU registers employed by an 8086 type CPU, the general type of CPU employed by many personal computers, and for which the presently described preferred embodiment is intended. Flags register 300 is a set of bit-wise flags that may be set or cleared during the execution of various types of instructions. These bits can be examined by other instructions to alter program flow or to perform other tasks. Registers 310 are general purpose and are used for a variety of tasks. Index registers 320 are typically used to indirectly reference memory. Stack pointer 330 is used to maintain a data storage stack in memory. Instruction pointer (program counter) 340 points to the location in memory at which the next instruction to be executed resides. Finally, segment registers 350 are used to prepend and additional 4 bits onto other memory addressing registers (16 bits wide), allowing them to access a broader range of memory. Because these registers are intimately involved in the execution of programs, they are all emulated by the monitor program of the preferred embodiment, so as to fully control the execution of a target program.

Viral Code

Fig. 4A illustrates the installation procedure typically employed by computer viruses. The virus execution begins at block 400 and proceeds to block 410, at which the virus determines if a copy of itself has already been installed in memory. If not, execution proceeds to block 420, where the virus the current value of interrupt vector 21h (the operating system entry point on 8086 type computers), and saves this value for later use. Next, at block 430, the virus sets the entry point to point to a procedure within the virus itself, after which at block 440 control is passed to the host program. If at block 410 the virus had determined that a copy had previously been installed, control would pass immediately to block 440.

Fig. 4B illustrates a typical viral procedure for replication. The beginning of such a procedure would be the replacement entry point stored by the viral code at step 430 of Fig. 4A. When a program later attempts to make an operating system call through int 21, the call would be directed to beginning block 450 of the viral procedure of Fig. 4B. The viral code would then execute, and at block 460 would determine if there was a file name associated with the operating system call. Such operating system calls are typically used by a normal program to open a file or execute another program. If there was a name associated with the operating system call, then at block 470 the viral code would replicate itself by writing its own executable code to the file that was the subject of the

operating system call, in some instances after having checked to ensure that this file was not already infected by the virus. After block 470, the viral code would then exit at block 480, passing control to the original interrupt handler, a pointer to which had been saved at block 420 of Fig. 4A. If at block 460 the viral code had determined that there was no filename associated with the operating system call, then execution would have passed directly from block 460 to block 480. In this manner the operating system continues to function normally except for a slight interruption while the viral code executes.

Emulation to Detect Viral Code

Fig. 5 illustrates the operation of monitor program 60 according to a preferred embodiment of the invention. The monitor program can be executed explicitly by the user with a designated target program, or in alternative embodiments can be executed automatically whenever an operating system call is placed to execute a program. At block 500, the monitor program loads the target program into memory, in exactly the same manner as the operating system would have loaded the target program, but rather than passing execution to the target program immediately, the monitor program retains control for a period of time, to evaluate the target program.

After the target program is loaded at block 500, at block 510 the monitor program initializes the emulated registers, which correspond to the registers used by CPU 10. These register variables are used by a set of instruction emulation routines that are capable of emulating the instructions of CPU 10. The emulated registers are initialized with the same values that the real registers would have had if the target program had been loaded by the operating system for execution.

After the emulation registers are initialized, the main emulation loop is entered. At block 520 the instruction pointed to by the emulated program counter register is fetched by the emulation software and the emulated program counter register is incremented by the size of the fetched instruction, so that it points to the next instruction. Control then proceeds to a set of evaluation procedures for the instruction. At block 530, the monitor program determines if the target program is attempting to access memory selected for controlled access. In the preferred embodiment, operating system procedures and data areas the address range of the monitor program are selected for controlled access. Optionally, any memory not belonging to the target program can be selected for controlled access. The memory access process is explained in more detail below with reference to Fig. 6.

After block 539, at block 540 the monitor program evaluates the instruction for attempted access to a controlled procedure, explained more fully below with reference to Fig. 7, and then also emulates the execution of the instruction. Following block 540 is block 550, at which the monitor program evaluates any possible modifications to the operating system entry points. The processes performed at block 550 are described in more detail below with reference to Figs. 8-10.

Following the emulation and evaluation blocks 530-550, at block 560 the monitor code determines if the target application has terminated. If so, emulation is terminated at block 570. The determination of step 560 can be according to whether the target program terminates of its own accord, or the determination can be set by a total number of instructions to be emulated or by a fixed period of time for emulation. If the target program has not terminated of its own accord at step 560, and if the monitor program has not forcibly terminated it, control returns to block 520, where the next cycle of the emulation loop is begun. The emulation termination at block 570 includes some "cleanup" on the part of the monitor program. This includes displaying to the user a status report of all operating system requests performed by the target program. This step may optionally also include reporting any memory accesses that have been performed outside of the area provided for the target program by the monitor program.

Controlling Access to Memory

The memory access monitoring process of block 530 is illustrated in further detail in Fig. 6. The described process involves remapping selected parts of memory, which effectively virtualizes those memory areas, making them inaccessible to the target program, and thus protected. In alternative embodiments, access by the target program to these areas of memory is simply denied by the monitor program.

From the starting point at block 600, the procedure passes to block 610, at which the monitor program determines if the current instruction is one whose function is to access memory. If so, then control passes to block 620, where the monitor program determines if the memory location to be accessed by the current instruction is in an area selected for controlled access. If so, then control passes to block 630, which implements a remapping of the memory address. The monitor program's representation of the instruction is modified to point to the mapping destination, so that the original memory location is protected from the target program.

In the preferred embodiment, the contents of the original memory location are copied to the mapping destination the first time the location is accessed by the target program. In other embodiments, the contents of the entire memory area selected for controlled access are copied into the mapping destination area when the monitor program first starts. In yet other embodiments, certain areas selected for controlled access can have their mapping destination areas initialized with null or dummy values. For example, it may be desirable that the content of the monitor program be protected and hidden from the target program, so that a virus cannot detect the presence of the monitor program.

After the remapping of block 630, at block 640 the attempted access to a controlled memory area is logged for later analysis and reporting to the user. After block 640, the memory access control procedure ends at block 650, which returns control to the main process of Fig. 5, at block 540. A negative determination at either of blocks 610 or 620 also results in control passing immediately to block 650.

Controlling Access to Procedures

In some instances, it is desirable to control access to certain procedures. For instance, operating system procedures, ROM procedures, and interrupt handling procedures can have powerful effects and can be subject to misuse by a virus. For these reasons, it is desirable to control access to them and substitute special purpose procedures in their place, to encapsulate viral code within the emulated environment.

After the memory access control procedure of block 530 of Fig. 5, control passes to block 540, which is illustrated in further detail in Fig. 7. From beginning block 700 control passes to block 710, at which the monitor program determines if the emulated program counter points to a controlled procedure entry point; a list of such entry points is maintained by the monitor program. If so, then at block 720 the attempted access to a controlled procedure is noted. This can be by displaying a message to the user on the screen, writing to a log file, etc.

Next, at block 730 the monitor program determines if the instruction is to be directly emulated. This determination is made according to information stored for each controlled procedure entry point: for certain such procedures a special case emulation may be desired rather than directly emulating the instructions of the procedure. If the procedure is not to be directly emulated, then control passes to block 740, where a special case emulation of the entry point instruction is performed. In

some instances this special case emulation will entail emulation of the entire controlled procedure at this point.

If at block 730 it were determined that the controlled procedure was to be directly emulated, or if at block 710 it were determined that the emulated instruction pointer did not indicate a controlled procedure entry point, then control would pass to block 750, where the instruction indicated by the emulated instruction pointer is emulated in the same manner as other instructions. Following the emulation according to either of blocks 740 or 750, control passes to block 760, which returns execution to the main process of Fig. 5.

Controlling Access to Operating System Entry Points

Block 550 is illustrated in further detail by Fig. 8. This control of operating system entry points need not be performed to obtain substantial benefits from this emulation of the target program; however, this process does a higher level of control over the target program and also allows for a more accurate evaluation of viral behavior on the part of the target program.

From beginning block 800 control passes to block 810, at which the monitor program examines a list of operating system entry points to determine if any have changed as a result of the instruction just emulated. This would indicate that the target program had replaced an interrupt handler with a routine of its own. If there is such a change, then it is logged at block 820. At block 820 a flag is also preferable set to indicate that the entry point has changed, so that the change will not be logged redundantly later. In some embodiments, the flag indicates the new value of the entry point, so the monitor program can determine if the entry point gets modified yet again.

After block 820, at block 830 the emulated instruction pointer, emulated code segment register, and emulated flag register are saved onto the emulated stack. Then the emulated stack pointer is decremented the corresponding 8 bytes, in the same manner as if a hardware interrupt had been received. Next, at block 840, the emulated code segment register and emulated instruction pointer are set to a special purpose monitor program routine to test the interrupt handler just installed by the target program. This interrupt handler testing routine is described below with reference to Fig. 9.

After block 840, execution passes to block 850, which returns control to the basic process of Fig. 5. This causes the interrupt handler routine of Fig. 9 to be emulated in the same step by step manner as the target program. This maintains the highest degree of encapsulation around the target program,

although if detecting viral replication is essentially the only concern, the interrupt handler testing routine of Fig. 9 may alternatively be executed in a more straightforward emulation without many of the execution safeguards described above.

If at block 810 the monitor program had determined that no operating system entry points had been changed, then control would have passed directly to block 850, and thus returned to the process of Fig. 5 to emulate the remainder of the target program.

Interrupt Handler Testing

The basic tack of the interrupt handler testing routine is to offer up a guinea pig file for "sacrifice" to a potential viral interrupt handler, and then test the guinea pig file for corruption. This requires that a "clean" guinea pig file already be at hand and also be disposable. This can be easily provided for by several methods, such as by creating the guinea pig file or copying the guinea pig file from a clean library copy at the very start of the monitor program. The guinea pig file should have a known content. It is preferably executable, but without its execution involving writing to other files. The guinea pig file can thus be essentially a null file that does nothing when executed, simply returning immediately.

As shown in Fig. 9, when the interrupt handler routine is entered at block 900, the first action is to open the guinea pig file, at block 910, after which the guinea pig file is closed at block 920. Next, at block 930 the interrupt handler testing routine examines the guinea pig file to determine if its content has been changed. Such would be the result of a virus having contaminated the interrupt handlers for opening or closing files. If a change is not detected at block 930, then at block 940 the guinea pig file is executed, after which at block 950 the guinea pig file is again examined by the interrupt handler testing routine to determine if its content has been changed by the execution interrupt handler.

If a positive determination had been made at either of blocks 930 or 950, then execution would pass from the respective block to block 960, at which the unauthorized access to the guinea pig file would be logged. After block 960, and also after a negative determination at block 950, execution passes to block 970, which executes an (emulated) IBET instruction. This is a return from interrupt instruction, which causes the values placed onto the emulated stack at block 830 of Fig. 8 to be restored to the emulated registers. This completes the interrupt handler testing, and returns the emulation to its last point of emulation in the target program.

For a more refined and definitive degree of analysis, block 960 can also initiate a routine to determine not just if the guinea pig was contaminated, but if it was contaminated in a way so as to contaminate other files; i.e., if it was infected with viral replication behavior. Such a routine is illustrated in Fig. 10. The process of Fig. 10 essentially creates a completely new emulation, with the modified guinea pig file serving as the target program. If this first guinea pig file now passes modification behavior on to a second guinea pig file, then the original target program has been shown to be contaminated with viral code having replicative behavior. To prevent needless additional recursion, the second level of emulation should be identified as such, through use of a flag, etc., so that if block 960 is reached during the second level of emulation, viral behavior is confirmed and the second level of emulation is terminated (rather than beginning another level of testing with yet another guinea pig file).

This replication detection process is illustrated in Fig. 10. After beginning at block 1000, at block 1010 the complete state of the current emulation is saved, and all operating system entry points, etc., are returned to their values at the beginning of the first emulation. Block 1010 also then includes the step of initiating emulation again, but with the guinea pig file specified as the target program. As noted above, this emulation level should be flagged as a second level emulation.

Block 1020 indicates the point at which the emulation of the guinea pig file has terminated, after which at block 1040 the first level emulation determines if the emulated guinea pig file had written to a second guinea pig file. This determination is most straightforward if a flag is simply passed from the second level emulation back to the first; it can also be by examining a checksum for the file. If the determination is positive, then at block 1050 the initial target program is confirmed and logged as being virus-contaminated. At block 1060 at the end of the process of Fig. 10, control is passed back to block 970 of Fig. 9, to continue emulation of the initial target program. Alternatively, reaching block 1050 can result in the entire emulation being terminated, as the target program has been confirmed as being virus-contaminated.

Alternative Embodiments

Rather than requiring the user to load the monitor program which then loads the target program, a "zero length loader" TSR version could be installed in a system and every program requested to be executed could be emulated. If no abnormal behavior is found in the first 'n' instructions, the monitor program could pass control to the CPU to allow the

target to execute at "full speed" and the end user would not have to be aware of the existence of the monitor program (other than a slight delay during the initial execution).

Another alternative approach would be where a recursive parser/emulator could effectively evaluate every single instruction of executable code in a program by noting the address of conditional branch instructions, and returning to that branch location, restoring the op/memory state of the machine at that instant, and continuing emulation as if the branch had taken the alternate route instead. Emulation continues until all instructions have been evaluated. This would be a time consuming process; however, the information revealed would definitively answer the question of whether the original code was virus infected.

It is also important to note that, although the described embodiment is oriented towards identifying viral behavior, the disclosed emulation techniques can be constructively employed to emulate program execution in all types of situations where potentially destructive or other predetermined program behavior is a concern.

It is to be understood that the above description is intended to be illustrative and not restrictive. Many other embodiments will be apparent to those of skill in the art upon reviewing the above description. For instance, the instructions of the emulated application program could be read directly from disk storage rather than being loaded into memory first. The scope of the invention should, therefore, be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.

Claims

1. A computer system configured to monitor the execution of a target program, said computer system comprising:
 - a processing unit having an instruction set;
 - instruction emulation means for emulating instructions in the target program corresponding to said instruction set;
 - monitor means, coupled to said instruction emulation means, for emulating execution of said target program and for monitoring said emulated target program execution to detect a predetermined behavior by said target program; and
 - means, coupled to said monitor means, for logging said predetermined behavior when detected.
2. The computer system of claim 1, wherein said computer system is configured to detect a computer virus associated with said target pro-

- gram, wherein said predetermined behavior is chosen to be indicative of replication of said computer virus.
3. The computer system of claim 1 wherein said instruction emulation means comprises:
 - a register emulator for emulating registers of said processing unit; and
 - a procedure controller for substituting emulation procedures for procedures accessed by said target program during emulation.
 4. The computer system of claim 1 wherein said instruction emulation means comprises:
 - a memory access controller for controlling access by said instructions to memory.
 5. In a computer system, a method for monitoring execution of a target program comprising the steps of:
 - emulating the target program; and
 - monitoring emulation of the target program to detect a predetermined behavior indicating presence of a computer virus.
 6. A computer system configured to monitor the execution of a target program, said computer system comprising:
 - a processing unit having an instruction set;
 - an instruction emulator for emulating instructions corresponding to the instruction set;
 - an entry point access controller for controlling access to operating system entry points; and
 - a logger for logging improper access by said instructions to operating system entry points.
 7. The computer system of claim 6 further comprising a procedure access controller for controlling access to procedures during instruction emulation, wherein said logger logs improper access by said instructions to procedures.
 8. The computer system of claim 6 further comprising a memory access controller for controlling access by said instructions to memory during instruction emulation wherein said logger logs improper access by said instructions to memory.
 9. The computer system of claim 6 wherein said entry point access controller includes an interrupt tester for checking if a viral interrupt has been installed.
 10. The computer system of claim 9 wherein said interrupt tester executes a guinea pig file and tests for modification of the guinea pig file to determine if a viral interrupt has been installed.
 11. The computer system of claim 9 wherein said interrupt tester opens and closes a guinea pig file and tests for modification of the guinea pig file to determine if a viral interrupt has been installed.
 12. The computer system of claim 10 wherein said interrupt tester executes the guinea pig file as a new target program, thereby creating a second guinea pig file, and tests for modification of the second guinea pig file to determine if a replicative viral interrupt has been installed.

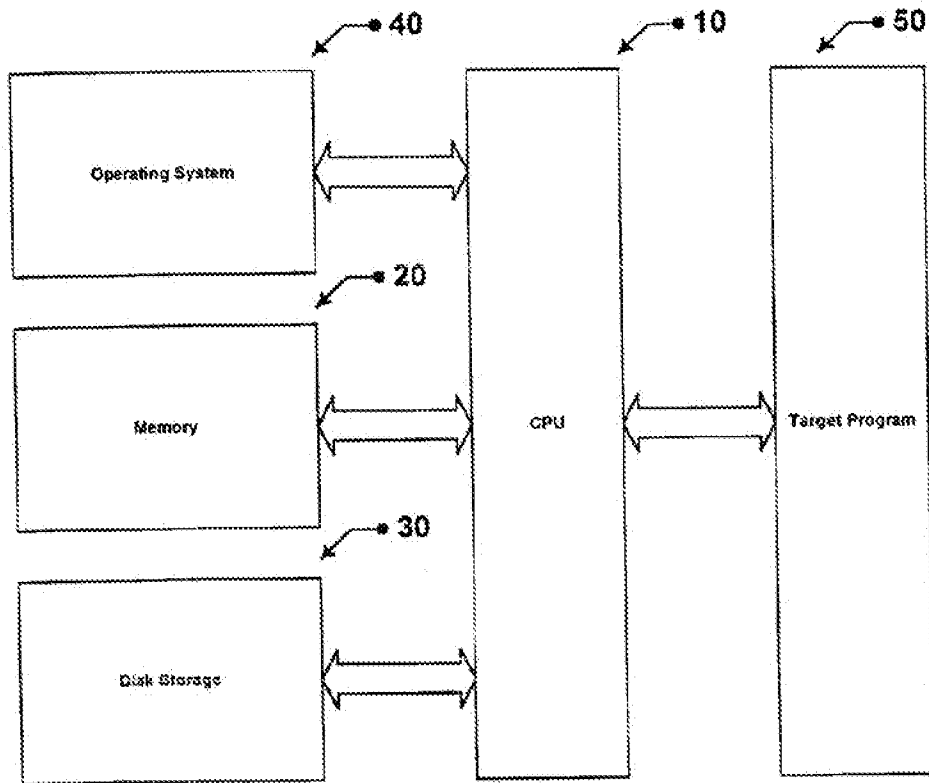


Figure 1A
(Prior Art)

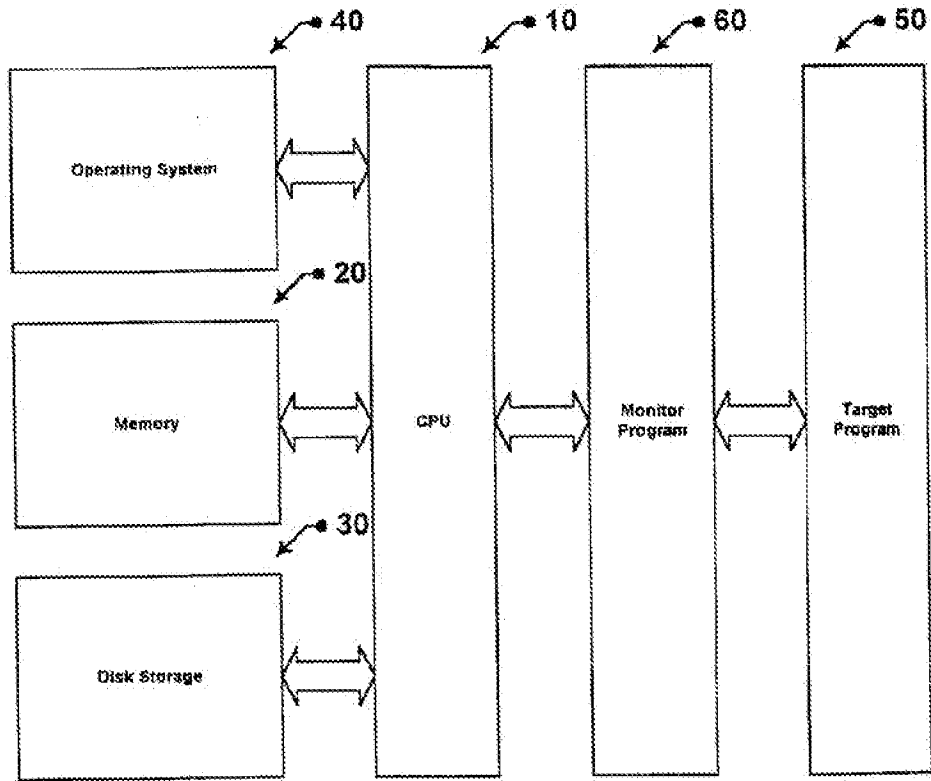


Figure 1B

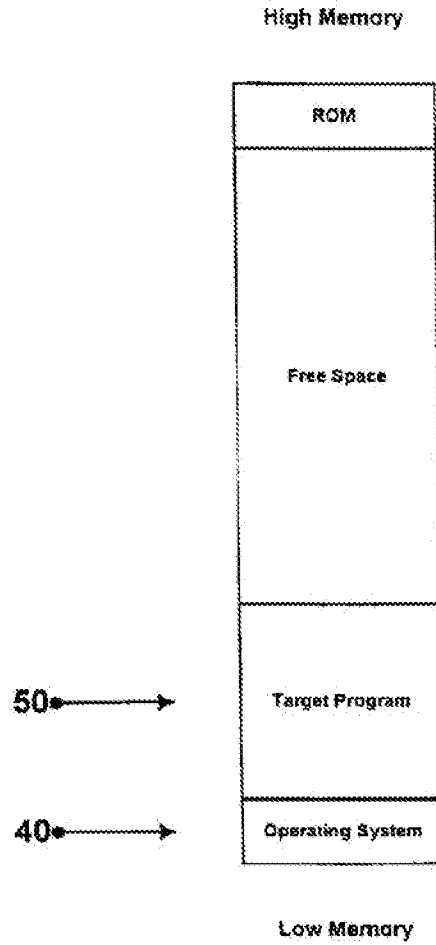


Figure 2A
(Prior Art)

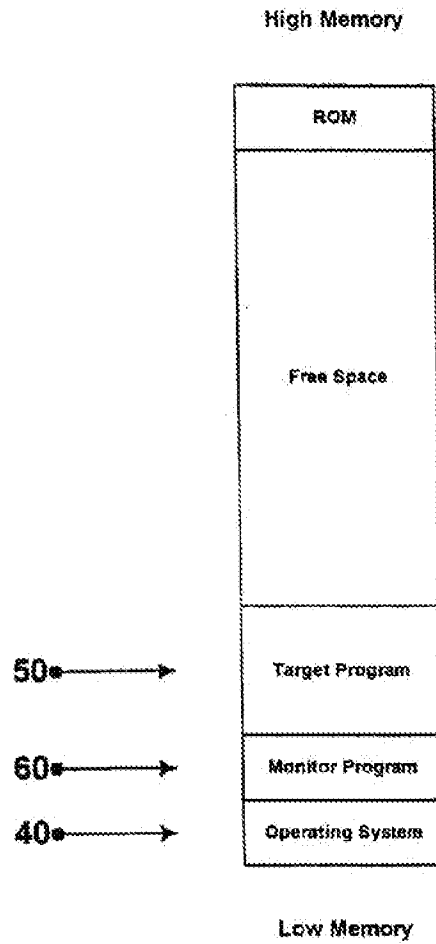


Figure 2B

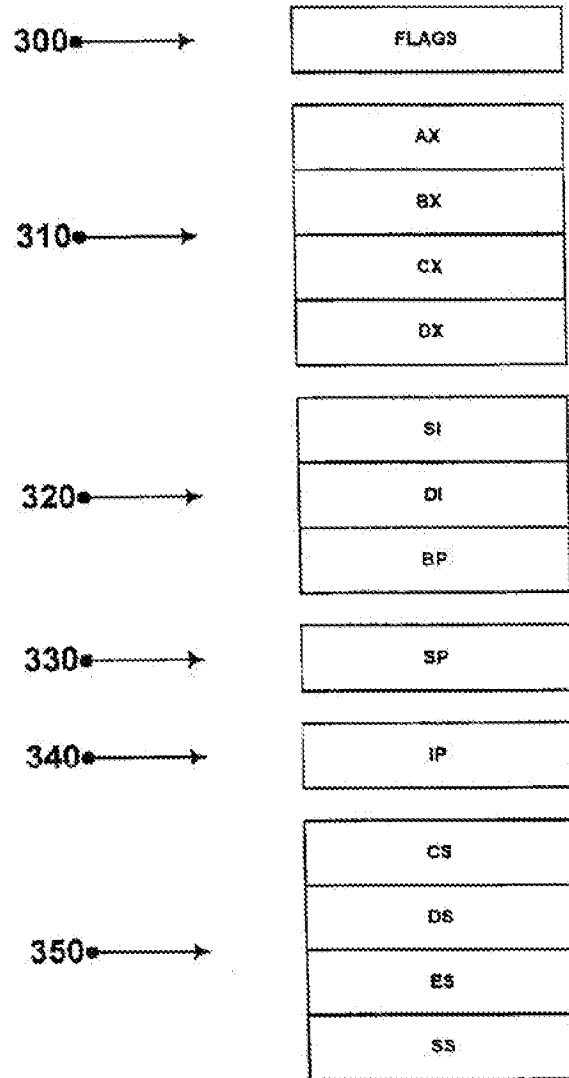


Figure 3
(Prior Art)

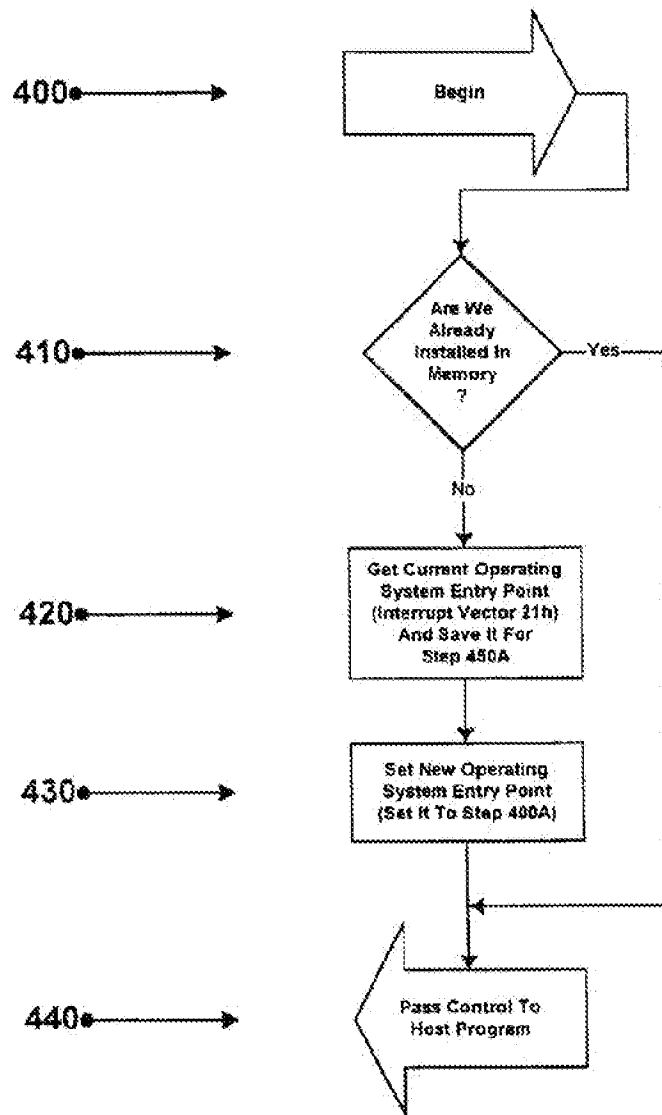


Figure 4A
(Prior Art)

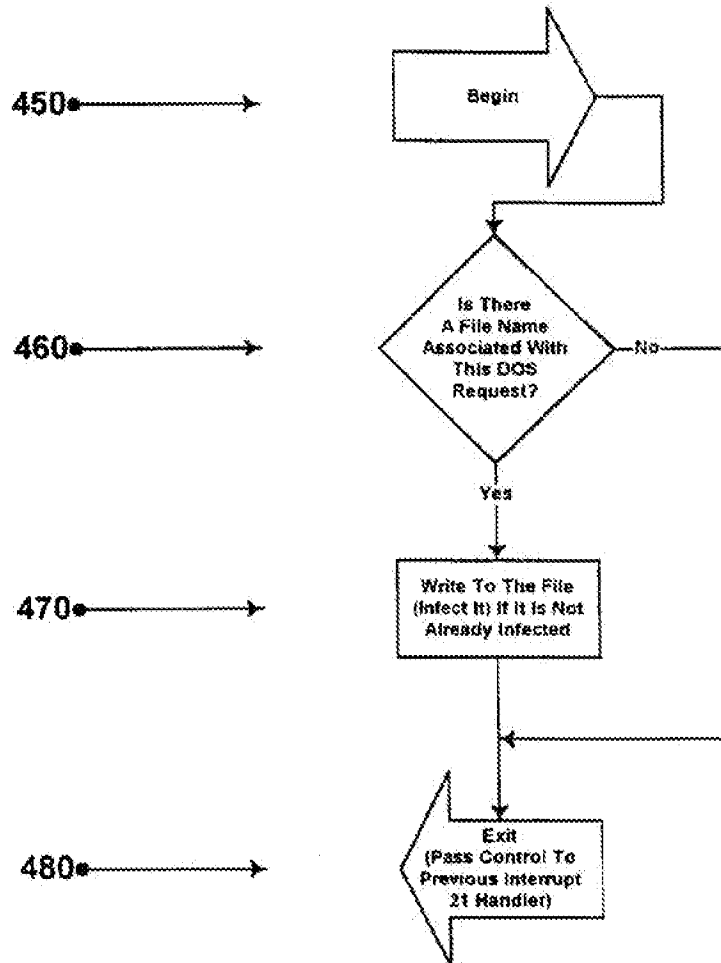


Figure 4B
(Prior Art)

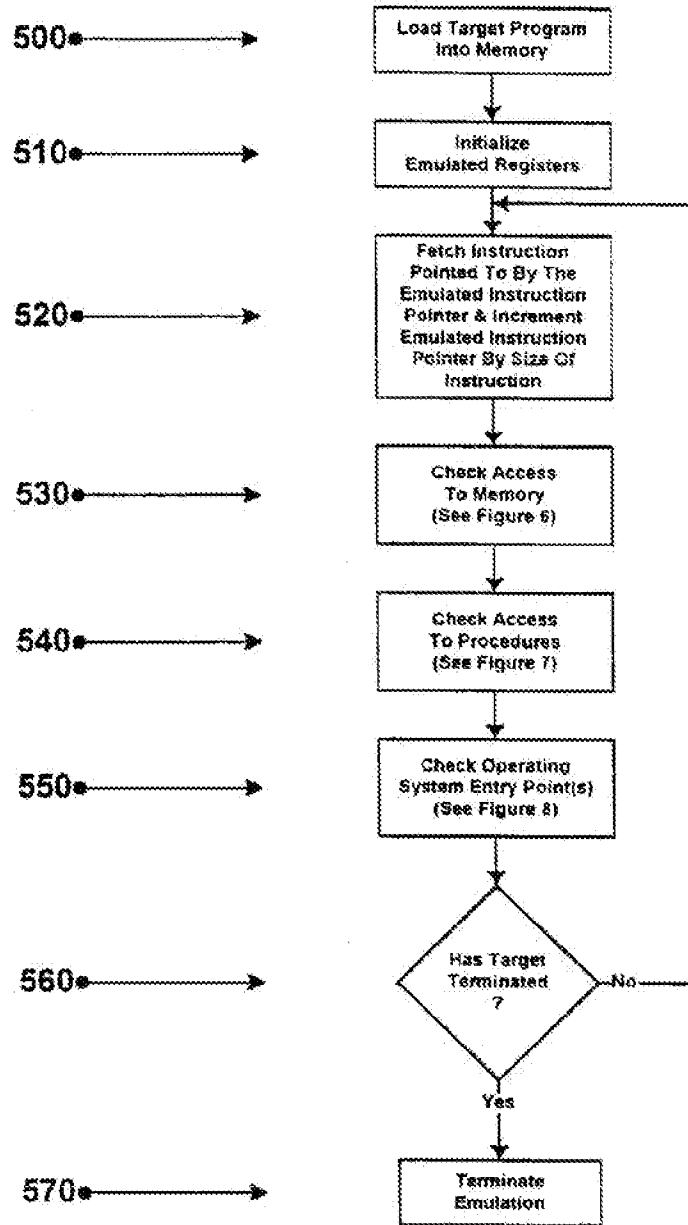


Figure 5

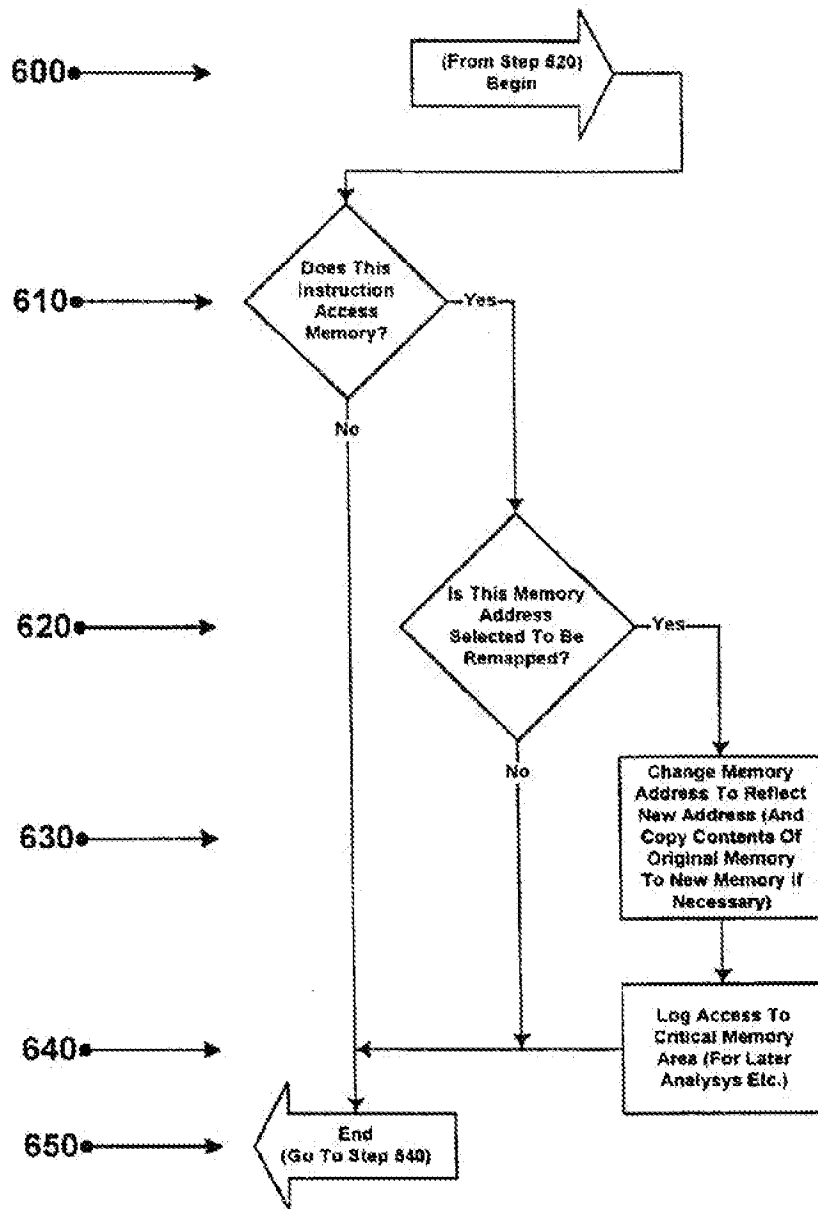


Figure 6

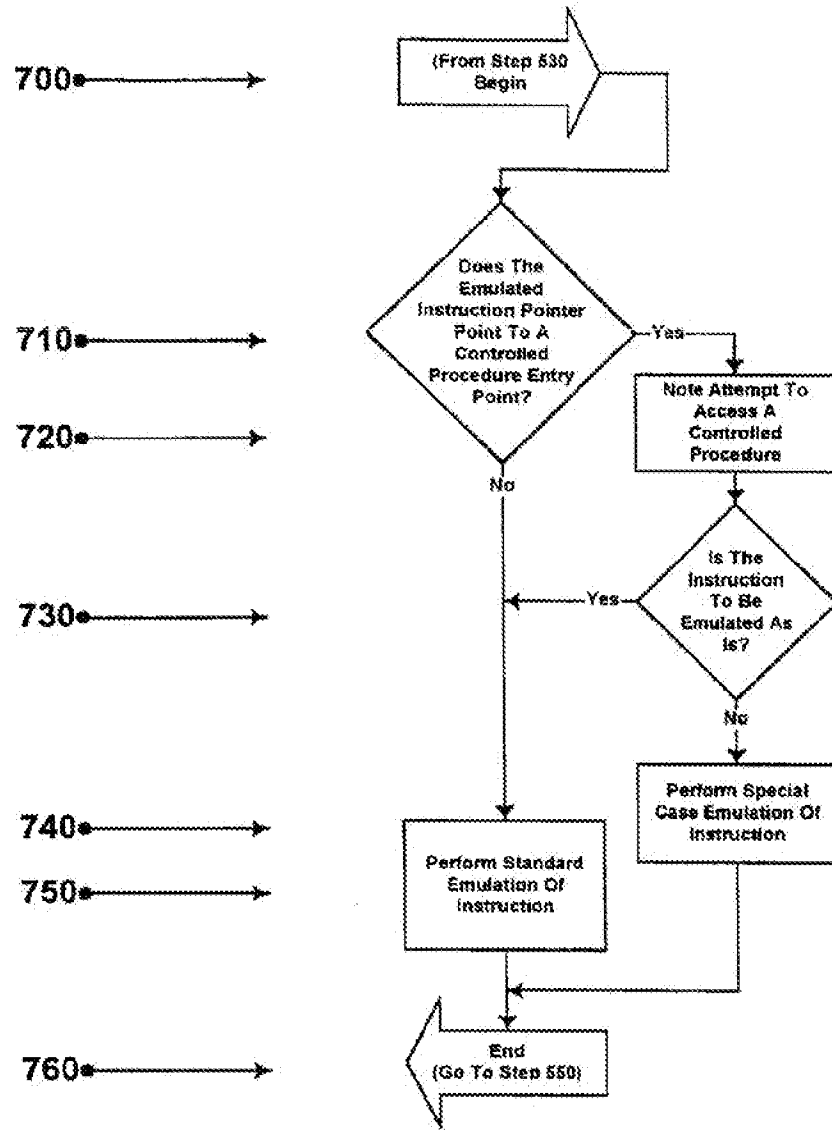
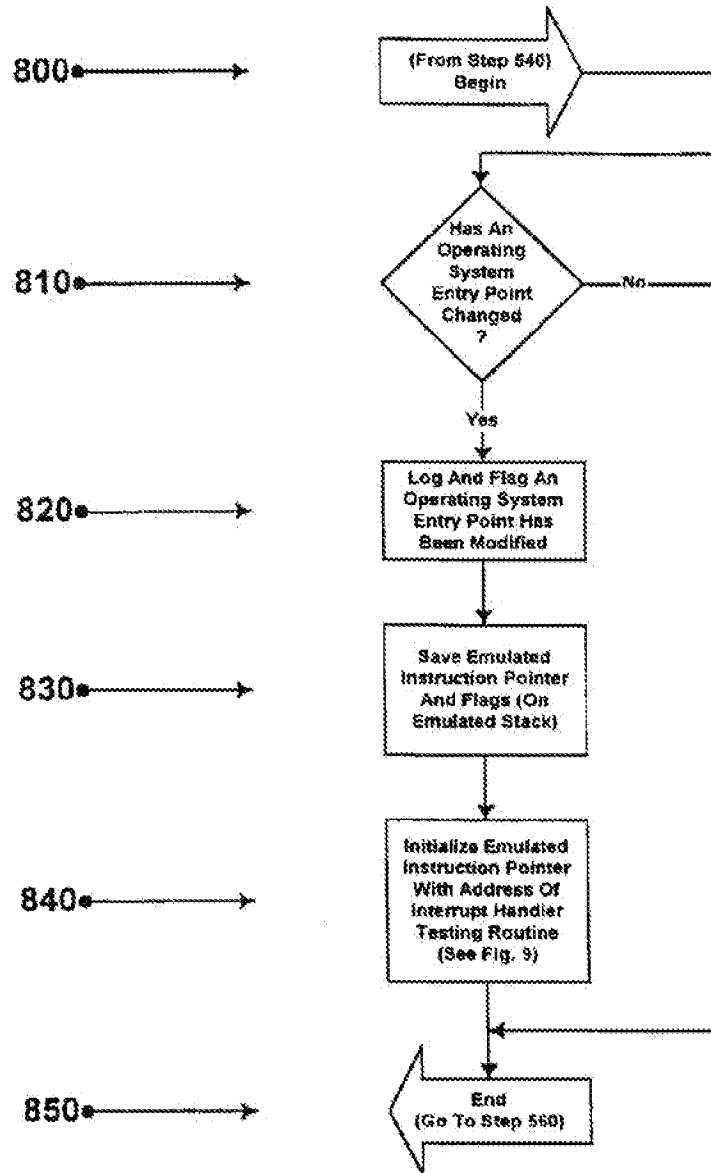


Figure 7

**Figure 8**

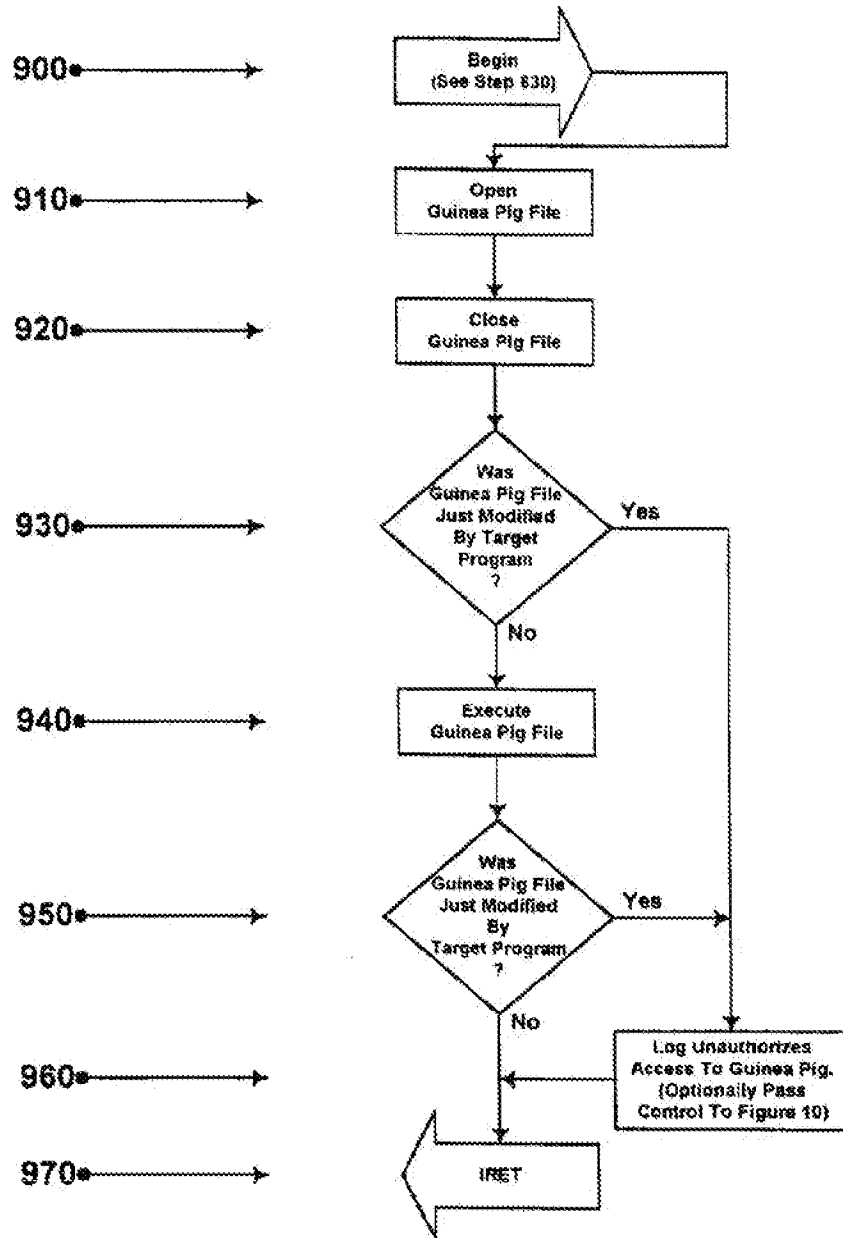
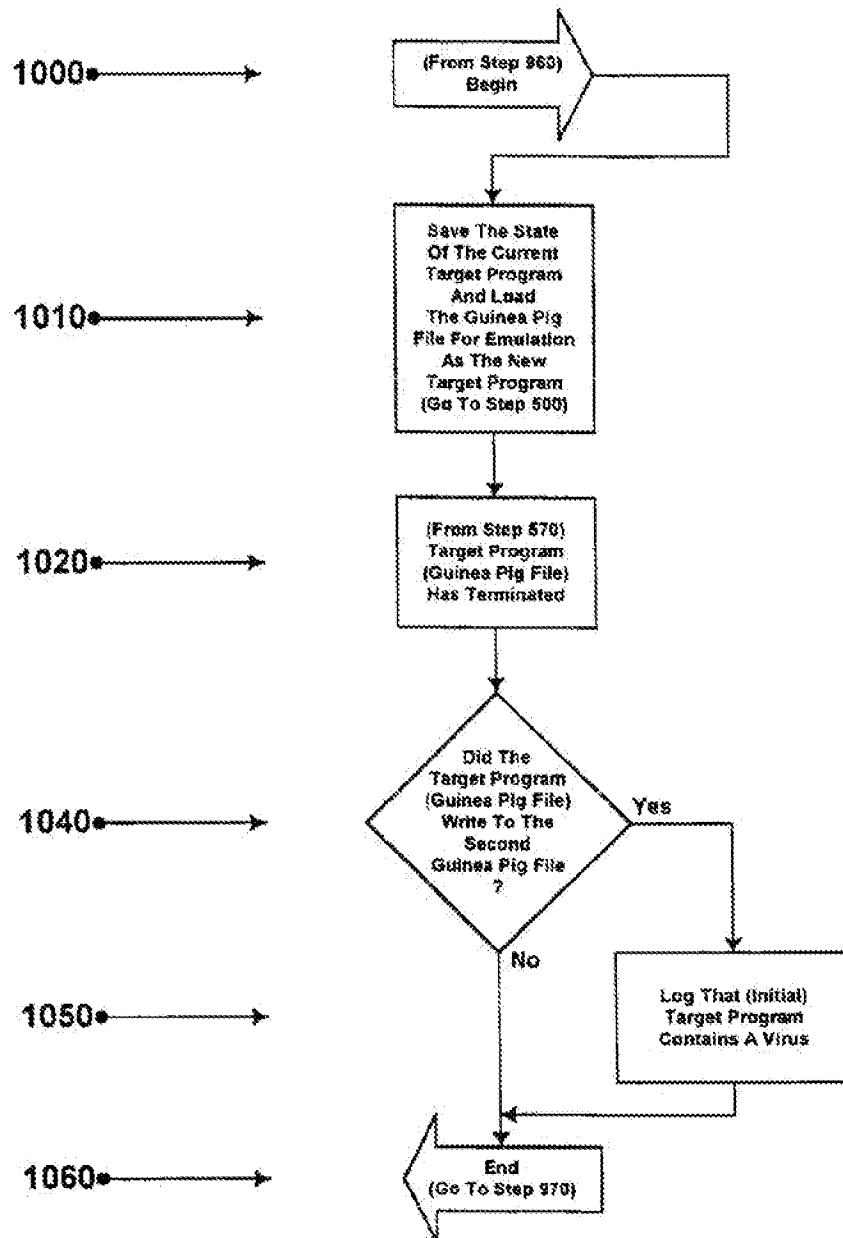


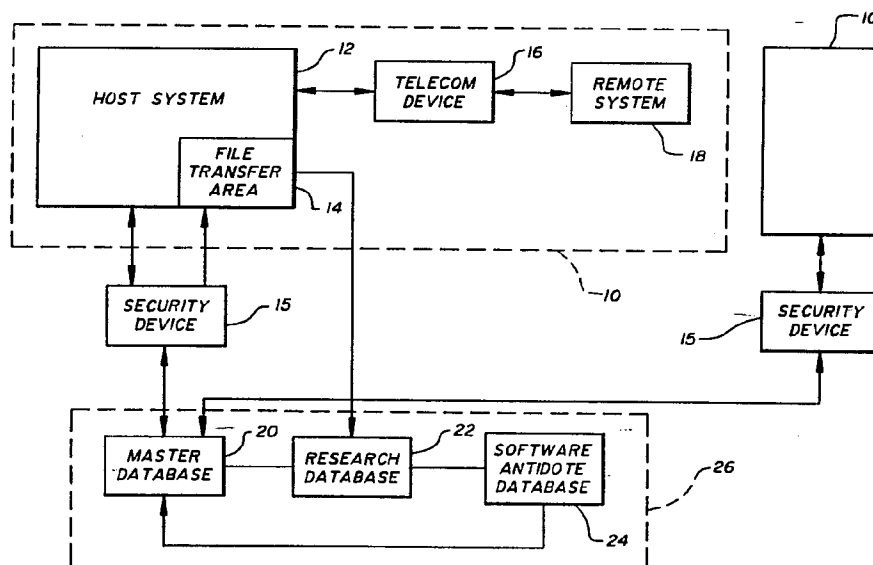
Figure 9

**Figure 10**



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁵ : H04L 9/00	A1	(11) International Publication Number: WO 93/25024 (43) International Publication Date: 9 December 1993 (09.12.93)
(21) International Application Number: PCT/US93/05029 (22) International Filing Date: 26 May 1993 (26.05.93) (30) Priority data: 07/888,909 26 May 1992 (26.05.92) US (71) Applicant: CYBERLOCK DATA INTELLIGENCE, INC. [US/US]; 906 Woodbrook Lane, Philadelphia, PA 19150 (US). (72) Inventor: BRANHAM, Reginald, K. ; 906 Woodbrook Lane, Philadelphia, PA 19150 (US). (74) Agent: HEIDELBERGER, Louis, M.; Reed Smith Shaw & McClay, 2500 One Liberty Place, 1650 Market Street, Philadelphia, PA 19103-7301 (US).		(81) Designated States: CA, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE). Published <i>With international search report.</i>

(54) Title: COMPUTER VIRUS MONITORING SYSTEM**(57) Abstract**

A method and an apparatus for preventing the infection of computer systems (10) by computer viruses is disclosed. The computer virus monitoring system provides an external security device (15) that stores copies of the host boot sector and file allocation table, and electronic fingerprints of executable files on the host system disk. The external security device (15) monitors writes to the host disk and informs a network monitoring host (26) if the boot sector, file allocation table or electronic fingerprints are altered. The network monitoring host (26) responds to such an emergency by saving the status of the host system (12) and transferring corrective software to remove the virus. The computer virus monitoring system further provides for periodic scanning of the host files to detect and eradicate known viruses.

FOR THE PURPOSES OF INFORMATION ONLY

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

AT	Austria	FR	France	MR	Mauritania
AU	Australia	GA	Gabon	MW	Malawi
BB	Barbados	GB	United Kingdom	NL	Netherlands
BE	Belgium	GN	Guinea	NO	Norway
BF	Burkina Faso	GR	Greece	NZ	New Zealand
BG	Bulgaria	HU	Hungary	PL	Poland
BJ	Benin	IE	Ireland	PT	Portugal
BR	Brazil	IT	Italy	RO	Romania
CA	Canada	JP	Japan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo	KR	Republic of Korea	SE	Sweden
CH	Switzerland	KZ	Kazakhstan	SK	Slovak Republic
CI	Côte d'Ivoire	LJ	Liechtenstein	SN	Senegal
CM	Cameroon	LK	Sri Lanka	SU	Soviet Union
CS	Czechoslovakia	LU	Luxembourg	TD	Chad
CZ	Czech Republic	MC	Monaco	TG	Togo
DE	Germany	MD	Madagascar	UA	Ukraine
DK	Denmark	ML	Mali	US	United States of America
ES	Spain	MN	Mongolia	VN	Viet Nam
FI	Finland				

COMPUTER VIRUS MONITORING SYSTEM

Field Of The Invention

This invention relates to detection of viruses or logic bombs in computer systems and for taking
5 responsive action upon such detection.

Background Of The Invention

Security of computer systems is becoming increasingly important. Computer viruses and logic bombs ("viruses") are proliferating, and their potential impact
10 is increasing with the increasing use of network systems. Efforts to address the problems posed by viruses have included backing up the contents of computer systems on magnetic tape or other media, and storing the backup in a safe location. Such systems are time consuming and
15 expensive to implement, and only serve to mitigate the effects of a problem after a problem has occurred. Others have developed software programs, resident on a computer system to be protected, to monitor the system for viruses. Such protection systems have certain drawbacks: for
20 instance, because they are resident on a system which may become infected, they are similarly subject to disruption or corruption. Such systems may also be cumbersome to update to respond to newly encountered viruses.

Summary Of The Invention

25 It is therefore an object of the invention to provide an improved computer virus monitoring system. In accordance with the invention, the system includes a monitoring computer which is coupled to the system to be protected. The monitoring computer monitors the boot
30 sector and file allocation table of the disk drives of the

-2-

host system to be protected, and creates and maintains identification data for each executable file. The monitor detects attempts to alter these files or attempts by a file to alter itself or another file and takes appropriate responsive action to protect the host system from unauthorized alteration. The monitor also includes means for scanning the host system to detect particular viruses, and to take appropriate protective action upon detection of a virus. In a particularly preferred embodiment, such monitors on a number of host systems are coupled by a communication network to facilitate analysis of viruses and distribution of antidotes therefor.

Other objects and features of the invention will be understood with reference to the following specification and claims and the drawings.

Brief Description Of The Drawings

Figure 1 is a block diagram showing the basic elements of a system in accordance with the invention.

Figure 2 is a flow diagram illustrating a virus protection method which may be implemented on the system of Figure 1.

Figure 3 is a flow diagram of a method of operating the host system of Figure 1 in accordance with the invention.

Figure 4 is a flow diagram illustrating a method of operation of the monitor of Figure 1 in accordance with the invention.

Figure 5 is a flow diagram illustrating a method of operation of the monitor in response to an emergency condition.

Figure 6 is a flow diagram illustrating a method of communication between monitoring system at a number of host locations and a central monitoring system.

-3-

Figure 7 is a block diagram illustrating the basic elements of a preferred embodiment of a security device in accordance with the invention.

Detailed Description of the Invention

5 Figure 1 is a diagram showing the functional elements of the preferred embodiment of the system of the present invention. A computer network 10 which is to be protected (the "protected network") includes a host system 12 operating at a customer field site that communicates
10 with one or more remote systems and communication nodes 18 via one or more telecommunication devices 16. In a typical application, host system 12 may be an IBM fileserver, an Apple fileserver, or other computer system. Telecommunications device 16 may be a modem, a network
15 card, or a satellite transceiver, for example.

Security device 15 monitors the operation of host system 12 in three different modes to automatically detect viruses and prevent adverse effects. Security device 15 monitors the boot sector and file allocation
20 table (FAT) of disk drives in host system 12. It creates and maintains identifying data as a "fingerprint" for each executable file. These files should never be altered; therefore, if an attempted alteration is detected, appropriate protective action may be taken. Finally,
25 security device 15 scans host system 12 periodically to detect particular viruses.

Security device 15 is coupled to host 12 via a parallel connection to the standard architectural bus of host 12 in the preferred embodiment; however, security
30 device 15 may be coupled to host 12 via a serial port in an alternate embodiment. Security device 15 is also coupled to a security network monitoring host 26, by a telephone line 19 or other communications link.

Desirably, a number of protected networks 10,
35 each with its own security device 15, are connected in a

-4-

network with a security network monitoring host computer system 26. Security network monitoring host 26 includes a master service database 20 to monitor standard operation of security devices 15. It also includes a research service database 22 which may be called by a protected network 10 in case of an emergency. Because research service database 22 includes storage media such as disk drives, it may become infected by a virus. Therefore, research database 22 is kept separate from the master service database 20. Security network monitoring host system 26 also includes an antidote software service database 24, in which the characteristics of newly encountered viruses are stored and in which solutions to eradicate them are maintained. Antidote software may be transferred via master database 20 to security devices 15 in the monitoring network as needed.

In a system such as that shown in Figure 1, it is important for security device 15 to be as secure and reliable as possible. Therefore, security device 15 is preferably a stand alone unit without a disk drive or any other alterable storage device except RAM, to minimize the possibility of infection. Security devices 15 are provided with battery backup to maintain operation during a power failure. An emergency routine is called when security device 15 detects tampering, such as an unauthorized attempt to open its housing. Communications between security network monitoring host 26 and security devices 15 or protected networks 10 are desirably protected by requiring device-specific electronic identification numbers (EINs) to initiate communication.

Figure 2 is a flow diagram showing a method for operating a host system 12 and a security device 15 to provide virus protection for executable files. In this method, an electronic binary "fingerprint" of identifying data is created and assigned for each executable file on host system disk drive pack. The identifying data is then after each write access monitored by security device 15.

-5-

If the identifying data changes, then security device 15 will halt the protected system and initiate a call to the research database 22 at security network monitoring host 26.

5 The process of Figure 2 for monitoring the software of the protected system starts in step 50 and new software is added to the protected system in step 52. In step 54 the fingerprint, which can be conventionally generated as a check sum, a cyclic redundancy check (CRC)
10 or other error detecting coding scheme, is created to provide an identifying code that is unique to the file. In a preferred embodiment, the fingerprint is generated by host system 12, although in alternate embodiments the fingerprint could be generated on security device 15 or on
15 a dedicated circuit card. The fingerprint is saved on host system 12 in step 58, and on security device 15 in step 56. Steps 60 and 62 provide an ongoing process, for determining the status of the software by monitoring the fingerprint. The host fingerprint is read in step 60. In
20 step 62 the host fingerprint is compared with the corresponding fingerprint in security device 15. If the host fingerprint has not changed, the process returns from step 62 to step 60 to continue the monitoring process. If the fingerprint has changed, then in step 64 security
25 device 15 halts operation of host system 12. Security device 15 then initiates a call to the system operator of host system 12 in step 68, and in the preferred embodiment, a call to the research database 22 of security network monitoring host 26 in step 70. Also in response
30 to detection in step 62 of a fingerprint change, security device 15 attempts in step 72 to destroy the virus which caused the fingerprint change.

 The process of destroying the virus involves comparison of the contents of files stored on host 12 with
35 characteristics of known viruses stored in security device 15 downloaded from master database 20 of network host 26. If characteristics of a known virus are detected, antidote

-6-

software, stored in antidote software database 24, is executed to destroy the virus in step 74. Upon successful detection and destruction of a virus, a report of this activity is generated in step 76, transmitted to the
5 research database 22 of security network monitoring host 26 in step 78, and transmitted to the system operator of host system 12 in step 80. Step 82 returns the process to step 54, in which a new fingerprint is created, and then saved. Host system 12 may thereafter continue its
10 operation free from the virus. If in step 74 the virus has not been successfully destroyed, a further call is initiated by security device 15 to research database 22 for further assistance in identification and destruction of the virus, while operation of host system 12 is
15 disabled to prevent consequential damage from the virus.

Figure 3 is a flow diagram of the operation of host system 12 relating to security device 15. Initialization steps of host system 12 include porting the boot sector and FAT of host system 12 to security device
20 15 in steps 90 and 92 respectively, creating a fingerprint file in step 94 and porting the fingerprint file to security device 15 in step 96. In the host monitoring process, security device 15 monitors write access signals in step 102, indicating an instruction to write to a disk.
25 When a write access signal is received, host system 12 provides information to security device 15 to use in tests to verify system integrity and virus free input. In step 104, host system 12 sends a write access interrupt to security device 15. In step 106, security device 15 reads
30 its interrupt status to determine whether the emergency interrupt flag is set. If an emergency interrupt is indicated, the keyboard is locked and host 12 is halted in step 108. In step 110 the write access is examined to determine whether it seeks to write to an existing
35 executable file; if so, in step 112 the keyboard is locked and host system 12 is halted. Otherwise, in step 114 the write access is examined to determine whether it seeks to

-7-

write a new executable file or batch. If so, in step 116 host 12 calls a scan routine from security device 15 and executes it. The scan routine examines the file to be written for viruses and the like. If the scan indicates the presence of a virus or the like, step 118 directs the process to step 120, where an emergency interrupt is sent to security device 15, and step 122, where the keyboard is locked and host system 12 is halted. If in step 118 no viruses or other problems have been detected, or if step 114 has determined that the write access is not for a new executable file, the file may be written to disk. In step 124, host system 12 updates the FAT on security device 15, creates new fingerprints in step 126, and updates the fingerprint data on security device 15 in step 128.

Host system 12 also waits for interrupt from security device 15 to run tests to verify the entire system is virus-free. This is done to determine at what time any system errors occur to help with data recovery in case that becomes necessary. Thus, in step 130 if a fingerprint interrupt is received from security device 15, host system 12 in step 132 ports its fingerprint data to security device 15. If host system 12 then receives a fingerprint emergency interrupt from security device 15 in step 134, it locks the keyboard and halts host system 12 in step 136; otherwise, control is passed to step 138 and host system 12 returns to its start status in 100. If in step 140 a scan interrupt is received from security device 15, host system 12 calls the scan routine from security device 15 and executes it in step 142. If no viruses or the like are detected in the scan, step 144 directs a return to the start status via step 138. If, however, the scan process results in the detection of a virus or the like, in step 146 an emergency interrupt is sent to security device 15 and host system 12 is halted and its keyboard locked in step 148.

Figure 4 is a flow diagram illustrating the operation of security device 15. Security device 15

-8-

initialization includes steps 150, 152, and 154, where boot sector, FAT, and fingerprints ("CRCs") of host 12 are copied and mapped into security device 15 memory.

The security device 15 monitoring process starts in step 156, and in step 158 security device 15 monitors the host until a write access interrupt is received from the host. In step 160, security device 15 compares the boot sector files stored in its memory with the boot sector files stored in host system 12. If these are not the same, step 162 directs the process to an emergency state in step 164. If the boot sector files are determined to be the same as in step 162, step 166 determines whether the write access seeks to write to a .EXE, .COM file, or the like. If so, security device 15 goes to an emergency state in step 168 and follows the emergency procedure shown in Figure 5. If not, step 170 determines whether the write access seeks to write a new executable file. If so, step 172 scans the file to be written to determine whether it contains any viruses. If the test is not passed, security device 15 is directed in step 174 to go to an emergency state in step 176. If the test in step 174 is passed, or if in step 170 the write access was not for a new executable file, the file may be written. In step 178, the FAT is updated on host system 12 and security device 15, and in step 180 fingerprint data for the file is created on the host 12 and updated fingerprint data is sent to security device 15.

In addition to monitoring initiated upon write accesses, security device 15 also automatically periodically monitors fingerprint data and scans for viruses. In step 182, if an internal clock in security device 15 indicates that it is time for a fingerprint check, in step 184 the fingerprint data stored in security device 15 is compared with the current fingerprint data of host system 12. If the fingerprint data is the same, step 194 directs the process to step 190 and a return to start

-9-

condition 154. If the fingerprint data is different, security device 15 goes to an emergency state in step 196.

In step 186, if security device 15 clock indicates that it is time to perform a scan, a scan is performed in step 188 to determine whether the contents of host system 12 contain any viruses or the like by comparison with identifying data stored in security device 15. If the scan detects no such code, step 192 directs the process to the start condition through step 190; if a virus is detected, security device 15 goes to an emergency state in step 198.

Figure 5 is a flow diagram illustrating emergency condition operation of security device 15, which starts at step 200. A security device 15 sends an emergency interrupt to host system 12 in step 202, instructing host system 12 in step 204 to close all files and return to the DOS prompt (or an equivalent prompt in a non-DOS system), and in step 206 to lock the keyboard and halt operation. This action isolates the virus in host system 12. Host system 12 remains locked until the proper unlock password is entered in step 208, in which event host system 12 will unlock the keyboard in step 210 and save the status to disk in step 212, display the status in step 214, and write the status to the printer in step 216.

In addition to sending an emergency interrupt to host system 12, in the preferred embodiment security device 15 is connected in a network, and in step 218 calls the monitoring host 26. In step 220, security device 15 identifies itself by sending a login code and the electronic identification number for the particular security device 15. If the electronic identification number is accepted, security device 15 will be permitted to login to the research database, a call is generated to the monitoring network system operator in step 222, and the status of the protected network 10 which generated the emergency condition is displayed in step 224. After the monitoring network system operator identifies and corrects

-10-

the problem which generated the emergency condition, the system operator resets security device 15 in step 222, and normal operation of protected network 10 resumes in step 228.

5 Figure 6 is a flow diagram illustrating the operation of network monitoring system 26. In step 250, network monitoring system 26 is responding to a connect attempt initiated by a security device 15, in step 218 of Figure 5. In step 252, network monitoring system 26
10 prompts security device 15 for its EIN and login code. In step 220, security device 15 sends its logon code and EIN to network monitoring system 26. In step 254, network monitoring system 26 checks the received EIN and logon code to verify that the communication was initiated by a
15 legitimate security device 15, not by a computer hacker. To ensure the security of the system, a preferred embodiment of the invention uses a 128 bit EIN. In a preferred embodiment, network monitoring host 26 waits only a limited amount of time, approximately 9 seconds, to
20 receive logon information after it detects a carrier signal on the telephone link. The combination of a 128 bit EIN and a limited time to respond effectively eliminates the possibility of a hacker logging onto the system. If network monitoring system 26 detects an
25 invalid EIN or a time-out condition, it disconnects the telephone hookup in step 256. If network monitoring system 26 detects a valid EIN in step 254, the logon procedure is executed in step 258.

 In step 260, network monitoring system 26
30 determines whether or not the call is an emergency call. If the call is an emergency call, the status of host system 12 is saved in research database 22 in step 272, displayed in step 274 and printed out in step 276, so that the virus can be isolated. In parallel with steps 272,
35 274 and 276, a call is generated by an emergency routine within host 12 to the system operator of network monitoring system 26 in step 278. In step 280, network

-11-

monitoring system 26 checks whether host system 12 has been reset. Once host system 12 has been reset, a logoff procedure is followed in step 282 and network monitoring system 26 disconnects telephone link 19 from security device 15 to research database 22 in step 284.

If, in step 260, network monitoring system 26 determines that the call is not an emergency call, it checks in step 262 whether the call is a report call. If the call is not a report call, the system operator is called in step 264 and network monitoring system 26 enters terminal mode. If the call is a report call, network monitoring system 26 creates a report file including the time and date in step 266, stores the report file in step 268 and prints the file in step 270. After the file is sent to the printer, network monitoring host 26 initiates a logoff procedure in step 282 and disconnects the telephone link from security device 15 in step 284.

Figure 7 is a block diagram of a preferred embodiment of security device 15. Security device 15 includes a processor 300 connected to a control bus 354, a data bus 356 and an address bus 358. Processor 300 communicates through buses 354, 356 and 358 to read only memory (ROM) 306, random access memory (RAM) 308, I/O interface 320, I/O interface 316 and liquid crystal display (LCD) driver 310. Processor 300 further includes an input from Clock 302 and from Reset Input 304. LCD driver 310 is coupled through LCD interface 312 to LCD 314. LCD 314 is used by security device 15 to display status information to a maintenance technician or an engineer, to aid in isolating a virus found in host system 12. Security device 15 includes ROM 306 and RAM 308 as storage devices. ROM 306 is utilized for storing the firmware which controls processor 300. RAM 308 is used by Processor 300 to store temporary information, including fingerprint files, boot sector information and file access table information. Security device 15 communicates with host 12 through I/O interface 320. I/O interface 320 is

-12-

preferably coupled to file transfer area 14 of host 12 via a parallel connection, although in another embodiment a serial port is utilized. Security device 15 is assigned a random address location in the memory space of host system 5 12, so that a Hacker cannot try to access RAM 308. Security device 15 is coupled to modem 318 through I/O interface 316. Security device 15 uses modem 318 to access the master database 20 of host of network monitoring system 26 through telephone line 19.

10 It is to be understood that the foregoing is merely illustrative of the principles of this invention, and that various modifications can be made by those skilled in the art without departing from the scope and spirit of the invention.

-13-

What Is Claimed Is:

1. A monitoring system for detecting software viruses in a host computer system having a host memory and a file storage memory, for storing an executable software file by means of a write access, said monitoring system comprising:
- means for generating an electronic fingerprint of the executable software file when the file is written to the file storage memory by means of a first write access;
- a security device coupled to the host computer, said security device comprising means for monitoring write accesses of the file storage memory, fingerprint memory means for storing a copy of the fingerprint and emergency interrupt response means for halting and locking the host computer in response to an emergency interrupt;
- means for transferring a first copy of the fingerprint to the fingerprint memory of the security device and for transferring a second copy of the fingerprint to the host memory; and
- comparison means for comparing the first copy of the fingerprint stored in fingerprint memory with the second copy of the fingerprint stored in the host memory when a subsequent write access of the executable software file occurs and for generating an emergency interrupt to the security device if the first and second copies are not identical.
2. The system of claim 1, wherein the host computer system includes the fingerprint generation means.

-14-

3. The system of claim 1, wherein the security device includes the fingerprint generation means.

4. The system of claim 2, wherein the electronic fingerprint is a checksum code.

5 5. The system of claim 2, wherein the electronic fingerprint is a cyclic redundancy check code.

6. The system of claim 1, wherein the host computer includes the comparison means.

7. The system of claim 1, wherein the security
10 device includes the comparison means.

8. A monitoring system for detecting software viruses in a protected network including a host computer coupled to a remote computer system, the host computer having a host memory and a host file storage memory for
15 storing a software file by means of a write access, the monitoring system comprising:

means for generating an electronic fingerprint of the software file when the file is written to the file storage memory;
20 a security device coupled to the host computer, said security device comprising means for monitoring write accesses of the file storage memory, fingerprint memory means for storing a copy of the fingerprint
25 and emergency interrupt response means for halting and locking the host computer in response to an emergency interrupt;

means for transferring a first copy of the fingerprint to the fingerprint memory of the security device and for transferring
30 a second copy of the fingerprint to the host memory; and

comparison means for comparing the first copy of the fingerprint stored in fingerprint memory with the second copy of
35 the fingerprint stored in the host memory when a subsequent write access of the

-15-

software file occurs and for generating an emergency interrupt to the security device if the first and second copies are not identical.

5 9. The system of claim 8, further comprising:
 means, in the security device, for storing a copy of the file allocation table of the host computer system; and

10 comparison means responsive to a write access of file storage memory, for comparing the copy of the file allocation table stored in the security device memory means with the file allocation table stored in the host system when a write access
15 occurs and for generating an emergency if the copy of the file access table is not identical to the file allocation table stored in the host system.

20 10. The system of claim 9, further comprising:
 means for copying the boot sector of the host computer system to the memory means in the security device; and

25 comparison means for comparing the copy of the boot sector stored in the security device memory with the boot sector stored in the host memory when a write access occurs and for generating an emergency interrupt if the copy of the boot sector is not identical to the boot sector
30 of the host system.

11. The system of claim 8, further comprising a network monitoring system coupled to the security device and to the host system, for monitoring a plurality of security devices.

35 12. The system of claim 11, wherein the network monitoring system further comprises a research database for compiling performance characteristics for each

-16-

security device and protected network, and an antidote software database for storing software to overcome known computer viruses.

13. The system of claim 12, wherein the
5 research database contains known computer virus code patterns.

14. The system of claim 13, further comprising means for scanning the host file storage memory to detect the known software viruses stored in the research
10 database.

15. A method for protecting a host computer system including a file storage memory from infection by a software virus, comprising the steps of:

generating an electronic fingerprint
15 for an executable software file being stored in the file storage memory;

storing a first copy of the electronic fingerprint in the host memory and storing a second copy of the electronic fingerprint
20 in an external security device;

monitoring write accesses to the file storage memory;

comparing the first and second copies of the fingerprint whenever the write access occurs; and
25

halting and locking the host computer when the first and second copies of the fingerprint differ, indicating the possible presence of a virus, so that the virus cannot damage the host computer system.
30

16. The method of claim 15, further comprising the steps of:

transferring a copy of the boot sector of the host computer system to the security device memory;
35

periodically comparing the copy of the boot sector with the host boot sector to

-17-

determine whether the host boot sector has changed; and

if the host boot selector has changed, halting and locking the host computer system to prevent possible damage by a virus.

5

17. The method of claim 16 further comprising the steps of:

10

transferring a copy of the file allocation table of the host computer system to the security device memory:

15

periodically comparing the file allocation table with the copy of the file allocation table stored in the security device memory to determine whether its fingerprint has changed; and

20

if the file allocation table has changed, halting and locking the host computer system to prevent possible damage by a virus.

18. The method of claim 17, further comprising the steps of:

25

storing known computer virus code patterns in a first database;

storing antidote software for said known computer virus code patterns in a second database;

30

periodically scanning the file storage memory of the host computer system for said known computer virus code patterns; and

halting the host computer system and executing said antidote software if a virus is detected.

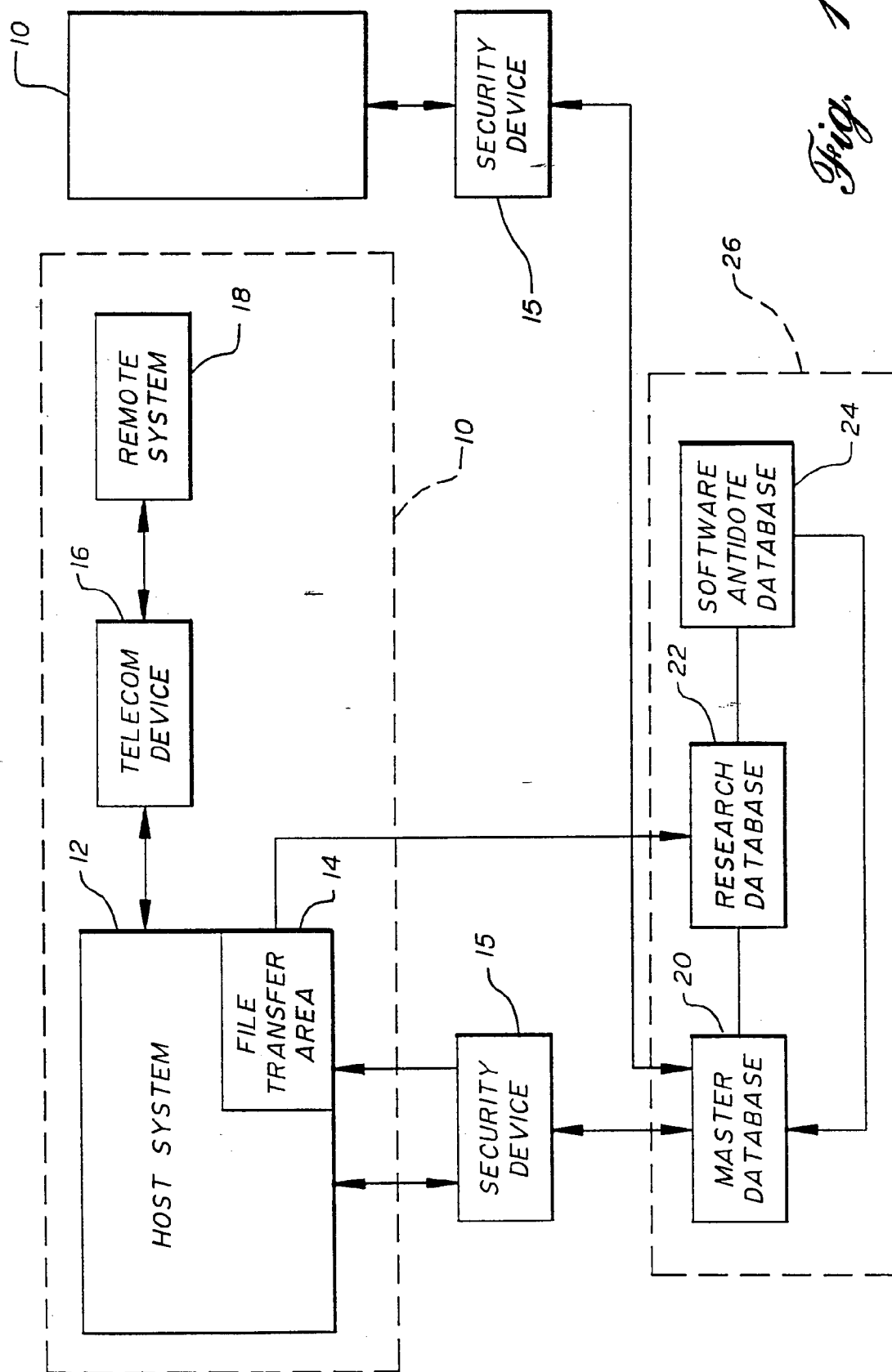
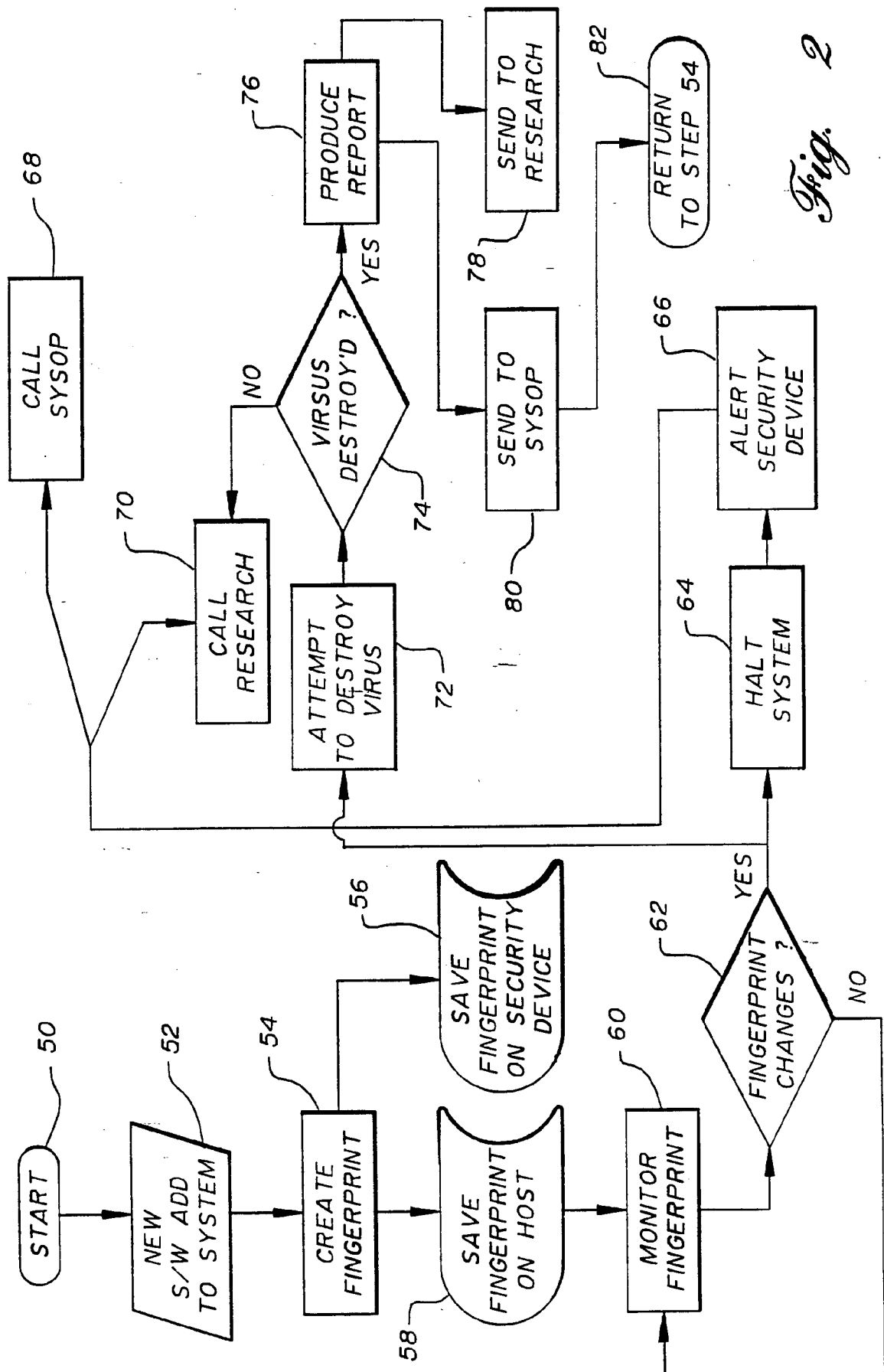


Fig. 1



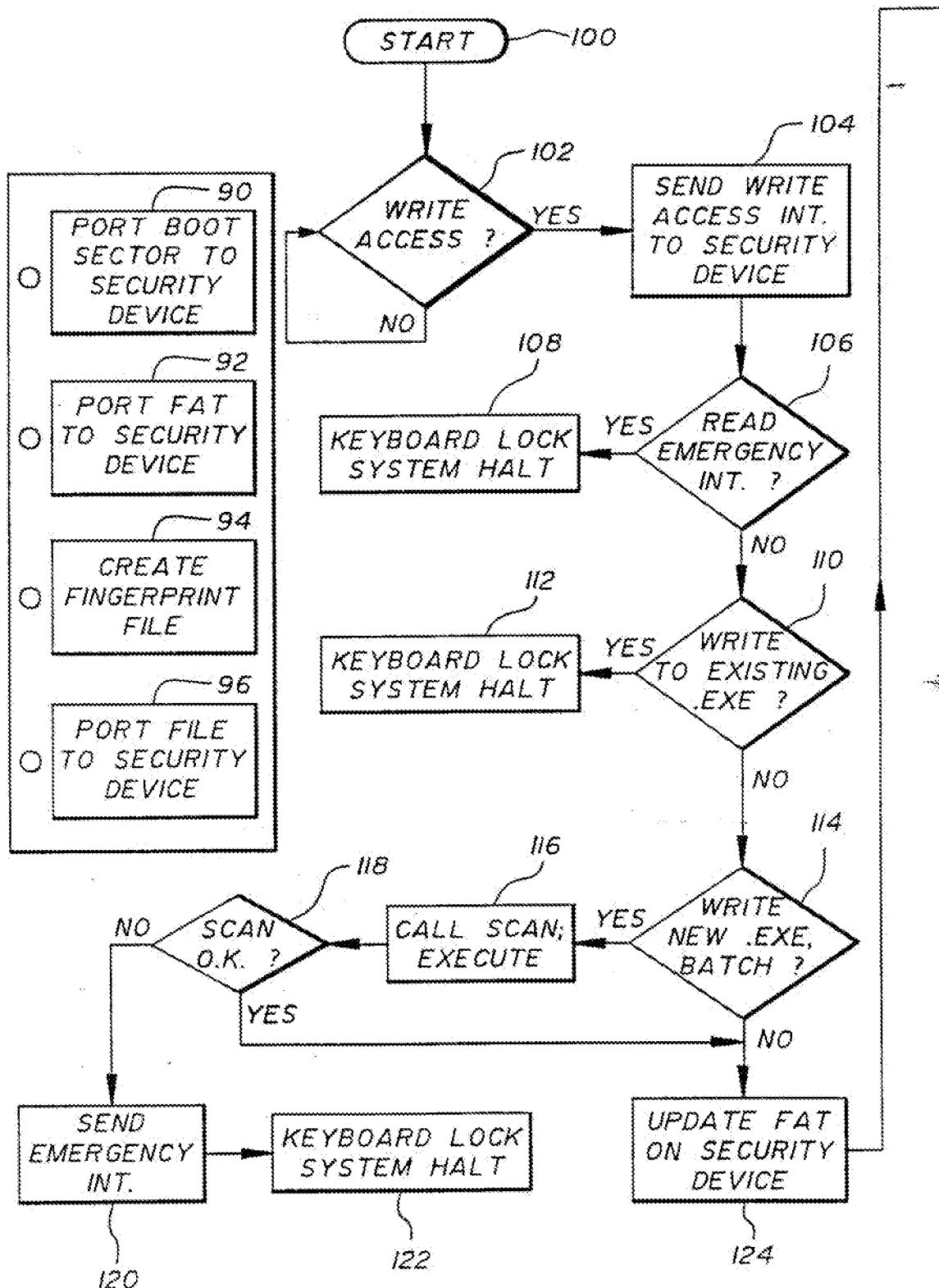
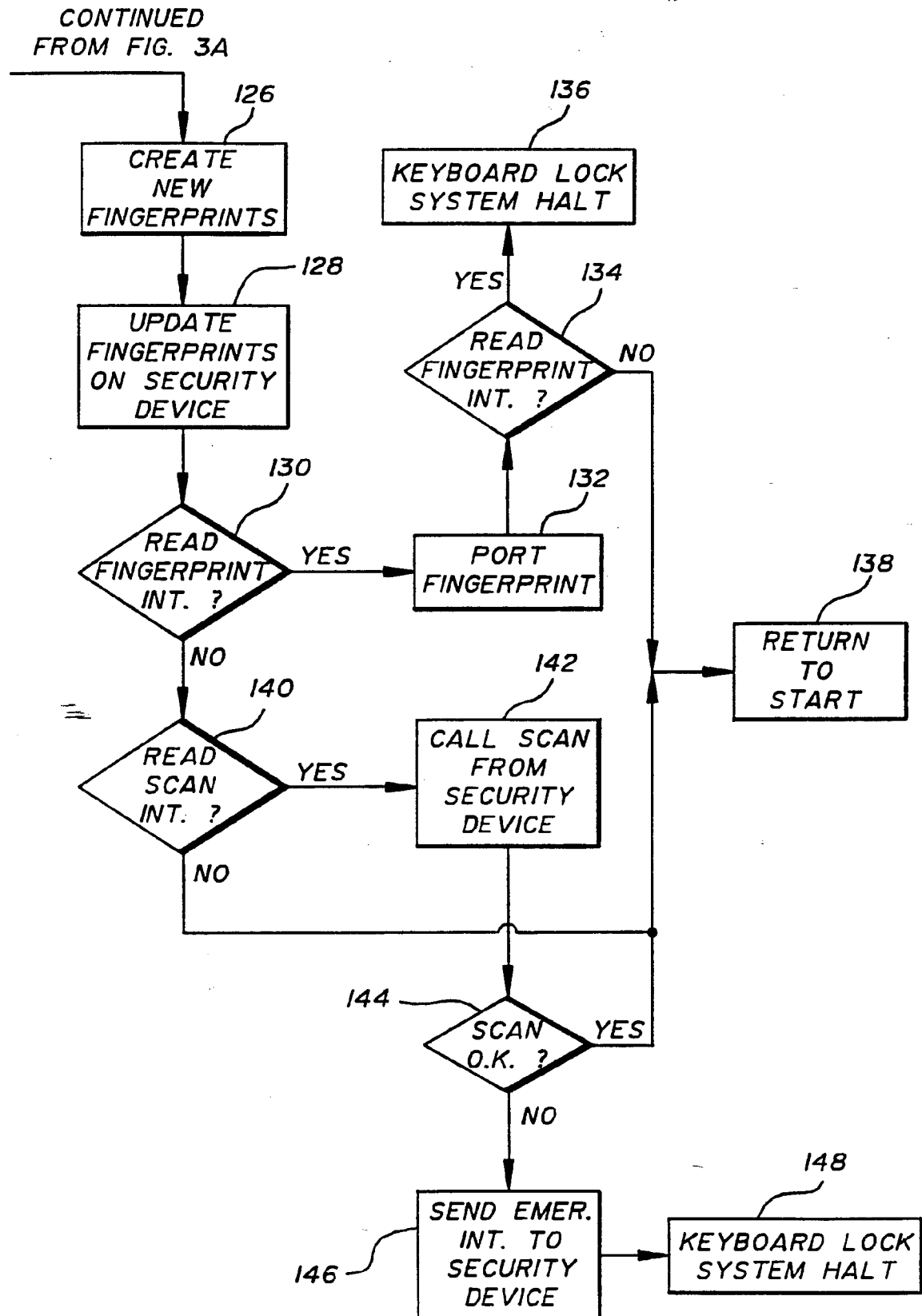
CONTINUED
ON FIG. 3B

Fig. 3A
SUBSTITUTE SHEET

*Fig. 3B*

SUBSTITUTE SHEET

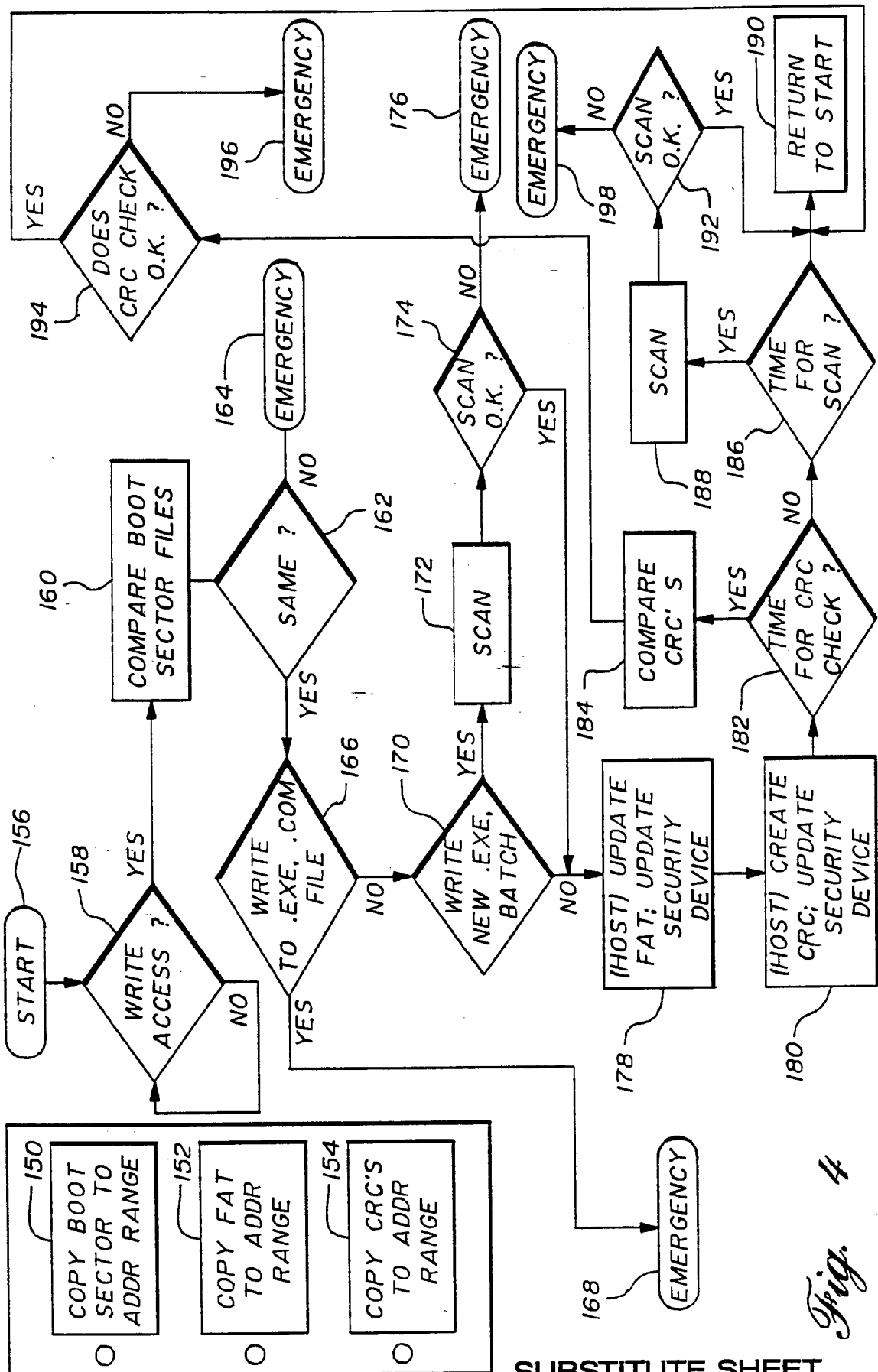


Fig. 4

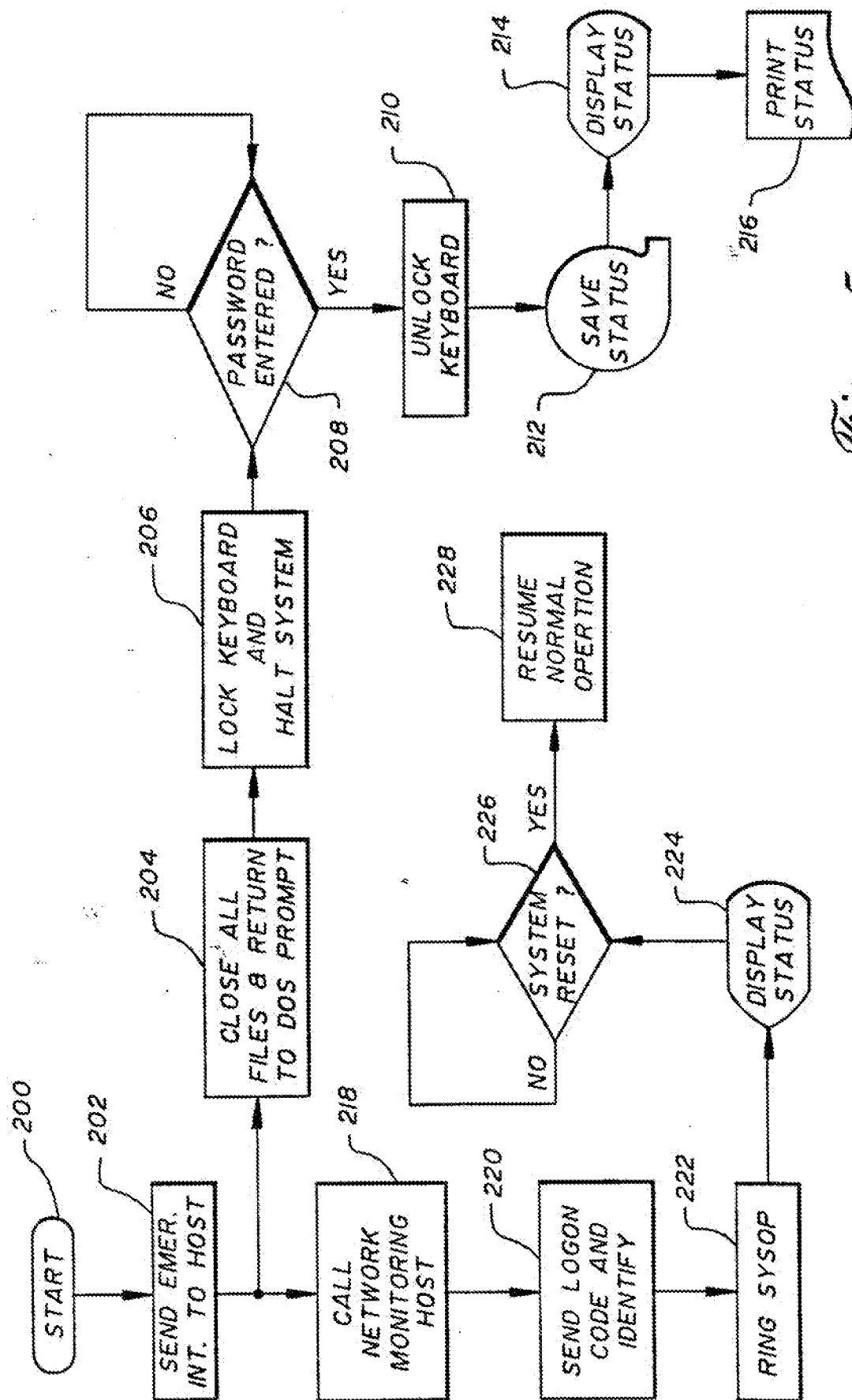
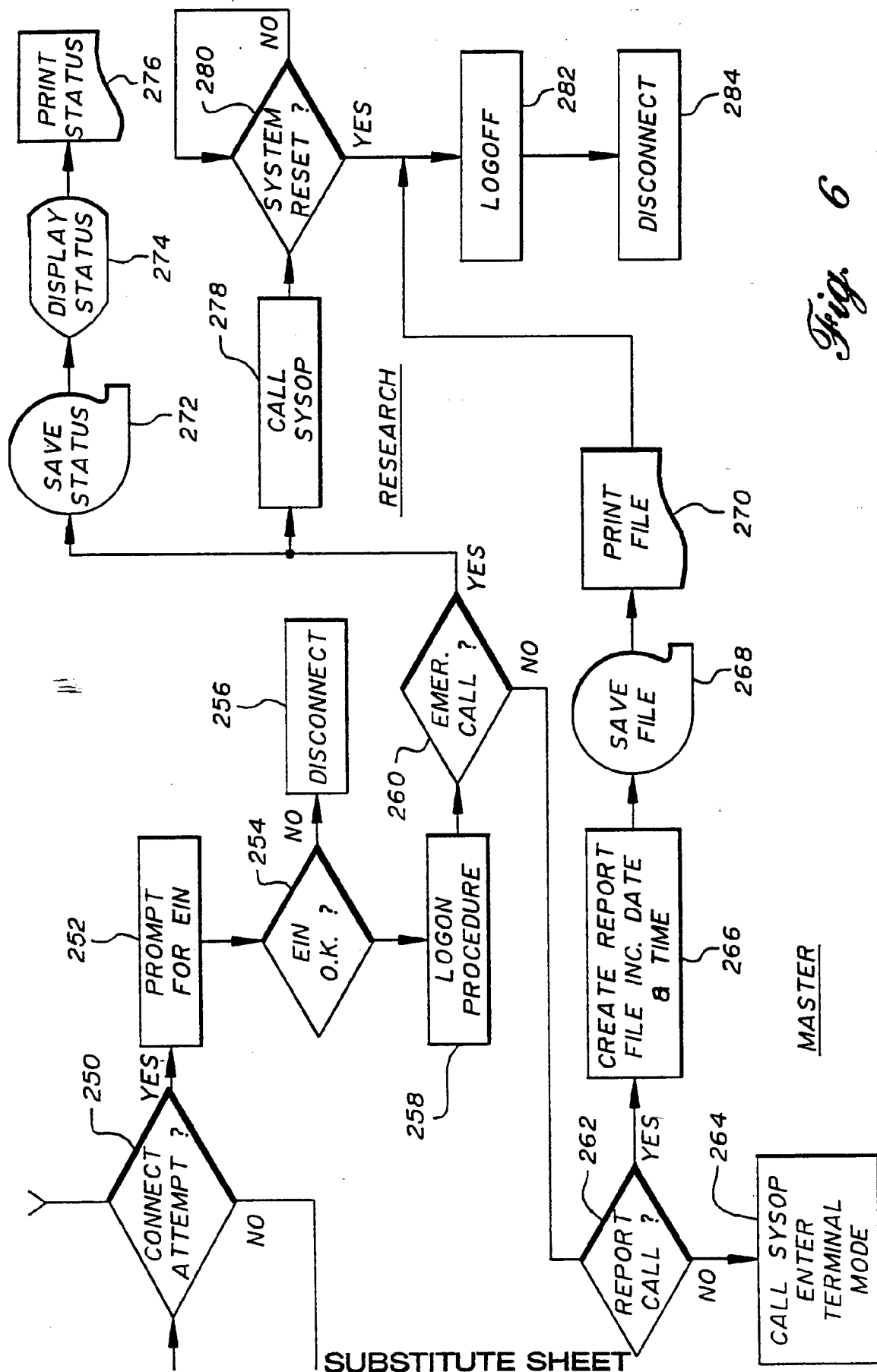
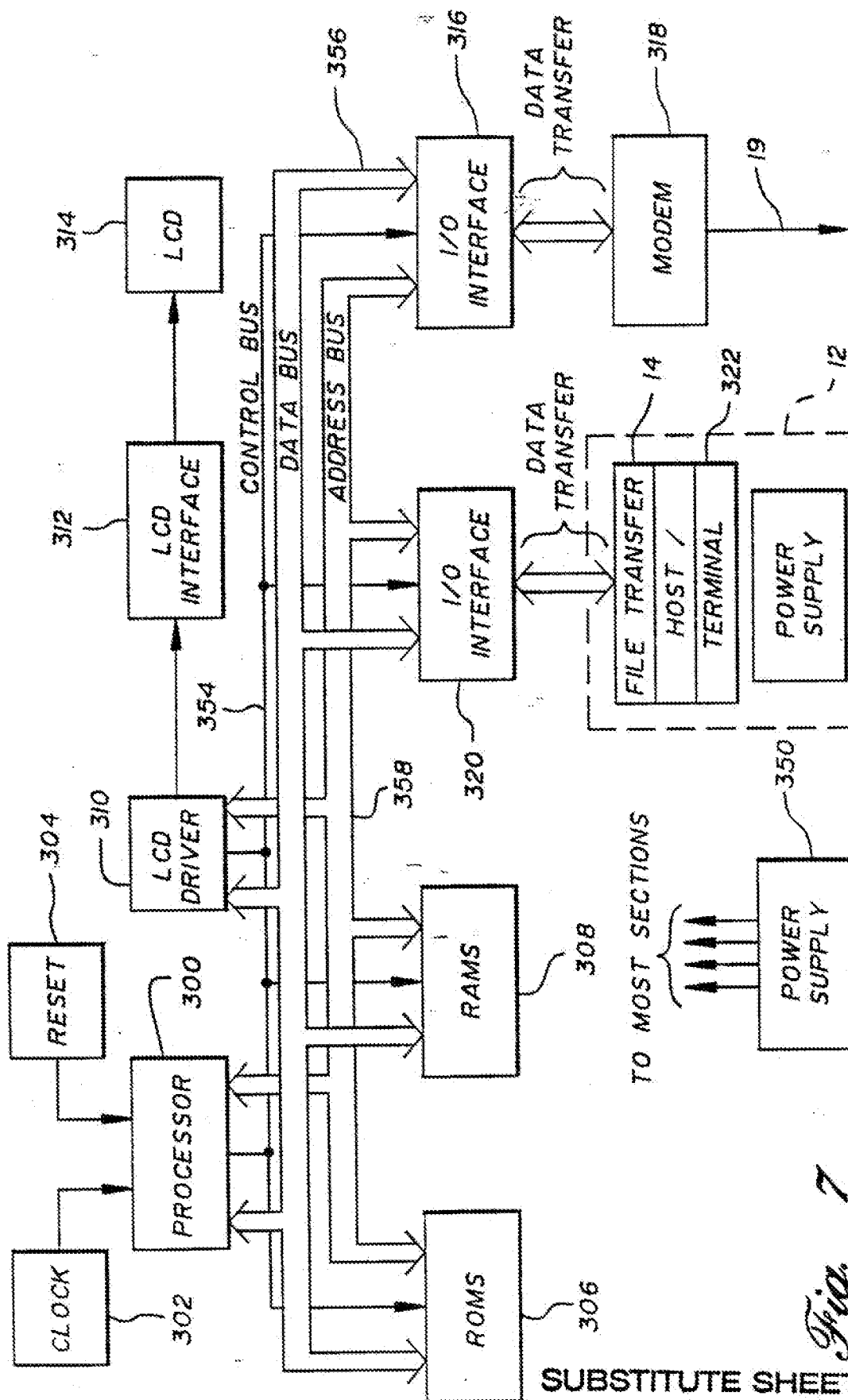


Fig. 5



8 / 8



SUBSTITUTE SHEET

Fig. 7

BLOCK DIAGRAM OF SECURITY DEVICE

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US93/05029

A. CLASSIFICATION OF SUBJECT MATTER

IPC(S) : H04L 9/00

US CL : 395/575

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. : 380/24,18,4,25; 371/14,19,67.1

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)

AUTOMATED PATENT SEARCH:

Search terms: Virus and (computer? or program? or microprocessor? or software)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US, A, 4,975,950 (Lentz) 04 December 1990, see Abstract; Figures 1 and 2; and col. 1, lines 1 - col. 4, lines 51.	1-18
P,Y	US, A, 5,121,345 (Lentz) 09 June 1992, see Abstract; Figures 1 and 2; and col. 1, lines 1 - col. 4, line 60.	1-18
P,A	US, A, 5,144,660 (Rose) 01 September 1992, see Abstract; col. 1- col. 2, line 17.	1-18
P,A	US, A, 5,163,088 (LoCascio) 10 November 1992, see Abstract; Figures 1 and 2; and col. 4, lines 39048.	1-18

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

* Special categories of cited documents:	* 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
* A document defining the general state of the art which is not considered to be part of particular relevance	* 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
* E earlier document published on or after the international filing date	* 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
* L document which may throw doubt on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	* G	document member of the same patent family
* 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		

Date of the actual completion of the international search

09 JULY 1993

Date of mailing of the international search report

21 SEP 1993

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Authorized officer

LY V. HUA

Facsimile No. NOT APPLICABLE

Telephone No. (703) 305-9684

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

Electronic Acknowledgement Receipt

EFS ID:	22598847
Application Number:	14732188
International Application Number:	
Confirmation Number:	1047
Title of Invention:	Malicious Mobile Code Runtime Monitoring System and Methods
First Named Inventor/Applicant Name:	Yigal Mordechai Edery
Customer Number:	115222
Filer:	Dawn-Marie Bey./Jeanne Paoletta-Bald
Filer Authorized By:	Dawn-Marie Bey.
Attorney Docket Number:	FIN0001-CON1-CIP1-C7-DIV1
Receipt Date:	11-JUN-2015
Filing Date:	
Time Stamp:	10:49:12
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Foreign Reference	bc_ref16.pdf	2725301 ee5003a8f0365daf576be35f53979d35bdea a941	no	28

Warnings:

Information:

2	Foreign Reference	bc_ref17.pdf	920792	no	23
			da9bbc9ab64d948a2bfa154cf2f01d7c6722e7dc		
Warnings:					
Information:					
3	Foreign Reference	finjan_ref9.pdf	4315235	no	41
			c1cf53563c86454844a1eed0e9abe9d1dad2d938		
Warnings:					
Information:					
4	Foreign Reference	bc_ref21.pdf	3040081	no	43
			840ffcedd0d5559e2ba2010a5924b01d3ba7d7fb		
Warnings:					
Information:					
5	Foreign Reference	bc_ref20.pdf	581027	no	15
			d83a5dad2462d9a078d400d4795202f5671dccb		
Warnings:					
Information:					
6	Foreign Reference	finjan_ref10.pdf	3291855	no	27
			042d1e4430fc6821824cd165b069eeae42f0fa5a		
Warnings:					
Information:					
7	Foreign Reference	pparweb_ref1.pdf	4323208	no	28
			c1af5c549364428e0c7538589d45a5aa21c70d35		
Warnings:					
Information:					
8	Foreign Reference	finjan_ref11.pdf	1116813	no	1
			a80ad3391dbb5cf9d565d6ce3a0f0f39c8f435e1		
Warnings:					
Information:					
9	Foreign Reference	finjan_ref12.pdf	3272767	no	27
			a8118162094be748a1ad66358a5676fda34528cb		
Warnings:					
Information:					
10	Foreign Reference	finjan_ref13.pdf	8929850	no	81
			3ca952026cf18ae84dc27b31dbe359c5ea8db2d2		
Warnings:					
Information:					

11	Foreign Reference	finjan_ref14.pdf	2565950	no	22
			46f8574ca20e9acd1da3692cadcea97f0158bb32		
Warnings:					
Information:					
12	Foreign Reference	soph_ref3.pdf	3585029	no	28
			502d8d76bf6892f24d119c9159e4478c67b9ecdc		
Warnings:					
Information:					
13	Non Patent Literature	finjan_ref35.pdf	23112336	no	179
			7ba60c4bc918692587445bd5cd609f609e0fc89e		
Warnings:					
Information:					
14	Non Patent Literature	finjan_ref36.pdf	4849019	no	19
			f4f62b047f1a49ca727a8f574c5d229c462626fc		
Warnings:					
Information:					
15	Non Patent Literature	finjan_ref37.pdf	1523373	no	12
			10baa7a33f6eef1b986c1b1528ab01ea79960e76		
Warnings:					
Information:					
Total Files Size (in bytes):			68152636		
This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503. <u>New Applications Under 35 U.S.C. 111</u> If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application. <u>National Stage of an International Application under 35 U.S.C. 371</u> If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course. <u>New International Application Filed with the USPTO as a Receiving Office</u> If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.					

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
22 November 2001 (22.11.2001)

PCT

(10) International Publication Number
WO 01/88673 A2

(51) International Patent Classification: **G06F 1/00**

(21) International Application Number: **PCT/IB01/01138**

(22) International Filing Date: **17 May 2001 (17.05.2001)**

(25) Filing Language: **English**

(26) Publication Language: **English**

(30) Priority Data:
60/205,591 17 May 2000 (17.05.2000) US

(71) Applicant: **FINJAN SOFTWARE, LTD.** [IL/IL]; Giborai
Israel Street, 42504 Netania (IL).

(72) Inventors: **EDERY, Yigal, Mordechai**, Hashikma 11,
POB 1115, 42815 Pardesia (IL); **VERED, Nimrod,**
Itzhak; Moshav Mismaret #81, 40695 Goosh Tel-Mond
(IL); **KROLL, David, R.**; 4856 Kingbrook Drive, San
Jose, CA 95124 (US).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU,
AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU,
CZ, DE, DK, DM, DZ, 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,
IT, LU, MC, NL, PT, SE, TR), 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.



WO 01/88673 A2

(54) Title: **MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS**

(57) Abstract: Protection systems and methods provide for protecting one or more personal computers ("PCs") and/or other intermittently or persistently network accessible devices or processes from undesirable or otherwise malicious operations of Java™ applets, ActiveX™ controls, JavaScript™ scripts, Visual Basic scripts, add-ins, downloaded/uploaded programs or other "Downloadables" or "mobile code" in whole or part. A protection engine embodiment provides, within a server, firewall or other suitable "re-communicator", for monitoring information received by the communicator, determining whether received information does or is likely to include executable code, and if so, causes mobile protection code (MPC) to be transferred to and rendered operable within a destination device of the received information, more suitably by forming a protection agent including the MPC, protection policies and a detected-Downloadable. An MPC embodiment further provides, within a Downloadable-destination, for initiating the Downloadable, enabling malicious Downloadable operation attempts to be received by the MPC, and causing (predetermined) corresponding operations to be executed in response to the attempts, more suitably in conjunction with protection policies.

MALICIOUS MOBILE CODE RUNTIME MONITORING
SYSTEM AND METHODS

BACKGROUND OF THE INVENTION

5

Field of the Invention

This invention relates generally to computer networks, and more particularly provides a system and methods for protecting network-connectable devices from undesirable downloadable operation.

10

Description of the Background Art

Advances in networking technology continue to impact an increasing number and diversity of users. The Internet, for example, already provides to expert, intermediate and even novice users the informational, product and service resources of over 100,000 interconnected networks owned by governments, universities, nonprofit groups, companies, etc. Unfortunately, particularly the Internet and other public networks have also become a major source of potentially system-fatal or otherwise damaging computer code commonly referred to as "viruses."

20

Efforts to forestall viruses from attacking networked computers have thus far met with only limited success at best. Typically, a virus protection program designed to identify and remove or protect against the initiating of known viruses is installed on a network firewall or individually networked computer. The

program is then inevitably surmounted by some new virus that often causes damage to one or more computers. The damage is then assessed and, if isolated, the new virus is analyzed. A corresponding new virus protection program (or update thereof) is then developed and installed to combat the new virus, and the
5 new program operates successfully until yet another new virus appears - and so on. Of course, damage has already typically been incurred.

To make matters worse, certain classes of viruses are not well recognized or understood, let alone protected against. It is observed by this inventor, for example, that Downloadable information comprising program code can include
10 distributable components (e.g. Java™ applets and JavaScript scripts, ActiveX™ controls, Visual Basic, add-ins and/or others). It can also include, for example, application programs, Trojan horses, multiple compressed programs such as zip or meta files, among others. U.S. Patent 5,983,348 to Shuang, however, teaches a protection system for protecting against only distributable components including
15 "Java applets or ActiveX controls", and further does so using resource intensive and high bandwidth static Downloadable content and operational analysis, and modification of the Downloadable component; Shuang further fails to detect or protect against additional program code included within a tested Downloadable. U.S. Patent 5,974,549 to Golan teaches a protection system that further focuses
20 only on protecting against ActiveX controls and not other distributable components, let alone other Downloadable types. U.S. patent 6,167,520 to Touboul enables more accurate protection than Shuang or Golan, but lacks the greater flexibility and efficiency taught herein, as do Shuang and Golan.

Accordingly, there remains a need for efficient, accurate and flexible protection of computers and other network connectable devices from malicious Downloadables.

5

SUMMARY OF THE INVENTION

The present invention provides protection systems and methods capable of protecting a personal computer ("PC") or other persistently or even intermittently network accessible devices or processes from harmful, undesirable, suspicious or other "malicious" operations that might otherwise be effectuated by remotely operable code. While enabling the capabilities of prior systems, the present invention is not nearly so limited, resource intensive or inflexible, and yet enables more reliable protection. For example, remotely operable code that is protectable against can include downloadable application programs, Trojan horses and program code groupings, as well as software "components", such as Java™ applets, ActiveX™ controls, JavaScript™/Visual Basic scripts, add-ins, etc., among others. Protection can also be provided in a distributed interactively, automatically or mixed configurable manner using protected client, server or other parameters, redirection, local/remote logging, etc., and other server/client based protection measures can also be separately and/or interoperably utilized, among other examples.

In one aspect, embodiments of the invention provide for determining, within one or more network "servers" (e.g. firewalls, resources, gateways, email relays or other devices/processes that are capable of receiving-and-transferring a

Downloadable) whether received information includes executable code (and is a "Downloadable"). Embodiments also provide for delivering static, configurable and/or extensible remotely operable protection policies to a Downloadable-destination, more typically as a sandboxed package including the mobile protection code, downloadable policies and one or more received Downloadables. Further client-based or remote protection code/policies can also be utilized in a distributed manner. Embodiments also provide for causing the mobile protection code to be executed within a Downloadable-destination in a manner that enables various Downloadable operations to be detected, intercepted or further responded to via protection operations. Additional server/information-destination device security or other protection is also enabled, among still further aspects.

A protection engine according to an embodiment of the invention is operable within one or more network servers, firewalls or other network connectable information re-communicating devices (as are referred to herein summarily one or more "servers" or "re-communicators"). The protection engine includes an information monitor for monitoring information received by the server, and a code detection engine for determining whether the received information includes executable code. The protection engine also includes a packaging engine for causing a sandboxed package, typically including mobile protection code and downloadable protection policies to be sent to a Downloadable-destination in conjunction with the received information, if the received information is determined to be a Downloadable.

A sandboxed package according to an embodiment of the invention is receivable by and operable with a remote Downloadable-destination. The sandboxed package includes mobile protection code ("MPC") for causing one or more predetermined malicious operations or operation combinations of a

5 Downloadable to be monitored or otherwise intercepted. The sandboxed package also includes protection policies (operable alone or in conjunction with further Downloadable-destination stored or received policies/MPCs) for causing one or more predetermined operations to be performed if one or more undesirable operations of the Downloadable is/are intercepted. The sandboxed package can

10 also include a corresponding Downloadable and can provide for initiating the Downloadable in a protective "sandbox". The MPC/policies can further include a communicator for enabling further MPC/policy information or "modules" to be utilized and/or for event logging or other purposes.

A sandbox protection system according to an embodiment of the invention

15 comprises an installer for enabling a received MPC to be executed within a Downloadable-destination (device/process) and further causing a Downloadable application program, distributable component or other received downloadable code to be received and installed within the Downloadable-destination. The protection system also includes a diverter for monitoring one or more operation

20 attempts of the Downloadable, an operation analyzer for determining one or more responses to the attempts, and a security enforcer for effectuating responses to the monitored operations. The protection system can further include one or more security policies according to which one or more protection system elements are

operable automatically (e.g. programmatically) or in conjunction with user intervention (e.g. as enabled by the security enforcer). The security policies can also be configurable/extensible in accordance with further downloadable and/or Downloadable-destination information.

5 A method according to an embodiment of the invention includes receiving downloadable information, determining whether the downloadable information includes executable code, and causing a mobile protection code and security policies to be communicated to a network client in conjunction with security policies and the downloadable information if the downloadable information is
10 determined to include executable code. The determining can further provide multiple tests for detecting, alone or together, whether the downloadable information includes executable code.

 A further method according to an embodiment of the invention includes forming a sandboxed package that includes mobile protection code ("MPC"),
15 protection policies, and a received, detected-Downloadable, and causing the sandboxed package to be communicated to and installed by a receiving device or process ("user device") for responding to one or more malicious operation attempts by the detected-Downloadable from within the user device. The MPC/policies can further include a base "module" and a "communicator" for
20 enabling further up/downloading of one or more further "modules" or other information (e.g. events, user/user device information, etc.).

 Another method according to an embodiment of the invention includes installing, within a user device, received mobile protection code ("MPC") and

protection policies in conjunction with the user device receiving a downloadable application program, component or other Downloadable(s). The method also includes determining, by the MPC, a resource access attempt by the Downloadable, and initiating, by the MPC, one or more predetermined operations
5 corresponding to the attempt. (Predetermined operations can, for example, comprise initiating user, administrator, client, network or protection system determinable operations, including but not limited to modifying the Downloadable operation, extricating the Downloadable, notifying a user/another, maintaining a local/remote log, causing one or more MPCs/policies to be downloaded, etc.)

10 Advantageously, systems and methods according to embodiments of the invention enable potentially damaging, undesirable or otherwise malicious operations by even unknown mobile code to be detected, prevented, modified and/or otherwise protected against without modifying the mobile code. Such protection is further enabled in a manner that is capable of minimizing server and
15 client resource requirements, does not require pre-installation of security code within a Downloadable-destination, and provides for client specific or generic and readily updateable security measures to be flexibly and efficiently implemented. Embodiments further provide for thwarting efforts to bypass security measures (e.g. by "hiding" undesirable operation causing information within apparently inert
20 or otherwise "friendly" downloadable information) and/or dividing or combining security measures for even greater flexibility and/or efficiency.

Embodiments also provide for determining protection policies that can be downloaded and/or ascertained from other security information (e.g. browser

settings, administrative policies, user input, uploaded information, etc.). Different actions in response to different Downloadable operations, clients, users and/or other criteria are also enabled, and embodiments provide for implementing other security measures, such as verifying a downloadable source, certification, authentication, etc. Appropriate action can also be accomplished automatically (e.g. programmatically) and/or in conjunction with alerting one or more users/administrators, utilizing user input, etc. Embodiments further enable desirable Downloadable operations to remain substantially unaffected, among other aspects.

10

15

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1a is a block diagram illustrating a network system in accordance with an embodiment of the present invention;

FIG. 1b is a block diagram illustrating a network subsystem example in accordance with an embodiment of the invention;

FIG. 1c is a block diagram illustrating a further network subsystem example in accordance with an embodiment of the invention;

FIG. 2 is a block diagram illustrating a computer system in accordance with an embodiment of the invention;

FIG. 3 is a flow diagram broadly illustrating a protection system host according to an embodiment of the invention;

FIG. 4 is a block diagram illustrating a protection engine according to an embodiment of the invention;

FIG. 5 is a block diagram illustrating a content inspection engine according to an embodiment of the invention;

FIG. 6a is a block diagram illustrating protection engine parameters according to an embodiment of the invention;

FIG. 6b is a flow diagram illustrating a linking engine use in conjunction with ordinary, compressed and distributable sandbox package utilization, according to an embodiment of the invention;

FIG. 7a is a flow diagram illustrating a sandbox protection system operating within a destination system, according to an embodiment of the invention;

FIG. 7b is a block diagram illustrating memory allocation usable in conjunction with the protection system of FIG. 7a, according to an embodiment of the invention;

FIG. 7c is a block diagram illustrating a mobile protection code according to an embodiment of the invention;

FIG. 8 is a flowchart illustrating a method for examining a Downloadable in accordance with the present invention;

FIG. 9 is a flowchart illustrating a server based protection method according to an embodiment of the invention;

FIG. 10a is a flowchart illustrating method for determining if a potential-Downloadable includes or is likely to include executable code, according to an embodiment of the invention;

FIG. 10b is a flowchart illustrating a method for forming a protection agent, according to an embodiment of the invention;

FIG. 11 is a flowchart illustrating a method for protecting a Downloadable destination according to an embodiment of the invention;

FIG. 12a is a flowchart illustrating a method for forming a Downloadable access interceptor according to an embodiment of the invention; and

FIG. 12b is a flowchart illustrating a method for implementing mobile protection policies according to an embodiment of the invention.

DETAILED DESCRIPTION

In providing malicious mobile code runtime monitoring systems and methods, embodiments of the invention enable actually or potentially undesirable operations of even unknown malicious code to be efficiently and flexibly avoided. Embodiments provide, within one or more "servers" (e.g. firewalls, resources, gateways, email relays or other information re-communicating devices), for receiving downloadable-information and detecting whether the downloadable-information includes one or more instances of executable code (e.g. as with a Trojan horse, zip/meta file etc.). Embodiments also provide for separately or interoperably conducting additional security measures within the server, within a Downloadable-destination of a detected-Downloadable, or both.

Embodiments further provide for causing mobile protection code ("MPC") and downloadable protection policies to be communicated to, installed and executed within one or more received information destinations in conjunction with a detected-Downloadable. Embodiments also provide, within an information-destination, for detecting malicious operations of the detected-Downloadable and causing responses thereto in accordance with the protection policies (which can correspond to one or more user, Downloadable, source, destination, or other parameters), or further downloaded or downloadable-destination based policies (which can also be configurable or extensible). (Note that the term "or", as used herein, is generally intended to mean "and/or" unless otherwise indicated.)

FIGS. 1a through 1c illustrate a computer network system 100 according to an embodiment of the invention. FIG. 1a broadly illustrates system 100, while FIGS. 1b and 1c illustrate exemplary protectable subsystem implementations corresponding with system 104 or 106 of FIG. 1a.

5 Beginning with FIG. 1a, computer network system 100 includes an external computer network 101, such as a Wide Area Network or "WAN" (e.g. the Internet), which is coupled to one or more network resource servers (summarily depicted as resource server-1 102 and resource server-N 103). Where external network 101 includes the Internet, resource servers 1-N (102, 103) might
10 provide one or more resources including web pages, streaming media, transaction-facilitating information, program updates or other downloadable information, summarily depicted as resources 121, 131 and 132. Such information can also include more traditionally viewed "Downloadables" or "mobile code" (i.e. distributable components), as well as downloadable application programs or other
15 further Downloadables, such as those that are discussed herein. (It will be appreciated that interconnected networks can also provide various other resources as well.)

Also coupled via external network 101 are subsystems 104-106. Subsystems 104-106 can, for example, include one or more servers, personal
20 computers ("PCs"), smart appliances, personal information managers or other devices/processes that are at least temporarily or otherwise intermittently directly or indirectly connectable in a wired or wireless manner to external network 101 (e.g. using a dialup, DSL, cable modem, cellular connection, IR/RF, or various

other suitable current or future connection alternatives). One or more of
subsystems 104-106 might further operate as user devices that are connectable to
external network 101 via an internet service provider ("ISP") or local area network
("LAN"), such as a corporate intranet, or home, portable device or smart appliance
5 network, among other examples.

FIG. 1a also broadly illustrates how embodiments of the invention are
capable of selectively, modifiably or extensibly providing protection to one or
more determinable ones of networked subsystems 104-106 or elements thereof
(not shown) against potentially harmful or other undesirable ("malicious") effects
10 in conjunction with receiving downloadable information. "Protected" subsystem
104, for example, utilizes a protection in accordance with the teachings herein,
while "unprotected" subsystem-N 105 employs no protection, and protected
subsystem-M 106 might employ one or more protections including those
according to the teachings herein, other protection, or some combination.

15 System 100 implementations are also capable of providing protection to
redundant elements 107 of one or more of subsystems 104-106 that might be
utilized, such as backups, failsafe elements, redundant networks, etc. Where
included, such redundant elements are also similarly protectable in a separate,
combined or coordinated manner using embodiments of the present invention
20 either alone or in conjunction with other protection mechanisms. In such cases,
protection can be similarly provided singly, as a composite of component
operations or in a backup fashion. Care should, however, be exercised to avoid
potential repeated protection engine execution corresponding to a single

Downloadable; such “chaining” can cause a Downloadable to operate incorrectly or not at all, unless a subsequent detection engine is configured to recognize a prior packaging of the Downloadable..

FIGS. 1b and 1c further illustrate, by way of example, how protection
5 systems according to embodiments of the invention can be utilized in conjunction with a wide variety of different system implementations. In the illustrated examples, system elements are generally configurable in a manner commonly referred to as a “client-server” configuration, as is typically utilized for accessing Internet and many other network resources. For clarity sake, a simple client-server
10 configuration will be presumed unless otherwise indicated. It will be appreciated, however, that other configurations of interconnected elements might also be utilized (e.g. peer-peer, routers, proxy servers, networks, converters, gateways, services, network reconfiguring elements, etc.) in accordance with a particular application.

15 The FIG. 1b example shows how a suitable protected system 104a (which can correspond to subsystem-1 104 or subsystem-M 106 of FIG. 1) can include a protection-initiating host “server” or “re-communicator” (e.g. ISP server 140a), one or more user devices or “Downloadable-destinations” 145, and zero or more redundant elements (which elements are summarily depicted as redundant client
20 device/process 145a). In this example, ISP server 140a includes one or more email, Internet or other servers 141a, or other devices or processes capable of transferring or otherwise “re-communicating” downloadable information to user devices 145. Server 141a further includes protection engine or “PE” 142a, which

is capable of supplying mobile protection code ("MPC") and protection policies for execution by client devices 145. One or more of user devices 145 can further include a respective one or more clients 146 for utilizing information received via server 140a, in accordance with which MPC and protection policies are operable
5 to protect user devices 145 from detrimental, undesirable or otherwise "malicious" operations of downloadable information also received by user device 145.

The FIG. 1c example shows how a further suitable protected system 104b can include, in addition to a "re-communicator", such as server 142b, a firewall 143c (e.g. as is typically the case with a corporate intranet and many existing or
10 proposed home/smart networks.) In such cases, a server 141b or firewall 143 can operate as a suitable protection engine host. A protection engine can also be implemented in a more distributed manner among two or more protection engine host systems or host system elements, such as both of server 141b and firewall 143, or in a more integrated manner, for example, as a standalone device.
15 Redundant system or system protection elements can also be similarly provided in a more distributed or integrated manner (see above).

System 104b also includes internal network 144 and user devices 145. User devices 145 further include a respective one or more clients 146 for utilizing information received via server 140a, in accordance with which the MPCs or
20 protection policies are operable. (As in the previous example, one or more of user devices 145 can also include or correspond with similarly protectable redundant system elements, which are not shown.)

It will be appreciated that the configurations of FIGS 1a-1c are merely exemplary. Alternative embodiments might, for example, utilize other suitable connections, devices or processes. One or more devices can also be configurable to operate as a network server, firewall, smart router, a resource server servicing deliverable third-party/manufacturer postings, a user device operating as a firewall/server, or other information-suppliers or intermediaries (i.e. as a "re-communicator" or "server") for servicing one or more further interconnected devices or processes or interconnected levels of devices or processes. Thus, for example, a suitable protection engine host can include one or more devices or processes capable of providing or supporting the providing of mobile protection code or other protection consistent with the teachings herein. A suitable information-destination or "user device" can further include one or more devices or processes (such as email, browser or other clients) that are capable of receiving and initiating or otherwise hosting a mobile code execution.

FIG. 2 illustrates an exemplary computing system 200, that can comprise one or more of the elements of FIGS. 1a through 1c. While other application-specific alternatives might be utilized, it will be presumed for clarity sake that system 100 elements (FIGS. 1a-c) are implemented in hardware, software or some combination by one or more processing systems consistent therewith, unless otherwise indicated.

Computer system 200 comprises elements coupled via communication channels (e.g. bus 201) including one or more general or special purpose processors 202, such as a Pentium® or Power PC®, digital signal processor

("DSP"), etc. System 200 elements also include one or more input devices 203 (such as a mouse, keyboard, microphone, pen, etc.), and one or more output devices 204, such as a suitable display, speakers, actuators, etc., in accordance with a particular application.

5 System 200 also includes a computer readable storage media reader 205 coupled to a computer readable storage medium 206, such as a storage/memory device or hard or removable storage/memory media; such devices or media are further indicated separately as storage device 208 and memory 209, which can include hard disk variants, floppy/compact disk variants, digital versatile disk
10 ("DVD") variants, smart cards, read only memory, random access memory, cache memory, etc., in accordance with a particular application. One or more suitable communication devices 207 can also be included, such as a modem, DSL, infrared or other suitable transceiver, etc. for providing inter-device communication directly or via one or more suitable private or public networks that can include but
15 are not limited to those already discussed.

Working memory further includes operating system ("OS") elements and other programs, such as application programs, mobile code, data, etc. for implementing system 100 elements that might be stored or loaded therein during use. The particular OS can vary in accordance with a particular device, features or
20 other aspects in accordance with a particular application (e.g. Windows, Mac, Linux, Unix or Palm OS variants, a proprietary OS, etc.). Various programming languages or other tools can also be utilized, such as C++, Java, Visual Basic, etc. As will be discussed, embodiments can also include

a network client such as a browser or email client, e.g. as produced by Netscape, Microsoft or others, a mobile code executor such as an OS task manager, Java Virtual Machine ("JVM"), etc., and an application program interface ("API"), such as a Microsoft Windows or other suitable element in accordance with the teachings herein. (It will also become apparent that embodiments might also be implemented in conjunction with a resident application or combination of mobile code and resident application components.)

One or more system 200 elements can also be implemented in hardware, software or a suitable combination. When implemented in software (e.g. as an application program, object, downloadable, servlet, etc. in whole or part), a system 200 element can be communicated transitionally or more persistently from local or remote storage to memory (or cache memory, etc.) for execution, or another suitable mechanism can be utilized, and elements can be implemented in compiled or interpretive form. Input, intermediate or resulting data or functional elements can further reside more transitionally or more persistently in a storage media, cache or more persistent volatile or non-volatile memory, (e.g. storage device 207 or memory 208) in accordance with a particular application.

FIG. 3 illustrates an interconnected re-communicator 300 generally consistent with system 140b of FIG. 1, according to an embodiment of the invention. As with system 140b, system 300 includes a server 301, and can also include a firewall 302. In this implementation, however, either server 301 or firewall 302 (if a firewall is used) can further include a protection engine (310 or 320 respectively). Thus, for example, an included firewall can process received

information in a conventional manner, the results of which can be further processed by protection engine 310 of server 301, or information processed by protection engine 320 of an included firewall 302 can be processed in a conventional manner by server 301. (For clarity sake, a server including a singular protection engine will be presumed, with or without a firewall, for the remainder of the discussion unless otherwise indicated. Note, however, that other embodiments consistent with the teachings herein might also be utilized.)

FIG. 3 also shows how information received by server 301 (or firewall 302) can include non-executable information, executable information or a combination of non-executable and one or more executable code portions (e.g. so-called Trojan horses that include a hostile Downloadable within a friendly one, combined, compressed or otherwise encoded files, etc.). Particularly such combinations will likely remain undetected by a firewall or other more conventional protection systems. Thus, for convenience, received information will also be referred to as a "potential-Downloadable", and received information found to include executable code will be referred to as a "Downloadable" or equivalently as a "detected-Downloadable" (regardless of whether the executable code includes one or more application programs, distributable "components" such as Java, ActiveX, add-in, etc.).

Protection engine 310 provides for detecting whether received potential-Downloadables include executable code, and upon such detection, for causing mobile protection code ("MPC") to be transferred to a device that is a destination of the potential-Downloadable (or "Downloadable-destination"). Protection

engine 310 can also provide protection policies in conjunction with the MPC (or thereafter as well), which MPC/policies can be automatically (e.g. programmatically) or interactively configurable in accordance user, administrator, downloadable source, destination, operation, type or various other parameters
5 alone or in combination (see below). Protection engine 310 can also provide or operate separately or interoperably in conjunction with one or more of certification, authentication, downloadable tagging, source checking, verification, logging, diverting or other protection services via the MPC, policies, other local/remote server or destination processing, etc. (e.g. which can also include
10 protection mechanisms taught by the above-noted prior applications; see FIG. 4).

Operationally, protection engine 310 of server 301 monitors information received by server 301 and determines whether the received information is deliverable to a protected destination, e.g. using a suitable monitor/data transfer mechanism and comparing a destination-address of the received information to a
15 protected destination set, such as a protected destinations list, array, database, etc. (All deliverable information or one or more subsets thereof might also be monitored.) Protection engine 310 further analyzes the potential-Downloadable and determines whether the potential-Downloadable includes executable code. If not, protection engine 310 enables the not executable potential-Downloadable 331
20 to be delivered to its destination in an unaffected manner.

In conjunction with determining that the potential-Downloadable is a detected-Downloadable, protection engine 310 also causes mobile protection code or "MPC" 341 to be communicated to the Downloadable-destination of the

Downloadable, more suitably in conjunction with the detected-Downloadable 343 (see below). Protection engine 310 further causes downloadable protection policies 342 to be delivered to the Downloadable-destination, again more suitably in conjunction with the detected-Downloadable. Protection policies 342 provide
5 parameters (or can additionally or alternatively provide additional mobile code) according to which the MPC is capable of determining or providing applicable protection to a Downloadable-destination against malicious Downloadable operations.

(One or more "checked", tag, source, destination, type, detection or other
10 security result indicators, which are not shown, can also be provided as corresponding to determined non-Downloadables or Downloadables, e.g. for testing, logging, further processing, further identification tagging or other purposes in accordance with a particular application.)

Further MPCs, protection policies or other information are also deliverable
15 to a the same or another destination, for example, in accordance with communication by an MPC/protection policies already delivered to a downloadable-destination. Initial or subsequent MPCs/policies can further be selected or configured in accordance with a Downloadable-destination indicated by the detected-Downloadable, destination-user or administrative information, or
20 other information providable to protection engine 310 by a user, administrator, user system, user system examination by a communicated MPC, etc. (Thus, for example, an initial MPC/policies can also be initially provided that are operable

with or optimized for more efficient operation with different Downloadable-destinations or destination capabilities.)

While integrated protection constraints within the MPC might also be utilized, providing separate protection policies has been found to be more efficient, for example, by enabling more specific protection constraints to be more easily updated in conjunction with detected-Downloadable specifics, post-download improvements, testing, etc. Separate policies can further be more efficiently provided (e.g. selected, modified, instantiated, etc.) with or separately from an MPC, or in accordance with the requirements of a particular user, device, system, administration, later improvement, etc., as might also be provided to protection engine 310 (e.g. via user/MPC uploading, querying, parsing a Downloadable, or other suitable mechanism implemented by one or more servers or Downloadable-destinations).

(It will also become apparent that performing executable code detection and communicating to a downloadable-Destination an MPC and any applicable policies as separate from a detected-Downloadable is more accurate and far less resource intensive than, for example, performing content and operation scanning, modifying a Downloadable, or providing completely Downloadable-destination based security.)

System 300 enables a single or extensible base-MPC to be provided, in anticipation or upon receipt of a first Downloadable, that is utilized thereafter to provide protection of one or more Downloadable-destinations. It is found, however, that providing an MPC upon each detection of a Downloadable (which

is also enabled) can provide a desirable combination of configurability of the MPC/policies and lessened need for management (e.g. given potentially changing user/destination needs, enabling testing, etc.).

Providing an MPC upon each detection of a Downloadable also facilitates
5 a lessened demand on destination resources, e.g. since information-destination resources used in executing the MPC/policies can be re-allocated following such use. Such alternatives can also be selectively, modifiably or extensibly provided (or further in accordance with other application-specific factors that might also apply.) Thus, for example, a base-MPC or base-policies might be provided to a
10 user device that is/are extensible via additionally downloadable "modules" upon server 301 detection of a Downloadable deliverable to the same user device, among other alternatives.

In accordance with a further aspect of the invention, it is found that improved efficiency can also be achieved by causing the MPC to be executed
15 within a Downloadable-destination in conjunction with, and further, prior to initiation of the detected Downloadable. One mechanism that provides for greater compatibility and efficiency in conjunction with conventional client-based Downloadable execution is for a protection engine to form a sandboxed package 340 including MPC 341, the detected-Downloadable 343 and any policies 342.
20 For example, where the Downloadable is a binary executable to be executed by an operating system, protection engine 310 forms a protected package by concatenating, within sandboxed package 340, MPC 341 for delivery to a Downloadable-destination first, followed by protection policies 342 and

Downloadable 343. (Concatenation or techniques consistent therewith can also be utilized for providing a protecting package corresponding to a Java applet for execution by a JVM of a Downloadable-destination, or with regard to ActiveX controls, add-ins or other distributable components, etc.)

5 The above concatenation or other suitable processing will result in the following. Upon receipt of sandboxed package 340 by a compatible browser, email or other destination-client and activating of the package by a user or the destination-client, the operating system (or a suitable responsively initiated distributed component host) will attempt to initiate sandboxed package 340 as a
10 single Downloadable. Such processing will, however, result in initiating the MPC 341 and -in accordance with further aspects of the invention- the MPC will initiate the Downloadable in a protected manner, further in accordance with any applicable included or further downloaded protection policies 342. (While system 300 is also capable of ascertaining protection policies stored at a Downloadable-
15 destination, e.g. by poll, query, etc. of available destination information, including at least initial policies within a suitable protecting package is found to avoid associated security concerns or inefficiencies.)

Turning to FIG. 4, a protection engine 400 generally consistent with protection engine 310 (or 320) of FIG. 3 is illustrated in accordance with an
20 embodiment of the invention. Protection engine 400 comprises information monitor 401, detection engine 402, and protected packaging engine 403, which further includes agent generator 431, storage 404, linking engine 405, and transfer engine 406. Protection engine 400 can also include a buffer 407, for temporarily

storing a received potential-Downloadable, or one or more systems for conducting additional authentication, certification, verification or other security processing (e.g. summarily depicted as security system 408) Protection engine 400 can further provide for selectively re-directing, further directing, logging, etc. of a
5 potential/detected Downloadable or information corresponding thereto in conjunction with detection, other security, etc., in accordance with a particular application.

(Note that FIG. 4, as with other figures included herein, also depicts exemplary signal flow arrows; such arrows are provided to facilitate discussion,
10 and should not be construed as exclusive or otherwise limiting.)

Information monitor 401 monitors potential-Downloadables received by a host server and provides the information via buffer 407 to detection engine 402 or to other system 400 elements. Information monitor 401 can be configured to monitor host server download operations in conjunction with a user or a user-
15 device that has logged-on to the server, or to receive information via a server operation hook, servlet, communication channel or other suitable mechanism.

Information monitor 401 can also provide for transferring, to storage 404 or other protection engine elements, configuration information including, for example, user, MPC, protection policy, interfacing or other configuration
20 information (e.g. see FIG. 6). Such configuration information monitoring can be conducted in accordance with a user/device logging onto or otherwise accessing a host server, via one or more of configuration operations, using an applet to acquire

such information from or for a particular user, device or devices, via MPC/policy polling of a user device, or via other suitable mechanisms.

Detection engine 402 includes code detector 421, which receives a potential-Downloadable and determines, more suitably in conjunction with inspection parameters 422, whether the potential-Downloadable includes executable code and is thus a "detected-Downloadable". (Code detector 421 can also include detection processors for performing file decompression or other "decoding", or such detection-facilitating processing as decryption, utilization/support of security system 408, etc. in accordance with a particular application.)

Detection engine 402 further transfers a detected-downloadable ("XEQ") to protected packaging engine 403 along with indicators of such detection, or a determined non-executable ("NXEQ") to transfer engine 406. (Inspection parameters 422 enable analysis criteria to be readily updated or varied, for example, in accordance with particular source, destination or other potential Downloadable impacting parameters, and are discussed in greater detail with reference to FIG. 5). Detection engine 402 can also provide indicators for delivery of initial and further MPCs/policies, for example, prior to or in conjunction with detecting a Downloadable and further upon receipt of an indicator from an already downloaded MPC/policy. A downloaded MPC/policy can further remain resident at a user device with further modules downloaded upon or even after delivery of a sandboxed package. Such distribution can also be provided in a configurable manner, such that delivery of a complete package or partial packages are

automatically or interactively determinable in accordance with user/administrative preferences/policies, among other examples.

Packaging engine 403 provides for generating mobile protection code and protection policies, and for causing delivery thereof (typically with a detected-
5 Downloadable) to a Downloadable-destination for protecting the Downloadable-destination against malicious operation attempts by the detected Downloadable. In this example, packaging engine 403 includes agent generator 431, storage 404 and linking engine 405.

Agent generator 431 includes an MPC generator 432 and a protection
10 policy generator 433 for “generating” an MPC and a protection policy (or set of policies) respectively upon receiving one or more “generate MPC/policy” indicators from detection engine 402, indicating that a potential-Downloadable is a detected-Downloadable. MPC generator 432 and protection policy generator 433 provide for generating MPCs and protection policies respectively in
15 accordance with parameters retrieved from storage 404. Agent generator 431 is further capable of providing multiple MPCs/policies, for example, the same or different MPCs/policies in accordance with protecting ones of multiple executables within a zip file, or for providing initial MPCs/policies and then further MPCs/policies or MPC/policy “modules” as initiated by further indicators
20 such as given above, via an indicator of an already downloaded MPC/policy or via other suitable mechanisms. (It will be appreciated that pre-constructed MPCs/policies or other processing can also be utilized, e.g. via retrieval from storage 404, but with a potential decrease in flexibility.)

MPC generator 432 and protection policy generator 433 are further configurable. Thus, for example, more generic MPCs/policies can be provided to all or a grouping of serviced destination-devices (e.g. in accordance with a similarly configured/administered intranet), or different MPCs/policies that can be
5 configured in accordance with one or more of user, network administration, Downloadable-destination or other parameters (e.g. see FIG. 6). As will become apparent, a resulting MPC provides an operational interface to a destination device/process. Thus, a high degree of flexibility and efficiency is enabled in providing such an operational interface within different or differently configurable
10 user devices/processes or other constraints.

Such configurability further enables particular policies to be utilized in accordance with a particular application (e.g. particular system uses, access limitations, user interaction, treating application programs or Java components from a particular known source one way and unknown source ActiveX
15 components, or other considerations). Agent generator 431 further transfers a resulting MPC and protection policy pair to linking engine 405.

Linking engine 405 provides for forming from received component elements (see above) a sandboxed package that can include one or more initial or complete MPCs and applicable protection policies, and a Downloadable, such that
20 the sandboxed package will protect a receiving Downloadable-destination from malicious operation by the Downloadable. Linking engine 405 is implementable in a static or configurable manner in accordance, for example, with characteristics of a particular user device/process stored intermittently or more persistently in

storage 404. Linking engine 405 can also provide for restoring a Downloadable, such as a compressed, encrypted or otherwise encoded file that has been decompressed, decrypted or otherwise decoded via detection processing (e.g. see FIG. 6b).

5 It is discovered, for example, that the manner in which the Windows OS initiates a binary executable or an ActiveX control can be utilized to enable protected initiation of a detected-Downloadable. Linking engine 405 is, for example, configurable to form, for an ordinary single-executable Downloadable (e.g. an application program, applet, etc.) a sandboxed package 340 as a
10 concatenation of ordered elements including an MPC 341, applicable policies 342 and the Downloadable or "XEQ" 343 (e.g. see FIG. 4).

Linking engine 405 is also configurable to form, for a Downloadable received by a server as a compressed single or multiple-executable Downloadable such as a zipped or meta file, a protecting package 340 including one or more
15 MPCs, applicable policies and the one or more included executables of the Downloadable. For example, a sandboxed package can be formed in which a single MPC and policies precede and thus will affect all such executables as a result of inflating and installation. An MPC and applicable policies can also, for example, precede each executable, such that each executable will be separately
20 sandboxed in the same or a different manner according to MPC/policy configuration (see above) upon inflation and installation. (See also FIGS. 5 and 6)

Linking engine is also configurable to form an initial MPC, MPC-policy or sandboxed package (e.g. prior to upon receipt of a downloadable) or an additional

MPC, MPC-policy or sandboxed package (e.g. upon or following receipt of a downloadable), such that suitable MPCs/policies can be provided to a Downloadable-destination or other destination in a more distributed manner. In this way, requisite bandwidth or destination resources can be minimized (via two
5 or more smaller packages) in compromise with latency or other considerations raised by the additional required communication.

A configurable linking engine can also be utilized in accordance with other requirements of particular devices/processes, further or different elements or other permutations in accordance with the teachings herein. (It might, for example be
10 desirable to modify the ordering of elements, to provide one or more elements separately, to provide additional information, such as a header, etc., or perform other processing in accordance with a particular device, protocol or other application considerations.)

Policy/authentication reader-analyzer 481 summarily depicts other
15 protection mechanisms that might be utilized in conjunction with Downloadable detection, such as already discussed, and that can further be configurable to operate in accordance with policies or parameters (summarily depicted by security/authentication policies 482). Integration of such further protection in the depicted configuration, for example, enables a potential-Downloadable from a
20 known unfriendly source, a source failing authentication or a provided-source that is confirmed to be fictitious to be summarily discarded, otherwise blocked, flagged, etc. (with or without further processing). Conversely, a potential-Downloadable from a known friendly source (or one confirmed as such) can be

transferred with or without further processing in accordance with particular application considerations. (Other configurations including pre or post Downloadable detection mechanisms might also be utilized.)

Finally, transfer engine 406 of protection agent engine 303 provides for
5 receiving and causing linking engine 405 (or other protection) results to be transferred to a destination user device/process. As depicted, transfer engine 406 is configured to receive and transfer a Downloadable, a determined non-executable or a sandboxed package. However, transfer engine 406 can also be provided in a more configurable manner, such as was already discussed for other
10 system 400 elements. (Any one or more of system 400 elements might be configurably implemented in accordance with a particular application.) Transfer engine 406 can perform such transfer, for example, by adding the information to a server transfer queue (not shown) or utilizing another suitable method.

Turning to FIG. 5 with reference to FIG. 4, a code detector 421 example is
15 illustrated in accordance with an embodiment of the invention. As shown, code detector 421 includes data fetcher 501, parser 502, file-type detector 503, inflater 504 and control 506; other depicted elements. While implementable and potentially useful in certain instances, are found to require substantial overhead, to be less accurate in certain instances (see above) and are not utilized in a present
20 implementation; these will be discussed separately below. Code detector elements are further configurable in accordance with stored parameters retrievable by data fetcher 501. (A coupling between data fetcher 501 and control 506 has been removed for clarity sake.)

Data fetcher 501 provides for retrieving a potential-Downloadable or portions thereof stored in buffer 407 or parameters from storage 404, and communicates such information or parameters to parser 502. Parser 502 receives a potential-Downloadable or portions thereof from data fetcher 501 and isolates
5 potential-Downloadable elements, such as file headers, source, destination, certificates, etc. for use by further processing elements.

File type detector 502 receives and determines whether the potential-Downloadable (likely) is or includes an executable file type. File-reader 502 can, for example, be configured to analyze a received potential-Downloadable for a file
10 header, which is typically included in accordance with conventional data transfer protocols, such as a portable executable or standard ".exe" file format for Windows OS application programs, a Java class header for Java applets, and so on for other applications, distributed components, etc. "Zipped", meta or other compressed files, which might include one or more executables, also typically
15 provide standard single or multi-level headers that can be read and used to identify included executable code (or other included information types). File type detector 502 is also configurable for analyzing potential-Downloadables for all potential file type delimiters or a more limited subset of potential file type delimiters (e.g. ".exe" or ".com" in conjunction with a DOS or Microsoft Windows OS
20 Downloadable-destination).

Known file type delimiters can, for example, be stored in a more temporary or more persistent storage (e.g. storage 404 of FIG. 4) which file type detector 502 can compare to a received potential-Downloadable. (Such delimiters can thus also

be updated in storage 404 as a new file type delimiter is provided, or a more limited subset of delimiters can also be utilized in accordance with a particular Downloadable-destination or other considerations of a particular application.) File type detector 502 further transfers to controller 506 a detected file type indicator
5 indicating that the potential-Downloadable includes or does not include (i.e. or likely include) an executable file type.

In this example, the aforementioned detection processor is also included as pre-detection processor or, more particularly, a configurable file inflator 504. File inflator 504 provides for opening or "inflating" compressed files in accordance
10 with a compressed file type received from file type detector 503 and corresponding file opening parameters received from data fetcher 501. Where a compressed file (e.g. a meta file) includes nested file type information not otherwise reliably provided in an overall file header or other information, inflator 504 returns such information to parser 502. File inflator 504 also provides any
15 now-accessible included executables to control 506 where one or more included files are to be separately packaged with an MPC or policies.

Control 506, in this example, operates in accordance with stored parameters and provides for routing detected non-Downloadables or Downloadables and control information, and for conducting the aforementioned
20 distributed downloading of packages to Downloadable-destinations. In the case of a non-Downloadable, for example, control 506 sends the non-Downloadable to transfer engine 406 (FIG. 4) along with any indicators that might apply. For an ordinary single-executable Downloadable, control 506 sends control information

to agent generator 431 and the Downloadable to linking engine 405 along with any other applicable indicators (see 641 of FIG. 6b). Control 506 similarly handles a compressed single-executable Downloadable or a multiple downloadable to be protected using a single sandboxed package. For a multiple-executable

5 Downloadable, control 506 sends control information for each corresponding executable to agent generator agent generator 431, and sends the executable to linking engine 405 along with controls and any applicable indicators, as in 643b of FIG. 6b. (The above assumes, however, that distributed downloading is not utilized; when used –according to applicable parameters- control 506 also operates

10 in accordance with the following.)

Control 506 conducts distributed protection (e.g. distributed packaging) by providing control signals to agent generator 431, linking engine 405 and transfer engine 406. In the present example, control 506 initially sends controls to agent generator 431 and linking engine 405 (FIG. 4) causing agent generator to generate

15 an initial MPC and initial policies, and sends control and a detected-Downloadable to linking engine 405. Linking engine 405 forms an initial sandboxed package, which transfer engine causes (in conjunction with further controls) to be downloaded to the Downloadable destination (643a of FIG. 6b). An initial MPC within the sandboxed package includes an installer and a

20 communicator and performs installation as indicated below. The initial MPC also communicates via the communicator controls to control 506 (FIG. 5) in response to which control 506 similarly causes generation of MPC-M and policy-M

modules 643c, which linking engine 405 links and transfer engine 406 causes to be sent to the Downloadable destination, and so on for any further such modules.

(It will be appreciated, however, that an initial package might be otherwise configured or sent prior to receipt of a Downloadable in accordance with
5 configuration parameters or user interaction. Information can also be sent to other user devices, such as that of an administrator. Further MPCs/policies might also be coordinated by control 506 or other elements, or other suitable mechanisms might be utilized in accordance with the teachings herein.)

Regarding the remaining detection engine elements illustrated in FIG. 5,
10 where content analysis is utilized, parser 502 can also provide a Downloadable or portions thereof to content detector 505. Content detector 505 can then provide one or more content analyses. Binary detector 551, for example, performs detection of binary information; pattern detector 552 further analyzes the Downloadable for patterns indicating executable code, or other detectors can also
15 be utilized. Analysis results therefrom can be used in an absolute manner, where a first testing result indicating executable code confirms Downloadable detection, which result is then sent to control 506. Alternatively, however, composite results from such analyses can also be sent to control 506 for evaluation. Control 506 can further conduct such evaluation in a summary manner (determining whether a
20 Downloadable is detected according to a majority or minimum number of indicators), or based on a weighting of different analysis results. Operation then continues as indicated above. (Such analysis can also be conducted in accordance with aspects of a destination user device or other parameters.)

FIG. 6a illustrates more specific examples of indicators/parameters and known (or "knowledge base") elements that can be utilized to facilitate the above-discussed system 400 configurability and detection. For clarity sake, indicators, parameters and knowledge base elements are combined as indicated "parameters."

5 It will be appreciated, however, that the particular parameters utilized can differ in accordance with a particular application, and indicators, parameters or known elements, where utilized, can vary and need not correspond exactly with one another. Any suitable explicit or referencing list, database or other storage structure(s) or storage structure configuration(s) can also be utilized to implement

10 a suitable user/device based protection scheme, such as in the above examples, or other desired protection schema.

Executable parameters 601 comprise, in accordance with the above examples, executable file type parameters 611, executable code parameters 612 and code pattern parameters 613 (including known executable file type indicators,

15 header/code indicators and patterns respectively, where code patterns are utilized). Use parameters 602 further comprise user parameters 621, system parameters 622 and general parameters 623 corresponding to one or more users, user classifications, user-system correspondences or destination system, device or processes, etc. (e.g. for generating corresponding MPCs/policies, providing other

20 protection, etc.). The remaining parameters include interface parameters 631 for providing MPC/policy (or further) configurability in accordance with a particular device or for enabling communication with a device user (see below), and other parameters 632.

FIG. 6b illustrates a linking engine 405 according to an embodiment of the invention. As already discussed, linking engine 405 includes a linker for combining MPCs, policies or agents via concatenation or other suitable processing in accordance with an OS, JVM or other host executor or other applicable factors that might apply. Linking engine 405 also includes the aforementioned post-
5 detection processor which, in this example, comprises a compressor 508. As noted, compressor 508 receives linked elements from linker 507 and, where a potential-Downloadable corresponds to a compressed file that was inflated during detection, re-forms the compressed file. (Known file information can be provided
10 via configuration parameters, substantially reversal of inflating or another suitable method.) Encryption or other post-detection processing can also be conducted by linking engine 508.

FIGS. 7a, 7b and 8 illustrate a "sandbox protection" system, as operable within a receiving destination-device, according to an embodiment of the
15 invention.

Beginning with FIG. 7a, a client 146 receiving sandbox package 340 will "recognize" sandbox package 340 as a (mobile) executable and cause a mobile code installer 711 (e.g. an OS loader, JVM, etc.) to be initiated. Mobile code installer 711 will also recognize sandbox package 340 as an executable and will
20 attempt to initiate sandbox package 340 at its "beginning." Protection engine 400 processing corresponding to destination 700 use of a such a loader, however, will have resulted in the "beginning" of sandbox package 340 as corresponding to the beginning of MPC 341, as noted with regard to the above FIG. 4 example.

Such protection engine processing will therefore cause a mobile code installer (e.g. OS loader 711, for clarity sake) to initiate MPC 341. In other cases, other processing might also be utilized for causing such initiation or further protection system operation. Protection engine processing also enables MPC 341 to effectively form a protection "sandbox" around Downloadable (e.g. detected-Downloadable or "XEQ") 343, to monitor Downloadable 343, intercept determinable Downloadable 343 operation (such as attempted accesses of Downloadable 343 to destination resources) and, if "malicious", to cause one or more other operations to occur (e.g. providing an alert, offloading the Downloadable, offloading the MPC, providing only limited resource access, possibly in a particular address space or with regard to a particularly "safe" resource or resource operation, etc.).

MPC 341, in the present OS example, executes MPC element installation and installs any policies, causing MPC 341 and protection policies 342 to be loaded into a first memory space, P1. MPC 341 then initiates loading of Downloadable 343. Such Downloadable initiation causes OS loader 711 to load Downloadable 343 into a further working memory space-P2 703 along with an API import table ("IAT") 731 for providing Downloadable 631 with destination resource access capabilities. It is discovered, however that the IAT can be modified so that any call to an API can be redirected to a function within the MPC. The technique for modifying the IAT is documented within the MSDN (Microsoft Developers Network) Library CD in several articles. The technique is also different for each operating system (e.g. between Windows 9x and Windows

NT), which can be accommodated by agent generator configurability, such as that given above.

MPC 341 therefore has at least initial access to API IAT 731 of Downloadable 632, and provides for diverting, evaluating and responding to attempts by

- 5 Downloadable 632 to utilize system APIs 731, or further in accordance with protection policies 342.

In addition to API diverting, MPC 341 can also install filter drivers, which can be used for controlling access to resources such as a Downloadable-destination file system or registry. Filter driver installation can be conducted as documented in
10 the MSDN or using other suitable methods.

Turning to FIG. 8 with reference to FIG. 7b, an MPC 341 according to an embodiment of the invention includes a package extractor 801, executable installer 802, sandbox engine installer 803, resource access diverter 804, resource access (attempt) analyzer 805, policy enforcer 806 and MPC de-installer 807.

- 15 Package extractor 801 is initiated upon initiation of MPC 341, and extracts MPC 341 elements and protection policies 342. Executable installer 802 further initiates installation of a Downloadable by extracting the downloadable from the protected package, and loading the process into memory in suspended mode (so it only loads into memory, but does not start to run). Such installation further causes
20 the operating system to initialize the Downloadable's IAT 731 in the memory space of the downloadable process, P2, as already noted.

Sandbox engine installer 803 (running in process space P1) then installs the sandbox engine (803-805) and policies 342 into the downloadable process

space P2. This is done in different way in each operating system (e.g. see above). Resource access diverter 804 further modifies those Downloadable-API IAT entries that correspond with protection policies 342, thereby causing corresponding Downloadable accesses via Downloadable-API IAT 731 to be
5 diverted resource access analyzer 805.

During Downloadable operation, resource access analyzer or "RAA" 805 receives and determines a response to diverted Downloadable (i.e. "malicious") operations in accordance with corresponding protection policies of policies 342. (RAA 805 or further elements, which are not shown, can further similarly provide
10 for other security mechanisms that might also be implemented.) Malicious operations can for example include, in a Windows environment: file operations (e.g. reading, writing, deleting or renaming a file), network operations (e.g. listen on or connect to a socket, send/receive data or view intranet), OS registry or similar operations (read/write a registry item), OS operations (exit OS/client, kill
15 or change the priority of a process/thread, dynamically load a class library), resource usage thresholds (e.g. memory, CPU, graphics), etc.

Policy enforcer 806 receives RAA 805 results and causes a corresponding response to be implemented, again according to the corresponding policies. Policy enforcer 806 can, for example, interact with a user (e.g. provide an alert, receive
20 instructions, etc.), create a log file, respond, cause a response to be transferred to the Downloadable using "dummy" or limited data, communicate with a server or other networked device (e.g. corresponding to a local or remote administrator), respond more specifically with a better known Downloadable, verify accessibility

or user/system information (e.g. via local or remote information), even enable the attempted Downloadable access, among a wide variety of responses that will become apparent in view of the teachings herein.

The FIG. 9 flowchart illustrates a protection method according to an embodiment of the invention. In step 901, a protection engine monitors the receipt, by a server or other re-communicator of information, and receives such information intended for a protected information-destination (i.e. a potential-Downloadable) in step 903. Steps 905-911 depict an adjunct trustworthiness protection that can also be provided, wherein the protection engine determines whether the source of the received information is known to be "unfriendly" and, if so, prevents current (at least unaltered) delivery of the potential-Downloadable and provides any suitable alerts. (The protection engine might also continue to perform Downloadable detection and nevertheless enable delivery or protected delivery of a non-Downloadable, or avoid detection if the source is found to be "trusted", among other alternatives enabled by the teachings herein.)

If, in step 913, the potential-Downloadable source is found to be of an unknown or otherwise suitably authenticated/certified source, then the protection engine determines whether the potential-Downloadable includes executable code in step 915. If the potential-Downloadable does not include executable code, then the protection engine causes the potential-Downloadable to be delivered to the information-destination in its original form in step 917, and the method ends. If instead the potential-Downloadable is found to include executable code in step 915 (and is thus a "detected-Downloadable"), then the protection engine forms a

sandboxed package in step 919 and causes the protection agent to be delivered to the information-Destination in step 921, and the method ends. As was discussed earlier, a suitable protection agent can include mobile protection code, policies and the detected-Downloadable (or information corresponding thereto).

5 The FIG. 10a flowchart illustrates a method for analyzing a potential-Downloadable, according to an embodiment of the invention. As shown, one or more aspects can provide useful indicators of the inclusion of executable code within the potential-Downloadable. In step 1001, the protection engine determines whether the potential-Downloadable indicates an executable file type, for example, by comparing one or more included file headers for file type
10 indicators (e.g. extensions or other descriptors). The indicators can be compared against all known file types executable by all protected Downloadable destinations, a subset, in accordance with file types executable or desirably executable by the Downloadable-destination, in conjunction with a particular user, in conjunction with available information or operability at the destination, various
15 combinations, etc.

Where content analysis is conducted, in step 1003 of FIG. 10a, the protection engine analyzes the potential-Downloadable and determines in accordance therewith whether the potential-Downloadable does or is likely to
20 include binary information, which typically indicates executable code. The protection engine further analyzes the potential-Downloadable for patterns indicative of included executable code in step 1003. Finally, in step 1005, the protection engine determines whether the results of steps 1001 and 1003 indicate

that the potential-Downloadable more likely includes executable code (e.g. via weighted comparison of the results with a suitable level indicating the inclusion or exclusion of executable code). The protection engine, given a suitably high confidence indicator of the inclusion of executable code, treats the potential-
5 Downloadable as a detected-Downloadable.

The FIG. 10b flowchart illustrates a method for forming a sandboxed package according to an embodiment of the invention. As shown, in step 1011, a protection engine retrieves protection parameters and forms mobile protection code according to the parameters. The protection engine further, in step 1013,
10 retrieves protection parameters and forms protection policies according to the parameters. Finally, in step 1015, the protection engine couples the mobile protection code, protection policies and received-information to form a sandboxed package. For example, where a Downloadable-destination utilizes a standard windows executable, coupling can further be accomplished via concatenating the
15 MPC for delivery of MPC first, policies second, and received information third. (The protection parameters can, for example, include parameters relating to one or more of the Downloadable destination device/process, user, supervisory constraints or other parameters.)

The FIG. 11 flowchart illustrates how a protection method performed by
20 mobile protection code ("MPC") according to an embodiment of the invention includes the MPC installing MPC elements and policies within a destination device in step 1101. In step 1102, the MPC loads the Downloadable without actually initiating it (i.e. for executables, it will start a process in suspended

mode). The MPC further forms an access monitor or "interceptor" for monitoring or "intercepting" downloadable destination device access attempts within the destination device (according to the protection policies in step 1103, and initiates a corresponding Downloadable within the destination device in step 1105.

- 5 If, in step 1107, the MPC determines, from monitored/intercepted information, that the Downloadable is attempting or has attempted a destination device access considered undesirable or otherwise malicious, then the MPC performs steps 1109 and 1111; otherwise the MPC returns to step 1107. In step 1109, the MPC determines protection policies in accordance with the access
10 attempt by the Downloadable, and in step 1111, the MPC executes the protection policies. (Protection policies can, for example, be retrieved from a temporary, e.g. memory/cache, or more persistent storage.)

- As shown in the FIG. 12a example, the MPC can provide for intercepting Downloadable access attempts by a Downloadable by installing the Downloadable
15 (but not executing it) in step 1201. Such installation will cause a Downloadable executor, such as a the Windows operating system, to provide all required interfaces and parameters (such as the IAT, process ID, etc.) for use by the Downloadable to access device resources of the host device. The MPC can thus cause Downloadable access attempts to be diverted to the MPC by modifying the
20 Downloadable IAT, replacing device resource location indicators with those of the MPC (step 1203).

The FIG. 12b example further illustrates an example of how the MPC can apply suitable policies in accordance with an access attempt by a Downloadable.

As shown, the MPC receives the Downloadable access request via the modified IAT in step 1211. The MPC further queries stored policies to determine a policy corresponding to the Downloadable access request in step 1213.

The foregoing description of preferred embodiments of the invention is provided by way of example to enable a person skilled in the art to make and use the invention, and in the context of particular applications and requirements thereof. Various modifications to the embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles, features and teachings disclosed herein. The embodiments described herein are not intended to be exhaustive or limiting. The present invention is limited only by the following claims.

15

20

WHAT IS CLAIMED IS:

1. A method, comprising:
 - receiving downloadable-information;
 - determining whether the downloadable-information includes executable
 - 5 code; and
 - causing mobile protection code to be communicated to at least one
 - information-destination of the downloadable-information, if the downloadable-
 - information is determined to include executable code.
- 10 2. The method of claim 1, wherein the receiving includes monitoring received
- information of an information re-communicator.
3. The method of claim 2, wherein the information re-communicator is a network
- server.
- 15 4. The method of claim 1, wherein the determining comprises analyzing the
- downloadable-information for an included type indicator indicating an executable
- file type.
- 20 5. The method of claim 1, wherein the determining comprises analyzing the
- downloadable-information for an included an included type detector indicating an
- archive file that contains at least one executable.

6. The method of claim 1, wherein the determining comprises analyzing the downloadable-information for an included file type indicator and an information pattern corresponding to one or more information patterns that tend to be included within executable code.

5

7. The method of claim 1, further comprising receiving one or more executable code characteristics of executable code that is capable of being executed by the information-destination, and wherein the determining is conducted in accordance with the executable code characteristics.

10

8. The method of claim 1, wherein the determining comprises performing one or more analyses of the downloadable-information, the analyses producing detection-indicators indicating whether a correspondence is detected between a downloadable-information characteristic and at least one respective executable
15 code characteristic, and evaluating the detection-indicators to determine whether the downloadable-information includes executable code.

9. The method of claim 8, wherein at least one of the detection-indicators indicates a level of downloadable-information characteristic and executable code
20 characteristic correspondence.

10. The method of claim 8, wherein the evaluating includes assigning a weighted level of importance to at least one of the indicators.

11. The method of claim 1, wherein the causing mobile protection code to be communicated comprises forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed
5 package to be communicated to the at least one information-destination.
12. The method of claim 10, wherein the sandboxed package is formed such that the mobile protection code will be executed by the information-destination before the downloadable-information.
- 10 13. The method of claim 11, wherein the sandboxed package further includes protection policies according to which the mobile protection code is operable.
14. The method of claim 13, wherein the sandboxed package is formed for receipt
15 by the information-destination such that the mobile protection code is received before the downloadable-information, and the downloadable information before the protection policies.
15. The method of claim 13, wherein the protection policies correspond with at
20 least one of the information-destination and a user of the information destination.
16. A system, comprising:
an information monitor for receiving downloadable-information;

a content inspection engine communicatively coupled to the information monitor for determining whether the downloadable-information includes executable code; and

a protection agent engine communicatively coupled to the content inspection engine for causing mobile protection code ("MPC") to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code.

10 17. The system of claim 16, wherein the information monitor intercepts received information received by an information re-communicator.

18. The system of claim 17, wherein the information re-communicator is a network server.

15

19. The system of claim 16, wherein the content inspection engine comprises a file type detector for determining whether the downloadable-information includes a file type indicator indicating an executable file type.

20 20. The system of claim 16, wherein the content inspection engine comprises a parser for parsing the downloadable-information and a content analyzer communicatively coupled to the parser for determining whether one or more downloadable-information elements of the downloadable-information correspond

with executable code elements are executable code elements.

21. The system of claim 16, wherein the content inspection engine comprises one or more downloadable-information analyzers for analyzing the downloadable-
5 information, each analyzer producing therefrom a detection indicator indicating whether a downloadable-information characteristic corresponds with an executable code characteristic, and an inspection controller communicatively coupled to the analyzers for determining whether the indicators indicate that the
downloadable-information includes executable code.

10

22. The system of claim 21, wherein at least one of the detection-indicators indicates a level of downloadable-information characteristic and executable code characteristic correspondence.

15 23. The system of claim 21, wherein the evaluating includes assigning a weighted level of importance to at least one of the detection-indicators.

24. The system of claim 16, wherein the sandboxed package engine comprises an MPC generator for providing the MPC, a linking engine coupled to the MPC
20 generator for forming a protection agent including the MPC and the downloadable-information, and a transfer engine for causing the protection agent to be communicated to the at least one information-destination.

25. The system of claim 24, wherein the protection agent engine further comprises a policy generator communicatively coupled to the linking engine for providing protection policies according to which the MPC is operable.
- 5 26. The system of claim 25, wherein the sandboxed package is formed for receipt by the information-destination such that the mobile protection code is executed before the downloadable-information.
27. The system of claim 26, wherein the protection policies correspond with
10 policies of at least one of the information-destination and a user of the information destination.
28. A system, comprising:
- means for receiving downloadable-information;
 - 15 means for determining whether the downloadable-information includes executable code; and
 - means for causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code.
- 20
29. A computer-readable storage medium storing program code for causing a computer to perform the steps of:
- receiving downloadable-information;

determining whether the downloadable-information includes executable code; and

causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-
5 information is determined to include executable code.

30. A method, comprising:

receiving, at an information re-communicator, downloadable-information, including executable code; and

10 causing mobile protection code to be executed by a mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code.

15 31. The method of claim 30, wherein the mobile code executor is a Java Virtual Machine.

32. The method of claim 30, wherein the mobile code executor is the operating system, running native code executables.

20

33. The method of claim 30, wherein the mobile code executor is ActiveX subsystem of the windows operating system

34. The method of claim 30, wherein the mobile code executor is the Microsoft Windows scripting host.

35. The method of claim 30, wherein the causing is accomplished by forming a
5 sandboxed package including the mobile protection code and the downloadable-
information, and causing the sandboxed package to be delivered to the
downloadable-information destination.

36. The method of claim 35, wherein the sandboxed package further includes
10 protection policies according to which the processing by the mobile protection
code is conducted.

37. A sandboxed package formed according to the method of claim 35.

15 38. A sandboxed package formed according to the method of claim 36.

39. The method of claim 36, wherein the forming comprises generating the mobile
protection code, generating the sandboxed package, and linking the mobile
protection code, protection policies and downloadable-information.

20

40. The method of claim 39, wherein the generating of at least one of the mobile
protection code and the protection policies is conducted in accordance with one or
more destination-characteristics of the destination.

41. The method of claim 40, wherein the destination-characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process.

5

42. The method of claim 35, wherein the causing the sandboxed package to be executed includes communicating the sandboxed package to a communication buffer of the information re-communicator.

10 43. The method of claim 30, wherein the re-communicator is at least one of a firewall and a network server.

44. The method of claim 30, wherein the sandboxed package has a same file type as the downloadable-information, thereby causing the mobile code executor to be
15 unaware that the protected package is not a normal downloadable.

45. The method of claim 44, wherein the sandboxed package is formed using concatenation of a mobile protection code, a policy, and a downloadable.

20 46. The method of claim 30, wherein executing the mobile protection code at the destination causes downloadable interfaces to resources at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code.

47. A system, comprising:

receiving means for receiving, at an information re-communicator,
downloadable-information, including executable code; and

5 mobile code means communicatively coupled to the receiving means for
causing mobile protection code to be executed by a mobile code executor at a
downloadable-information destination such that one or more operations of the
executable code at the destination, if attempted, will be processed by the mobile
protection code.

10

48. The system of claim 47, wherein the mobile code executor is a Java Virtual
Machine.

49. The system of claim 47, wherein the mobile code executor is an operating
15 system, running native code executables.

50. The system of claim 47, wherein the mobile code executor is an ActiveX
subsystem of the windows operating system.

20 51. The system of claim 47, wherein the mobile code executor is a Microsoft
Windows scripting host.

52. The system of claim 47, wherein the causing is accomplished by forming a

sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable-information destination.

5 53. The system of claim 52, wherein the sandboxed package further includes protection policies according to which the processing by the mobile protection code is conducted.

54. The system of claim 53, wherein the forming comprises generating the mobile
10 protection code, generating the protection policies, and linking the mobile protection code, protection policies and downloadable-information.

55. The system of claim 54, wherein the generating of at least one of the mobile protection code and the protection policies is conducted in accordance with one or
15 more destination-characteristics of the destination.

56. The system of claim 55, wherein the destination-characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process.

20

57. The system of claim 46, wherein the causing the sandboxed package to be executed includes communicating the sandboxed package to a communication buffer of the information re-communicator.

58. The system of claim 47, wherein the re-communicator is at least one of a firewall and a network server.

5 59. The system of claim 47, wherein executing the mobile protection code at the destination causes downloadable interfaces a resource at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code.

10 60. A computer-readable storage medium storing program code for causing a computer to perform the steps of:

receiving, at an information re-communicator, downloadable-information, including executable code; and

causing mobile protection code to be executed by a mobile code executor
15 at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code.

61. A method, comprising:

20 receiving mobile protection code ("MPC") and a Downloadable at a Downloadable-destination;

causing, by the MPC, one or more operations attempted by the Downloadable to be received by the MPC;

receiving, by the MPC, an attempted operation of the Downloadable; and
initiating, by the MPC, a protection policy corresponding to the attempted
operation.

5 62. The method of claim 61, wherein the receiving comprises receiving a
sandboxed package that includes the MPC, the Downloadable and one or more
protection policies.

63. The method of claim 62, wherein the sandboxed package is configured such
10 that the MPC is executed first, the Downloadable is executed by the MPC and the
protection policies are accessible to the MPC.

64. The method of claim 61, wherein the causing comprises modifying, by the
MPC, interfaces of a corresponding downloadable to resources at the destination.

15

65. The method of claim 64, wherein the modifying is accomplished by initiating a
loading of the Downloadable, thereby causing a mobile code executor to provide
and initialize the interfaces, modifying one or more interface elements to divert
corresponding attempted Downloadable operations to the MPC, and initiating
20 execution of the Downloadable.

66. The method of claim 64, wherein the interfaces comprise an import address
table ("IAT") of a native code executable downloadable.

67. The method of claim 64, wherein modifying the interfaces installs a filter-driver between the downloadable and the resources.

5 68. A system, comprising:

a mobile code executor for initiating received mobile code; and

a sandboxed package capable of being received and initiated by the mobile code executor, the sandboxed package including a Downloadable and mobile protection code ("MPC") for causing one or more Downloadable operations to be intercepted and for processing the intercepted operations, if the Downloadable attempts to initiate the operations.

69. The system of claim 60, wherein the MPC comprises:

an MPC installer for causing MPC elements to be installed;

15 a Downloadable installer communicatively coupled to the MPC element installer for installing the Downloadable;

a resource access diverter communicatively coupled to the MPC installer for causing the Downloadable operations to be intercepted;

a resource access analyzer communicatively coupled to the MPC installer for receiving an intercepted Downloadable operation and determining a protection policy corresponding to the intercepted Downloadable operation; and

20 a policy enforcer communicatively coupled to the resource access analyzer for processing the intercepted Downloadable operation.

70. The system of claim 69, wherein the resource access diverter modifies one or more elements of an interface usable by the Downloadable to effectuate the Downloadable operations.

5

71. The system of claim 69, wherein the mobile code executor is a Java Virtual Machine.

72. The system of claim 69, wherein the mobile code executor is an operating system, running native code executables.

10

73. The system of claim 69, wherein the mobile code executor is an ActiveX subsystem of the windows operating system.

74. The system of claim 69, wherein the mobile code executor is an Microsoft Windows scripting host.

15

75. A system, comprising

receiving means for receiving mobile protection code ("MPC") and a

Downloadable at a Downloadable-destination;

20

monitoring means for causing, by the MPC, one or more operations

attempted by the Downloadable to be received by the MPC;

second receiving means receiving, by the MPC, an attempted operation of

the Downloadable; and

initiating means for initiating, by the MPC, a protection policy corresponding to the attempted operation.

- 5 76. A computer-readable storage medium storing program code for causing a computer to perform the steps of:

receiving mobile protection code ("MPC") and a Downloadable at a Downloadable-destination;

causing, by the MPC, one or more operations attempted by the

- 10 Downloadable to be received by the MPC;

receiving, by the MPC, an attempted operation of the Downloadable; and

initiating, by the MPC, a protection policy corresponding to the attempted operation.

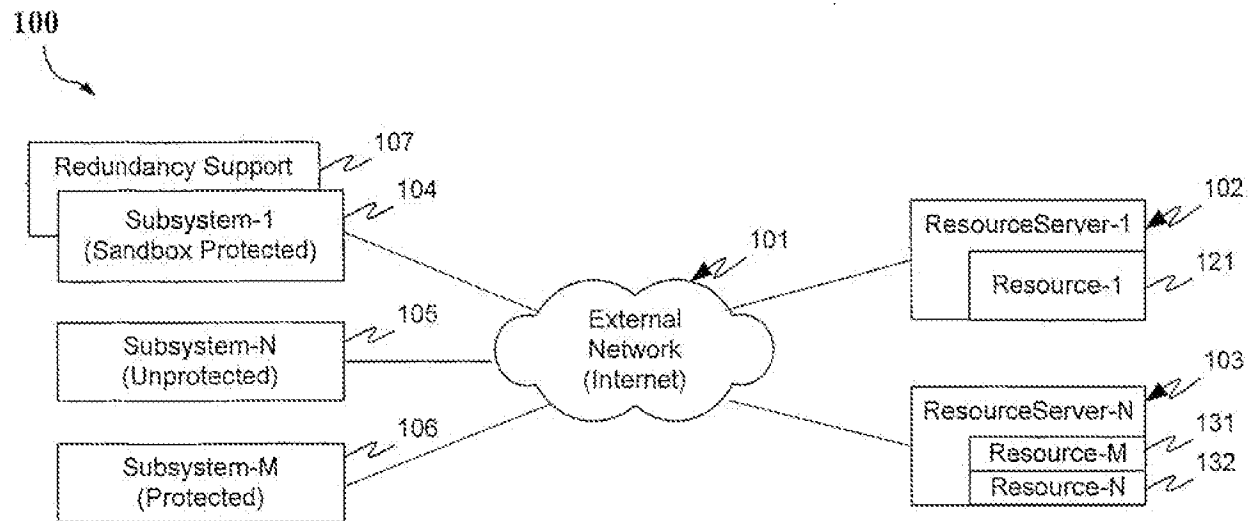


FIG. 1a

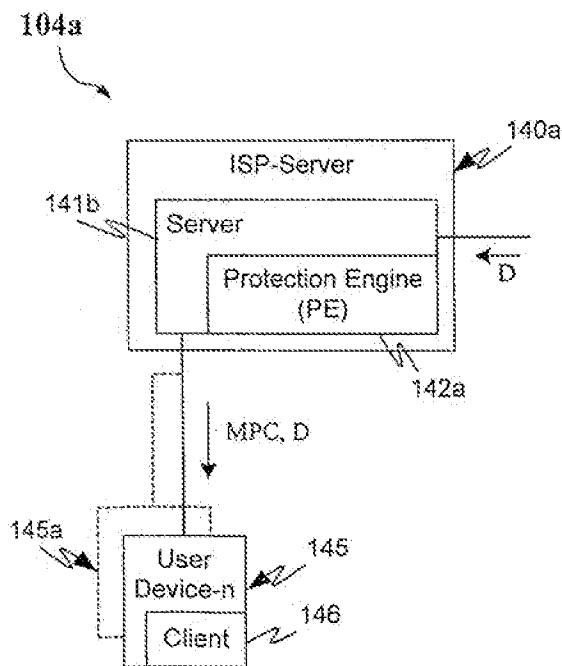


FIG. 1b

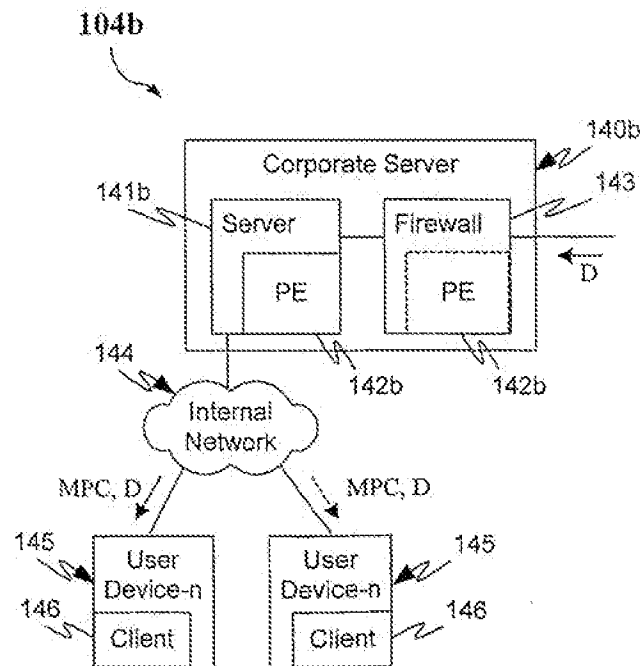


FIG. 1c

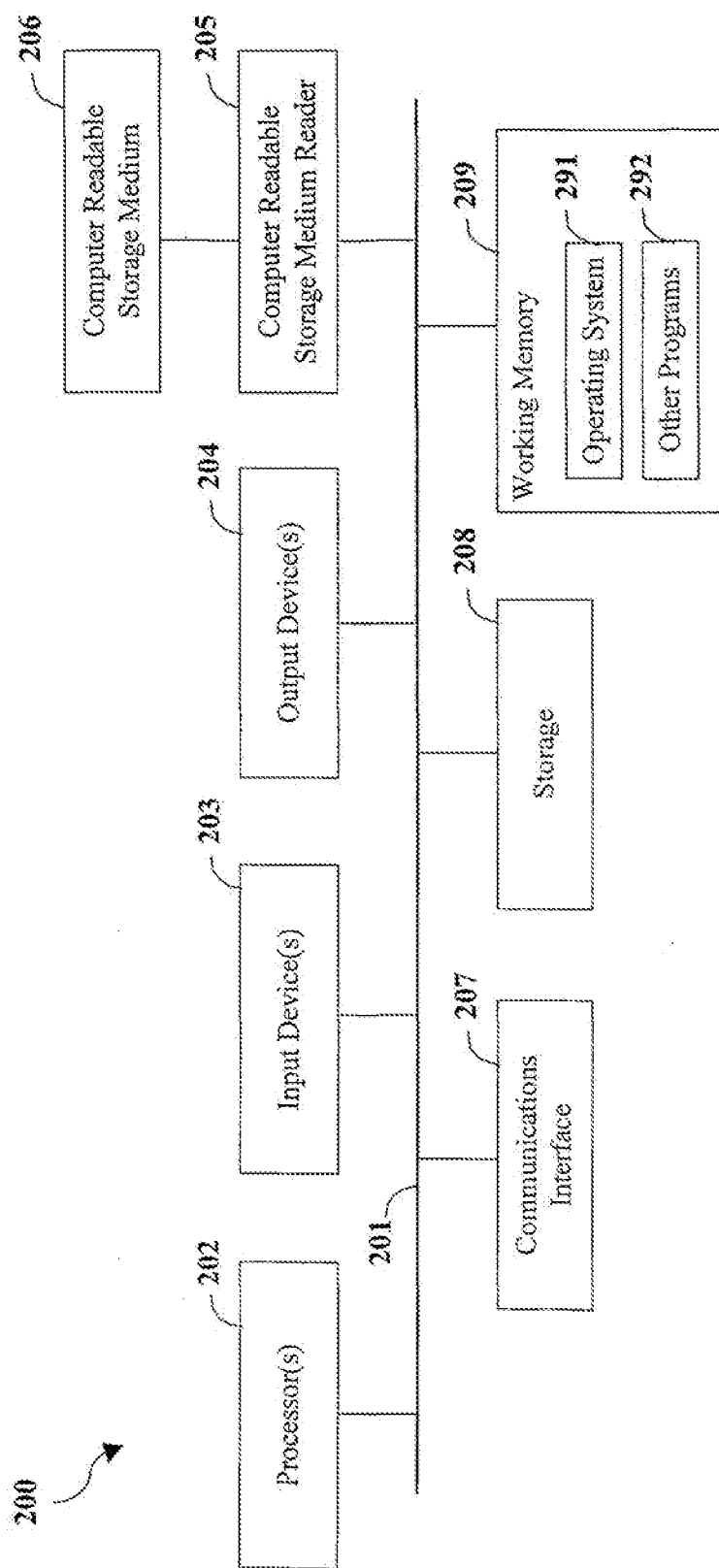


FIG. 2

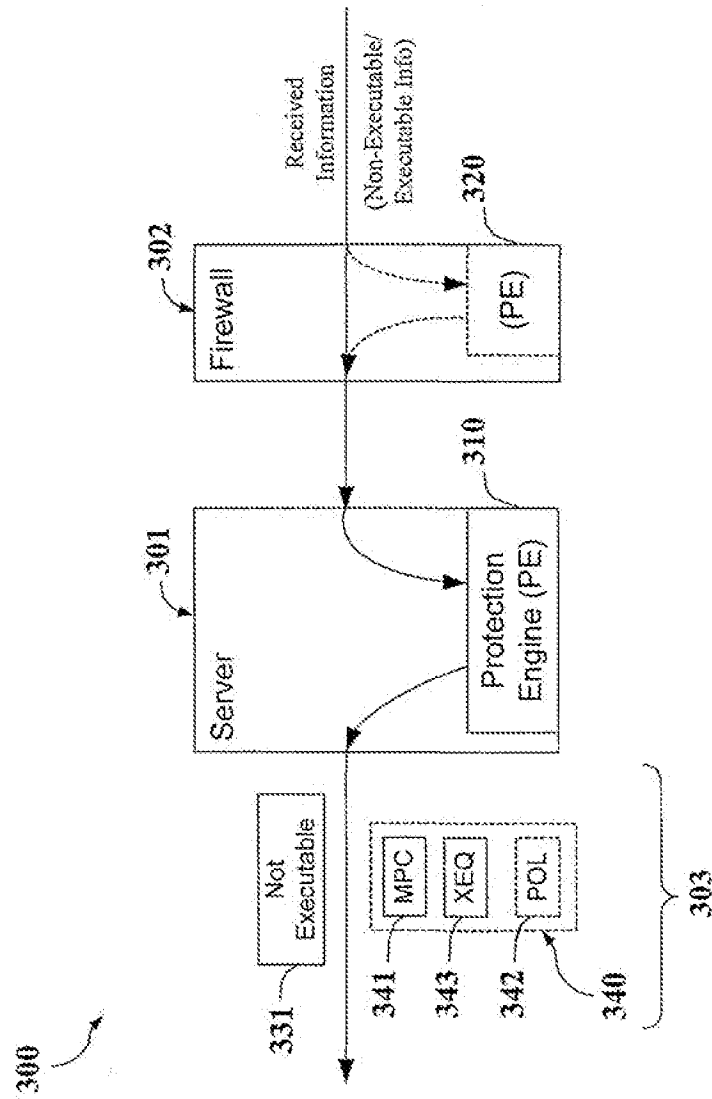


FIG. 3

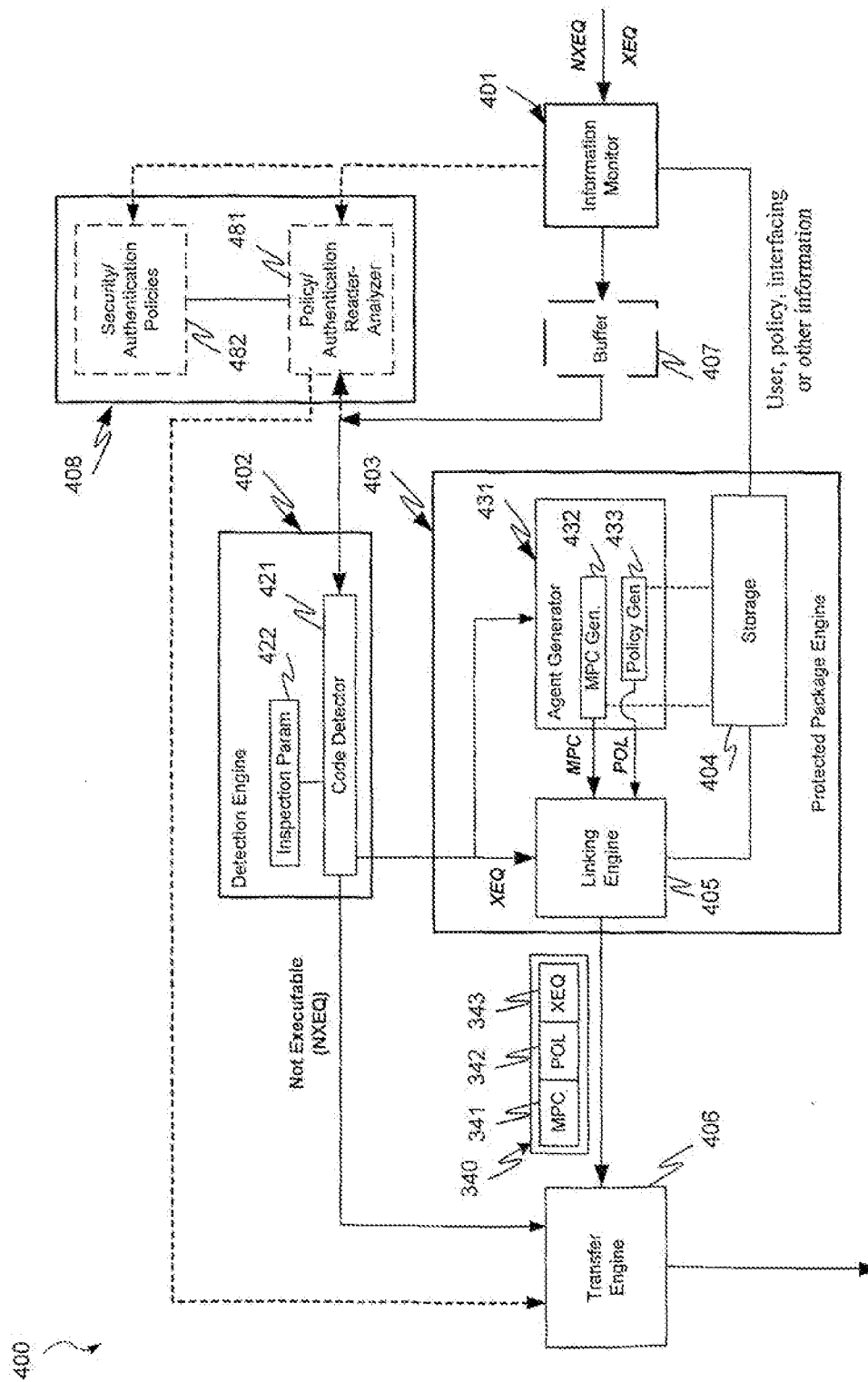


FIG. 4

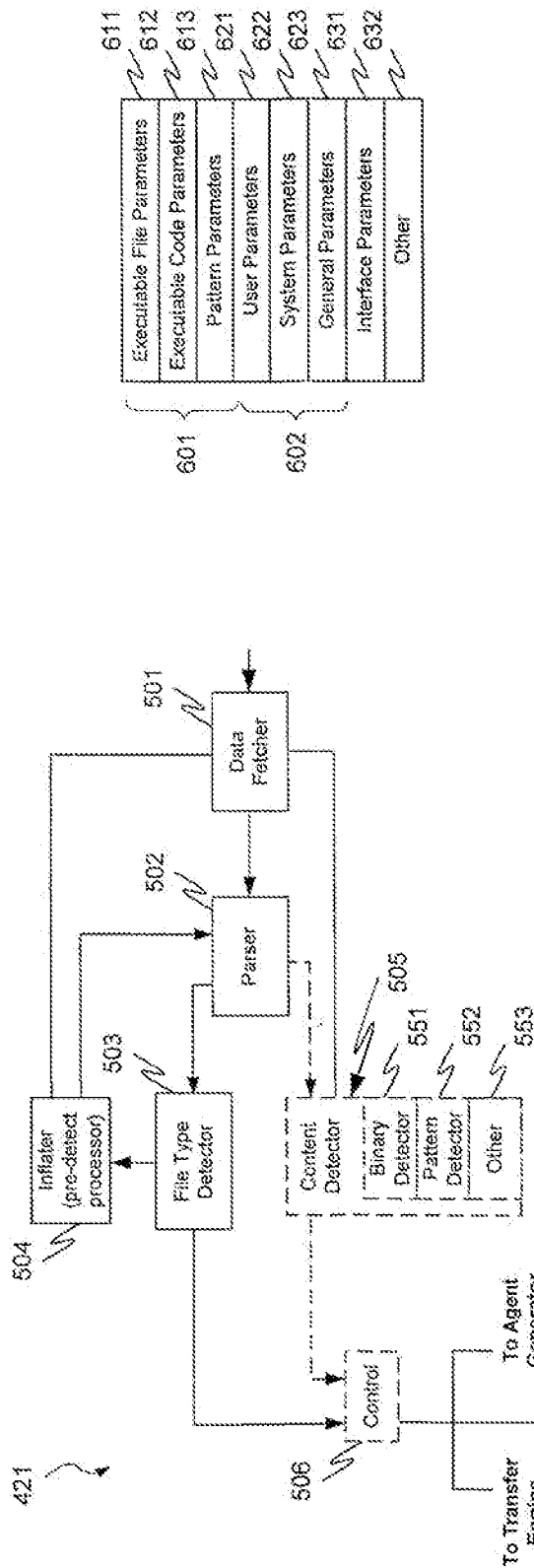


FIG. 6a

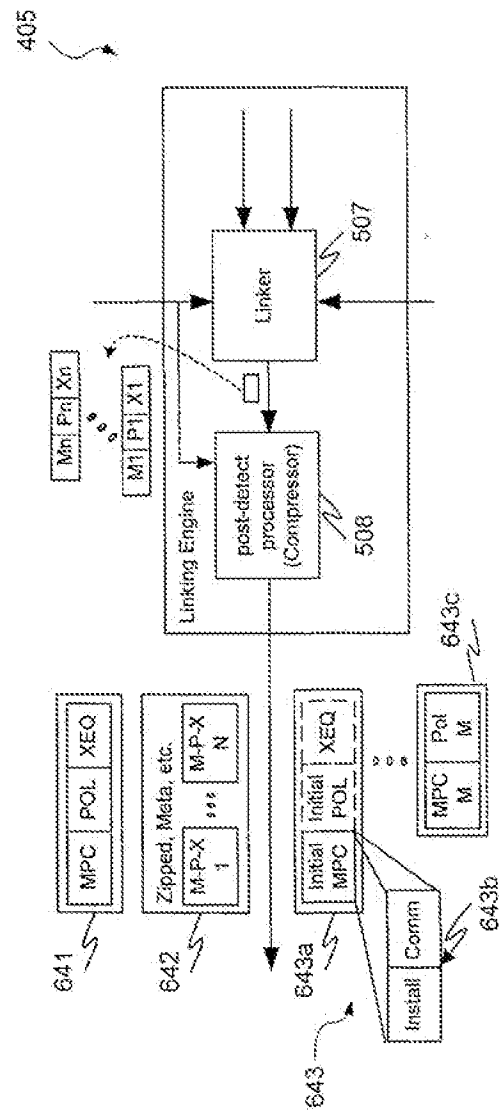


FIG. 6b

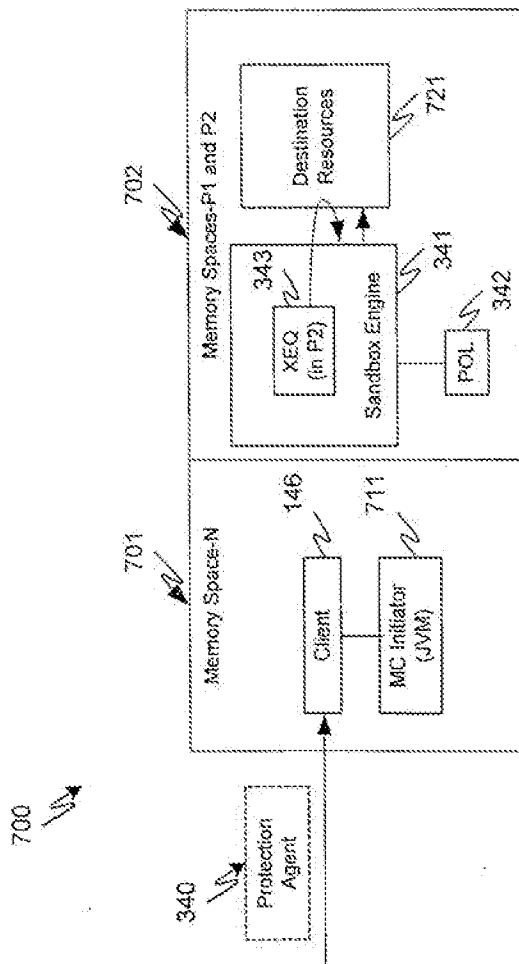


FIG. 7a

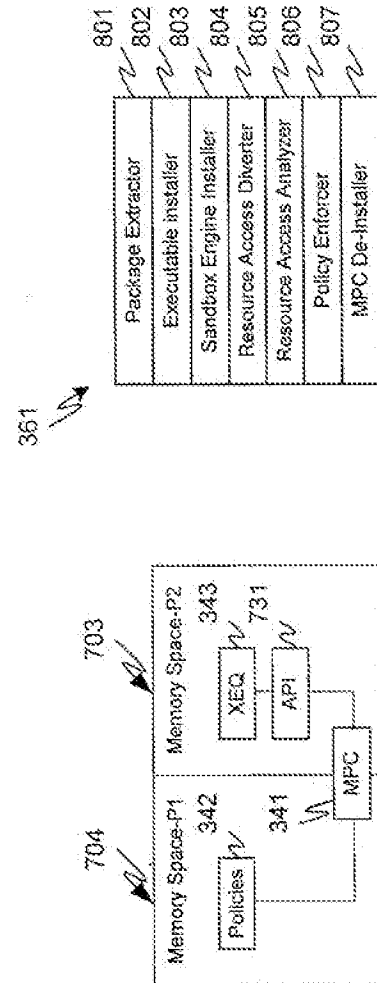


FIG. 7b

FIG. 8

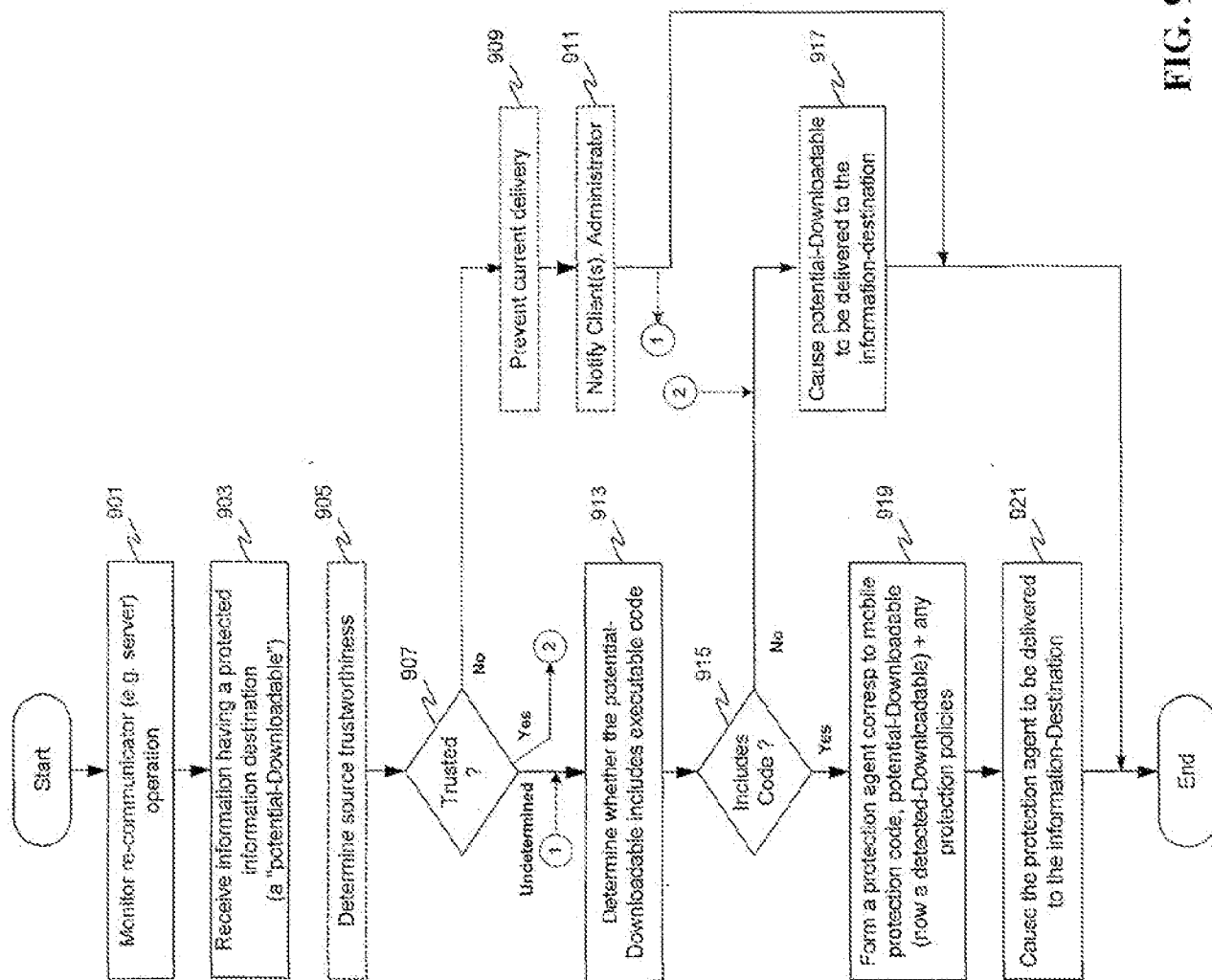


FIG. 9

913

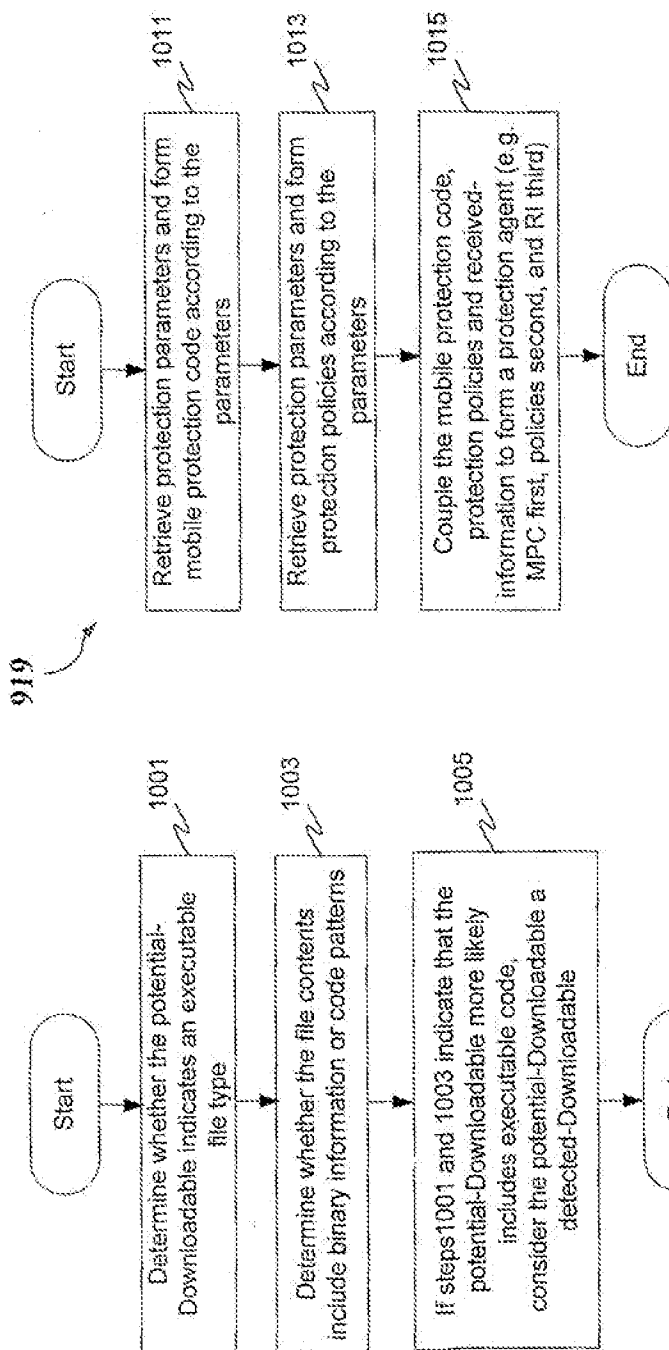


FIG. 10A

919

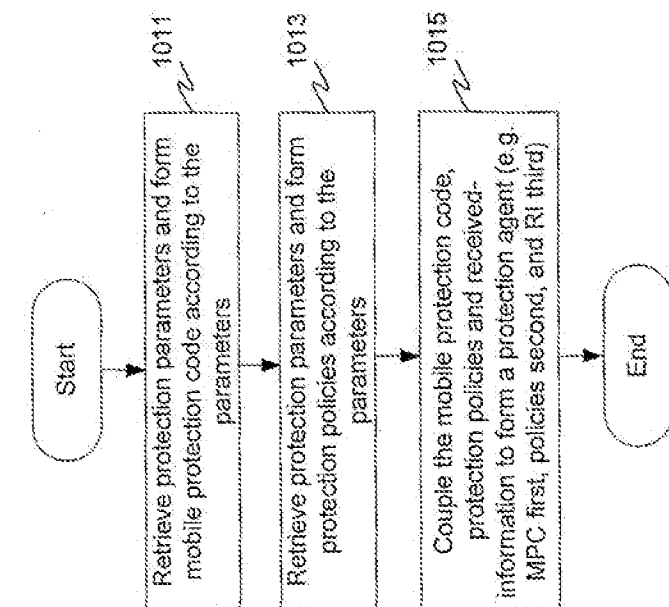


FIG. 10B

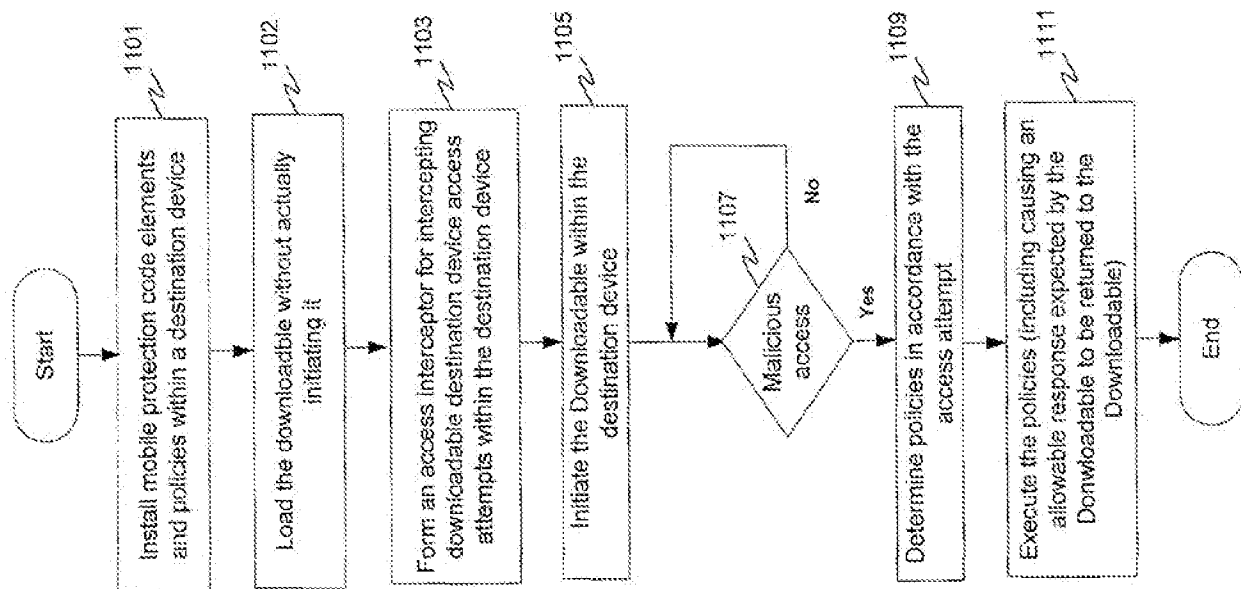


FIG. 11

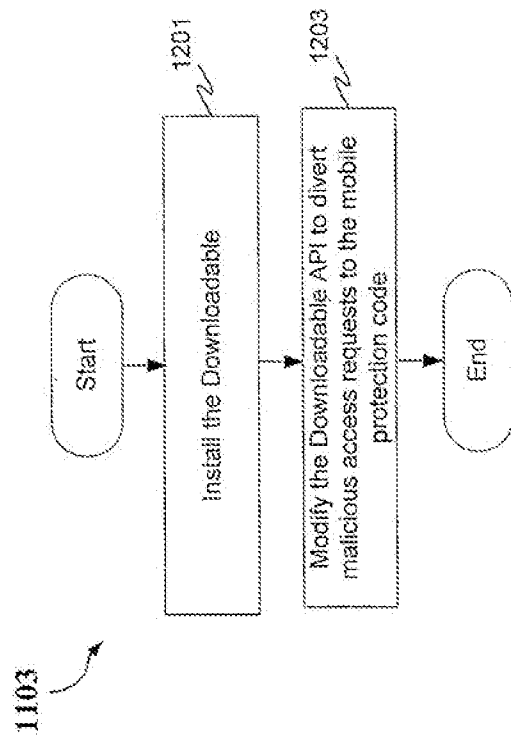


FIG. 12a

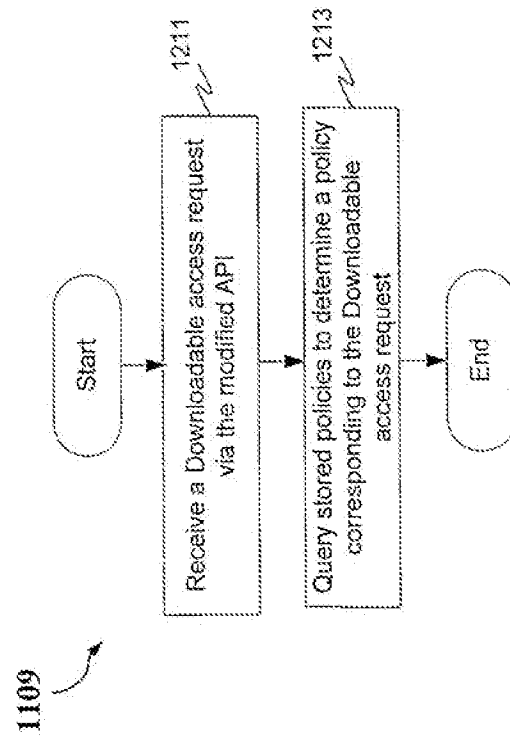


FIG. 12b



Espacenet

Bibliographic data: JP2001256045 (A) — 2001-09-21

METHOD AND DEVICE FOR CHECKING COMPUTER VIRUS

Inventor(s): SATO KAZUHIKO; TERANISHI TOSHIHARU; SOMA FUMIHIKO; USHIGAWA YOKO; TAKEBE EISUKE; TOKUSA YUKINOBU; HOSOKAWA MASANOBU ± (SATO KAZUHIKO, ; TERANISHI TOSHIHARU, ; SOMA FUMIHIKO, ; USHIGAWA YOKO, ; TAKEBE EISUKE, ; TOKUSA YUKINOBU, ; HOSOKAWA MASANOBU)

Applicant(s): NIPPON TELEGRAPH & TELEPHONE ± (NIPPON TELEGRAPH & TELEPHONE EAST CORP)

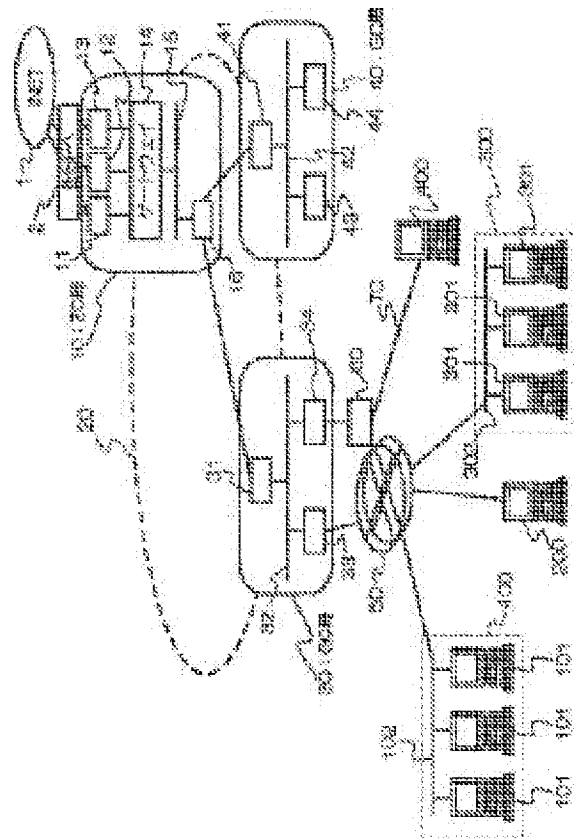
Classification: - **international:** **G06F13/00; G06F21/22; G06F9/06; H04L12/66;**
(IPC1-7): G06F13/00; G06F9/06; H04L12/66
- **cooperative:**

Application number: JP20000071200 20000314

Priority number(s): JP20000071200 20000314

Abstract of JP2001256045 (A)

PROBLEM TO BE SOLVED: To provide a computer virus check system capable of reducing damages of infection due to computer virus via a network. **SOLUTION:** A storage device to temporarily store packet data to be transferred between a ZC station 10 and the Internet 1 and a gateway 14 with a function to assemble plural pieces of the packet data stored in the storage device in file data by making them correspond to IP addresses in the Internet 1 and local IP addresses in the network 20, to judge whether or not the computer virus exist in the assembled data and to transfer the data in which no computer virus is judged to exist to a subscriber terminal are installed for each user ID of each subscriber terminal in the ZC station 10 connected with GC stations 30, 40 to store plural subscriber terminals (101, 200, 301, 400) to be connected with the Internet 1.



(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号
特開2001-256045
(P2001-256045A)

(43)公開日 平成13年9月21日(2001.9.21)

(51)Int.Cl. ⁷	識別記号	F I	データ ⁸ (参考)
G 0 6 F 9/06	5 5 0	G 0 6 F 9/06	5 5 0 Z 5 B 0 7 6
13/00	3 5 1	13/00	3 5 1 Z 5 B 0 8 9
H 0 4 L 12/66		H 0 4 L 11/20	B 5 K 0 3 0
			9 A 0 0 1

審査請求 未請求 請求項の数 6 O L (全 8 頁)

(21)出願番号 特願2000-71200(P2000-71200)

(22)出願日 平成12年3月14日(2000.3.14)

(71)出願人 399040405

東日本電信電話株式会社
東京都新宿区西新宿三丁目19番2号

(72)発明者 佐藤 和彦

東京都新宿区西新宿三丁目19番2号 東日本電信電話株式会社内

(72)発明者 寺西 俊晴

東京都新宿区西新宿三丁目19番2号 東日本電信電話株式会社内

(74)代理人 100064908

弁理士 志賀 正武

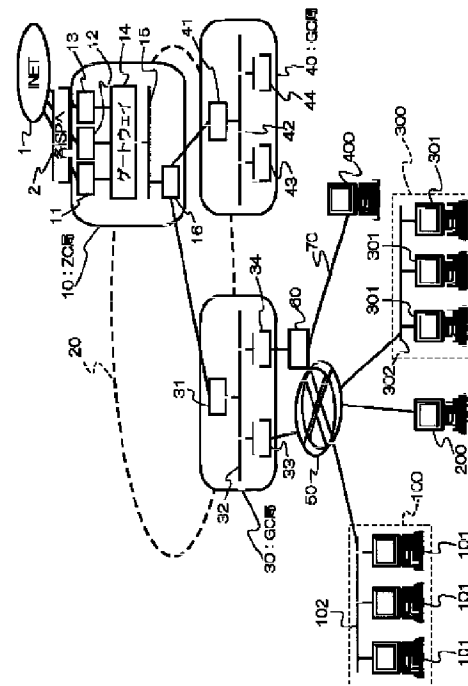
最終頁に続く

(54)【発明の名称】 コンピュータウイルスチェック方法及び装置

(57)【要約】

【課題】 ネットワークを介したコンピュータウイルスによる感染被害を減少させることができるコンピュータウイルスチェックシステムを提供する。

【解決手段】 インターネット1へ接続する複数の加入者端末(101, 200, 301, 400)を収容するGC局30, 40に接続されているZC局10内に、各加入者端末のユーザID毎に、インターネット1との間で転送されるパケットデータを一時記憶する記憶装置と、記憶装置に記憶された複数のパケットデータをインターネット1内のIPアドレスとネットワーク20内の地域IPアドレスに対応させてファイルデータに組み立てて、組み立てたデータ内にコンピュータウイルスが存在するか否かを判定し、コンピュータウイルスが存在しないと判定したデータを加入者端末へ転送する機能を備えるゲートウェイ14を設置する。



【特許請求の範囲】

【請求項1】 複数の第1の網アドレスを用いて通信を行う第1の通信網と、

第1の網アドレスとは割り当て形態が異なる複数の第2の網アドレスのうちのいずれかが割り当てられて、第1の通信網との間で通信を行う通信端末と、

第1の通信網と通信端末との間に介在し、第2の網アドレスを用いて通信を行う第2の通信網とを備える通信システム内のコンピュータウイルスチェック方法であって、

第2の通信網において転送される複数の分割データを、その分割データの転送先又は転送元となる第1及び第2の網アドレスに対応させて一時記憶する第1の過程と、第1の過程で記憶された複数の分割データを第1及び第2の網アドレスに対応させて組み立てる第2の過程と、第2の過程で組み立てられた組立データ内にコンピュータウイルスが存在するか否かを判定する第3の過程と、第3の過程でコンピュータウイルスが存在しないと判定された組立データに対応する分割データを転送先へ転送する第4の過程とを有することを特徴とするコンピュータウイルスチェック方法。

【請求項2】 前記第2の網アドレスが、通信端末の各ユーザに対して割り当てられたユーザ識別符号毎に決定されるものであることを特徴とする請求項1記載のコンピュータウイルスチェック方法。

【請求項3】 前記第1の過程において、前記複数の分割データを一時記憶する際に、分割データの配列と、その分割データの第1の通信網における通信プロトコルの種別とをあわせて記憶することを特徴とする請求項1又は2記載のコンピュータウイルスチェック方法。

【請求項4】 前記第1の過程において、分割データの第1の通信網における通信プロトコルの種別を検知し、検知した種別があらかじめ定めた所定の種別である場合にのみ、前記第2～第4の過程を実行することを特徴とする請求項1又は2記載のコンピュータウイルスチェック方法。

【請求項5】 前記第3の過程でコンピュータウイルスが存在すると判定された場合に、分割データの転送先又は転送元の通信端末へその旨を通知するとともに、コンピュータウイルスが存在すると判定された組立データに対応する分割データ及び組み立てデータに対して、破棄、一時保存、又は、コンピュータウイルスの除去処理を行ってからの転送、のいずれかの処理を行う第5の過程を有することを特徴とする請求項1～3のいずれか1項に記載のコンピュータウイルスチェック方法。

【請求項6】 複数の第1の網アドレスを用いて通信を行う第1の通信網と、

第1の網アドレスとは割り当て形態が異なる複数の第2の網アドレスのうちのいずれかが割り当てられて、第1の通信網との間で通信を行う通信端末と、

第1の通信網と通信端末との間に介在し、第2の網アドレスを用いて通信を行う第2の通信網とを備える通信システム内のコンピュータウイルスチェック装置であって、

第2の通信網において転送される複数の分割データを、その分割データの転送先又は転送元となる第1及び第2の網アドレスに対応させて一時記憶する記憶手段と、記憶手段によって記憶された複数の分割データを第1及び第2の網アドレスに対応させて組み立てる組立手段と、

組立手段によって組み立てられた組立データ内にコンピュータウイルスが存在するか否かを判定する判定手段と、

判定手段によってコンピュータウイルスが存在しないと判定された組立データに対応する分割データを転送先へ転送する転送手段とを備えることを特徴とするコンピュータウイルスチェック装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ネットワークを介して転送されるパケットデータに対してコンピュータウイルスチェックを行うためのコンピュータウイルスチェック方法及び装置に関する。

【0002】

【従来の技術】近年、パーソナルコンピュータの出荷台数が増加し、各個人がパーソナルコンピュータを用いてインターネット等の世界的なネットワークを利用する機会が増加してきている。このためネットワークの利用率が増大し、インターネット等の世界的なネットワークの拡大が進んでいる。このネットワークが拡大にともなう、ネットワークを介してコンピュータウイルスが氾濫する問題が生じてきている。企業内においては、企業内LAN (Local Area Network) の浸透にともない、コンピュータウイルスチェックを、エンドユーザ側の各端末や、LAN内のゲートウェイ等の所定の装置において複数の端末単位で行うようになっている。また、初心者ユーザの急増にともなう、各個人のレベルでは、コンピュータウイルス等のネットワークにおける問題に対する危機管理意識の不足が問題となってきた。

【0003】インターネット等のネットワークへアクセスの際には、公衆網等に接続された端末では、例えば、公衆網 (アクセス網) を介して、ネットワークへの接続を提供するISP (Internet Service Provider) とPPP (Point-to-Point Protocol) 接続することで、IP (Internet Protocol) を用いたネットワークへの接続を行うことが可能となる。この場合、各ユーザは、提供されるサービスの内容が異なる複数のISPのいずれかに加入している。各ユーザのうち、ウイルス対策に対して関心・意識の有るユーザが、ウイルス対策用のソフトウェアを利用して、ユーザ単位でコンピュータウイルス

チェックを実施している。

【0004】

【発明が解決しようとする課題】コンピュータウイルスチェック用のソフトウェア（アンチウイルスソフト）では、コンピュータウイルスの識別情報を定義する定義ファイルの内容を逐次更新するとともに、定義ファイル内のデータを参照することで、新たな、あるいは変化した新種のコンピュータウイルスを発見する。したがって定義ファイルの更新頻度が低くなった場合、最新のウイルスが発見されずに検査から洩れてしまう場合がある。また、通常、受け身的に送られてくるウイルスに対してチェックを行うだけで、能動的にウイルスの発生源追跡までは行わない。また、無意識にウイルス感染したファイルをネットワーク等を介して広めてしまう危険性がある。特に、インターネット内を複数のサーバを介して次々に転送されるインターネットデータの安全性には不安がある。一方、LAN構成のものは、一般的に、メールサーバ上でウイルスチェックのみを行っており、柔軟性がない。

【0005】各個人では、アンチウイルスソフトをパーソナルコンピュータにインストールし、定期的に定義ファイルを更新し、チェックする必要がある。したがって、アンチウイルスソフトの購入費用が必要である。また、アンチウイルスソフトをインストール・設定・不良時の修復等の出来ない人は、ウイルスチェックを行う事ができなかった。さらに、ハードディスクにアンチウイルスソフトおよびその定義ファイル用のインストール領域が必要である。定義ファイル更新は有料化の動向のため、定義ファイル更新の費用が発生する。

【0006】各企業内では、企業内の各人がアンチウイルスソフトをインストールし、定期的に定義ファイルを更新しチェックするか、チェック用のサーバにてチェックする必要がある。したがって、いずれの形態であっても、アンチウイルスソフト購入費用が必要である。社内ネットワーク（NW）に設けられたチェック用のサーバによってチェックする場合でも、社内NWに接続されている全員がそれぞれの端末にアンチウイルスソフトをインストールし、さらに定義ファイル更新しないと無意識にウイルスをNW内でばらまく危険性がある。サーバにてウイルスチェックを実施する場合は、サーバが感染した場合には、全社員の端末がウイルスに感染する危険にさらされる。また、チェック用サーバまたは各端末のハードディスクにアンチウイルスソフトおよびその定義ファイル用のインストール領域が必要である。上記と同様に、定義ファイル更新は有料化の動向のため、定義ファイル更新の費用が発生する。

【0007】本発明は、従来のコンピュータウイルスチェックの方法における上述した種々の課題を解決することを目的とするものであって、より具体的には、従来に比べ、ネットワークを介したデータ転送にともなうコン

ピュータウイルスによる感染被害を減少させることができるコンピュータウイルスチェック方法及び装置を提供することを目的とする。

【0008】

【課題を解決するための手段】上記課題を解決するため、請求項1記載の発明は、複数の第1の網アドレスを用いて通信を行う第1の通信網と、第1の網アドレスとは割り当て形態が異なる複数の第2の網アドレスのうちのいずれかが割り当てられて、第1の通信網との間で通信を行う通信端末と、第1の通信網と通信端末との間に介在し、第2の網アドレスを用いて通信を行う第2の通信網とを備える通信システム内のコンピュータウイルスチェック方法であって、第2の通信網において転送される複数の分割データを、その分割データの転送先又は転送元となる第1及び第2の網アドレスに対応させて一時記憶する第1の過程と、第1の過程で記憶された複数の分割データを第1及び第2の網アドレスに対応させて組み立てる第2の過程と、第2の過程で組み立てられた組立データ内にコンピュータウイルスが存在するか否かを判定する第3の過程と、第3の過程でコンピュータウイルスが存在しないと判定された組立データに対応する分割データを転送先へ転送する第4の過程とを有することを特徴とする。

【0009】請求項2記載の発明は、前記第2の網アドレスが、通信端末の各ユーザに対して割り当てられたユーザ識別符号毎に決定されるものであることを特徴とする。請求項3記載の発明は、前記第1の過程において、前記複数の分割データを一時記憶する際に、分割データの配列と、その分割データの第1の通信網における通信プロトコルの種別とを合わせて記憶することを特徴とする。請求項4記載の発明は、前記第1の過程において、分割データの第1の通信網における通信プロトコルの種別を検知し、検知した種別があらかじめ定めた所定の種別である場合にのみ、前記第2～第4の過程を実行することを特徴とする。請求項5記載の発明は、前記第3の過程でコンピュータウイルスが存在すると判定された場合に、分割データの転送先又は転送元の通信端末へその旨を通知するとともに、コンピュータウイルスが存在すると判定された組立データに対応する分割データ及び組み立てデータに対して、破棄、一時保存、又は、コンピュータウイルスの除去処理を行ってから転送、のいずれかの処理を行う第5の過程を有することを特徴とする。

【0010】請求項6記載の発明は、複数の第1の網アドレスを用いて通信を行う第1の通信網と、第1の網アドレスとは割り当て形態が異なる複数の第2の網アドレスのうちのいずれかが割り当てられて、第1の通信網との間で通信を行う通信端末と、第1の通信網と通信端末との間に介在し、第2の網アドレスを用いて通信を行う第2の通信網とを備える通信システム内のコンピュータ

ウイルスチェック装置であって、第2の通信網において転送される複数の分割データを、その分割データの転送先又は転送元となる第1及び第2の網アドレスに対応させて一時記憶する記憶手段と、記憶手段によって記憶された複数の分割データを第1及び第2の網アドレスに対応させて組み立てる組立手段と、組立手段によって組み立てられた組立データ内にコンピュータウイルスが存在するか否かを判定する判定手段と、判定手段によってコンピュータウイルスが存在しないと判定された組立データに対応する分割データを転送先へ転送する転送手段とを備えることを特徴とする。

【0011】

【発明の実施の形態】以下、図面を参照して本発明によるコンピュータウイルスチェック方法および装置の実施の形態について説明する。図1は、本発明によるウイルスチェックシステムの実施形態の一例を示すブロック図である。図1において、インターネット1は、複数のインターネットサービスプロバイダ（ISP）群2を介して、デジタルネットワーク20内の上階位の交換局であるZC局（ゾーンセンタ局）10に接続されている。ZC局10は、それぞれが複数の加入者端末を収容する下階位の交換局であるGC局（グループセンタ局）30、40および図示していない他の複数のGC局に光ファイバ等のデジタル伝送路（中継線）で接続されている。

【0012】図1に示す構成では、ZC局10は、ISP群2内の各ISPに対してそれぞれ専用線で接続されているルータ（あるいは交換機）11、12、13と、ルータ11、12、13に接続されているゲートウェイ（あるいはゲートウェイサーバ）14と、ネットワーク20側の内部回線網15を介してルータ16に接続されている。GC局30は、ZC局10のルータ16にデジタル中継線を介して接続されているルータ31と、内部回線網32を介してルータ31にそれぞれ接続されているルータ33、34とを備えて構成されている。GC局40は、ZC局10のルータ16にデジタル中継線を介して接続されているルータ41と、内部回線網42を介してルータ41にそれぞれ接続されているルータ43、44とを備えて構成されている。

【0013】GC局30のルータ33は、ISDN（サービス総合デジタル通信網）、高速データ通信網等の交換回線（公衆網）50を介して、例えば1加入者である企業100内でLAN102を介して接続される複数のパーソナルコンピュータ等の通信端末101、101、101に接続されている。交換回線50には、さらに、通信端末200や、SOHO（Small Office Home Office）等の加入者300においてLAN302を介して接続される複数の通信端末301、301、301に接続されている。一方、ルータ34は、光回線終端装置60を介して、光ファイバによって構成されている直通回線（専用線）70で、通信端末400に接続されている。

なお、GC局40の各ルータ43、44も、GC局30の各ルータ33、34と同様に、図示していない複数の加入者回線を収容しているものとする。

【0014】図1の構成において、インターネット1では、グローバルなIPアドレスを用いた通信が行われる。一方、デジタルネットワーク20内では、インターネット1内で用いられるグローバルなIPアドレスとは異なる形態の地域的なIPアドレス（地域IPアドレス）を用いた通信が行われる。地域IPアドレスは、GC局30、40が収容する各加入者回線に対応して設定されるアドレスであって、インターネット1で用いられるグローバルなIPアドレスに対して動的または静的に対応づけられるものである。また、GC局30、40が収容する各加入者回線に対しては、電話番号、ISDN番号等の加入者識別符号（ユーザID）が割り当てられている。したがって、デジタルネットワーク20内では、グローバルなIPアドレスと地域IPアドレスの両方のアドレス情報が、データの転送先または転送元のアドレス情報として、転送される各パケットデータに、1または複数のパケットデータ単位で付与されることになる。

【0015】次に、図3および図4を参照して、図1のゲートウェイ14（あるいは後述する図2示すゲートウェイ35、36）の構成および機能について説明する。図3は、図1に示すゲートウェイ14に対応するゲートウェイ500の基本構成を示すブロック図であり、図4はその動作を示すシステムフロー図である。図3に示すゲートウェイ500は、パケットデータを入力するパケットキャプチャ501と、コンピュータウイルスの定義情報を記憶・管理するウイルスパターンデータベース（DB）502と、ウイルスパターンDB502内のウイルス定義情報を参照して、照合を行い、記憶装置504に記憶されているファイルデータが、定義済みの既知のウイルスに感染しているかどうかを確認する既知ウイルスチェッカ503と、パケットキャプチャ501から入力されたパケットデータ、およびパケットデータをパケット分割される前の形式に組み立てた（復元した）ファイルデータを一時保存する記憶装置504と、未知ウイルス類推DB506に記憶・管理されている未知ウイルスを類推するための情報を参照して、照合し、記憶装置504に記憶されているファイルデータが、未知のウイルスに感染しているかどうかを類推・確認する未知ウイルスチェッカ505と、未知ウイルス類推DB506と、記憶装置504に保存されているパケットデータを転送するパケット転送（部）507とから構成されている。

【0016】上記構成において、パケットキャプチャ501は、図1の構成においては、ルータ11～13あるいはルータ16から転送されてきたパケットデータを入力し、パケット転送507は、図1の構成においては、

ルータ 16あるいはルータ 11～13へパケットデータを転送する。その際、パケットキャプチャ 501は、入力された複数のパケットデータを、そのパケットデータのパケット配列と、そのパケットデータの送信先および送信元 IP アドレスと、インターネットにおけるそのパケットデータの通信プロトコルと、各加入者線（図 1 の符号 100, 200, 300, 400）のユーザ ID とに対応づけて、記憶装置 504 に保存する。ここで記憶される通信プロトコルには、少なくとも、例えば、WWW（World Wide Web）のクライアントとサーバが通信を行うためのプロトコルである HTTP（HyperText Transfer Protocol）、ファイル転送を行うためのプロトコルである FTP（File Transfer Protocol）、電子メールをダウンロードするためのプロトコルである POP（Post Office Protocol）、電子メールを送信するためのプロトコルである SMTP（Simple Mail Transfer Protocol）、その他動画配信に使用される各種プロトコル等のアプリケーション層における情報転送の制御を定義するもの等である。また、既知ウイルスチェッカ 503 および未知ウイルスチェッカ 505 は、ウイルスチェックを行う際に、各ファイルの通信プロトコルを確認して、例えば、電子メールに対してはその本文と添付ファイルに対してウイルスチェックを行うが、動画配信ファイルに対してはウイルスチェックを行わない、というような通信プロトコルによるウイルスチェックの実行・非実行を選択する選択機能を有している。

【0017】次に、図 4 を参照して、ゲートウェイ 500 の動作について説明する。図 4 は、インターネット 1 内の図示していないサーバから、各加入者（図 1 の符号 100, 200, 300, 400）へパケットデータを転送する場合の処理を示したシステムフロー図である。まず、パケットキャプチャ 501 において、転送されてきたパケットデータの入力処理を行って、各パケットデータに対応する、ユーザ ID、パケット配列、通信プロトコルを確認するとともに（S601）、半導体メモリ等のバッファに一時保存した後、ユーザ ID 単位で記憶装置 504 に保存する（S602）。次に、記憶装置 504 に記憶されたパケットデータが、パケットデータに付与されている転送元、転送先の IP アドレスと、ユーザ ID とに対応させて、かつ、そのパケットデータのパケット配列を参照することで、画像ファイル、実行ファイル、スクリプトファイル等の所定のファイル形式のデータに組み立てられる（S603）。次に、既知ウイルスチェッカ 503 および未知ウイルスチェッカ 505 が、組み立てられたファイルに対してウイルス照合処理を行って、ウイルスに冒されているか否かを確認する（S604）。ここで、ウイルスに冒されていないと確認されたファイルに関しては、対応するパケットデータが、パケット転送 507 において、ユーザ ID 先へとパケット転送される（S605）。一方、ウイルスに冒さ

れていると確認されたファイルに関しては、ユーザ毎にあらかじめ登録されている処理の設定内容に従って、そのファイルに対して強制破棄、一時保存、ウイルス除去のいずれかのユーザ設定処理が行われる（S606）。強制破棄を行う場合には、ファイルを破棄するとともに、ユーザ ID 先へファイルを破棄した旨の通知が行われる（S607）。一時保存を行う場合には、そのファイルを記憶装置 504 内でそのまま一時保存するか、あるいは他の領域に一時保存し直すとともに、ユーザ ID 先へファイルを一時保存した旨の通知が行われる（S607）。この場合、ユーザ ID 元からの指示に従って、破棄やウイルス除去および転送等の処理がさらに行われる。そして、ウイルス除去を行う場合には、ウイルス除去を行った後、ユーザ ID 先へウイルス除去を行った旨の通知が行われ（S609）、続いて、ユーザ ID 先へ除去後のファイルデータがパケット転送される（S610）。

【0018】以上の動作によって、図 3 のゲートウェイ 500 によれば、例えば図 1 に示すシステムにおいては、地域 IP を利用して ISP 群 2 を介してインターネット 1 へ接続する各加入者 100, 200, 300, 400 に対して、インターネット 1 から各加入者 100, 200, 300, 400 へパケット転送されるファイルデータに対して、デジタルネットワーク 20 内でウイルスチェックを行うサービスを提供することが可能となる。なお、各加入者（図 1 の符号 100, 200, 300, 400）からインターネット 1 へパケット転送されるファイルデータに対してウイルスチェックを行う場合の動作は、図 4 において、処理 S605 および処理 S610 のパケットデータの転送先がインターネット 1 内のサーバあるいはクライアントに変更される。

【0019】次に、図 2 を参照して、図 1 に示すウイルスチェックシステムの実施形態の変形例について説明する。図 1 と図 2 で同一の構成については同一の符号を付けて説明を省略する。図 2 に示すウイルスチェックシステムの実施形態では、ZC 局 10a（図 1 の ZC 局 10 に対応）内にウイルスチェック機能を有するゲートウェイ（図 1 のゲートウェイ 14 相当のもの）を設けるのではなく、GC 局 30a, 40a（図 1 の GC 局 30, 40 に対応）その他のすべての GC 局内に、ウイルスチェック機能を有するゲートウェイ 35, 45 を追加して設けている。ゲートウェイ 35, 45 は、そのウイルスチェックのための基本構成が図 3 にゲートウェイ 500 として示されるものであって、ゲートウェイ 35 がルータ 31 と内部回線網 32 との間に設けられていて、ゲートウェイ 45 がルータ 41 と内部回線網 42 との間に設けられている。図 2 に示す実施形態では、デジタルネットワーク 20 内の各 GC 局すべてにウイルス対策用のゲートウェイを設置するので、デジタルネットワーク 20 に接続されるすべてのユーザに対して、ウイルスチェッ

インストール用の記憶領域が軽減あるいは不要となる。また、NW外部からのウイルスに対してはウイルスチェックや定義ファイルの更新の必要がなく、ウイルスチェックしないLAN内ユーザも感染の恐れがなく、また、知らず知らずのうちに感染源となる可能性も低くなる。NWからのウイルス侵入はチェック済みのため、FD等の媒体からの侵入の場合を除き、アンチウイルスソフトのインストール必要性がなく、サーバも領域も不要となる。

【００２２】
【発明の効果】本発明によれば、第１の通信網（インターネット１）と通信端末との間に介在し、第２の網アドレス（地域ＩＰアドレス）を用いて通信を行う第２の通信網（デジタルネットワーク２０）において、コンピュータウイルスのチェックを行うことができるので、従来に比べ、ネットワークを介したコンピュータウイルスによる感染被害を減少させることが可能となる。

【図面の簡単な説明】
【図１】 本発明によるコンピュータウイルスシステムの
実施形態のシステム構成を示すブロック図である。
【図２】 本発明によるコンピュータウイルスシステム
の他の実施形態のシステム構成を示すブロック図であ
る。

【図4】 図3のゲートウェイ500の動作を説明するためのシステムフロー図である。

1 インターネット
10 ZC局
20 デジタルネットワーク
30, 40, 30a, 40a GC局
14, 35, 45, 500 ゲートウェイ（ウイルスス
チェック機能付き）

501 入力

502 ウイルスパターンDB

503 既知ウイルスチェッカ

504 パケットキャプチャ

505 ファイル相立一時保存

506 未知ウイルスチェッカ

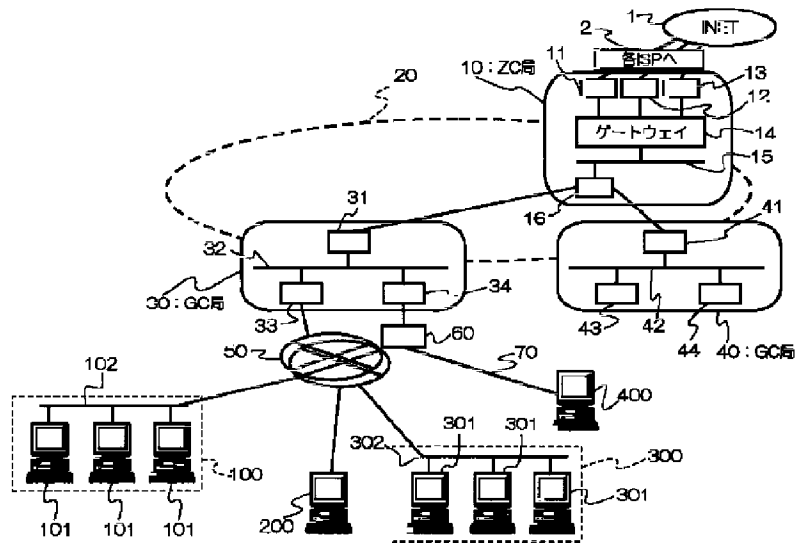
507 未知ウイルス検出DB

508 パケット転送

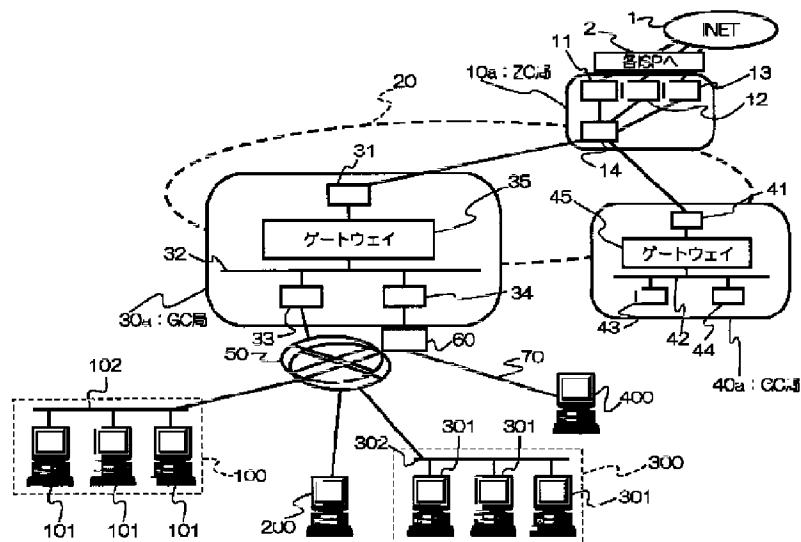
509 出力

500(14, 35, 45): ギートウェイ

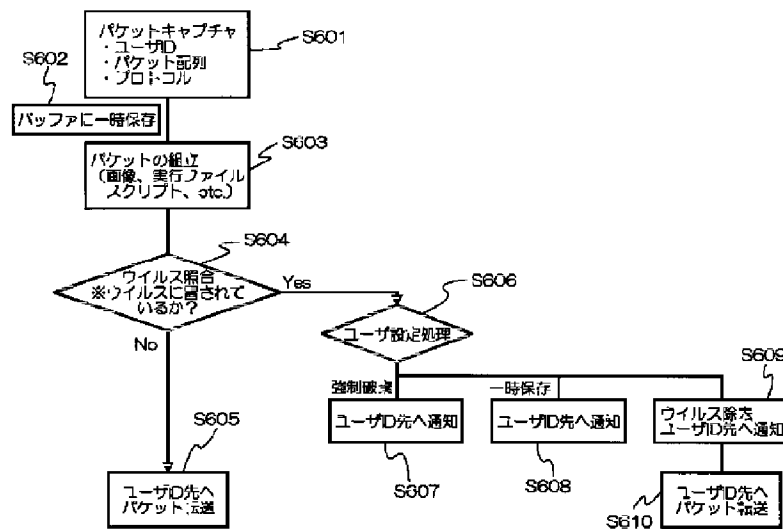
【 図 1 】



【图2】



【図4】



フロントページの続き

(72)発明者 相馬 文彦
東京都新宿区西新宿三丁目19番2号 東日
本電信電話株式会社内
(72)発明者 牛川 陽子
東京都新宿区西新宿三丁目19番2号 東日
本電信電話株式会社内
(72)発明者 建部 英輔
東京都新宿区西新宿三丁目19番2号 東日
本電信電話株式会社内

(72)発明者 徳佐 幸展
東京都新宿区西新宿三丁目19番2号 東日
本電信電話株式会社内
(72)発明者 細川 真伸
東京都新宿区西新宿三丁目19番2号 東日
本電信電話株式会社内
Fターム(参考) 5B076 FD08 FD09
5B089 GA31 GB01 KA17 KB13 KC47
5K030 GA12 HD03 JA10 KA02 LD11
9A001 CC03 CC06 CZ08 JJ25 LL04



(12) **EUROPEAN PATENT APPLICATION**

(43) Date of publication:
 12.09.2001 Bulletin 2001/37

(51) Int. Cl.: **G06F 1/00**

(21) Application number: **00104966.7**

(22) Date of filing: **08.03.2000**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU
MC NL PT SE
 Designated Extension States:
AL LT LV MK RO SI
 (71) Applicant: **UNIVERSITE CATHOLIQUE DE**
LOUVAIN
1346 Louvain la Neuve (BE)

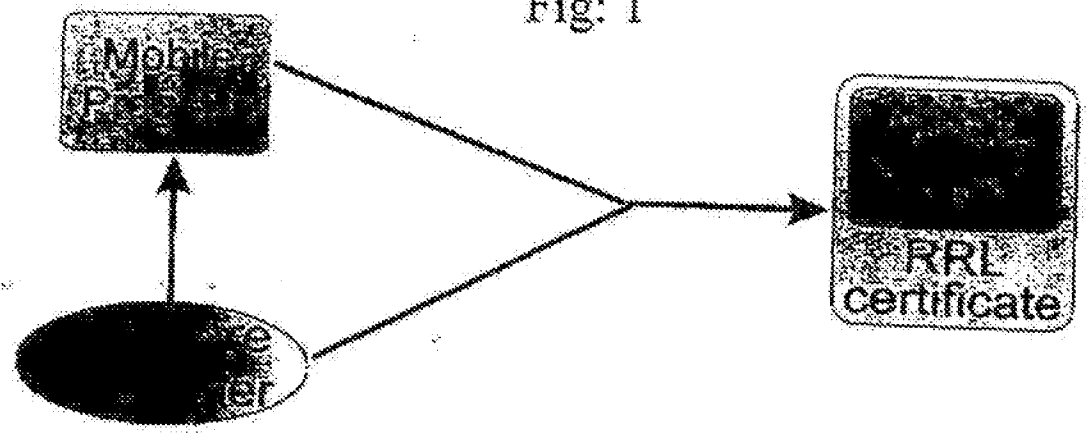
(72) Inventor: **Mas Ribés, Joan-Maria**
43206 Reus (ES)
 (74) Representative:
Kirschner, Klaus Dieter, Dipl.-Phys.
Schneiders & Behrendt
Rechtsanwälte - Patentanwälte
Sollner Strasse 38
81475 München (DE)

(54) **Mobile code and method for resource management for mobile code**

(57) A mobile code linked to a certificate including at least a resource requirements list (RRL) including those resources needed by the mobile code to be properly executable plus those resources that are known a priori to be accessed when executing the mobile code. The unique certificate contains additionally an issuer of the certificate information identifying the entity issuing the certificate, a subject information identifying the mo-

bile code to which the certificate is referred, and a validity period information stating the period of time within which the certificate is valid. When downloading or uploading a mobile code the RRL is transferred to the user informing the user of the resource requirements of the mobile code. An execution environment is provided in an execution peer of the user, the execution environment defining at least the resource access policy that will be applied to the mobile code on execution.

Fig: 1



Description

[0001] This invention relates to a mobile code and method for resource management for mobile code.

[0002] Nowadays, with the recent explosive growth of the Internet, the number of computer interconnected in a global communications network grows exponentially. Many view the Internet as a universal communications medium that can replace telephone, television and radio. The potential is there, but progress has been hampered by the open design of the network itself. It is still too easy to intercept, monitor and forge messages on the Internet, and people are reluctant to use the network for financially or legally sensitive data.

[0003] Computer networks are evolving at a very fast pace, and this evolution proceeds along several aspects. Network links are constantly improved, and technological developments lead to increased computational power in network nodes. With increase in size and performance of computer networks, network connectivity has become a basic feature of computers and many products in the consumer electronics industry. On the other hand, users can exploit network connectivity independently of their physical location. In this new scenario, mobile users can move together with their hosts across different locations and still find their working environment.

[0004] The problems faced by users of the Internet fall into two main categories: privacy and authentication. Privacy involves transmitting messages that cannot be altered or read en route, while authentication allows each party to a communication to be sure of the identity of the other. Cryptography holds the promise of a solution to these problems.

[0005] These developments lead to a widespread diffusion of services and applications, making it necessary to increase the customizability of services. Thereby, different classes of users are then enabled to tailor the functionality and interface to a service according to their specific configuration, needs and preferences. Finally, the dynamic nature of both the underlying network infrastructure and market requirements demand higher levels of extensibility and flexibility.

[0006] There exist already a number of patent publications related to security aspects and authorizations for mobile programs. The systems described in these patent publications have, however some serious drawbacks. First, whenever certification is used, the systems require the existence of a hierarchic certification infrastructure in place. Second, all the systems deal with authorization. And finally, these patent publications all talk about low-level resource access such as file permissions, program execution, and network access. Some examples of these patent publications are discussed below.

[0007] The US 5,412,717 A relates to a computer system security method and apparatus having program authorization information data structures. The authoriza-

tion information is about low level resource access at operating system level. The only external resources available are the possibility to call another executable. Furthermore, the system needs to be implemented at an operating system level. The invention states that if all authorizations defined in the "Program Authorization Information" are not granted, the program can not be executed.

[0008] The US 5,882,904 A relates to a code certification for network transmission. A system is described to support the distribution of software over networks or off-line along with an Access Control List (ACL) for the program itself and a certificate that 'makes' the program secure for execution. In this case, the code production system submits the program and the ACL for the program to a certification authority, which sends back a certificate on the code and another one on the ACL for the program. At distribution time, the code is sent along with the ACL, the certificate on the code (which in fact is more a signature than a certificate) and another certificate on the ACL (again, this is more a signature by a CA over the ACL than a certificate). The contents of the ACL define the rights and authorizations a program has. In case not all of these authorizations are granted by the executing system or user, the program cannot run.

[0009] The US 5,892,904 A shows a system for certifying executable objects. The patent deals exclusively with program certification for network transmission. This certification guarantees program integrity, gives descriptive information on the program itself and information on the entity that certifies the program. This patent does not deal with any kind of authorization nor resource access.

[0010] The US 5,919,247 A relates to a method for the distribution of code and data updates over any network. Applications are seen as channels that can be subscribed to and updated. Whenever a user subscribes to a channel, the associated application is downloaded to the local disk and can be executed any number of times. On the other hand, there is the possibility to define an updating rate in which applications will be updated if necessary. This method basically deals with software downloading and updating and lacks, however, some important aspects on software downloading such as security and payment/licensing.

[0011] The US 5,978,484 A describes a system in which code to be sent through the network is associated with a "privilege request code", i.e. a set of rights the code may exercise, and digitally signed by a certification authority. A run-time system prevents the code from exercising unauthorized accesses. A certification hierarchy needs to be in place so that the user or executing system can verify the certificate attached to the program. The first drawback of the system is that it assumes the existence of a certification hierarchy in such a way that any user on the network can verify the validity of a given certificate. Such an infrastructure is not in place nowadays and will most likely never exist. On the other

hand, it makes the distributing authority dependent on a certification authority, which is a strong requirement.

[0012] There are also a number of scientific publications dealing mobile code handling. Examples are: D. Balfanz and L. Gong, "Experience with Secure Multi-Processing in Java", Technical Report, Princeton University, September 1997; and G. Back and W. Hsieh, "Drawing the Red Line in Java", In Proceedings of the 7th Workshop on Hot Topics in Operating Systems, March 1999, IEEE Computer Society; and G. Back, P. Trullmann, L. Stoller, W.C. Hsieh and J. Lepreau, "Java Operating Systems: Design and Implementation", Technical Report UUCS-98-015, University of Utah, August 1999; and G. Czajkowski and T. von Eicken, "JRes: A Resource Accounting Interface for Java", In Proceedings of the 1998 ACL OOPSLA Conference, Vancouver, BC, October 1998; and L. Gong, M. Mueller, H. Pralulchandra and R. Schemers, "Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2", In Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, CA, December 1997; and T. Tock, D. Sturman and R. Campbell, "Security, Delegation, and Extensibility", Technical Report, University of Illinois, November 1994; and T. von Eicken, C. Chang, G. Czajkowski, C. Hawblitzel, D. Hu and D. Spoonhower, "J-Kernel: a Capability-Based Operating System for Java", To appear in Secure Internet Programming: Security Issues for Distributed and Mobile Objects, Springer-Verlag Lecture Notes in Computer Science, 1999; and D. S. Wallach, D. Balfanz, D. Dean and E.W. Felten, "Extensible Security Architectures for Java", In Proceedings of the 16th Symposium on Operating Systems Principles, October 1997, Saint-Malo, France.

[0013] A few years ago, Java, developed by Sun Microsystems, triggered most of the attention and expectations on code mobility. Being able to run on any platform, Java has become a preferred research and development language for code mobility. Since then, most code mobility research literature refers to Java even if the paradigms, methodologies or concepts exposed are general and independent of any language. The Java 1.2 approach to the security of mobile code is focused exclusively on control access to resources on the machine onto which the application is executed. Classes are grouped in domains defined on the basis of the origin of the code. The address of the server providing the code or the public key associated with the signature over the code define such domains. A user can then associate to each domain an access control list containing the resources made available to classes within a domain. The Java runtime maintains a mapping from objects to their protection domains and then to their permissions. Each resource manages the permissions by itself. Nevertheless, it has some important drawbacks. Precisely, privileges are assigned to code based on their origin and totally independent of the application it implements. There is no support for resource accounting, monitoring

or reclamation, which are required from a system point of view. Furthermore, mobile code usually requires awareness of the location it is executed, and the resources and its state available to it.

[0014] Another totally different approach to resource management comes from research carried out in the past in the field of operating systems applied to type-safe languages such as Java. Type-safe languages provide the same functionality as a MMU (memory management unit) in classical operating systems, but within a single address space. The MMU is in charge of isolating address-spaces of different processes running on the same machine, and user and kernel execution modes.

[0015] Operating systems implemented with type-safe languages have several advantages over traditional operating systems with hardware-based MMU. One of the most time expensive operations on computers is context switching between user and kernel mode. These switches occur every time a user-space application makes a system call. Any operations on the file system, network access or keyboard read causes produces a context switch. Type-safe languages prevent from accessing variables or objects in an illegal way, opposed to the possibility in other languages like C/C++, in which one can access and modify the processes' memory. This feature makes unnecessary the use of MMU and boosts the performance of the system by avoiding context switching.

[0016] However, the concept of operating system limits the possibilities of such systems. The different prototypes deal exclusively with fair allocation of resources to different processes running on a machine, and provide applications with different ways to manage these resources. They lack, nevertheless, the possibility to externally define the resources available to an application.

[0017] Code mobility is exploited on an Internet scale, conceived to operate in large-scale settings with heterogeneous hosts connected by links of different bandwidths. This conception is opposed to distributed systems providing object migration that have been designed having in mind small-scale networks with high bandwidths. Mobile code is location and environment-aware and it takes actions based on such knowledge. Nevertheless, mobile code has some non-negligible risks regarding its security. A program going from computer to computer with the same privileges for the provider and the user is a non-acceptable risk for system administrators and users. Unless some precautions are taken, mobile code could read files, delete them, access the network impersonating the user or abuse of any of the resources the user has access to.

[0018] In view of the above, it is an object of the invention to provide a secured and scalable resource management at user level when using the code.

[0019] For achieving the above object, a mobile code comprises a resource usage needs section containing at least a resource requirements list (RRL) including

those resources needed by the mobile code to be properly executable plus those resources that are known a priori to be accessed when executing the mobile code. The invention provides a secure resource management for mobile code on the receiving and executing peer. A programmer or software provider/distributor attaches a RRL containing a description of the resources required by the application in order to correctly run. This information is a list of the different resources the mobile code will eventually access. The semantics of this Resource Requirements List is "the programmer of this mobile code states that the application needs to access the resources in the RRL". The goal of the RRL is not to transfer authorization but to provide a basis for the resource management.

[0020] According to a preferred aspect of the invention, the resource usage needs section of the mobile code is a certificate which is unique for each different mobile code. Out of security reasons, it is preferred to include the RRL in a certificate linked to the mobile code. For example the "most important" certificate is the certificate which is attached, for example, via a soft link by means of a hash function on the mobile code. The RRL can be contained in this certificate.

[0021] According to a preferred aspect of the invention, the resource usage needs section of the mobile code contains, in addition to the resource requirements list, any of the following information: a) issuer of the certificate information identifying the entity issuing the certificate, b) subject information identifying the mobile code to which the certificate is referred, and c) validity period information stating the period of time within which the certificate is valid. Any of this information subjects serve to further improve the ability of the system to manage resources.

[0022] According to a further preferred aspect of the invention, the information as to the issuer of the certificate is a digest of the public key of the entity having produced the mobile code. By using a digest of the public key of the entity having produced the mobile code, the safety of this information is further improved as it is made more difficult to forge the identity of this entity.

[0023] According to a further preferred aspect of the invention, the information as to the issuer of the certificate is a public key of the entity having produced the mobile code. Using the public key as an identification of the entity having produced the mobile code along with the signature on the certificate, identifies and authenticates the producer and gives a high level of security to this identification information.

[0024] According to a further preferred aspect of the invention, the subject information is a hash of the mobile code. To use a hash of the mobile code as subject information ensures again a high level of security in relation to this information. As security is an important aspect in the handling of mobile code, the importance of the last mentioned aspects of the invention is substantial.

[0025] According to a further preferred aspect of the invention, the resource requirements' list contains any of the following information: a) name of the resource information specifying the type of resource, b) action on the resource specifying as to how the resource should be used, c) upper quantitative limit information stating the maximum usage of the resource from a quantitative point of view, and d) upper qualitative limit information stating the maximum usage of the resource from a qualitative point of view.

[0026] The more information is given about the resource requirements, the better is the basis for deriving a successful and tailored management. Therefore, if anyone or several or all of this information is provided, the results management is correspondingly improved.

[0027] According to a further preferred aspect of the invention, the mobile code and the certificate are logically linked by means of the code hash. This ensures that the mobile code and the certificate containing the information necessary for performing a good resource management are not separated in any stage of their co-existence so that the mobile code can, at any time, rely on the resource management based on the logically linked certificate.

[0028] According to a further preferred aspect of the invention, certificate or a sequence of certificates is linked to the mobile code and the RRL list, the certificate or certificates transferring trust from a principal trusted by the software consumer to the RRL certificate issuer. The certificate or the sequence of the certificate contains one or several certificates transferring authorization from a executing peer to the principal who signed the certificate containing RRL. If the certificate or the certificate sequences is/are valid, the run-time execution environment will define the resource location policy. This system contributes very much to the success of the transfer and usage of the mobile code.

[0029] Furthermore, a certificate containing the RRL contains a digest of the mobile code that is used to verify its integrity which is another security feature.

[0030] According to a further preferred aspect of the invention, the mobile code comprises Java programs and applications. As mentioned before, Java provides programs and applications which are not restricted to special platforms which means that also the resource management will be platform independent.

[0031] According to a further preferred aspect of the invention, the format of the certificate or certificates is SPKI. As stated below, the SPKI is a preferred format when putting the invention to practice as SPKI provides all the features which are desirable for the invention in an efficient and elegant way.

[0032] According to a further preferred aspect of the invention, an execution program is provided in an execution environment of the user, the execution program defining at least the resource access policy that will be applied to the mobile code on execution. Such execution program will be the most suitable tool to define the re-

source access policy which also has the advantage that the implementation of the resource access policy will be done by a program which is adapted to the RRL transmitted with the mobile code.

[0033] For achieving the above object a method for resource management for mobile code using a mobile code as discussed above comprises, in the case of downloading upon request a mobile code from a principal (software provider or distributor) to a user, in the negotiation phase in the beginning of the downloading process, a RRL list is transferred from the principal to the user informing the user of the resource requirements of the mobile code. Whenever a peer requests to download mobile code, the RRL information is used in the negotiation protocol the goal of which is to ensure that the receiving peer has all resources required for the execution of the mobile code. Exactly this information is provided by this method in a most advantageous way. Whenever a peer requests to download or upload mobile code, the RRL information can be used in a negotiation protocol. The goal of this negotiation protocol is for the sender peer to ensure that the receiving peer has all resources the mobile application requires to execute. [0034] According to a further preferred aspect of the invention, in the negotiation phase, the downloading process further includes user and/or platform authentication, specifying restrictions imposed by the mobile code distributor as to the user and/or platform involved, and/or payment/licensing evaluation, comprising the financial aspects of the mobile code transfer. The platform authentication offers some guarantees for the software producer/distributor that is a contribution to the deal is acknowledged and the mobile code is used in the proper way.

[0035] According to a further preferred aspect of the invention, after the negotiation phase, the mobile code is downloaded. This ensures that the mobile code is downloaded and only then downloaded if all the basic requirements for its execution have already been checked and verified as being available.

[0036] According to a further preferred aspect of the invention, the mobile code or upgrades thereof are distributed from a service provider to a plurality of users, and wherein, in the case of upgrading, additional information is transmitted specifying which files need to be deleted, replaced or added. The mobile code and methods described so far can not only be used in a negotiation between two entities but also for distributing mobile code from a service provider to a plurality of users. It is advantageous that, for this application of invention, only a minimum of additional information is required which can be put into the resource usage needs section or the certificate containing the RRL.

[0037] For achieving the above object a method for resource management for mobile code using a mobile code as discussed above comprises, in the case of uploading upon request a mobile code from a resource owner to a user using a mobile code, in the negotiation

phase in the beginning of the uploading process, a RRL list is transferred from the resource owner to the user informing the user of the resource requirements of the mobile code. Here again, the same advantages are achieved as with the downloading process.

[0038] According to a further preferred aspect of the invention, in the negotiation phase, the uploading process further includes user and/or platform authentication information specifying restrictions imposed by the resource owner as to the user and/or platform involved, and/or payment/licensing evaluation information comprising the financial aspects of the mobile code transfer. Also in the uploading process, such information is valuable to conclude an acceptable deal and to optimize the resource management.

[0039] According to a further preferred aspect of the invention, after the negotiation phase, the mobile code is uploaded. This is again to make sure that the actual transfer of the mobile code is effected only after all the security and resource management information checks have been made.

[0040] For achieving the above object, in a method for transferring mobile code through an active network for resource management for mobile code using a mobile code of as discussed above, the network comprising a plurality of active network nodes, the active flow is composed of the following information: a) a mobile code that needs to be executed in a node which is crossed by the active flow; b) a RRL list issued by the entity that sends the mobile code to the network; c) a certificate or a sequence of certificates whose first entry is a certificate from the network operator to the starting entity, and d) the data themselves. This method ensures in a most advantageous way that the mobile code with the resource usage needs section can also be used and transferred in an environment of active networks playing an ever increasing role in the global program and data transfer.

[0041] According to a further preferred aspect of the invention, when the active flow crosses a network operator boundary from an operator X to an operator Y, the exit node of the operator X adds a certificate to the sequence issued by network operator Y authorizing operator X to send active flows through its network. This is a simple and, therefore, advantageous way to ensure a safe transfer of the mobile code with the resource usage needs section within the active networks.

[0042] According to a further preferred aspect of the invention, an execution program is provided in an execution environment of the user; the execution program defining at least the resource access policy that will be applied to the mobile code on execution. As the certificate sequence with resource usage information is attached to the mobile code, this information can be used by the receiving peer to define the resource management policy on the mobile code at run-time.

[0043] According to a further preferred aspect of the invention, the execution program is transmitted together with the mobile code. Also the execution program could

also be provided separately or on other storage media to the user, the transfer of the execution program together with the mobile code is an advantageous way of handling this matter.

[0044] According to a further preferred aspect of the invention, the method comprises any of the following steps: a) verifying that the mobile code integrity has not been compromised, b) reducing the certificate chain associated with the mobile code to verify trust transfer from the execution environment to the supplier, and c) create a process-like structure for the mobile code which isolates the mobile code from other programs running on the same execution environment.

[0045] Before executing the mobile code, the receiving peer reduces the certificate sequence that comes along with the mobile code. If the certificate or sequence of certificates is valid, the run-time execution environment will define the resource allocation policy based on the RRL along with the type of access to the resource and an upper limit on its usage. Any or all of these steps contribute to a smooth execution of the mobile code. Furthermore, the mobile programs are isolated one from each other. Also the access to resources is done through the execution environment avoiding influence or interference of mobile code and programs among each other.

[0046] According to a further preferred aspect of the invention, the resource allocation policy is defined by an intersection between the sequence of certificates, one of which contains the RRL, and the ACL local to the executing peer. In other words, authorization to access resources at run-time will be defined on the executing peer based on the RRL and the ACL of each peer and/or user. If the certificate or the certificate sequence of the certificates is valid, the run-time execution environment will define the resource location policy based on the RRL and the ACL. The ACL contains a list of principals known to the executing peer along with a maximum resource usage list per principal. Unknown principals can have a default maximum resource usage list too.

[0047] According to a further preferred aspect of the invention, the mobile code or the execution program or its reduced program is configured to discover that a given resource is available through the execution environment and to request access to it, and thus to dynamically request access to other resources, and wherein the execution environment will decide on run time whether to grant or to deny such access. One advantageous feature of the mobile code is its ability to discover the resources and other applications present or running on the target machine to be able to communicate or work with them. This gives rise to new security concerns for both the calling and the called code. Each one of them might impose its own access control based on an authenticated message exchange system, which helps to run the mobile code in a safe way. Another functionality of the execution environment is the dynamic allocation of resources not listed in the RRL. More specifically, the mo-

bile code can dynamically discover resources on the executing peer. Therefore, the resource usage policy can be made dynamically updateable.

[0048] According to a further preferred aspect of the invention, for resources not listed in the RRL, if the resource is a built-in resource in the execution environment, the execution program will check its "run-time resource access policy" and determine, whether to grant access or not to the resource. This method takes advantage of the presence of the built-in resource and the general ability thereof.

[0049] According to a further preferred aspect of the invention, if the resource is another mobile code, this can define its own access policy stating to whom access should be granted, the advantage being that any resources which are available to anyone are integrated in the process in the execution environment almost automatically.

[0050] According to a further preferred aspect of the invention, wherein the execution program monitors and/or accounts for and/or reclaims the resources whenever its usage limit is exceeded depending on the level of trust the user has on the source of the mobile code, the resources made available to the application can be trusted to never exceed the allocated amount.

[0051] In the invention, resource needs are described and it is up to the executing environment to decide which ones are granted and which ones are not, based on their ACL and the trust path between themselves and the certifying programmer. This reflects a more generalized vision of resource as "anything a program can interact with" which is a much broader concept than the once present in the state of art. A main advantage of the invention is that it provides secure fine-grained access to resources, both quantitative and qualitative, for mobile code and that it is not restricted to provide an all or nothing access control to resources. Furthermore, in the invention, there is no need for a certificate infrastructure in order to validate the certificates or certificate sequences.

[0052] The invention also differs from the state of art specifically in that the mobile code comes along with a non-exhaustive list of required resources. The list is nevertheless only intended for execution environment information. The mobile programs could run with fewer/greater resources granted or discover new resources on the fly.

[0053] The execution environment embodying the invention allows, apart from controlling and managing access to resources, for collaboration between different programs running on this execution environment.

[0054] Embodiments of the invention are now described with reference to the attached drawings in which:

Figure 1 is a block diagram view of the software producer system depicting the phases involved in the production of a mobile pro-

- gram,
- Figure 2 shows a download upon request case in which a software consumer requests to download a mobile program from a software distributor;
- Figure 3 shows an upload upon request case in which a resource requester contacts a resource owner and asks to upload a mobile program that will act as personalized interface with the resource;
- Figure 4 shows a broadcast / multicast of mobile programs or upgrades case in which a service provider broadcasts mobile programs to offer new services to its clients or upgrades/patches;
- Figure 5 shows an active flow crossing the active network between two execution environments;
- Figure 6 shows an execution environment for mobile code.

[0055] A software producer is the entity responsible for producing a mobile code or program. This principal can be a programmer, a department within a company, an organization, etc. The mobile code is any code or application that can be sent/received through the net and is, thus, susceptible of attacking the executing peer. The mobile code can also be a local code that has arrived at the peer through a network or applications on CDROM and distributed to the users.

[0056] The first step in the process is to attach a certificate to the mobile code stating which are the resource usage needs for the given program: the software producer writes a mobile program that wants to diffuse over the Internet. To do so it needs to attach to the mobile program a certificate detailing the resource usage needs of the mobile program. This certificate is unique for each different mobile application and contains the following information:

a) Issuer of the certificate:

This is a unique identifier for the software producer. This needs not to represent a whole organization: it can be a programmer within a company, a research group or an open software group. Practically, it will be a digest (or hash) of the public key of the software producer, or the key itself.

b) Subject:

A value that uniquely identifies the mobile program to which the certificate is referred. In cryptographic words, this will be a hash of the mobile program.

c) Validity period:

This states from when to when the given certificate and thus the information contained in it is valid. This field allows for producing demo software with short validity periods, or release software with long ones.

d) Resource Requirements List (RRL):

This list should contain those resources needed by

the mobile program without which it is unable to execute, plus those resources that the software producer knows a priori that will be accessed. For each entry of the list there should be the following information which describes precisely the access to the resource:

d1) Name of the resource:

This name can be general specifying the type of resource, or more detailed, for example the resource manufacturer. The name can have constructor like 'any', or 'prefix'. For example, 'C:\Temp\' stands for any file in the temporary directory.

d2) Action of the resource:

A statement as to how the resource should be used. For example, if accessing a webcam, actions supported could be read (the images), zoom, on, off, focus and move.

d3) Upper quantitative limit:

This statement relates to the maximum usage of the resource from a quantitative point of view, for example writing 150Mbytes to disk or allocating 30Mbytes of memory.

d4) Upper qualitative limit:

This statement relates to the maximum usage of the resource from a qualitative point of view, for example a network connection with 10Mbits/sec, or a 4Mbits/sec video decoder.

[0057] With all the previous information, the software producer creates a certificate and attaches it to the mobile program. Here, "attach" should not be understood as a physical link, but a logical one. Precisely, a characteristic of cryptographic hashing functions is that for two different inputs, the result will be different. Moreover, it is computationally impossible, given an input, to find another one that generates the same output. Thus, mobile program and certificate are logically linked.

[0058] The certificate fields described above are those required. However, a certificate can contain some optional information such as the certification authority (entity capable of generating certificates) of the issuer, an address with detailed information on the mobile application, etc.

[0059] It should be noted that the RRL certificate is only a requirements list issued by the programmer of the mobile program. As can be seen in the following section, this certificate alone provides no security at all. Upon software distribution, the mobile program and the RRL certificate will be accompanied by a sequence of certificates transferring trust from a principal trusted by the software consumer to the RRL certificate issuer.

[0060] The distribution of mobile applications and programs can follow different patterns. In this section, different scenarios of mobile software distribution are presented. It should be noted that this section does not deal with classical software download from the Internet (ftp,

http, etc), but only with mobile applications that take advantage of the invention.

[0061] Figure 2 shows the interactions between a software distributor and a software consumer in the 'download upon request' case. A user or device contacts a software distributor and requests to download a specific piece of software. When the software distributor receives such a request, it starts a negotiation phase previous to the downloading of the mobile program. This negotiation comprehends several sub-phases:

a) User and/or platform authentication:

A software distributor may, and probably will, impose restrictions as to whom or where the software is being downloaded. Software producers or distributors may require software to be downloaded onto secure platforms that provide some guarantees as of there will not be any interference on program execution.

b) Resource requirements:

In this phase, the software distributor informs the consumer of the resource requirements on the mobile program. The objective of this phase is to avoid the downloading of software that will not be able to execute due to lack of resources. Note that the RRL is not exhaustive, since, by definition, mobile code should be able to discover resources present on the executing platform. The software consumer answers back to the distributor with a list of principals it trusts and to whom it will grant access to the resources. It is the distributor's responsibility to provide a sequence of certificates transferring trust from one of those principals to the principal that issued the RRL, along with the RRL certificate and the mobile program.

c) Payment / licensing / evaluation:

Since not all software is free of charge, this phase deals with the financial aspects of software distribution. Here, software distributor and consumer reach an agreement, possibly with proof of payment or license, before the downloading of the mobile program. Note that the consumer may be requesting an evaluation software. In this case, the only difference will be that the RRL certificate will have a short validity period, and platform authentication as described in the previous phase becomes mandatory in order to avoid illegal usage of the software.

[0062] The last step in the process is the actual download of the mobile program, the RRL certificate and a sequence of certificates that transfer trust from the software consumer down to the principal that issued the RRL certificate. Along with these data, the software distributor will most likely send a description of the mobile code with information such as name, version, etc. Software integrity is assured by the subject field in the RRL certificate which contains for example the result of a hash function on the mobile program file. If privacy is

needed, any protocol cryptographic protocol may be used.

[0063] Figure 3 shows a case in which a computer or device wants to access a resource residing on a remote computer. A resource requester contacts a resource owner and asks to upload a mobile program that will act as personalized interface with the resource. Examples of this are analyzing images of an electronic microscope or convert data from a compressed format to postscript before printing which means an application wanting to get some specific information from an electronic microscope or printing a compressed image. However, the requester may not want to access directly the resource, but use a specific interface providing the desired functionality. This is done by sending a mobile application to the resource owner system which, in the first case, extracts locally the information from the microscope images and sends it back to the application or, in the second case, converts from a compressed image format to postscript before sending to the printer, increases the performance of the application.

[0064] The protocols between peers are basically the same as in previous case of the communication between a software distributor and a consumer, with the exception that here there is a request to upload mobile code instead of downloading. As for the negotiation phase, user and platform authentication will be used here by the resource owner, since it can have its own policy as of who can upload software to the system. On the other hand, the payment/licensing phase can be used here whenever the resource requester should pay to access the resource. An example would be sending a mobile program that queries a remote database for which a subscription is required.

[0065] Figure 4 shows the case of a service provider with several subscribers broadcast or multicasts mobile programs to all or some of its clients. This mobile code can be whether a new mobile program that the service provider wishes to install on all its client platforms, or an upgrade/patch to already existing applications of the subscribers' systems.

[0066] Given the nature of the broadcast scenario, in this case there is not the possibility of an interactive protocol between service provider and consumers. Therefore, when the service provider broadcasts the mobile program along with some extra information:

a) Installation / upgrade information:

The installation information is basically the same information about the mobile program sent in the earlier cases. In the case of upgrading, the service provider needs to specify which files need to be deleted, replaced or added.

b) Certificate sequence:

If, in this scenario, the receiving systems are subscribed to a service and thus there is already a trust relationship, the service provider needs only to provide the sequence of certificates transferring trust

from itself to the programmer. The service provider itself may be also a software producer, in which case the certificate sequence will be empty.
c) RRL certificates and mobile program as in previous cases.

[0067] The case in which a service provider or software distributor sends a mobile program to a single receiver is a special case of the one presented above.

[0068] Active networks are a hot topic of research nowadays. The idea behind active networks is the possibility to configure each node of the network as a data flow traverses it. The active flow carries the data along with code that is executed by each active node and that does any processing on the flow. This processing can be from deciding which link the flow should follow up to reducing the quality of a video flow depending on the capacity of the link.

[0069] Figure 5 shows a scenario in which a flow between two execution environments, i.e. computers, crosses several active nodes or routers from different network operators. Any negotiation between active nodes belonging to the same or different network operators are not possible in this case. An active flow is composed of the following information:

- a) The mobile code that needs to be executed in every node the flow crosses.
- b) RRL certificate issued by the originating execution environment, the entity that sends the mobile code to the network.
- c) A sequence of certificates whose first entry is a certificate from the network operator X to the execution environment. This certificate allows the flow to cross all active nodes belonging to operator X. When the flow crosses an operator boundary, the exit node of operator X adds a certificate to the sequence issued by network operator Y authorizing operator X to send flows through its network.
- d) The data itself.

[0070] The certificates between network operators reflect real-world deals between network operators. An operator Y may authorize operator X to cross its network, but imposing some limits to the resources available to mobile code sent. In this case, there is a particular need for the active node to control the resources made available to "foreign" mobile programs.

[0071] The last phase involved in the present invention is mobile code security during execution and secured resource management. The mobile program has gotten to the executing system, or it is already present on the system. The execution environment, that is the software in charge of executing a mobile program, needs to meet some requirements so that the security of the system is not compromised (see Fig. 6). When a mobile program is launched, the execution environment performs the following steps:

- a) Verify that the mobile program integrity has not been compromised. This is done by computing the hash function on the mobile program and verifying that the result is the same as in the RRL certificate.
- b) Reduce the certificate chain associated with the mobile program to verify that trust is passed from the executing environment to the programmer or the issuer of the RRL certificate. To do this, the execution environment needs to access its own access control list (ACL) or the ACL of the user.
- c) Define the resource access policy that will be applied to the mobile program on execution. This resource access policy is the intersection between the RRL and the ACL plus certificate sequence reduction. Note that this resource access policy refers only to those resources specified in the RRL and the ACL. Mobile programs can dynamically request access to other resources: the execution environment will decide on run-time whether to grant or deny such access.
- d) Create a process-like structure for the mobile program, which isolates the program from other programs running on the same execution environment. The process abstraction also enforces the program to go through the execution environment in order to access any resource.

[0072] Whenever a mobile program requests access or tries to access a resource, the execution environment checks in the resource access policy of the process whether it has access to the resource or not. If it does, it will provide a capability that will monitor, account for and reclaim the resource whenever its usage limit is exceeded. There are, nevertheless, exception to this: low level resources, that is CPU time and memory, cannot be managed through capabilities; the execution environment manages them directly.

[0073] As stated above, the mobile code has the ability to discover the system on which it is being executed and take advantage of the resources available. This means that a program can discover that a given resource is available through the execution environment and request access to it. This resource can be a built-in resource in the execution environment or a software-based resource, i.e. any other mobile program that allows being called.

[0074] If the resource is a built-in one in the system, the execution environment will check its "run-time resource access policy" and determine whether to grant access or not to the resource. If, on the other hand, the resource is another mobile program (a video decoder or a decryption service for example) that gives access to anyone (it has not defined a its own resource access policy), access is granted too.

[0075] In case the software-based resource defines its own access policy, the execution environment will query the resource itself as to whether access is granted or not. This means that mobile programs available as

resources on a system have the ability to manage and define who (that is which mobile programs) can access them.

[0076] As stated above, security and privacy is a major concern with the handling of mobile code to cope with these requirements, and relies on cryptography. There are many different ways to implement cryptographic features in a program or on data. However, one particular format, the Simple Public Key Infrastructure or SPKI-format is particularly adapted for the purposes of the invention as will be described below.

[0077] Cryptography provides a means whereby two people can communicate openly in such a way that a third party is unable to determine or alter what is being said. By assuring privacy, cryptography indirectly provides authentication because only the communicating parties know how to encrypt and decipher each other's messages. A form of cryptography known as public-key cryptography appears to be best suited to fulfilling the requirements of the Internet. Each user of a public-key cryptosystem holds a pair of related keys. Anything encoded with one key can only be decoded by its counterpart. Each user keeps one key secret and makes the other publicly available. Thus, other people can employ the user's public key to send messages that only the user can read, or the user can "sign" a message with her private key to authenticate it - other people can apply the user's public key to verify that the message came from the user. Crucial to the operation of a global public-key cryptosystem on the Internet is a practical and reliable means of having access to the public keys, called a Public Key Infrastructure or PKI.

[0078] Much recent work has focused on moving away from identity-based PKIs to a more general system based on attributes or credentials. SPKI and SDSI (Simple Distributed Security Infrastructure) are two of such efforts. These two initiatives merged later into SPKI, given that their approach to security infrastructures and certificates were almost identical. SPKI is designed to "facilitate the construction of secure systems" and "provides simple, clear terminology for defining access-control lists and security policies". It is also an attempt to move away from identity-based certification and towards a system based on roles and credentials.

[0079] SPKI calls its entities "principals" and defines them to be digital signature verification keys. Thus, SPKI principals are public keys that can make declarations by issuing verifiable signed statements. Those signed statements come mainly in the form of certificates. SPKI provides for so called SPKI authorization certificates as a basic form of certificates which transfer some specific authorization or permission from one principal to another. Because a certificate merely transfers authorizations, rather than creating them, it is required to inject authorizations into a chain of certificates. This is done by means of ACL-entries (ACL = Access Control List). An ACL-entry lives on the machine of the verifier, leading to the observation that all authorization flow is in a

circuit - from the verifying machine's ACL, possibly through certificates and then back to the verifying machine. Alternatively, one might say that the only root of an authorization certificate chain is the verifier.

[0080] SPKI allows its principals to define groups, or sets, of principals by means of name certificates. Each group has a name and a set of members. The name is local to some principal, which is the "owner" of the group. Only a group's owner may change its definition. A group can be an explicit list of the group's members (either as a list of principals and/or names of principals), or it can be defined in terms of other groups. Any principal can define his own groups and export them via his servers in much the same way as name bindings. The servers can issue membership certificates based on the groups' definitions.

[0081] If, from a practical point of view, mobile applications are programmed in the Java language, and programs and applications can be distributed using a specific file format that packages all files that compose the application. Moreover, this format fits the requirements of code certification, since a single file can easily be hashed to create a certificate.

[0082] As for the certificate format, SPKI certificates fit the above expressed requirements. Moreover, the fact that there is no need for an infrastructure of certification authorities in place will make the present invention easy to deploy.

Claims

1. Mobile code comprising a resource usage needs section containing at least a resource requirements list including those resources needed by the mobile code to be properly executable plus those resources that are known a priori to be accessed when executing the mobile code.
2. Mobile code according to claim 1, wherein the resource usage needs section of the mobile code is a certificate which is unique for each different mobile code.
3. Mobile code according to claim 1 or 2, wherein the resource usage needs section of the mobile code or the certificate contains, in addition to the resource requirements list, any of the following information:
 - a) issuer of the certificate information identifying the entity issuing the certificate,
 - b) subject information identifying the mobile code to which the certificate is referred, and
 - c) validity period information stating the period of time within which the certificate is valid.
4. Mobile code according to claim 3, wherein the re-

source requirements list contains any of the following information:

- a) name of the resource information specifying the type of resource,
- b) action on the resource specifying as to how the resource should be used,
- c) upper quantitative limit information stating the maximum usage of the resource from a quantitative point of view, and
- d) upper qualitative limit information stating the maximum usage of the resource from a qualitative point of view.

5. Mobile code according to any of the preceding claims, wherein an execution program is provided in an execution environment of the user, the execution program defining at least the resource access policy that will be applied to the mobile code on execution.

6. Method for resource management for mobile code using a mobile code of any of the claims 1 to 5, wherein:

(a) in the case of downloading upon request a mobile code from a principal to a user, in a the negotiation phase in the beginning of the downloading process, a RRL list is transferred from the principal to the user informing the user of the resource requirements of the mobile code, and

(b) in the case of uploading upon request a mobile code from a resource owner to a user, in a the negotiation phase in the beginning of the uploading process, a RRL list is transferred from the resource owner to the user informing the user of the resource requirements of the mobile code.

7. Method according to claim 6, wherein, in the negotiation phase, the downloading process further includes user and/or platform authentication, specifying restrictions imposed by the mobile code distributor as to the user and/or platform involved, and/or payment/licensing evaluation, comprising the financial aspects of the mobile code transfer; and wherein, in the negotiation phase, the uploading process further includes user and/or platform authentication information specifying restrictions imposed by the resource owner as to the user and/or platform involved, and/or payment/licensing evaluation information comprising the financial aspects of the mobile code transfer.

8. A method for transferring mobile code through an active network for resource management for mobile code using a mobile code of any of the claims 1 to

5, the network comprising a plurality of active network nodes, wherein the active flow is composed of the following information:

- a) a mobile code that needs to be executed in a node which is crossed by the active flow,
- b) a RRL-list issued by the entity that sends the mobile code to the network,
- c) a certificate or a sequence of certificates whose first entry is a certificate from the network operator to the starting entity, and
- e) the data themselves.

9. Method according to claim 8, further comprising any of the following steps:

- a) verifying that the mobile code integrity has not been compromised,
- b) reducing the certificate chain associated with the mobile code to verify trust transfer from the execution environment to the supplier, and
- c) create a process-like structure for the mobile code which isolates the mobile code from other programs running on the same execution environment.

10. Method according to claim 8 or 9, wherein the mobile code or the execution program or its reduced program is configured to discover that a given resource is available through the execution environment and to request access to it thus to dynamically request access to other resources, and wherein the execution environment will decide on run time whether to grant or to deny such access.