IN THE UNITED STATES DISTRICT COURT
WESTERN DISTRICT OF TEXAS
AUSTIN DIVISION

K.MIZRA LLC, Plaintiff, v. AMAZON.COM, INC. and AMAZON WEB SERVICES, INC., Defendants.	Case No. 1:26-cv-00316 **JURY TRIAL DEMANDED**

COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff K.Mizra LLC ("K.Mizra") files this Complaint for patent infringement against Defendants Amazon.com, Inc. ("Amazon.com") and Amazon Web Services, Inc. ("AWS") (collectively, "Defendants" or "Amazon"), alleging as follows:

I. INTRODUCTION

1. K.Mizra is a patent licensing company run by experienced management. The company focuses on high value, high quality patents with a global reach. It owns patent portfolios originating with a wide array of inventors, including portfolios developed by well-known multinationals such as IBM, Intel, Rambus and others, as well as from research institutes such as Nederlandse Organisatie voor Toegespast Natuurwetenschappelijk Onderzoek (Netherlands Organization for Applied Scientific Research, or "TNO"). By focusing on high quality patents, K.Mizra provides a secondary market for inventors to recoup their research and development investments and to continue their innovations. K.Mizra offers licenses to its patents on reasonable terms and in this way plays an important part in the development of the technologies that improve all our lives.

2. K.Mizra remains ready, willing, and able to provide commercially-reasonable licenses for its various patented technologies to all entities who wish or need to use them internally or in connection with products or services offered to others. As outlined below, Amazon is one such entity.

II. THE PARTIES

3. K.Mizra is a Delaware limited liability company with a mailing address of 777 Brickell Avenue, #500-96031, Miami, Florida 33131, and operates in Florida. K.Mizra is the owner by assignment of the Asserted Patents.

4. Defendant Amazon.com, Inc. is a Delaware corporation with a principal place of business at 410 Terry Ave. North, Seattle, Washington 98109-5210. Amazon may be served with process via its registered agent Corporation Service Company, MC-CSC1, 300 Deschutes Way SW, Suite 208, Tumwater, Washington 98501 or via its listed registered agent of Corporation Service Company, 251 Little Falls Drive, Wilmington, Delaware 19808.

5. Defendant Amazon Web Services, Inc. is a Delaware corporation with a principal place of business at 410 Terry Ave. North, Seattle, Washington 98109. Amazon Web Services, Inc. may be served through its registered agent Corporation Service Company, 211 E. 7th Street, Suite 620, Austin, Texas 78701. On information and belief, Amazon Web Services, Inc. is registered to do business in the State of Texas and has been since at least May 3, 2006. On information and belief, Amazon Web Services, Inc. is a wholly-owned subsidiary of Amazon.com, Inc.

III. JURISDICTION AND VENUE

6. This is an action for patent infringement under the patent laws of the United States, 35 U.S.C. §§ 1 *et seq.*, including 35 U.S.C. §§ 271, 281, and 284, among others. The Court has

subject-matter jurisdiction over the claims raised in this action pursuant to 28 U.S.C. §§ 1331 and 1338(a).

7. This Court has personal jurisdiction over Defendants as they have committed acts within this District giving rise to this action and have established minimum contacts with this forum such that the exercise of jurisdiction over Defendants would not offend traditional notions of fair play and substantial justice. Defendants, directly and through subsidiaries or intermediaries, have committed and continue to commit acts of infringement in this District and elsewhere by, among other things, importing, offering to sell, and selling products and/or services that infringe the Asserted Patents.

8. Venue is proper in this District pursuant to 28 U.S.C. § 1400(b) because Defendants have committed acts of infringement in this District and Defendants have a regular and established place of business in this District at Amazon Tech Hub located at 11501 Alterra Parkway, Austin, Texas.

IV. GENERAL ALLEGATIONS

A. Asserted Patents

9. U.S. Patent No. 8,234,705 ("the '705 Patent"), titled "Contagion Isolation and Inoculation," was legally issued by the USPTO to inventors Dr. Roskind and Mr. Emigh on July 31, 2012. A true and correct copy of the '705 Patent is attached hereto as Exhibit 1. The '705 Patent claims priority to U.S. Provisional Application No. 60/613,909, filed on September 27, 2004 ("the '909 Application").

10. U.S. Patent No. 8,144,717 ("the '717 Patent"), titled "Initialization of a Wireless Communication Network," was legally issued by the USPTO to inventors Marinus Johannes Blange and Miodrag Djurica on March 27, 2012. A true and correct copy of the '717 Patent is

attached hereto as Exhibit 2. The '717 Patent claims priority to PCT Application No. PCT/NL2007/000001, filed on January 2, 2007 ("the '001 Application"), and to U.S. Patent Application No. 12/159,753 ("the '753 Application").

11. U.S. Patent No. 8,438,120 ("the '120 Patent"), titled "Machine Learning Hyperparameter Estimation," was legally issued by the USPTO to inventor Stephan Alexander Raaijmakers on May 7, 2013. A true and correct copy of the '120 Patent is attached hereto as Exhibit 3. The '120 Patent claims priority to PCT Application No. PCT/NL2008/050247, filed on April 25, 2008 ("the '247 Application"), and to U.S. Patent Application No. 12/597,257 ("the '257 Application").

12. U.S. Patent No. 9,235,259 ("the '259 Patent"), titled "Method for Detecting Audio Ticks in a Noisy Environment," was legally issued by the USPTO to inventors Mark van Staalduinen, Victor Bastiaan Klos and Peter Jan Otto Doets on January 12, 2016. A true and correct copy of the '259 Patent is attached hereto as Exhibit 4. The '259 Patent claims priority to PCT Application No. PCT/NL2010/050795, filed on November 26, 2010 ("the '795 Application"), and to U.S. Patent Application No. 13/512,139 ("the '139 Application").

13. U.S. Patent No. 9,602,649 ("the '649 Patent"), titled "Event Disambiguation," was legally issued by the USPTO to inventors Jeroen Laarakkers, Mattijs Oskar van Deventer and Victor Bastiaan Klos on March 21, 2017. A true and correct copy of the '649 Patent is attached hereto as Exhibit 5. The '649 Patent claims priority to PCT Application No. PCT/NL2010/050478, filed on July 23, 2010 ("the '478 Application"), and to U.S. Patent Application No. 13/386,461 ("the '461 Application").

14. U.S. Patent No. 8,782,282 ("the '282 Patent"), titled "Network Management System," was legally issued by the USPTO to inventors Eileen Zhou, Roger Liu, Vijoy Choyi,

Moshe Itah and John Z. Yu on July 15, 2014. A true and correct copy of the '282 Patent is attached hereto as Exhibit 6. The '282 Patent claims priority to U.S. Patent Application No. 10/742,573, filed on December 19, 2003 ("the '573 Application").

15. K.Mizra is the sole owner by assignment of the '705 Patent, '717 Patent, '120 Patent, '259 Patent, '649 Patent and '282 Patent (collectively, "the Asserted Patents") with the full and exclusive right to bring suit to enforce them. K.Mizra is also entitled to sue to collect damages for all past infringement of the Asserted Patents.

1. <u>The '705 Patent</u>

16. The '705 Patent was involved in an unsuccessful *Inter Partes* Review Proceeding ("IPR") and several now-resolved federal court litigations, and was originally invented by two highly respected and prolific individual inventors, James A. Roskind and Aaron T. Emigh.

17. The '705 Patent was originally owned by Dr. Roskind and Mr. Emigh's company, Radix Labs, LLC. Dr. Roskind and Mr. Emigh were then, and remain today, focused on innovation, conducting new research, developing new technologies, and creating new and innovative computer products.

18. Dr. Roskind, one of the two inventors of the '705 Patent, has bachelor's, master's, and doctorate degrees from the Massachusetts Institute of Technology in both electrical engineering and computer science, and is the named inventor of over 300 U.S. patents. He has worked for Netscape as the Chief Architect and as the Netcenter Security Architect and was a co-founder for Infoseek, a company that was eventually acquired by Disney for $770 million. He was also a key developer of Google's "transport protocol" that provides the tech giant billions of dollars in value every year.

19. Mr. Emigh, the other named inventor of the '705 Patent, graduated from the University of California, Santa Cruz with degrees in linguistics and computer and information sciences, and is the named inventor of over 140 patents. Prior to working with Dr. Roskind, Mr. Emigh worked in various positions developing software, including working as a software manager, architect, and engineer for Unicom and working as a manager for the software development and technical marketing groups for Philips TriMedia. He has founded or co-founded many companies, in addition to Radix Labs, LLC, including CommerceFlow, Inc., which was acquired by eBay for its technology that Mr. Emigh helped develop.

20. After the '705 Patent issued, Dr. Roskind and Mr. Emigh recouped their research and development investment by selling their rights thereto and continued on in their individual technology development pursuits. K.Mizra ultimately acquired the '705 Patent and licensed it to many of the who's-who of the tech world. Some of the accused infringers chose to test the validity of the '705 Patent before settling their lawsuits involving the '705 Patent. For instance, a few accused infringers of the '705 Patent previously sought IPR by the Patent Trial and Appeal Board ("PTAB"). A Final Written Decision ("Decision") in the IPR found that the petitioners had not shown, by a preponderance of the evidence, that the asserted claims were unpatentable. The IPR Decision was appealed to the U.S. Court of Appeals for the Federal Circuit ("CAFC"), resulting in a procedurally focused remand to the PTAB. Prior to the issuance of the mandate that would have sent the IPR back to the United States Patent and Trademark Office ("USPTO") for further consideration, the parties agreed to move to dismiss the appeal.

a. Prior licensing and litigation of the '705 Patent

21. The '705 Patent has been owned by several entities, in addition to Radix Labs, LLC and K.Mizra, with some of those entities issuing to third parties certain rights to the technologies covered thereby.

22. K.Mizra has been involved in a number of actions it had to institute to protect its patent rights, including actions involving the '705 Patent. Most of those actions resulted in the execution of confidential patent license agreements.

b. Network security problems in 2004 solved by the '705 Patent

23. The technology described in the '705 Patent was invented by Dr. Roskind and Mr. Emigh, two colleagues living in the same area who had similar interests in innovating computer-related technologies. In 2003, the inventors decided to create a business—Radix Labs, LLC—which focused on developing intellectual property related to various computer technologies, including computer network security technologies. The inventors focused on conceiving and reducing to practice inventions that they knew were needed (or soon would be needed) in the computer networking industry and then on drafting patent applications to capture and protect their technological innovations. In September 2004, the inventors filed the '909 Application to which the '705 Patent claims priority. The '909 Application described technology that focused on securing a computer network against the threats to which it was exposed when computer endpoints (e.g., laptop computers) were connected to a computer network. The '909 Application, and by natural extension the '705 Patent, also focuses on remedying identified threats and quarantining those threats to mitigate any damage to the secured network.

24. Claims of the '705 Patent are directed to technological solutions that address specific challenges grounded in computer network security. Maintaining the security of computer

systems and networks is a tremendous concern for modern enterprises, since a breach of an internal network can have severe repercussions, including major financial losses, data theft, disclosure of sensitive information, network disruptions, data corruption, etc. The inventors of the '705 Patent understood that while a network security appliance or hardware can be adept at keeping out unwanted external intrusions from the network, the most exploitable vulnerabilities of most networks are the end-user computers that roam throughout various public and private network domains, potentially exposing those computers to infection and then accessing and potentially infecting the entire and presumably secure computer network.

25. For example, the '705 Patent explains that "[l]aptop and wireless computers and other mobile systems pose a threat to elements comprising and/or connected to a network service provider, enterprise, or other protected network to which they reconnect after a period of connection to one or more networks and/or systems that are not part of the service provider, enterprise, or other protected network. By roaming to unknown domains, such as the Internet, and/or connecting to such domains through public, wireless, and/or otherwise less secure access nodes, such mobile systems may become infected by computer viruses, worms, backdoors, and/or countless other threats and/or exploits and/or have unauthorized software installed; have software installed on the mobile system by an operator of the protected network for the protection of the mobile system and/or the protected network removed or altered without authorization and/or have configurations, settings, security data, and/or other data added, removed, and/or changed in authorized ways and/or by unauthorized person[s]." (*See, e.g.*, Ex. 1 at 1:14–31.)

26. The solution to these problems—as specified and claimed in the '705 Patent—was an advanced departure from the conventional network access control solutions then in use and was then, as it remains today, patent eligible, highly valuable, novel, and non-obvious technology.

c. The '705 Patent claims are presumed valid and patent-eligible

27. K.Mizra asserts that at least, and without limitation, Claim 19 of the '705 Patent has been directly infringed, either literally or under the doctrine of equivalents. K.Mizra reserves the right to assert additional claims of the '705 Patent, including both independent and dependent claims, pursuant to the Court's (and other applicable) rules and procedures and as discovery progresses. These claims are referred to herein as the "Asserted '705 Claims."

28. None of the Asserted '705 Claims are directed to abstract ideas, and each employs inventive concepts and is directed to patent-eligible subject matter. All claims of the '705 Patent are also presumed to be valid and enforceable against Amazon and others.

29. Indeed, the '705 Patent's specification and claims demonstrate that the need satisfied by the inventions of the Asserted '705 Claims was long-felt in the industry and thus unconventional. As one example, the '705 Patent explains that mobile end user devices such as laptops and wireless computers pose a threat to protected network elements because those devices may access unsecure systems and thereby "become infected by computer viruses, worms, backdoors, and/or countless other threats and/or exploits and/or have unauthorized software installed; have software installed on the mobile system by an operator of the protected network for the protection of the mobile system and/or the protected network removed or altered without authorization; and/or have configurations, settings, security data, and/or other data added, removed, and/or changed in unauthorized ways and/or by unauthorized person[s]." (Ex. 1 at 1:23-31.) Similarly, stationary systems such as desktop computers "may become infected, e.g., due to receipt and execution of malicious code via a network or other communication and/or a diskette and/or other removable media." (*Id.* at 1:31-34.) This poses a danger to a protected network because, when the user device connects to the protected network, "a system may infect or otherwise

harm resources associated with the protected network before measures can be taken to detect and prevent the spread of such infections or harm." (*Id.* at 1:34-38.) The '705 Patent's specification further provides that "[t]herefore, there is a need for a reliable way to ensure that a system does not infect or otherwise harm other network resources when connected to a protected network." (*Id.* at 1:38-41.)

30. The specification (including the provisions quoted above), the figures (including those included below), and the text related to the figures further illustrate the complex, tiered network system architecture of the inventions captured by the Asserted '705 Claims. These figures include the following:

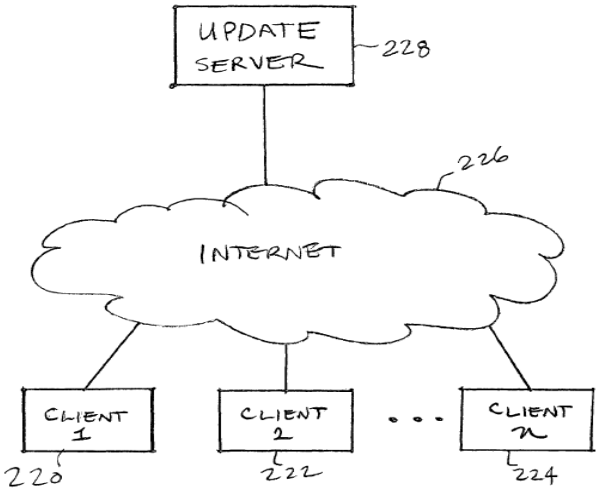
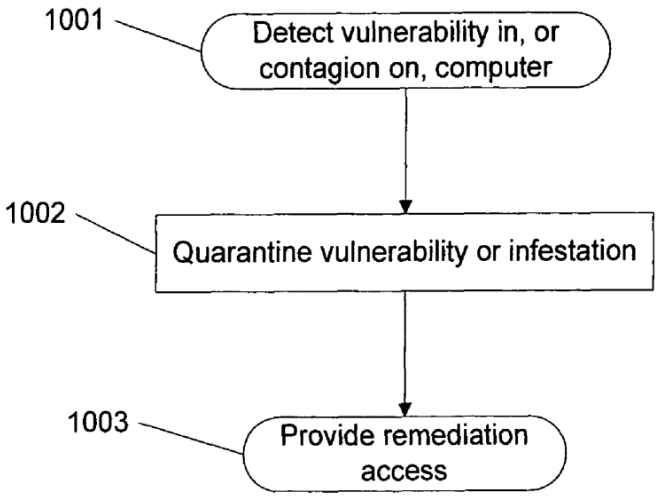


FIG. 2B

(*See id.* at Fig. 2B.)



(*See id.* at Fig. 10A.)

31. The foregoing demonstrates that the inventions of the Asserted '705 Claims focus on specific tamperproof hardware that must interact with unique software to improve network access control technology and protect a secure computer network and the data stored thereon from infected devices. Thus, the Asserted '705 Claims are eligible as a matter of law for patent protection under step one of *Alice Corp. v. CLS Bank Int'l*, 573 U.S. 208, 216 (2014) ("*Alice*").

32. All actions and steps recited in the Asserted '705 Claims, including the act of quarantining endpoints or other computers, if necessary, requires the involvement of various hardware components running dedicated software both before, during, and after the selection and isolation of an object. Said another way, a claim directed to allowing a machine to automatically and dynamically select and isolate an unsafe device attempting to access a secure network is not simply adding a generic computer component to a fundamentally human process. Rather, it is removing the once-necessary human intervention from a fundamentally mechanical process, an "improvement in the functioning of a" networked system that simply cannot be considered directed to an abstract concept. *Enfish LLC v. Microsoft Corp.*, 822 F.3d 1327, 1339 (Fed. Cir. 2016).

33. As the specification confirms, the improvement captured by the Asserted '705 Claims is not simply quarantining an infected device, but it is instead a multi-faceted network system involving multiple interrelated software and hardware components to protect a network from known and unknown threats. Specifically, the specification of the '705 Patent discloses that to reduce the burdens of having to manually identify, connect to, isolate, and remove malicious software from an infected device, the networked system can direct an unclean computer attempting to connect to the secure network, known as the host computer, to a form of remediation, such as downloading a software patch or a software update, removing material from the host computer and/or enabling certain settings, etc. present on the host computer. (*See* Ex. 1 at 1:14–41.) Indeed, the inventions of the Asserted '705 Claims are each tethered to these advances over the art in the 2005 time frame, reciting methods and systems that automatically and dynamically detect an insecure condition by contacting a trusted computing base, receiving a response therefrom, determining whether that response contains a valid identification of cleanliness, and configuring and implementing a remediation action based on what is discovered about the state of an endpoint or "host" computer. (*See, e.g.*, *id.*, Claims 12 and 19.) More specifically, the Asserted '705 Claims require a system configured to communicate with a "trusted computing base" to determine when a response includes a valid digitally signed attestation of cleanliness, and to control access to the network accordingly. These Asserted '705 Claims are thus directed to a machine-implemented solution resolving a machine-specific problem: a machine's difficulty in detecting, isolating, and remediating infected endpoint devices (*e.g.*, host computers) to prevent contagion of and damage to the larger computer network.

34. The Asserted '705 Claims are thus directed to a machine-implemented process for (1) determining whether the host computer is required to be quarantined, (2) isolating and

inoculating the contagions (including directing the host to software programs and/or code designed to identify undesirable and/or unauthorized states) by quarantining the host, (3) limiting access to the network by the host computer so that the unsafe condition thereof can be remedied, and (4) allowing for remediation of an unsafe or infected host computer. As such, the Asserted '705 Claims recite inventions with specific applications or improvements to technologies in the marketplace and cannot be considered abstract or patent ineligible under relevant law.

35. In a recently resolved action, Citrix, a competitor of Twingate, moved a Federal Court in Florida to dismiss the claims brought against it for infringement of the '705 Patent on the grounds of patent invalidity under 35 U.S.C. 101. That motion was denied, with the court issuing a substantial opinion finding that neither *Alice* Step 1 nor *Alice* Step 2 rendered the there involved '705 Patent claims ineligible.

d. Failed IPRs of the '705 Patent

36. Fortune 100 companies accused of infringing the '705 Patent have previously filed petitions for IPRs, alleging that the claims of the '705 Patent should be held invalid as either anticipated or obvious considering art not previously considered. Ultimately, the PTAB instituted an IPR against the '705 Patent, with similar third party IPRs that were subsequently filed being joined to the first-filed and instituted IPR.

37. The PTAB eventually issued its decision holding that no claims of the '705 Patent were unpatentable, finding that no asserted prior art reference alone or in combination satisfied the limitation of "providing . . . an IP address of a quarantine server configured to serve the quarantine notification page" that was present in all claims of the '705 Patent.

38. The IPR Decision was then appealed to the CAFC, which reversed the PTAB's Decision on a few narrow procedural issues involving proof that the asserted prior art references would be combined by a person having ordinary skill in the art, as alleged by the petitioners.

39. A motion to dismiss the IPR was subsequently filed and the IPR has been dismissed with prejudice by the PTAB.

40. More recently, Netskope, Citrix and Google filed petitions seeking to have new IPR proceedings instituted against the '705 Patent. The Citrix petition was voluntarily withdrawn, while the Netskope and Google petitions were discretionarily denied.

2. <u>The TNO Patents</u>

41. The '717 Patent, '120 Patent, '259 Patent and '649 Patent (collectively, "the TNO Patents") were all developed at TNO, which is a prominent statutory research organization in the Netherlands focusing on applied science. Founded in 1932, TNO develops solutions to technological and social challenges facing Europe and the world, partnering with thousands of multinational corporations, universities, public sector organizations, and other subject matter experts to develop and market critical innovations. The named inventors of the TNO Patents include highly respected luminaries in their respective technical fields, including several holders of Ph.Ds.

a. The '717 Patent claims are presumed valid and patent-eligible

42. K.Mizra asserts that at least, and without limitation, Claim 1 of the '717 Patent has been directly infringed, either literally or under the doctrine of equivalents by Amazon. K.Mizra reserves the right to assert additional claims of the '717 Patent, including both independent and dependent claims, pursuant to the Court's (and other applicable) rules and procedures and as discovery progresses. These claims are referred to herein as the "Asserted '717 Claims."

43. None of the Asserted '717 Claims are directed to abstract ideas, and each employs inventive concepts and are directed to patent-eligible subject matter. All claims of the '717 Patent are also presumed to be valid and enforceable against Amazon and others.

44. Indeed, the '717 Patent's specification and claims demonstrate that the need satisfied by the inventions of the Asserted '717 Claims was long-felt in the industry and thus unconventional. As one example, the '717 Patent explains that stations within wireless networks were required to transmit information before routing topology had been defined, *i.e.* when plural routes were still possible. The '717 Patent explains that this created complications because wireless stations would have to support different modes of communication while operating in low-power states, which was difficult to achieve. The '717 Patent further explains that wireless links between nodes were hindered by barriers and that using the same frequency for upward and downward link routes required synchronization, which led to further complications by requiring wireless stations to support synchronization and powerful radio frequency signals, again at low-power states. The '717 Patent thus provides "a wireless network wherein routes for communication can be determined without the need to forward information via stations for which the routes have not yet been defined," a method of operating such a network, and "stations for flexible use in wireless networks." (Ex. 2 at 3:6-15.)

45. The specification (including the provisions quoted above), the figures (including those included below), and the text related to the figures further illustrate the innovative network architecture captured by the Asserted '705 Claims. These figures include the following:

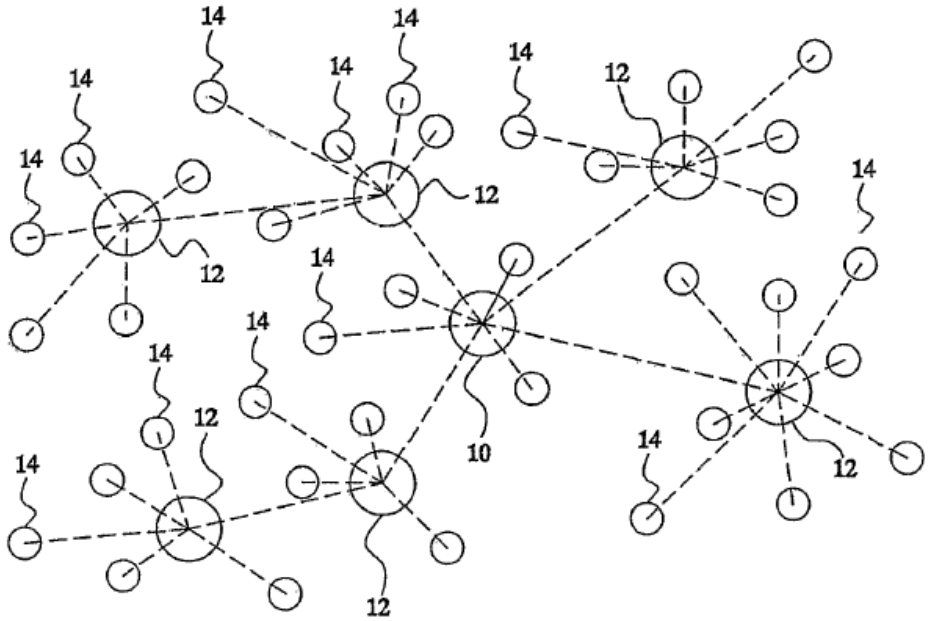


Fig.1

(*See id.* at Fig. 1.)

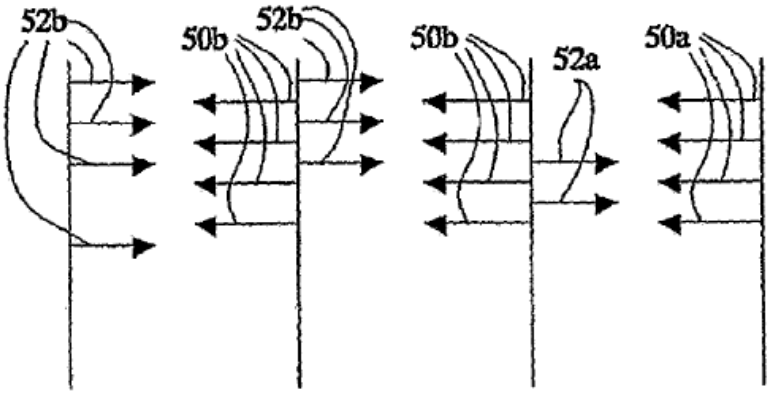


Fig.5a

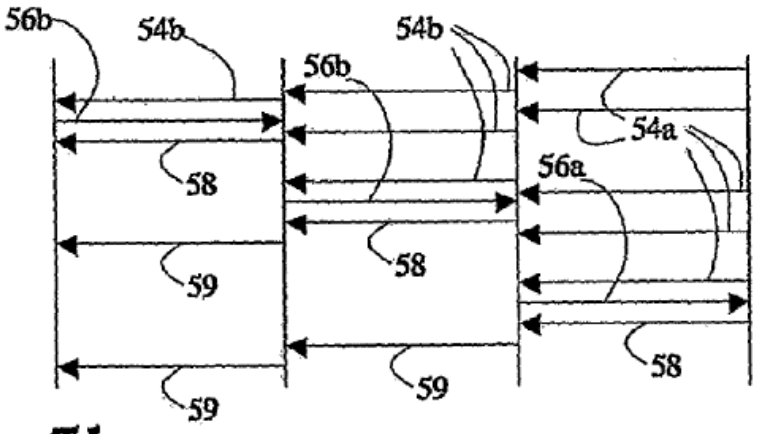


Fig.5b

(*See id.* at Fig. 5.)

46. The foregoing figures, which illustrate a wireless network and signaling protocol, demonstrate that the inventions of the Asserted '717 Claims focus on specific, highly efficient ways for a station to join a wireless network, thereby improving the operation and performance of the wireless network, as a whole. In particular, an unassociated station transmits an association request message and switches to an associated state upon reception of an association grant from

an association unit in response to one of its request messages. The association grant establishes an operating route associated with the newly associated station for exchanging operating messages with the association unit. Thereafter, the station can transmit association grants in response to received association request messages from additional stations seeking to join the network. This improves the efficiency and operability of the wireless network by facilitating the addition of new stations to the network. Thus, the Asserted '717 Claims are eligible as a matter of law for patent protection under step one of *Alice*.

47. All actions and steps recited in the Asserted '717 Claims, including the acts of transmitting association request messages, transmitting association grant messages, establishing operating routes and exchanging operating messages, and enabling stations to transmit association grant messages, require the involvement of various hardware components running dedicated software both before, during, and after the association of a station. Said another way, a claim directed to allowing a wireless network to automatically and dynamically associate a new station with the network and facilitate the addition of other stations is not simply adding a generic computer component to a fundamentally human process. Rather, it is an "improvement in the functioning of a" networked system that simply cannot be considered directed to an abstract concept. *Enfish*, 822 F.3d at 1339.

48. As the specification confirms, the improvement captured by the Asserted '717 Claims is not simply associating various devices into a wireless network, but is instead a specific network architecture whereby a particular unit operates to receive access requests and provide access grants, as well as providing operation information so that newly added wireless stations can transmit access grants to additional stations. Specifically, the specification of the '717 Patent discloses that to improve the efficiency of adding stations to a wireless network, a station seeking

to join the network operates initially in a non-associated state transmitting access requests. (*E.g.*, Ex. 2 at 8:25-32.) Subsequently, the station receives an access grant message and switches to an associated state. (*Id.* at 9:3-13.) In this state, the station establishes a dedicated communication route with the association unit or with a second station acting as an intermediary between the newly added station and the association unit, thereby ensuring efficient communication within the wireless network. (*Id.*) The newly added station can thereafter relay association grant messages to additional new stations, further enhancing the scope and efficiency of the wireless network. (*Id.* at 9:41-53.)

49. The '717 Patent specification further emphasizes that, once stations are added to the network via association grant messages, they have "well defined routes" to communicate within the network. (*Id.* at 9:63-10:6.) Thus, rather than requiring specific access routes in order to join the network initially, which creates needless complexity and power usage, the '717 Patent teaches that well defined routes are established after the station has been added to the network via an efficient method.

50. The Asserted '717 Claims are thus directed to a machine-implemented process for efficiently adding a new station to a wireless network and configuring that station to facilitate the addition of further stations to the network, improving the overall operation of the network itself. As such, the Asserted '717 Claims recite inventions with specific applications or improvements to technologies in the marketplace and cannot be considered abstract or patent ineligible under relevant law.

b. The '120 Patent claims are presumed valid and patent-eligible

51. The '120 Patent is directed to a critical advancement in machine learning, namely, an effective method of estimating hyperparameters given large potential datasets. The sole named

inventor of the '120 Patent, Stephan Raaijmakers, is a pioneer in artificial intelligence, holding a Ph.D. in Machine Learning from Tilburg University and a chair on Communicative AI at Leiden University. Dr. Raaijmakers is also the author of the book *Deep Learning for Natural Language Processing*, distributed by Simon & Schuster.

52. K.Mizra asserts that at least, and without limitation, Claim 1 of the '120 Patent has been directly infringed, either literally or under the doctrine of equivalents. K.Mizra reserves the right to assert additional claims of the '120 Patent, including both independent and dependent claims, pursuant to the Court's (and other applicable) rules and procedures and as discovery progresses. These claims are referred to herein as the "Asserted '120 Claims."

53. None of the Asserted '120 Claims are directed to abstract ideas, and each employs inventive concepts and is directed to patent-eligible subject matter. All claims of the '120 Patent are also presumed to be valid and enforceable against Amazon and others.

54. Indeed, the '120 Patent's specification and claims demonstrate that the need satisfied by the inventions of the Asserted '120 Claims was long-felt in the industry and thus unconventional. The '120 Patent is directed to the use of "hyperparameters" to train machine learning systems. "Hyperparameters" are parameters that can be used to determine, for example, decision criteria to be used in classifying objects. (Ex. 3 at 1:20-23.) Hyperparameters in turn must be determined using particular methods. The '120 Patent specification references two papers discussing a method of estimating parameters called "cross-entropy" and explains why this approach is not suitable for determining hyperparameters in the machine learning context. (*Id.* at 3:44-59.) Rather, the '120 Patent describes an iterative method of determining hyperparameters by drawing a random sample of hyperparameter vectors from a set of possible hyperparameter vectors, selecting from the sample a hyperparameter vector producing the best result, updating the

estimate using the selected hyperparameter vector, and repeating the process to determine the best hyperparameters. (*Id.* at 2:1-19.)

55. The '120 Patent thus identifies the conventional method of determining parameters and explains a specific technical improvement, namely:

> The present invention is based upon the insight that methods similar to the cross-entropy method can be adapted for hyperparameter estimation by adding a best performance preservation feature. The present invention benefits from the additional insight that the best performance in a particular iteration may be preserved by restricting the sample using the best performance of the previous iterations.

(*Id.* at 6:61-67.) The '120 Patent thus specifically explains a technological solution to improving known methods of machine learning parameter estimation, and the Asserted '120 Claims directed to those improvements are patent eligible. *See, e.g.*, *Aon Re, Inc. v. Zesty.AI, Inc.*, 791 F. Supp. 3d 531, 539 (D. Del. 2025) ("Although the patent does not purport to invent machine learning algorithms, computer hardware, or the general concept of aerial imagery analysis or risk assessment, it does specifically define how these known components must be structured to achieve a supposed technological improvement in accuracy and efficiency to the combination of those components."); *cf. Recentive Analytics, Inc. v. Fox Corp.*, 134 F.4th 1205, 1213 (Fed. Cir. 2025) (holding claims ineligible where "the claims do not delineate steps through which the machine learning technology achieves an improvement").

56. Because the Asserted '120 Claims are directed to a specific improvement in machine learning technology, the claims recite inventions with specific applications or improvements to technologies in the marketplace and cannot be considered abstract or patent ineligible under relevant law.

c. The '259 Patent claims are presumed valid and patent-eligible

57. K.Mizra asserts that at least, and without limitation, Claim 1 of the '259 Patent has been directly infringed, either literally or under the doctrine of equivalents. K.Mizra reserves the right to assert additional claims of the '259 Patent, including both independent and dependent claims, pursuant to the Court's (and other applicable) rules and procedures and as discovery progresses. These claims are referred to herein as the "Asserted '259 Claims."

58. None of the Asserted '259 Claims are directed to abstract ideas, and each employs inventive concepts and is directed to patent-eligible subject matter. All claims of the '259 Patent are also presumed to be valid and enforceable against Amazon and others.

59. Indeed, the '259 Patent's specification and claims demonstrate that the need satisfied by the inventions of the Asserted '259 Claims was long-felt in the industry and thus unconventional. As one example, the '259 Patent explains that computing devices configured to detect certain sounds ("ticks") encountered problems in noisy environments where random sounds may be difficult to distinguish from sounds the devices are configured to detect. (Ex. 4 at 1:26-35.) The '259 Patent specification identifies prior art approaches involving "musical beat detection" and explains why these conventional techniques were inadequate to solve the identified problem. (*Id.* at 1:36-2:3.) The '259 Patent thus describes and claims a specific audio detection and processing system to solve this identified problem.

60. The specification (including the provisions quoted above), the figures (including those included below), and the text related to the figures further illustrate the complex audio detection and processing of the inventions captured by the Asserted '259 Claims. These figures include the following:

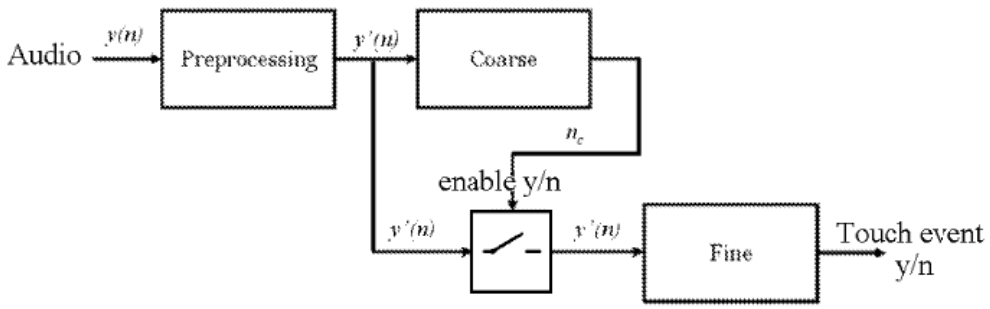


FIG. 1

(*See id.* at Fig. 1.)

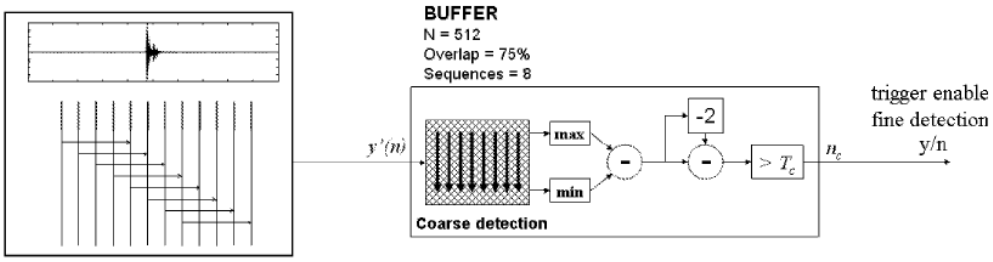


FIG. 3

(*See id.* at Fig. 3.)

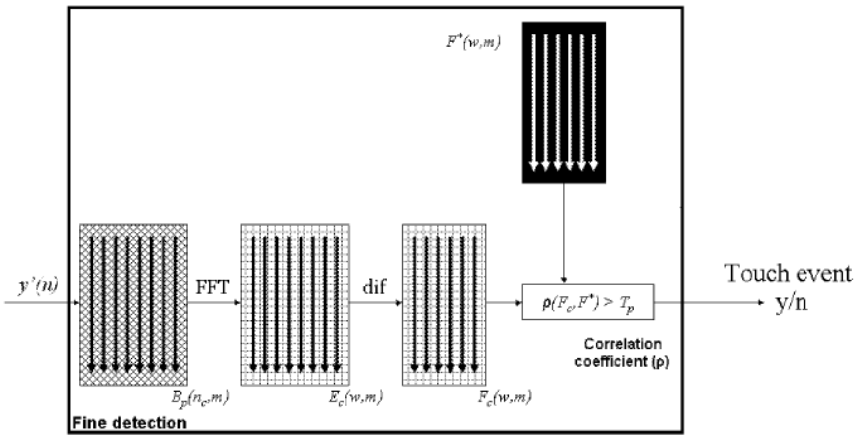


FIG. 4

(*See id.* at Fig. 4.)

61. The foregoing demonstrates that the inventions of the Asserted '259 Claims focus on specific multi-tiered hardware that must interact with unique software to conduct a specific, unconventional audio processing operation in order to properly detect an audio signal. Thus, the Asserted '259 Claims are eligible as a matter of law for patent protection under step one of *Alice*.

62. All actions and steps recited in the Asserted '259 Claims, including the acts of applying an audio signal to a course tick detection processor and enabling further processing by a fine tick detection processor, if necessary, require the involvement of various hardware components running dedicated software both before, during, and after the detection of an audio signal. Said another way, a claim directed to allowing a machine to automatically and dynamically detect a specific audio input in a noisy environment is not simply adding a generic computer component to a fundamentally human process. Rather, the claims "assert a specific and tangible improvement to computer functionality, namely the processing of audio data, that is not abstract." *Hybrid Audio, LLC v. Asus Computer Int'l*, No. 3:17-CV-05947-JD, 2019 WL 3037540, at *4 (N.D. Cal. July 11, 2019).

63. As the specification confirms, the improvement captured by the Asserted '259 Claims is not simply receiving an audio signal, but it is instead a multi-tiered processing system that first subjects a detected signal to coarse processing to determine whether further processing is necessary, and only conducts fine processing if the coarse processor determines that the detected signal is likely a desired "tick." Figure 1 above illustrates the overall system, Figure 3 illustrates the coarse processor, and Figure 4 illustrates the fine processor. (Ex. 4 at 4:13-18.) The specification provides explanations of the coarse detection operation (*id.* at 5:33-6:14) and the fine detection operation (*id.* at 6:15-7:50). The specification explains that the claimed system enables precise detection of desired audio signals while minimizing processor and power use by limiting

fine processing to signals that have first been identified as potential useful signals by a lower-power coarse processing unit. (*Id.* at 7:51-57.) The Asserted '259 Claims are thus directed to a machine-implemented solution resolving a machine-specific problem: a machine's difficulty in detecting audio signals in a noisy environment.

64. The Asserted '259 Claims are thus directed to a machine-implemented process for (1) filtering audio signals in a noisy environment to identify potential "ticks" that require further processing and input and (2) performing fine processing to definitively identify such "ticks" once the initial processing has determined that "ticks" may be present in the audio inputs. As such, the Asserted '259 Claims recite inventions with specific applications or improvements to technologies in the marketplace and cannot be considered abstract or patent ineligible under relevant law.

d. The '649 Patent claims are presumed valid and patent-eligible

65. K.Mizra asserts that at least, and without limitation, Claim 17 of the '649 Patent has been directly infringed, either literally or under the doctrine of equivalents. K.Mizra reserves the right to assert additional claims of the '649 Patent, including both independent and dependent claims, pursuant to the Court's (and other applicable) rules and procedures and as discovery progresses. These claims are referred to herein as the "Asserted '649 Claims."

66. None of the Asserted '649 Claims are directed to abstract ideas, and each employs inventive concepts and is directed to patent-eligible subject matter. All claims of the '649 Patent are also presumed to be valid and enforceable against Amazon and others.

67. Indeed, the '649 Patent's specification and claims demonstrate that the need satisfied by the inventions of the Asserted '649 Claims was long-felt in the industry and thus unconventional. As one example, the '649 Patent explains that known methods of pairing electronic devices using sensory identifiers (such as audio or visual information) were inadequate because

these methods lacked a way to ensure that the devices were in fact proximate to each other. (Ex. 5 at 1:9-25.) In other words, pairing mechanisms that relied on sensory identifiers were prone to errors because sensory information could erroneously be obtained from non-proximate devices. (*Id.*) The '649 Patent thus describes and claims a specific technology to enable devices to be paired using sensory identifiers while ensuring proximity.

68. The specification (including the provisions quoted above), the figures (including those included below), and the text related to the figures further illustrate the complex audio detection and processing of the inventions captured by the Asserted '649 Claims. These figures include the following:

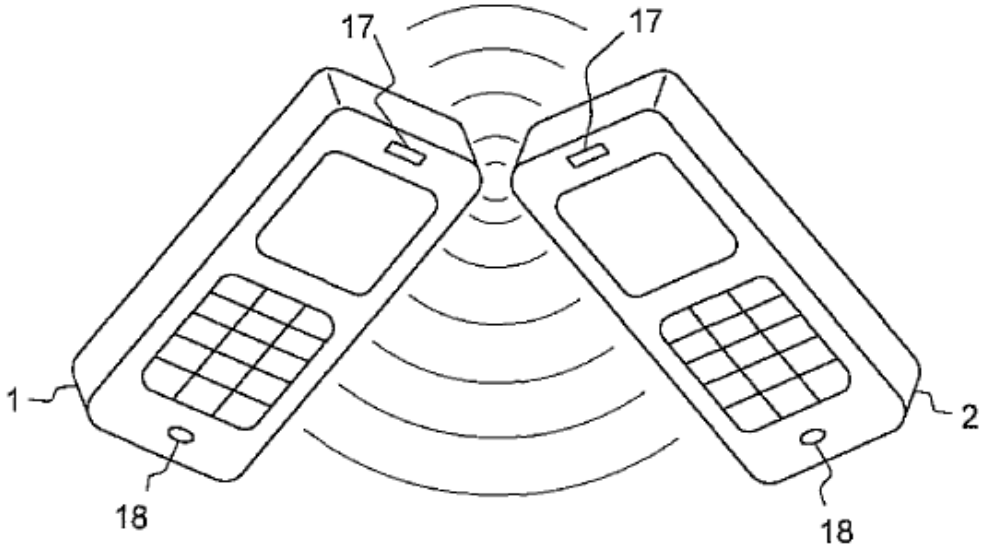


FIG. 1

(*See id.* at Fig. 1.)

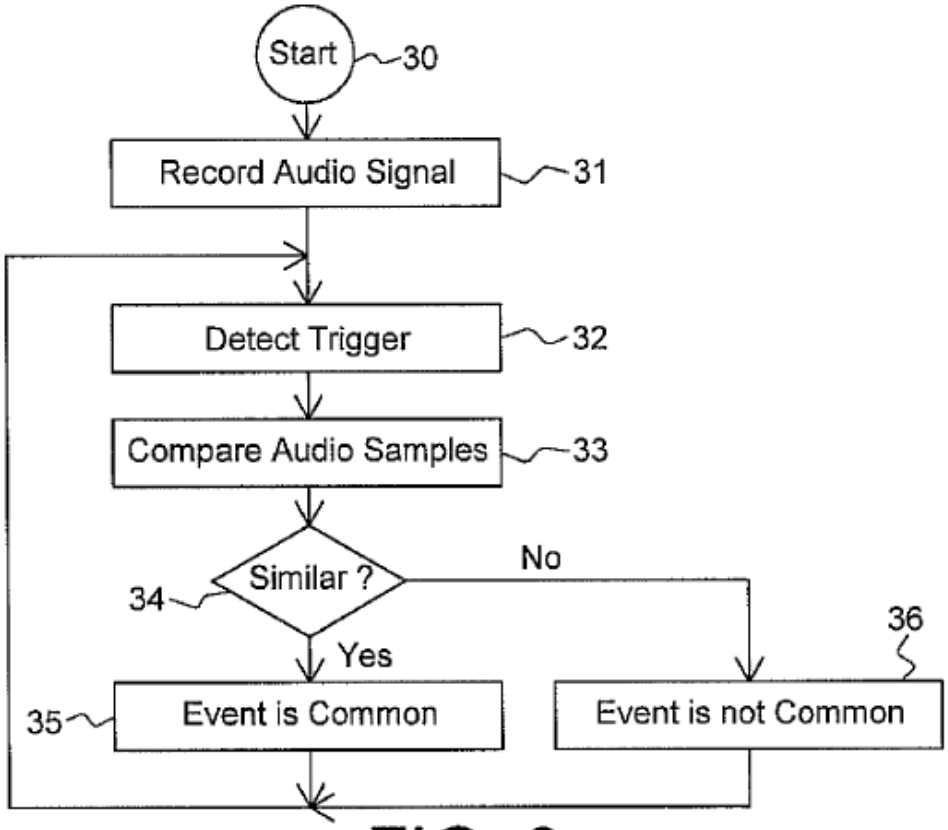


FIG. 3

(*See id.* at Fig. 3.)

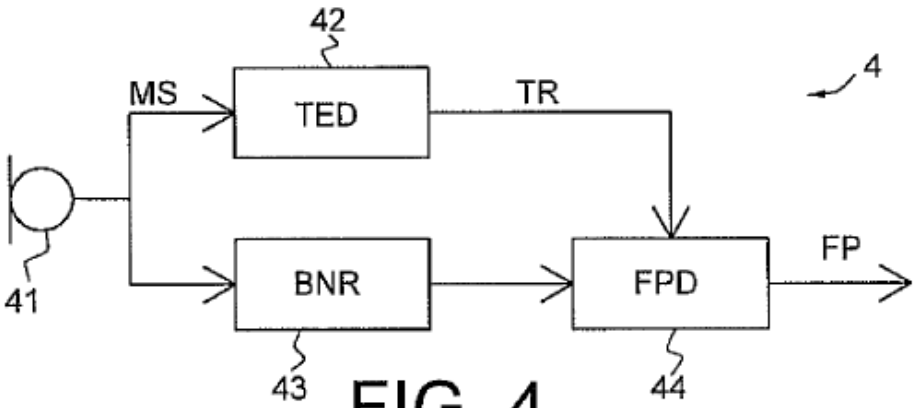


FIG. 4

(*See id.* at Fig. 4.)

69. The foregoing demonstrates that the inventions of the Asserted '649 Claims focus on a specific processing technique to ensure proximity between two devices by comparing audio samples and pairing the devices only if the comparison indicates sufficient similarity to confirm proximity. Thus, the Asserted '649 Claims are eligible as a matter of law for patent protection under step one of *Alice*.

70. All actions and steps recited in the Asserted '649 Claims, including the acts of recording audio signals in each device, detecting a sensory identifier, using the detected sensory identifier for matching the devices, selecting a common time interval of audio samples, comparing the audio samples, and deciding that the devices are proximate only if the respective audio samples are sufficiently similar, require the involvement of various hardware components running dedicated software both before, during, and after the pairing process. Said another way, a claim directed to allowing devices to automatically and dynamically detect and pair to one another only after determining that pairing conditions are satisfied is not simply adding a generic computer component to a fundamentally human process. Rather, like the '259 Patent, the '649 Patent Claims "assert a specific and tangible improvement to computer functionality, namely the processing of audio data, that is not abstract." *Hybrid Audio*, 2019 WL 3037540 at *4.

71. As the specification confirms, the improvement captured by the Asserted '649 Claims is not simply pairing devices based on a sensory identifier, but an enhanced proximity-based pairing process that avoids erroneous pairing attempts (false matches) by ensuring that devices are truly in proximity and meant to be paired. The '649 Patent specification explains that a sensory identifier that acts as a pairing trigger may be sampled by both devices, and the samples compared to ensure that the two devices detected the same sensory identifier. (Ex. 5 at 6:3-18.) The specification further explains that the pairing process may involve specific hardware

components, including a microphone, a "touch event detector," a "background noise recorder," and a "fingerprint determination unit," as shown in Figure 4 above. (*Id.* at 7:31-37.) The Asserted '649 Claims are thus directed to a machine-implemented solution resolving a machine-specific problem: reliably pairing electronic devices while avoiding false matches.

72. The Asserted '649 Claims are thus directed to a machine-implemented process for (1) recording audio signals in electronic devices, (2) detecting sensory identifiers, and (3) using the detected sensory identifiers to match the devices by comparing audio samples to ensure that the devices are in proximity. As such, the Asserted '649 Claims recite inventions with specific applications or improvements to technologies in the marketplace and cannot be considered abstract or patent ineligible under relevant law.

3. <u>The '282 Patent</u>

73. The '282 Patent was initially assigned to Hammerhead Systems, Inc., who thereafter assigned them to Brixham Solutions Ltd. Brixham Solutions Ltd. assigned the '282 Patent to Global Innovation Aggregators LLC, who in turn assigned it to K.Mizra. Hammerhead Systems was a networking company that dissolved in 2009.

74. K.Mizra asserts that at least, and without limitation, Claim 1 of the '282 Patent has been directly infringed, either literally or under the doctrine of equivalents. K.Mizra reserves the right to assert additional claims of the '282 Patent, including both independent and dependent claims, pursuant to the Court's (and other applicable) rules and procedures and as discovery progresses. These claims are referred to herein as the "Asserted '282 Claims."

75. The '282 Patent teaches novel systems and methods for improving scalability and efficiency in handling large volumes of events occurring in network and network management tasks. The '282 Patent's claimed systems and methods combine specific hardware and software

components in unconventional ways. Conventional network management systems (NMS) systems faced several challenges as the size of data communications networks grew in size. As networks expanded, NMS's required increased processing power to manage network performance, identify and fix problems or malfunctions in the network, and plan for further network growth without becoming overwhelmed by the number of events occurring in the network that are generated by network nodes and elements.

76. Through its novel technological innovation, the '282 Patent provides systems and methods for a scalable and robust NMS architecture that can efficiently manage increasing network demands by implementing server clusters, load balancing, and distributed gateway and adapter systems. This allows for incremental addition of network management resources without interrupting network operations, thereby optimizing performance and scalability.

77. The '282 Patent is directed to improvements to computer network technology, more particularly, "a network management system that is robust, scalable, and capable of efficiently handling a growing network." (Ex. 6 at 1:23-25.) A network management system "refers to a system responsible for managing a network, and facilitates the communication between a carrier's OSS and NEs in the network." (*Id.* at 2:37-40.) Prior network management systems ("NMSs") "face[d] many challenges as networks grow in size[,]" including the "thousands of events [that] can be generated when a fault is discovered[,]" resulting in it "easily be[ing] overwhelmed." (*Id.* at 1:17-22.) The '282 Patent specification notes that "[i]t would be desirable to develop a network management system that is robust, scalable, and capable of efficiently handling a growing network." (*Id.* at 1:22-25.) The problem the inventors sought to address was thus rooted in computer networking technology, *i.e.*, the technical limitations of prior art NMSs in their scalability, efficiency, and robustness. (*See id.*) This problem—NMSs with limited scalability,

efficiency, and robustness—is not a problem that arises in longstanding human activity, but is rather rooted in machine-based networking.

78. Recognizing this problem, the inventors provided a "scalable network management system[,]" (*id.* at 2:12-13), with, *e.g.*, a distributed architecture. Specifically, the inventors achieved, *e.g.*, "[h]igh scalability . . . by using server clustering and distributed adapter architecture, with the load balancing scheme to connect the server cluster and NB and SB adapters in northbound and southbound directions respectively." (*Id.* at 7:65-8:2.) In this distributed architecture, "multiple instances of such application servers can run simultaneously to distribute a workload and protect each other [where] [e]ach server instance by itself is a complete functioning NMS server that runs independently[.]" (*Id.* at 8:12-15.) An example of this is shown below:

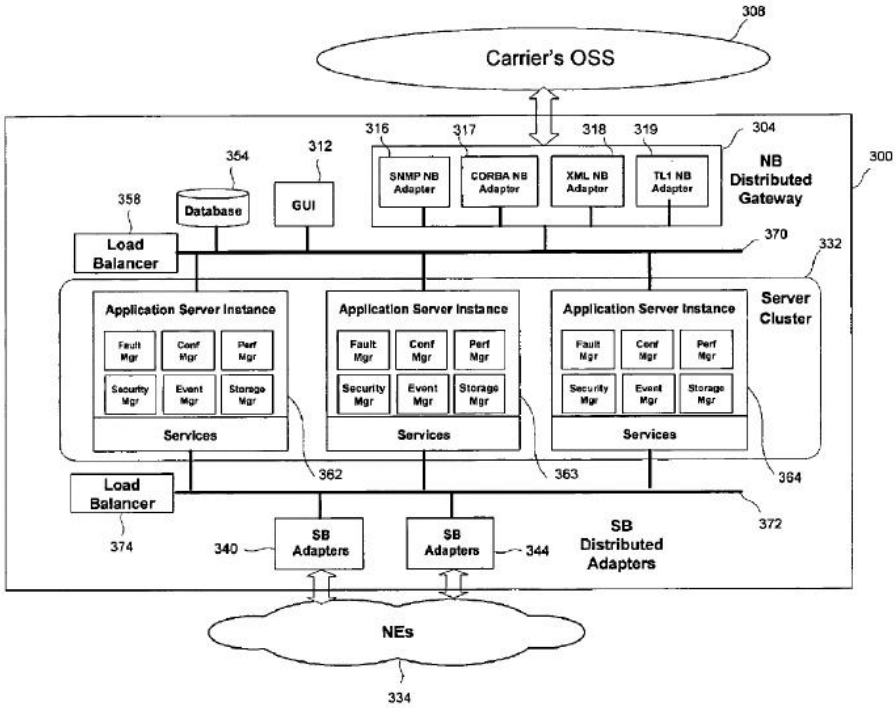


FIG. 3A: NB distributed gateway, SB distributed adapters, server cluster, & load balancers

(*Id.* at Fig. 3A.) The disclosed distributed architecture provides a specific, technical solution to a technical problem and tangible technical benefits rather than abstract ideas, including the ability

to increase NMS processing power, efficiency, and flexibility. (*Id.* at 9:47-60.) The '282 Patent notes that other technical benefits also flow from embodiments of the improved distributed architecture, including:

- "With this architecture of an NB gateway decoupled from NMS application servers, multiple NB gateways, each of which includes a certain number of NB adapters, can be designed and distributed on different workstations for scalability and performance optimization." (*Id.* at 9:39-44.)
- "An instance of the server can also be shut down for any reason without interrupting the server functionality as a whole. When a server is shutdown, its associated software modules (NB gate way 304, GUI312, or SB adapters 340 or 344) automatically re-establish the association with another server instance (one of existing servers 362-364 or a standby server) based on certain criterion, such as selecting the lightest loaded server." (*Id.* at 9:58-65.)
- "[T]he distributed SB adapter-based NMS architecture can Support a flexible network partitioning to manage different management domains based on the amount of management traffic in each domain in a large scale network." (*Id.* at 9:67-10:2.)

The improvement over and difference from prior conventional NMSs is a technical one: an NMS with the improved distributed architecture can be enabled to re-establish an association with another server to ensure functionality during an interruption. Thus, the Asserted '282 Claims are eligible as a matter of law for patent protection under step one of *Alice*.

79. All actions and steps recited in the Asserted '282 Claims, including the acts of receiving, at a first application server instance selected from a plurality of application server

instances based on a load balancing process, first adapter processed information from a first adapter; processing, by the first application server instance, the first adapter processed information based on an event management service to produce application processed information; sending, by the first application server instance, the application processed information to a gateway device, wherein the gateway device is one of a plurality of gateway devices respectively associated with the plurality of application server instances and is configured to transfer the application processed information to a second adapter of a plurality of second adapters configured to process the application processed information and transfer the second adapter processed information to an operation support system device; and in response to determining that the first application server instance has become disabled, facilitating establishing an association between the first adapter and a second application server instance of the plurality of application server instances and between the gateway device and the second application server instance require the involvement of various hardware components running dedicated software both before, during, and after the establishment of an association with a disabled server instance. Said another way, a claim directed to allowing a machine to automatically and dynamically manage associations within a network to ensure that the network operates properly and efficiently with minimal interruptions is not simply adding a generic computer component to a fundamentally human process. Rather, it is a fundamentally mechanical process, an "improvement in the functioning of a" networked system that simply cannot be considered directed to an abstract concept. *Enfish LLC v. Microsoft Corp.*, 822 F.3d 1327, 1339 (Fed. Cir. 2016).

80. During patent examination, the examiner allowed claim 1 because the closest prior art references "do not sufficiently teach or disclose all of the recited limitations of the amended independent claims, and claim 1 in particular, including the recited feature of 'in response to

determining that the first application server instance has become disabled, facilitating establishing an association between the first adapter and a second application server instance of the plurality of application server instances and between the gateway device and the second application server instance ….'" (Ex. 7 (3/17/2014 Notice of Allowance, '282 Patent File History) at 2.) That the Patent Office found the closest prior art did not disclose these additional inventive elements supports their being non-routine and unconventional. Accordingly, claim 1 of the '282 Patent recites additional elements that were not "well-understood, routine, or conventional," which provide "something more" sufficient to transform any alleged abstract idea into eligible subject matter.

B. <u>Amazon's Accused Instrumentalities And Services</u>

81. Amazon is not and has never been a licensee of any of the Asserted Patents nor had or has any rights to use technologies covered by the Asserted Patents. Amazon thus has no ownership or other rights (and is entitled to no rights) relating to the Asserted Patents.

82. Amazon has been making, selling, using, and offering for sale various products and services that infringe the Asserted Patents in violation of 35 U.S.C. § 271. These include, but are not limited to, the following:

- AWS Verified Access ('705 Patent);
- Thread-enabled Amazon devices, including without limitation Thread-enabled Amazon Echo devices ('717 Patent);[1]
- Amazon SageMaker AI ('120 Patent);
- Alexa Emergency Assist ('259 Patent);

[1] The Thread Specification is incorporated by reference.

- Amazon Echo devices incorporating Echo Spatial Perception ('649 Patent); and
- Amazon CloudWatch ('282 Patent)

(collectively, the "Accused Instrumentalities"). The sale, offer for sale, use and/or manufacture in the United States of the Accused Instrumentalities constitutes infringement of at least and without limitation, the asserted claims directly, either literally or under the doctrine of equivalents.

C. K.Mizra's Efforts To Work With Amazon

83. K.Mizra has contacted Amazon on several occasions seeking to discuss Amazon's infringement. For example, K.Mizra contacted Amazon at least as early as January 6, 2025 in a letter addressing Amazon's potential infringement of the '705 Patent, including a claim chart showing Amazon's infringement of the '705 Patent. (*See* Ex. 8.)

84. Thereafter, counsel for K.Mizra communicated with Amazon to resolve this dispute, but no agreement has been reached.

COUNT I
(Patent Infringement under 35 U.S.C. § 271 of the '705 Patent)

85. K.Mizra incorporates paragraphs 1 through 84 as though fully set forth herein.

86. The '705 Patent includes 19 claims.

87. Amazon has directly infringed one or more claims of the '705 Patent by making, importing, using, offering for sale, and/or selling AWS Verified Access, all in violation of 35 U.S.C. § 271(a).

88. Based on publicly available information, AWS Verified Access satisfies every element of at least Claim 19 of the '705 Patent.

89. Claim 19 of the '705 Patent states:

[preamble] A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

[A] detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network,

[B] wherein detecting the insecure condition includes:

[B1] contacting a trusted computing base associated with a trusted platform module within the first host,

[B2] receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness,

[C] wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

[D] when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network,

[E] wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes

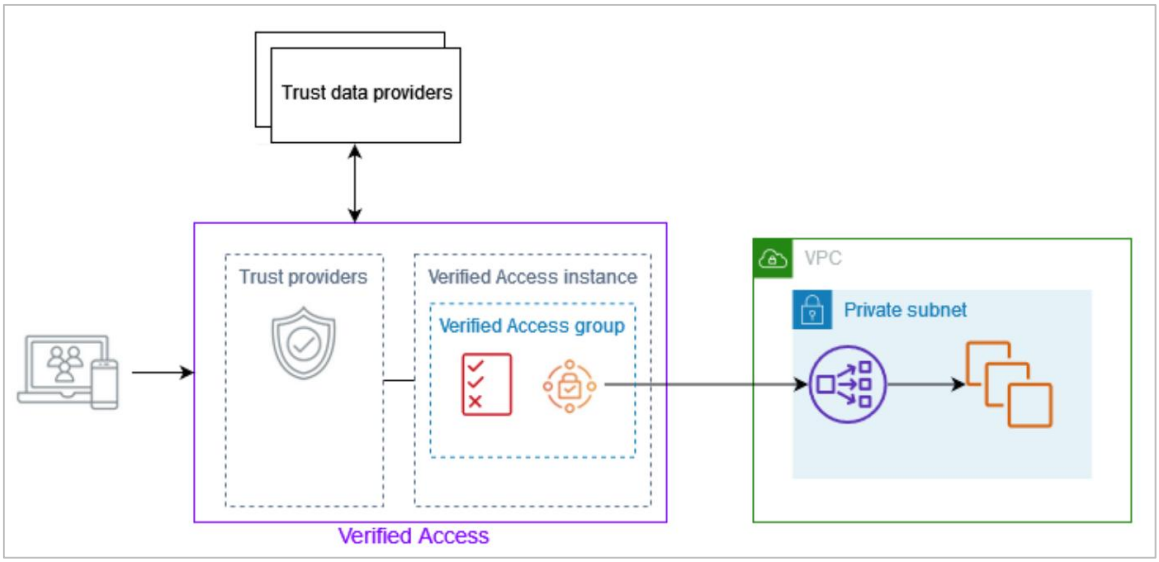
[E1] receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request,

[E2] and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition; and

[F] permitting the first host to communicate with the remediation host.

(Ex. 1 at 22:14-49.)

90. As for the preamble of Claim 19, to the extent that it is determined to be limiting, AWS Verified Access provides the features described in the preamble, which recites a "computer program product for protecting a network." Amazon's AWS Verified Access provides for secure access to protected network resources.



(*See* How to use AWS Verified Access logs to write and troubleshoot access policies, available at https://aws.amazon.com/blogs/security/how-to-use-aws-verified-access-logs-to-write-and-troubleshoot-access-policies/ (last accessed January 2026 and incorporated by reference).) For example, Amazon touts that its AWS Verified Access product "enables secure access to applications" and "evaluates each application request and gives users access to each application only when they meet the specified security requirements." (*Id.*) Amazon offers its AWS Verified Access service for sale as follows:

Pricing table for HTTP(S) applications

There are two pricing dimensions that determine your Verified Access bill for HTTP(S) applications 1) hourly charges for associated applications, and 2) per-GB charges for data processed.

Application hours

You are charged per hour for each application associated with an active HTTP(S) Verified Access endpoint. Each partial application hour (app-hour) consumed is billed as a full hour. For example, if you associate 10 applications with 10 Verified Access endpoints, each for an hour, you will be charged 10 app-hours.

GB of data processed

You are also charged per GB for data processed by Verified Access. When users request access or connect to your applications, Verified Access processes the data flowing between users and the application. For example, if all your applications using Verified Access require 10 GB of total data processing, you will be charged for that 10 GB of data processed.

Tiered pricing for app hours	Price
1 to 148,800 app hours (covers up to 200 apps in a month)	$0.27/hr
148,800+ app hours	$0.20/hr
Note: 200 apps associated for 31 days equals 148,800 app hours	
Per GB of data processed	$0.02

Example 2:

You created a Verified Access in US East (Ohio) and associated 300 applications. These 300 applications were associated for a month with 31 days. This will result in 223,200 app hours. Also, each application required 1 GB for data processing. The total charge will be $55,062.00, and the below table provides the details.

First 148,800 app hours	148,800 hours * $0.27	$40,176.00
Next 74,400 (223,200-148,800) app hours	74,400 hours * $0.20	$14,880.00
Data processing	1 GB * 300 applications * $0.02	$6.00
Total charge		**$55,062.00**

(*See* AWS Verified Access Pricing, available at https://aws.amazon.com/verified-access/pricing/) (last accessed January 2026 and incorporated by reference).) Accordingly, and to the extent that the preamble of Claim 19 is somehow limiting, AWS Verified Access would meet the limitation.

91. Limitation A of Claim 19 requires "detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network." AWS Verified Access also meets all the requirements of limitation A of Claim 19. Amazon's AWS Verified Access "enhances [owners'] security posture by allowing [owners] to define fine-grained access policies based on a user's identity and device security state, and enforcing policies on every access request."

Why AWS Verified Access?

AWS Verified Access provides secure access to corporate applications and resources without a VPN. It enhances your security posture by allowing you to define fine-grained access policies based on a user's identity and device security state, and enforcing policies on every access request. It simplifies security operations by allowing administrators to create, group, and manage access policies for applications and resources with similar security requirements from a single interface. Verified Access logs each access attempt, so you can efficiently respond to security and connectivity incidents.

Improve your security posture by defining granular access policies based upon user's identity and device security status for your corporate applications including web-based applications accessed using browsers and infrastructure resources such as databases and EC2 instances.

Application owners can apply zero trust access controls to their corporate applications by constantly verifying each access request against granular, conditional access policies. This ensures that access is granted per application only when specific security requirements like user identity and device security posture are met and maintained.

Get comprehensive logging and visibility into access attempts to quickly identify and resolve security and connectivity incidents.

(Secure Remote Access – AWS Verified Access – AWS, available at https://aws.amazon.com/verified-access/ (last accessed January 2026 and incorporated by reference).) AWS Verified Access allows owners to "apply zero trust access controls to their corporate applications by constantly verifying each access request against granular, conditional access policies," which "ensures that access is granted per application only when specific security

requirements like user identity and device security posture are met and maintained. (*Id.*) AWS Verified Access supports device trust providers such as Jamf, CrowdStrike and JumpCloud.

> If you plan to incorporate device trust context into your access policies, then you will need either the AWS Verified Access browser extension, or another partner's browser extension. Verified Access currently supports Google Chrome and Mozilla Firefox browsers.
>
> We currently support three device trust providers: Jamf (which supports macOS devices), CrowdStrike (which supports Windows 11 and Windows 10 devices), and JumpCloud (which supports both Windows and MacOS).
>
> - If you're using **Jamf** trust data in your policies, your users must download and install the AWS Verified Access browser extension from the Chrome web store or Firefox Add-on site on their devices.
> - If you are using **CrowdStrike** trust data in your policies, first your users need to install the AWS Verified Access Native Messaging Host (direct download link). This component is required to get the trust data from the CrowdStrike agent running on users' devices. Then, after installing this component, users must install the AWS Verified Access browser extension from the Chrome web store or Firefox Add-on site on their devices.
> - If you're using **JumpCloud**, your users must have the JumpCloud browser extension from the Chrome web store or Firefox Add-on site installed on their devices.

(Third-party trust provider context for Verified Access trust data, available at https://docs.aws.amazon.com/verified-access/latest/ug/trust-data-third-party-trust.html (last accessed January 2026 and incorporated by reference).)

92. Limitation B1 of Claim 19 requires that "detecting [an] insecure condition includes . . . contacting a trusted computing base associated with a trusted platform module within the first host." AWS Verified Access meets these requirements by, for example, accessing data or security applications located on the device attempting to access a network resource. For example, using JumpCloud, AWS Verified Access accesses a "Secure Enclave" on a macOS device or a "Trusted Platform Module (TPM)" on a Windows or Linux device.

> AWS and JumpCloud have partnered to offer a VPN alternative for securely accessing company applications over the web using JumpCloud Go™, JumpCloud SSO (SAML or OIDC) and AWS Verified Access (AVA). With this collaboration, JumpCloud can authenticate and authorize devices and identities in one console and provide access control for AWS Verified Access for managed devices. See our blog for more information about our partnership.

(Integrate with AWS Verified Access, available at https://jumpcloud.com/support/integrate-with-aws-verified-access (last accessed January 2026 and incorporated by reference).)

JumpCloud Go supports the following device types that meet these hardware requirements:

- macOS devices with a Secure Enclave.
- Windows devices with a Trusted Platform Module (TPM) 2.0.
- Linux devices (GNOME-based distros) with a Trusted Platform Module (TPM) 2.0.

The JumpCloud agent has to be installed and running on macOS, Windows, and Linux devices. See Install the JumpCloud Agent.

(Get Started: JumpCloud Go™, available at https://jumpcloud.com/support/get-started-jumpcloud-go (last accessed January 2026 and incorporated by reference).)

93. Limitation B2 of Claim 19 requires that "detecting the insecure condition" also includes "receiving a response and determining whether the response includes a valid digitally signed attestation of cleanliness." AWS Verified Access also meets all the requirements of limitation B2. For example, when a remote device ("first host") first initiates a request to access corporate resources (*i.e.*, a protected network), AWS Verified Access (in conjunction with the JumpCloud service), the device undergoes an enrolment process whereby it is assigned an ID Token. Thereafter, a DUK (Device User Key) is created for the device, which is added to a "JWT (JSON Web Token) that is signed by the JumpCloud Agent certificate unique to the device and used by the JumpCloud Agent to authenticate all traffic back to JumpCloud."

JumpCloud Go anchors its trust to both the device and the user. During the enrollment process the user authenticates to JumpCloud's OpenID Connect service and is issued an ID Token for the JumpCloud Go application. A new DUK (Device User Key) is created from the Secure Enclave (on Apple devices) or the Trusted Platform Module (TPM) (on Windows systems). This is added to a JWT (JSON Web Token) that is then signed by the JumpCloud Agent certificate unique to the device and used by the JumpCloud Agent to authenticate all traffic back to JumpCloud.

(JumpCloud Go™, available at https://jumpcloud.com/wp-content/uploads/2023/11/202310-WP-JumpCloudGo.pdf (last accessed January 2026 and incorporated by reference).) The JWT's signature is subject to a multi-step validation and inspection process:

> The JumpCloud Go enrollment request is validated by JumpCloud's DURT (Device User Refresh Token) API through the following process:
>
> 1. The JWT's signature is validated by first ensuring the certificate in the x5c claim in the JWT header is a valid JumpCloud Agent certificate in good standing.
> 2. The JWT signature is checked to ensure it was signed by the private key of the same agent certificate.
> 3. The JWT is inspected to ensure it hasn't expired and is otherwise valid.
> 4. The embedded ID Token is validated back to the JumpCloud OpenID Connect service via its embedded signature using the OIDC JWKS.

(*Id.*)

94. Limitation C of Claim 19 requires that "the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch, or a patch level associated with a software component on the first host." AWS Verified Access meets these requirements. For example, AWS Verified Access utilizes a "trust provider," which "is a service that sends information about users and devices to AWS Verified Access, including "device information such as installed security patches or anti-virus software version."

> A trust provider is a service that sends information about users and devices to AWS Verified Access. This information is called trust context. It can include attributes based on user identity, such as an email address or membership in the "sales" organization, or device information such as installed security patches or anti-virus software version.

(Trust providers for Verified Access, available at https://docs.aws.amazon.com/verified-access/latest/ug/trust-providers.html (last accessed January 2026 and incorporated by reference).)

95. Limitation D of Claim 19 requires that "when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host,

including by preventing the first host from sending data to one or more other hosts associated with the protected network." AWS Verified Access meets these requirements by, *e.g.*, denying all application requests that do not meet the specified security requirements.

> When a user tries to access an application, Verified Access gets their data from the trust provider and evaluates it against the policies that you set for the application. Verified Access grants access to the requested application only if the user meets your specified security requirements. All application requests are denied by default, until a policy is defined.
>
> In addition, Verified Access logs every access attempt, to help you respond quickly to security incidents and audit requests.

> **Verified Access instances** – An instance evaluates application requests and grants access only when your security requirements are met.
>
> **Verified Access endpoints** – Each endpoint represents an application. In the diagram above, the application is hosted on EC2 instances that are targets of a load balancer.
>
> **Verified Access group** – A collection of Verified Access endpoints. We recommend that you group the endpoints for applications with similar security requirements to simplify policy administration. For example, you can group the endpoints for all your sales applications together.
>
> **Access policies** – A set of user-defined rules that determine whether to allow or deny access to an application. You can specify a combination of factors, including user identity and device security state. You create a group access policy for each Verified Access group, which is inherited by all endpoints in the group. You can optionally create application-specific policies and attach them to specific endpoints.
>
> **Trust providers** – A service that manages user identities or device security state. Verified Access works with both AWS and third-party trust providers. You must attach at least one trust provider to each Verified Access instance. You can attach a single identity trust provider and multiple device trust providers to each Verified Access instance.
>
> **Trust data** – The security-related data for users or devices that your trust provider sends to Verified Access. Also referred to as *user claims* or *trust context*. For example, the email address of a user or the operating system version of a device. Verified Access evaluates this data against your access policies when it receives each request to access an application.

(How Verified Access Works, available at https://docs.aws.amazon.com/verified-access/latest/ug/how-it-works.html (last accessed January 2026 and incorporated by reference).)

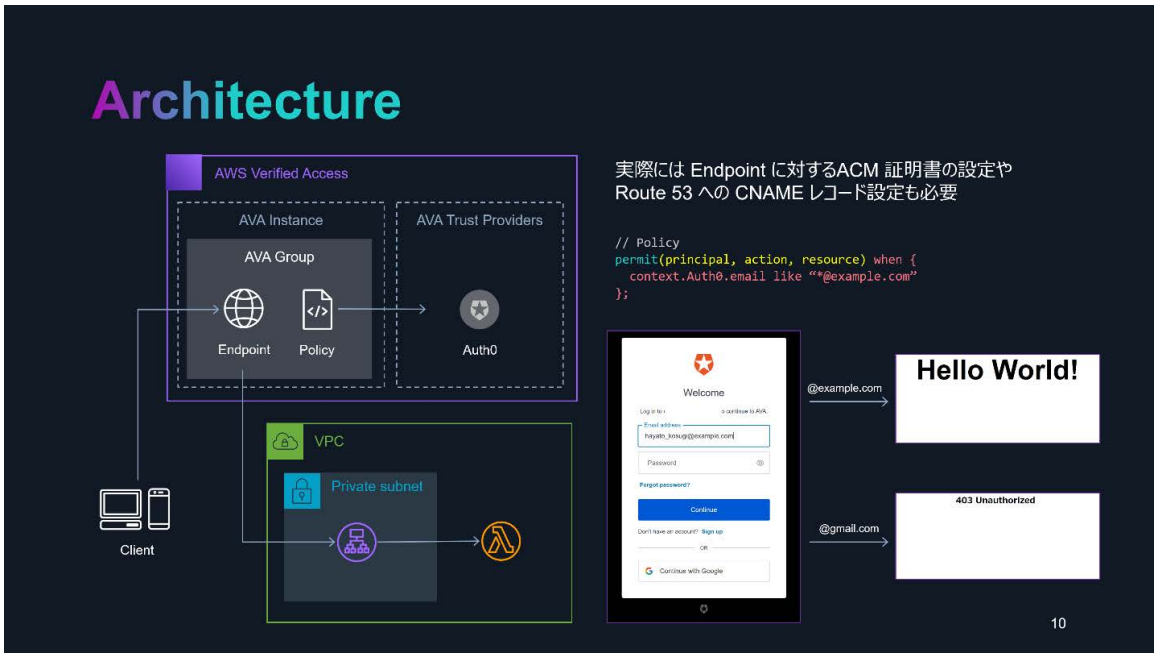
96. Limitation E1 of Claim 19 requires that "preventing the first host from sending data to one or more other hosts associated with the protected network includes . . . receiving a service request sent by the first host [and] serving a quarantine notification page to the first host when the service request comprises a web server request." AWS Verified Access meets these requirements, *e.g.*, because a user attempting to access an application protected by AWS Verified Access that does not meet the security requirements will receive a 403 Unauthorized error message.

> Use a web browser with the Verified Access extension to test access to the application. Before signing in, you should be redirected to the Okta sign-in page. After successful authentication, if the device does not have a ZTA score above the threshold in the group policy, then the user receives a 403 Unauthorized error.

(Integrating AWS Verified Access with device trust providers, available at https://aws.amazon.com/blogs/networking-and-content-delivery/integrating-aws-verified-access-with-device-trust-providers/ (last accessed January 2026 and incorporated by reference).)

97. Limitation E2 of Claim 19 requires that "preventing the first host from sending data to one or more other hosts associated with the protected network includes" "in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition." AWS Verified Access meets all the requirements of limitation E2 of Claim 19. For example, in connection with a 403 Unauthorized error message, AWS Verified Access provides an IP address of a quarantine server configured to serve the quarantine notification page (*e.g.*, a 403 Unauthorized error message). Accordingly, AWS Verified Access meets limitation E2 of Claim 19.

98. Limitation F of Claim 19 requires "permitting the first host to communicate with the remediation host." AWS Verified Access meets this limitation, for example, by permitting communication with "AVA Trust Providers" that act as remediation hosts.



(*See* https://qiita.com/hayao_k/items/5f3b8a88a0ea75828f95 (last accessed January 2026 and incorporated by reference).) Accordingly, AWS Verified Access meets limitation F of Claim 19.

99. Additionally, and/or alternatively, Amazon has indirectly infringed and continues to indirectly infringe one or more of the claims of the '705 Patent, in violation of 35 U.S.C. § 271(b) by actively inducing users of AWS Verified Access to directly infringe one or more claims of the '705 Patent. For example, (a) Amazon had actual knowledge of or was willfully blind to the existence of the '705 Patent and its infringement thereof no later than January 6, 2025, when it received the letter attached as Exhibit 8, and (b) Amazon intentionally causes, urges, or encourages users of AWS Verified Access to take action that, when taken, directly infringe one or more claims of the '705 Patent. Amazon's encouragement is accomplished by promoting, advertising, and instructing customers and potential customers to use AWS Verified Access, including infringing uses thereof. Amazon knows (based on the claim chart previously provided by K.Mizra and/or after reading this Complaint should know) that its actions will induce users of AWS Verified

Access to directly infringe one or more claims of the '705 Patent, and users thereof directly infringe one or more claims of the '705 Patent. For instance, at a minimum, Amazon has supplied and continues to supply AWS Verified Access to customers while knowing that use thereof will infringe one or more claims of the '705 Patent.

100. Due to Amazon's knowledge of the '705 Patent and of its infringement thereof no later than January 6, 2025, Amazon's infringement of the '705 Patent has been and continues to be willful since no later than that date.

101. Amazon's acts of infringement have occurred within this District and elsewhere throughout the United States.

102. As a result of Amazon's infringing conduct, K.Mizra has suffered damages. Amazon is liable to K.Mizra in an amount that adequately compensates K.Mizra for Amazon's infringement in an amount that is no less than a fully paid-up, lump-sum, reasonable royalty, together with interest and costs as fixed by this Court under 25 U.S.C. § 284.

COUNT II
(Patent Infringement under 35 U.S.C. § 271 of the '717 Patent)

103. K.Mizra incorporates paragraphs 1 through 102 as though fully set forth herein.

104. The '717 Patent includes 20 claims.

105. Amazon has directly infringed one or more claims of the '717 Patent at least by using Thread-enabled Amazon devices in a Thread-enabled network, including without limitation Thread-enabled Amazon Echo devices ("the '717 Accused Instrumentalities") in violation of 35 U.S.C. § 271(a).

106. Based on publicly available information, the '717 Accused Instrumentalities satisfies every element of at least Claim 1 of the '717 Patent.

107. Claim 1 of the '717 Patent states:

> [preamble] A wireless communication network that comprises
>
> [A] an association unit and
>
> [B] a plurality of stations,
>
> [C] each station being configured to start up in a not-associated state and to transmit association request messages in the not-associated state,
>
> [D] each station being configured to switch to an associated state upon reception of an association grant in response to one of its association request messages,
>
> [E] the association request message and/or the association grant in response thereto establishing an operating route associated with the station for exchanging operating messages with the association unit during subsequent operation,
>
> [F] the operating route associated with the station running through a source of the association grant in response to said one of the association request messages and,
>
> [G] if the source of the association grant is not the association unit, the operating route between the source of the association grant and the association unit previously established for the source of the association grant in response to association request messages and/or the association grants for intermediate stations, in which network
>
> [H] the association unit is configured to transmit association grants as responses to received ones of the association request messages;
>
> [I] at least part of the stations is configured to become active to transmit association grants as responses to received ones of the association request messages, but only after switching to the associated state upon an association grant from the association unit or one of the at least part of the stations switched to the associated state directly or indirectly based on an association grant from the association unit.

(Ex. 2 at 14:28-52.)

108. As for the preamble of Claim 1, to the extent that it is determined to be limiting, the '717 Accused Instrumentalities provide the features described in the preamble, which recites a wireless communication network. A Thread-enabled device is configured to operate within a wireless communication network.

109. Limitation A of Claim 1 requires an association unit. A Thread-enabled wireless network will include a "Leader" responsible for managing router ID assignment and acting as a central arbiter of the network configuration state. The Leader is an association unit.

110. Limitation B of Claim 1 requires a plurality of stations. The '717 Accused Instrumentalities are "Thread Devices" (IPv6-capable devices having at least one Thread Interface and implementing the Thread Specification) and thus stations.

111. Limitation C of Claim 1 requires "each station being configured to start up in a not-associated state and to transmit association request messages in the not-associated state." The '717 Accused Instrumentalities meet this limitation by, *e.g.*, being configured to start up in a "Detached" state according to the Thread Specification and transmit "Parent Request" signals according to the Thread Specification.

112. Limitation D of Claim 1 requires "each station being configured to switch to an associated state upon reception of an association grant in response to one of its association request messages." The '717 Accused Instrumentalities meet this limitation by, *e.g.*, switching to an "Attached" state according to the Thread Specification upon receipt of a "Child ID Response" signal from the Leader according to the Thread Specification.

113. Limitation E of Claim 1 requires "the association request message and/or the association grant in response thereto establishing an operating route associated with the station for exchanging operating messages with the association unit during subsequent operation." Use of the

'717 Accused Instrumentalities in a Thread-enabled network meets this limitation by, *e.g.*, establishing a "Route64 TLV" link between the association unit and the station according to the Thread Specification following receipt of the Child ID Response.

114. Limitation F of Claim 1 requires "the operating route associated with the station running through a source of the association grant in response to said one of the association request messages." Use of the '717 Accused Instrumentalities in a Thread-enabled network meets this limitation. For example, the Response TLV runs through the Parent Router.

115. Limitation G of Claim 1 requires "if the source of the association grant is not the association unit, the operating route between the source of the association grant and the association unit previously established for the source of the association grant in response to association request messages and/or the association grants for intermediate stations." Use of the '717 Accused Instrumentalities in a Thread-enabled wireless network meets this limitation. For example, access to the network may be provided by a device other than the Parent, in which case the Thread Device will still be able to access the Parent via the operating route.

116. Limitation H of Claim 1 requires that "the association unit is configured to transmit association grants as responses to received ones of the association request messages." A Thread-enabled wireless network including the '717 Accused Instrumentalities meets this limitation because, *e.g.*, a router in such a network is configured to allocate Child IDs to new '717 Accused Instrumentalities in the network.

117. Limitation I of Claim 1 requires that "at least part of the stations is configured to become active to transmit association grants as responses to received ones of the association request messages, but only after switching to the associated state upon an association grant from the association unit or one of the at least part of the stations switched to the associated state directly

or indirectly based on an association grant from the association unit." The '717 Accused Instrumentalities meet this limitation because, *e.g.*, they are configured to act as routers within a Thread-enabled wireless network.

118. Additionally, and/or alternatively, Amazon has indirectly infringed and continues to indirectly infringe one or more of the claims of the '717 Patent, in violation of 35 U.S.C. § 271(b) by actively inducing users of the '717 Accused Instrumentalities to directly infringe one or more claims of the '717 Patent. For example, (a) Amazon had actual knowledge of or was willfully blind to the existence of the '717 Patent and its infringement thereof no later than the filing of this Complaint, and (b) Amazon intentionally causes, urges, or encourages users of the '717 Accused Instrumentalities to take action that, when taken, directly infringe one or more claims of the '717 Patent. Amazon's encouragement is accomplished by promoting, advertising, and instructing customers and potential customers to use the '717 Accused Instrumentalities, including infringing uses thereof. Amazon knows (based on the claim chart previously provided by K.Mizra and/or after reading this Complaint should know) that its actions will induce users of the '717 Accused Instrumentalities to directly infringe one or more claims of the '717 Patent, and users thereof directly infringe one or more claims of the '717 Patent. For instance, at a minimum, Amazon has supplied and continues to supply the '717 Accused Instrumentalities to customers while knowing that use thereof will infringe one or more claims of the '717 Patent.

119. Due to Amazon's knowledge of the '717 Patent and of its infringement thereof no later than the filing of this Complaint, Amazon's infringement of the '717 Patent has been and continues to be willful since no later than that date.

120. Amazon's acts of infringement have occurred within this District and elsewhere throughout the United States.

121. As a result of Amazon's infringing conduct, K.Mizra has suffered damages. Amazon is liable to K.Mizra in an amount that adequately compensates K.Mizra for Amazon's infringement in an amount that is no less than a fully paid-up, lump-sum, reasonable royalty, together with interest and costs as fixed by this Court under 25 U.S.C. § 284.

COUNT III
(Patent Infringement under 35 U.S.C. § 271 of the '120 Patent)

122. K.Mizra incorporates paragraphs 1 through 121 as though fully set forth herein.

123. The '120 Patent includes 16 claims.

124. Amazon has directly infringed one or more claims of the '120 Patent by making, importing, using, offering for sale, and/or selling Amazon SageMaker AI, all in violation of 35 U.S.C. § 271(a).

125. Based on publicly available information, Amazon SageMaker AI satisfies every element of at least Claim 1 of the '120 Patent.

126. Claim 1 of the '120 Patent states:

> [preamble] A method of determining hyperparameters of a classifier in a machine learning system by iteratively producing an estimate of a target hyperparameter vector, each iteration comprising the steps of::
>
> [A] drawing a random sample of hyperparameter vectors from a set of possible hyperparameter vectors,
>
> [B] updating the estimate of the target hyperparameter vector by using the random sample, and
>
> [C] selecting, from the random sample of hyperparameter vectors, a hyperparameter vector producing a best result in the present and any previous iterations, and
>
> [D] wherein the step of updating the estimate of the target hyperparameter vector uses said hyperparameter vector producing the best result.

(Ex. 3 at 7:14-28.)

127. As for the preamble of Claim 1, to the extent that it is determined to be limiting, Amazon SageMaker AI provides the features described in the preamble, which recites a "method of determining hyperparameters of a classifier in a machine learning system by iteratively producing an estimate of a target hyperparameter vector." Amazon SageMaker AI performs a "hyperparameter tuning" function called "automatic model tuning (AMT)."

> Amazon SageMaker AI automatic model tuning (AMT) finds the best version of a model by running many training jobs on your dataset. Amazon SageMaker AI automatic model tuning (AMT) is also known as hyperparameter tuning. To do this, AMT uses the algorithm and ranges of hyperparameters that you specify. It then chooses the hyperparameter values that creates a model that performs the best, as measured by a metric that you choose.

("Automatic model tuning with SageMaker AI, available at https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning.html (last accessed January 2026 and incorporated by reference).) AMT "finds the best version of a model by running many training jobs on [a] dataset," "launch[ing] training jobs to find an optimal configuration of hyperparameters." (*Id.*)

> A hyperparameter is a high-level parameter that influences the learning process during model training. To get the best model predictions, you can optimize a hyperparameter configuration or set hyperparameter values. The process of finding an optimal configuration is called hyperparameter tuning. To configure and launch a hyperparameter tuning job, complete the steps in these guides.

> The hyperparameter tuning job will launch training jobs to find an optimal configuration of hyperparameters. These training jobs should be configured using the SageMaker AI `CreateHyperParameterTuningJob` API.

(Configure and Launch a Hyperparameter Tuning Job, available at https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-ex-tuning-job.html

(last accessed January 2026 and incorporated by reference).) Accordingly, and to the extent that the preamble of Claim 1 is somehow limiting, Amazon SageMaker AI would meet the limitation.

128. Limitation A of Claim 1 requires "drawing a random sample of hyperparameter vectors from a set of possible hyperparameter vectors." Amazon SageMaker AI meets this limitation. For example, Amazon SageMaker AI includes a "random search" function in which "hyperparameter tuning chooses a random combination of hyperparameter values in the ranges that you specify for each training job it launches."

> **Random search**
>
> When using random search, hyperparameter tuning chooses a random combination of hyperparameter values in the ranges that you specify for each training job it launches. The choice of hyperparameter values doesn't depend on the results of previous training jobs. As a result, you can run the maximum number of concurrent training jobs without changing the performance of the tuning.

(Understand the hyperparameter tuning strategies available in Amazon SageMaker AI, available at https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-how-it-works.html (last accessed January 2026 and incorporated by reference).)

129. Limitation B of Claim 1 requires "updating the estimate of the target hyperparameter vector by using the random sample," and limitation C of Claim 1 requires "selecting, from the random sample of hyperparameter vectors, a hyperparameter vector producing a best result in the present and any previous iterations." Amazon SageMaker AI meets these limitations. For example, Amazon SageMaker AI "chooses the hyperparameter values that result in a model that performs the best," and "[w]hile the hyperparameter tuning job is in progress, the best training job is the one that has returned the best objective metric so far."

> The following section explains how to use an algorithm resource to run a hyperparameter tuning job in Amazon SageMaker AI. A hyperparameter tuning job finds the best version of a model by running many training jobs on your dataset using the algorithm and ranges of hyperparameters that you specify. It then chooses the hyperparameter values that result in a model that performs the best, as measured by a metric that you choose. For more information, see Automatic model tuning with SageMaker AI.

(Use an Algorithm to Run a Hyperparameter Tuning Job, available at https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-mkt-algo-tune.html (last accessed January 2026 and incorporated by reference).)

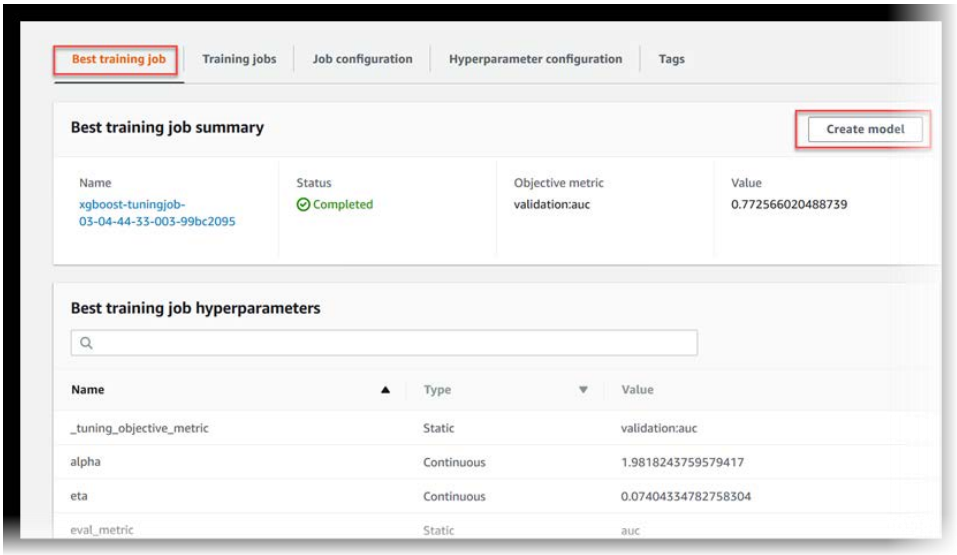
> A hyperparameter tuning job uses the objective metric that each training job returns to evaluate training jobs. While the hyperparameter tuning job is in progress, the best training job is the one that has returned the best objective metric so far. After the hyperparameter tuning job is complete, the best training job is the one that returned the best objective metric.

(Configure and Launch a Hyperparameter Tuning Job, available at https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-ex-tuning-job.html (last accessed January 2026 and incorporated by reference).) This suggests that, in the "random search" tuning method, the optimal hyperparameter combination is chosen from the random sample in each training job/iteration. Amazon further explains that "[u]sing information from previous hyperparameter tuning jobs can help increase the performance of the new hyperparameter tuning job by making the search for the best combination of hyperparameters more efficient," further indicating that Amazon SageMaker AI selects the best hyperparameter combinations for each training job based on the previous iteration.

> Use warm start to start a hyperparameter tuning job using one or more previous tuning jobs as a starting point. The results of previous tuning jobs are used to inform which combinations of hyperparameters to search over in the new tuning job. Hyperparameter tuning uses either Bayesian or random search to choose combinations of hyperparameter values from ranges that you specify. For more information, see Understand the hyperparameter tuning strategies available in Amazon SageMaker AI. Using information from previous hyperparameter tuning jobs can help increase the performance of the new hyperparameter tuning job by making the search for the best combination of hyperparameters more efficient.

(Run a Warm Start Hyperparameter Tuning Job, available at https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-warm-start.html (last accessed January 2026 and incorporated by reference).)

130. Limitation D of Claim 1 requires that " the step of updating the estimate of the target hyperparameter vector uses said hyperparameter vector producing the best result." Amazon SageMaker AI meets this limitation, as the best training job hyperparameters are stored separately, indicating that the best training job hyperparameters ("the estimate of the target hyperparameter vector") are updated whenever a new optimal set is found.



(Configure and Launch a Hyperparameter Tuning Job, available at https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-ex-tuning-job.html (last accessed January 2026 and incorporated by reference).)

131. Additionally, and/or alternatively, Amazon has indirectly infringed and continues to indirectly infringe one or more of the claims of the '120 Patent, in violation of 35 U.S.C. § 271(b) by actively inducing users of Amazon SageMaker AI to directly infringe one or more claims of the '120 Patent. For example, (a) Amazon had actual knowledge of or was willfully blind to the existence of the '120 Patent and its infringement thereof no later than the filing of this Complaint, and (b) Amazon intentionally causes, urges, or encourages users of Amazon SageMaker AI to take action that, when taken, directly infringe one or more claims of the '120 Patent. Amazon's encouragement is accomplished by promoting, advertising, and instructing customers and potential customers to use Amazon SageMaker AI, including infringing uses thereof. Amazon knows (based on the claim chart previously provided by K.Mizra and/or after reading this Complaint should know) that its actions will induce users of Amazon SageMaker AI to directly infringe one or more claims of the '120 Patent, and users thereof directly infringe one or more claims of the '120 Patent. For instance, at a minimum, Amazon has supplied and continues to supply Amazon SageMaker AI to customers while knowing that use thereof will infringe one or more claims of the '120 Patent.

132. Due to Amazon's knowledge of the '120 Patent and of its infringement thereof no later than the filing of this Complaint, Amazon's infringement of the '120 Patent has been and continues to be willful since no later than that date.

133. Amazon's acts of infringement have occurred within this District and elsewhere throughout the United States.

134. As a result of Amazon's infringing conduct, K.Mizra has suffered damages. Amazon is liable to K.Mizra in an amount that adequately compensates K.Mizra for Amazon's infringement in an amount that is no less than a fully paid-up, lump-sum, reasonable royalty, together with interest and costs as fixed by this Court under 25 U.S.C. § 284.

COUNT IV
(Patent Infringement under 35 U.S.C. § 271 of the '259 Patent)

135. K.Mizra incorporates paragraphs 1 through 134 as though fully set forth herein.

136. The '259 Patent includes 23 claims.

137. Amazon has directly infringed one or more claims of the '259 Patent by making, importing, using, offering for sale, and/or selling Alexa Emergency Assist, all in violation of 35 U.S.C. § 271(a).

138. Based on publicly available information, Alexa Emergency Assist satisfies every element of at least Claim 1 of the '259 Patent.

139. Claim 1 of the '259 Patent states:

> [preamble] Method for detecting ticks on a device with which an audio signal is received in a noisy environment, from the audio signal which includes a stream of audio samples (y(n), (y′(n)), the method comprising the steps of:
>
> [A] applying the audio signal to a coarse tick detection processor arranged to decide whether the audio signal likely includes a tick; and
>
> [B] only if the course tick processor decides that the audio signal likely includes a tick, enabling a fine tick detection processor to decide, by more thorough processing of said audio signal, whether the audio signal represents a tick on the device,
>
> [C] wherein the fine tick detection is performed using a comparison with a previously trained set of reference properties that are characteristic for the device with which the audio signal is received.

(Ex. 4 at 7:14-28.)

140. As for the preamble of Claim 1, to the extent that it is determined to be limiting, Alexa Emergency Assist provides the features described in the preamble, which recites a "[m]ethod for detecting ticks on a device with which an audio signal is received in a noisy environment, from the audio signal which includes a stream of audio samples (y(n), (y′(n))." For

example, Alexa Emergency Assist can detect the sound of smoke alarms, carbon monoxide alarms, and glass breaking.

> **1. What sounds can Alexa detect?**
>
> Alexa Emergency Assist can detect and notify you about the sound of smoke alarms and carbon monoxide alarms, and glass breaking.

(Alexa Emergency Assist – Sound Detection FAQs, available at https://www.amazon.com/gp/help/customer/display.html?nodeId=TZhqo3nwWAuKfX0BBT (last accessed January 2026 and incorporated by reference).) Further, the previous version of Alexa Emergency Assist, Alexa Guard, "allow[ed] customers to adjust the sensitivity to accommodate the noise in their home environments." (The science behind an Amazon Echo feature that helped save a puppy, available at https://www.amazon.science/latest-news/the-science-behind-an-amazon-echo-feature-that-helped-save-a-puppy (last accessed January 2026 and incorporated by reference).) According to Amazon staff member "Roger B.," Alexa Emergency Assist includes "the top features . . . from Alexa Guard" in an improved form.

> Roger B. (Amazon Staff)
> February 27, 2024 at 6:05 PM
>
> Hello and welcome to the Forum, @sakota!
>
> I'm sorry you're unable to activate Alexa Guard on your Echo device.
>
> We have discontinued Alexa Guard and transitioned some Guard features to a new subscription service, Alexa Emergency Assist.
>
> **When was Alexa Guard removed?**
> Alexa Guard was removed in January 2024.
>
> **What is Alexa Emergency Assist?**
> Alexa Emergency Assist is a new safety service. We've combined the top features you know and love from Alexa Guard and improved them to make it even easier for you to get 24/7 hands-free help from first responders when you need it most. For example, you no longer need to remember to turn on Away mode — with Alexa Emergency Assist, Alexa can detect smoke and CO alarms and send an alert to you 24/7. We've also added features like Emergency Contacts, the people who will be notified when you or someone in your house says, "Alexa, call for help", and Critical Information, which allows you to pre-save important details like your gate/door code or medical information to help local police, fire, or ambulance responders. We think these features combined with access to professionally-trained Urgent Response agents that can triage and request dispatch 24/7, 365 days a year, are something that will give you even more peace of mind. Learn more here.

(Alexa Guard, available at https://amazonforum.my.site.com/s/question/0D56Q0000Cw9TunSQE/alexa-guard (last accessed January 2026 and incorporated by reference).) Thus, on information and belief, Alexa Emergency Assist includes the environmental noise features of Alexa Guard.

141. Limitation A of Claim 1 requires "applying the audio signal to a coarse tick detection processor arranged to decide whether the audio signal likely includes a tick." Alexa Emergency Assist meets this limitation. The Guard/Emergency Assist technology "relies on two models applied in a two-step system, one on the device, another in the cloud." (The science behind an Amazon Echo feature that helped save a puppy, available at https://www.amazon.science/latest-news/the-science-behind-an-amazon-echo-feature-that-helped-save-a-puppy (last accessed January 2026 and incorporated by reference).) "[T]he majority of the audio data is processed and discarded by the neural network on the device," indicating that the first-step device-side detection involves "a coarse tick detection processor arranged to decide whether the audio signal likely includes a tick." (*Id.*)

142. Limitation B of Claim 1 requires, "only if the course tick processor decides that the audio signal likely includes a tick, enabling a fine tick detection processor to decide, by more thorough processing of said audio signal, whether the audio signal represents a tick on the device." Alexa Emergency Assist meets this limitation. For example, in the Guard/Emergency Assist technology, "audio is sent to the cloud only after running it through a device-side model." (*Id.*) Further, as noted above, "the majority of the audio data is processed and discarded by the neural network on the device," and "[o]nly potential triggers make it to the cloud" for the cloud-side processing. (*Id.*) The cloud-side processing (fine tick detection processor) conducts a "second verification step to confirm the on-device detection," using "a much more powerful recognition

system to filter out false triggers that might be linked to ambient noise around the home." (*Id.*) Thus, the Guard/Emergency Assist technology includes enabling a fine tick detection processor to decide, by more thorough processing of said audio signal, whether the audio signal represents a tick on the device.

143. Limitation C of Claim 1 requires that "the fine tick detection is performed using a comparison with a previously trained set of reference properties that are characteristic for the device with which the audio signal is received." On information and belief, the Guard/Emergency Assist cloud-side verification process involves a previously trained set of reference properties that are characteristic for the device with which the audio signal is received. For example, the device-side detection and processing "utilizes a recurrent neural network — a type of deep learning model that uses sequential data or time series data to learn — on the Echo device itself. The on-device detection works by converting the audio input into features that feed into a recurrent neural network (RNN)." (*Id.*) Further, Amazon "rented a warehouse and contracted a construction crew to break hundreds of windows . . . to build an authentic data set" to train the Guard/Emergency Assist technology. (*Id.*) Accordingly, on information and belief, the cloud-side verification process is performed using a comparison with a previously trained set of reference properties that are characteristic for the device with which the audio signal is received.

144. Additionally, and/or alternatively, Amazon has indirectly infringed and continues to indirectly infringe one or more of the claims of the '259 Patent, in violation of 35 U.S.C. § 271(b) by actively inducing users of Alexa Emergency Assist to directly infringe one or more claims of the '259 Patent. For example, (a) Amazon had actual knowledge of or was willfully blind to the existence of the '259 Patent and its infringement thereof no later than the filing of this Complaint, and (b) Amazon intentionally causes, urges, or encourages users of Alexa Emergency Assist to

take action that, when taken, directly infringe one or more claims of the '259 Patent. Amazon's encouragement is accomplished by promoting, advertising, and instructing customers and potential customers to use Alexa Emergency Assist, including infringing uses thereof. Amazon knows (based on the claim chart previously provided by K.Mizra and/or after reading this Complaint should know) that its actions will induce users of Alexa Emergency Assist to directly infringe one or more claims of the '259 Patent, and users thereof directly infringe one or more claims of the '259 Patent. For instance, at a minimum, Amazon has supplied and continues to supply Alexa Emergency Assist to customers while knowing that use thereof will infringe one or more claims of the '259 Patent.

145. Due to Amazon's knowledge of the '259 Patent and of its infringement thereof no later than the filing of this Complaint, Amazon's infringement of the '259 Patent has been and continues to be willful since no later than that date.

146. Amazon's acts of infringement have occurred within this District and elsewhere throughout the United States.

147. As a result of Amazon's infringing conduct, K.Mizra has suffered damages. Amazon is liable to K.Mizra in an amount that adequately compensates K.Mizra for Amazon's infringement in an amount that is no less than a fully paid-up, lump-sum, reasonable royalty, together with interest and costs as fixed by this Court under 25 U.S.C. § 284.

COUNT V
(Patent Infringement under 35 U.S.C. § 271 of the '649 Patent)

148. K.Mizra incorporates paragraphs 1 through 147 as though fully set forth herein.

149. The '649 Patent includes 26 claims.

150. Amazon has directly infringed one or more claims of the '649 Patent by making, importing, using, offering for sale, and/or selling Amazon Echo devices incorporating Echo Spatial Perception, all in violation of 35 U.S.C. § 271(a).

151. Based on publicly available information, Amazon Echo devices incorporating Echo Spatial Perception satisfy every element of at least Claim 1 of the '649 Patent.

152. Claim 1 of the '649 Patent states:

> [preamble] A method of identifying proximate devices, the method comprising the steps of:
>
> [A] recording, in each device, an audio signal so as to produce respective audio samples,
>
> [B] detecting, in each device, a detected sensory identifier, and
>
> [C] transmitting the respective audio samples from one or more of the devices to a server or another of the devices, wherein the server or device receiving the respective audio samples comprises a processor, a memory and software configured to perform the remaining steps of
>
> [D] using said detected sensory identifier for matching the devices, using said detecting as a trigger to select a common time interval of audio samples at a predetermined relative temporal position offset with respect to a time point of the detected sensory identifier;
>
> [E] comparing respective audio samples recorded during the common time interval by different ones of the devices with each other, and
>
> [F] deciding that the devices are proximate only if the respective audio samples are sufficiently similar.

(Ex. 5 at 9:18-39.)

153. As for the preamble of Claim 1, to the extent that it is determined to be limiting, Amazon Echo devices incorporating Echo Spatial Perception provides the features described in the preamble, which recites a "method of identifying proximate devices." Echo Spatial Perception

"enable[s] devices to collaboratively determine which is closes to the user" and thus performs a "method of identifying proximate devices."

> Amazon has officially moved ESP to the cloud, which should ensure that all devices, Amazon or non-Amazon, can use it. Users don't even have to download any software updates to begin using the feature; it'll come on board automatically. Literally any Alexa device able to connect to the internet can now start using the feature. ESP will enable these devices to collaboratively determine which is closest to the user. Amazon was short on specifics, but presumably this is done by basing which noise-canceling array detects the strongest audio signal.

(How Amazon Ensures Only The Closest Alexa Device Responds To Your Voice Commands, available at https://www.techtimes.com/articles/232691/20180726/how-amazon-ensures-only-the-closest-alexa-device-responds-to-your-voice-commands.htm (last accessed January 2026 and incorporated by reference).)

154. Limitation A of Claim 1 requires "recording, in each device, an audio signal so as to produce respective audio samples." The Echo Spatial Perception process meets this limitation because any given Echo device in proximity will record audio input following detection of the chosen wake word.

> **Fact: By default, Echo devices are designed to record audio only after the device detects your chosen wake word (or if the action button is pushed).**
>
> Until your Echo device detects your chosen wake word—current options include “Alexa,” “Amazon,” “Echo,” “Ziggy,” and “Computer”—Alexa does not record or store what you or any other person in the room has said. This is made possible by built-in technology called “keyword spotting” that matches spoken audio to the wake word’s acoustic patterns. Echo devices are designed to listen for the sound waves of the wake word while ignoring everything else. This is why your device can detect the wake word without listening in on your personal conversations.

(Busting the four biggest myths about Amazon's Alexa, available at https://www.aboutamazon.eu/busting-the-four-biggest-myths-about-amazons-alexa (last accessed January 2026 and incorporated by reference).)

155. Limitation B of Claim 1 requires "detecting, in each device, a detected sensory identifier." The Echo Spatial Perception process meets this limitation because each Echo device will detect a "wake word" (sensory identifier). (*Id.*)

156. Limitation C of Claim 1 requires "transmitting the respective audio samples from one or more of the devices to a server or another of the devices, wherein the server or device receiving the respective audio samples comprises a processor, a memory and software." The Echo Spatial Perception process meets this limitation because spoke audio inputs are transmitted to "Amazon's secure cloud" by Echo devices.

> When you speak to Alexa, your request is sent to Amazon's secure cloud so that Alexa can respond to you. Customers are in control of their voice recordings. You can see exactly what Alexa heard and sent to Amazon's cloud in the Alexa app by visiting Settings > Alexa Privacy > Review Voice History. Or you can look online at your Alexa privacy settings. Listen to the actual audio recorded or manage your voice history by choosing to delete specific recordings—sorting by date, device, or profile. You can delete them all at once, have them deleted on an ongoing basis, or not have them saved at all.

(*Id.*) On information and belief, Amazon's cloud servers comprise a processor, a memory and software configured to perform the remaining steps described below. In particular, Echo Spatial Perception is implemented on the cloud, and thus Amazon's cloud servers perform the Echo Spatial Perception process.

> *There are millions of customers with more than one Alexa-enabled device in their home, and for these customers one of the most important features we've built is Echo Spatial Perception (ESP). The ESP feature provides a frustration-free experience by identifying the best device to respond when interacting with Alexa in a multi-device environment. The technology determines which device is closest to the customer, and helps Alexa respond through that device.... Today we're announcing that ESP has moved to the cloud, introducing several benefits to developers. With Cloud ESP, all existing and new Alexa-enabled devices get the ESP feature with no requirement for device-side software changes. Cloud ESP also offers superior accuracy – even in noisy environments – and because Alexa is always getting smarter, the feature will continue to improve over time.*

(Echo Spatial Perception moves to the cloud, available at https://vowe.net/archives/017276.html (last accessed January 2026 and incorporated by reference).)

157. Limitation D of Claim 1 requires "using said detected sensory identifier for matching the devices, using said detecting as a trigger to select a common time interval of audio samples at a predetermined relative temporal position offset with respect to a time point of the detected sensory identifier." Amazon's cloud servers performing the Echo Spatial Perception process perform this step by, for example, using the volume and proximity of spoken requests to determine which Echo device is closest to the user.

> I'd like to inform you that, your firmware version is updated to recent one, which requires for ESP (Echo Spatial Perception) (i.e. 4008). With ESP, If you have more than one Alexa device using the same wake word (for example, two Echo devices using "Alexa" as a wake word), Alexa chooses the device closest to you to respond to your request using ESP (Echo Spatial Perception). ESP uses the volume and proximity of the spoken request. Nearby Alexa devices that happen to recognize the same wake word won't respond to the request.

(Amazon Echo rolls out Echo Spatial Perception, available at https://community.smartthings.com/t/amazon-echo-rolls-out-echo-spatial-perception/59873/34?page=2 (last accessed January 2026 and incorporated by reference).)

> Amazon has officially moved ESP to the cloud, which should ensure that all devices, Amazon or non-Amazon, can use it. Users don't even have to download any software updates to begin using the feature; it'll come on board automatically. Literally any Alexa device able to connect to the internet can now start using the feature. ESP will enable these devices to collaboratively determine which is closest to the user. Amazon was short on specifics, but presumably this is done by basing which noise-canceling array detects the strongest audio signal.

(How Amazon Ensures Only The Closest Alexa Device Responds To Your Voice Commands, available at https://www.techtimes.com/articles/232691/20180726/how-amazon-ensures-only-the-closest-alexa-device-responds-to-your-voice-commands.htm (last accessed January 2026 and incorporated by reference).) This indicates that all Echo devices that detect the wake word align their audio processing to the same timeframe (*e.g.*, the period of the user's query following the wake word) so that their recordings can be compared on an equal basis. In other words, the wake word detection triggers the process of evaluating that request audio across devices. Accordingly, the Echo Spatial Perception process meets limitation D of claim 1.

158. Limitation E of Claim 1 requires "comparing respective audio samples recorded during the common time interval by different ones of the devices with each other." Amazon's cloud servers performing the Echo Spatial Perception process perform this step. As discussed above with respect to limitation D, the Echo Spatial Perception process involves a comparison of audio samples based on, *e.g.*, volume to determine the closest device. Further, "Cloud ESP offers superior accuracy – even in noisy environments," indicating that the system performs a more complex comparison and analyzes audio characteristics beyond volume.

> *There are millions of customers with more than one Alexa-enabled device in their home, and for these customers one of the most important features we've built is Echo Spatial Perception (ESP). The ESP feature provides a frustration-free experience by identifying the best device to respond when interacting with Alexa in a multi-device environment. The technology determines which device is closest to the customer, and helps Alexa respond through that device.... Today we're announcing that ESP has moved to the cloud, introducing several benefits to developers. With Cloud ESP, all existing and new Alexa-enabled devices get the ESP feature with no requirement for device-side software changes. Cloud ESP also offers superior accuracy – even in noisy environments – and because Alexa is always getting smarter, the feature will continue to improve over time.*

(Echo Spatial Perception moves to the cloud, available at https://vowe.net/archives/017276.html (last accessed January 2026 and incorporated by reference).)

159. Limitation F of Claim 1 requires "deciding that the devices are proximate only if the respective audio samples are sufficiently similar." An Amazon cloud server performing the Echo Spatial Perception process meets this limitation because the process identifies devices to be distinguished based on proximity only if the devices respond to the same audio input. In other words, Echo Spatial Perception decides that two devices are proximate only if they detect/record the same audio samples.

160. Additionally, and/or alternatively, Amazon has indirectly infringed and continues to indirectly infringe one or more of the claims of the '649 Patent, in violation of 35 U.S.C. § 271(b) by actively inducing users of Amazon Echo devices incorporating Echo Spatial Perception to directly infringe one or more claims of the '649 Patent. For example, (a) Amazon had actual knowledge of or was willfully blind to the existence of the '649 Patent and its infringement thereof no later than the filing of this Complaint, and (b) Amazon intentionally causes, urges, or encourages users of Amazon Echo devices incorporating Echo Spatial Perception to take action that, when taken, directly infringe one or more claims of the '649 Patent. Amazon's encouragement is accomplished by promoting, advertising, and instructing customers and potential customers to use Amazon Echo devices incorporating Echo Spatial Perception, including infringing uses

thereof. Amazon knows (based on the claim chart previously provided by K.Mizra and/or after reading this Complaint should know) that its actions will induce users of Amazon Echo devices incorporating Echo Spatial Perception to directly infringe one or more claims of the '649 Patent, and users thereof directly infringe one or more claims of the '649 Patent. For instance, at a minimum, Amazon has supplied and continues to supply Amazon Echo devices incorporating Echo Spatial Perception to customers while knowing that use thereof will infringe one or more claims of the '649 Patent.

161. Due to Amazon's knowledge of the '649 Patent and of its infringement thereof no later than the filing of this Complaint, Amazon's infringement of the '649 Patent has been and continues to be willful since no later than that date.

162. Amazon's acts of infringement have occurred within this District and elsewhere throughout the United States.

163. As a result of Amazon's infringing conduct, K.Mizra has suffered damages. Amazon is liable to K.Mizra in an amount that adequately compensates K.Mizra for Amazon's infringement in an amount that is no less than a fully paid-up, lump-sum, reasonable royalty, together with interest and costs as fixed by this Court under 25 U.S.C. § 284.

COUNT VI
(Patent Infringement under 35 U.S.C. § 271 of the '282 Patent)

164. K.Mizra incorporates paragraphs 1 through 163 as though fully set forth herein.

165. The '282 Patent includes 22 claims.

166. Amazon has directly infringed one or more claims of the '282 Patent by making, importing, using, offering for sale, and/or selling Amazon CloudWatch, all in violation of 35 U.S.C. § 271(a).

167. Based on publicly available information, Amazon CloudWatch satisfies every element of at least Claim 1 of the '282 Patent.

168. Claim 1 of the '282 Patent states:

> [preamble] A method, comprising:
>
> [A] receiving, at a first application server instance selected from a plurality of application server instances based on a load balancing process, first adapter processed information from a first adapter, wherein the first adapter processed information comprises event information received by the first adapter from a network element and processed by the first adapter based on a first communication protocol;
>
> [B] processing, by the first application server instance, the first adapter processed information based on an event management service to produce application processed information;
>
> [C] sending, by the first application server instance, the application processed information to a gateway device, wherein the gateway device is one of a plurality of gateway devices respectively associated with the plurality of application server instances and is configured to transfer the application processed information to a second adapter of a plurality of second adapters configured to process the application processed information based on a second communication protocol to produce second adapter processed information and transfer the second adapter processed information to an operation support system device; and
>
> [D] in response to determining that the first application server instance has become disabled, facilitating establishing an association between the first adapter and a second application server instance of the plurality of application server instances and between the gateway device and the second application server instance.

(Ex. 6 at 10:59-11:22.)

169. As for the preamble of Claim 1, to the extent that it is determined to be limiting, Amazon CloudWatch provides the features described in the preamble because Amazon CloudWatch performs a "method."

170. Limitation A of Claim 1 requires "receiving, at a first application server instance selected from a plurality of application server instances based on a load balancing process, first adapter processed information from a first adapter, wherein the first adapter processed information comprises event information received by the first adapter from a network element and processed by the first adapter based on a first communication protocol." Amazon CloudWatch performs this step. The Amazon CloudWatch agent can be installed on "Amazon EC2 instances" that will then be capable of "Elastic Load Balancing," which enables the launch of multiple EC2 instances and distribution of traffic between them.

> You can collect metrics from servers by installing the CloudWatch agent on the server. You can install the agent on both Amazon EC2 instances and on-premises servers. You can also install the agent on computers running Linux, Windows Server, or macOS. If you install the agent on an Amazon EC2 instance, the metrics the agent collects are in addition to the metrics enabled by default on Amazon EC2 instances. For information about installing the CloudWatch agent on an instance, see Collect metrics, logs, and traces using the CloudWatch agent. You can use this section to learn about metrics the CloudWatch agent collects.

(Metrics collected by the CloudWatch agent, available at https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/metrics-collected-by-CloudWatch-agent.html (last accessed January 2026 and incorporated by reference).)

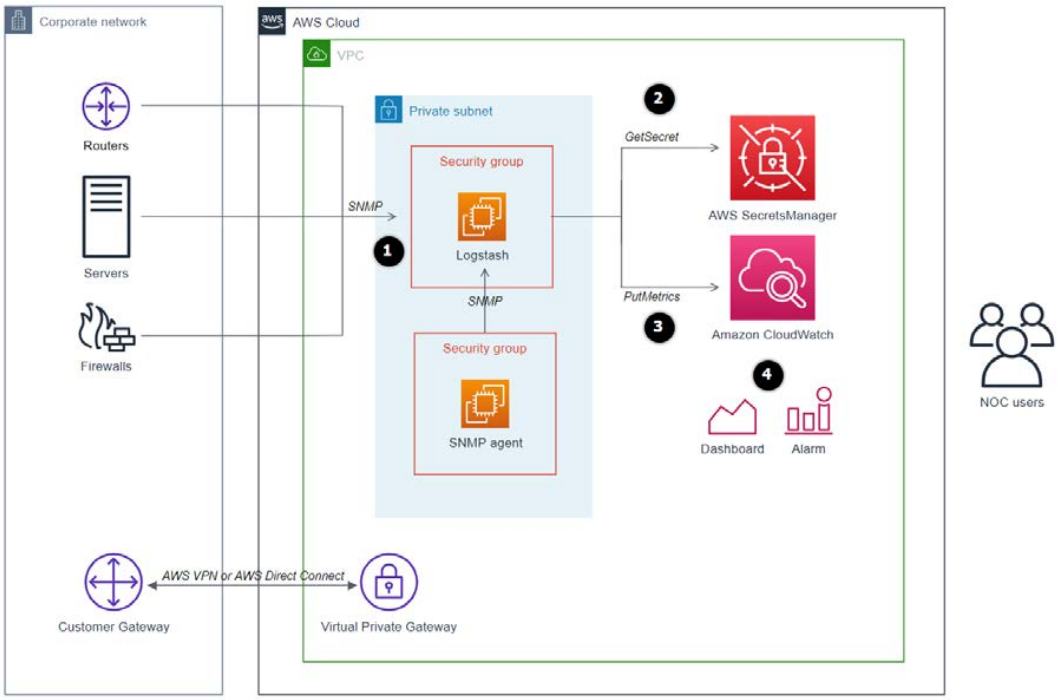
> **AWS HIGH AVAILABILITY FOR EC2 INSTANCES**
>
> If you are running instances on Amazon EC2, Amazon provides several built-in capabilities to achieve high availability:
>
> - **Elastic Load Balancing**—you can launch several EC2 instances and distribute traffic between them.
> - **Availability Zones**—you can place instances in different AZs.
> - **Auto Scaling**—use auto-scaling to detect when loads increase, and then dynamically add more instances.

(Understanding AWS High Availability: Compute, SQL and Storage, available at https://www.netapp.com/learn/understanding-aws-high-availability-compute-sql-and-storage/ (last accessed January 2026 and incorporated by reference).)

171. CloudWatch is integrated with an SNMP agent such Logstash ("first adapter"), which collects information from the monitored devices such as routers ("network elements"), and, based on a communication protocol such as SNMP, converts it into metric information concerning monitored devices ("first adapter processed information").

> In this solution, Logstash acts as a relay server that sits between the managed devices that you want to monitor and Amazon CloudWatch. I use two Logstash plugins: SNMP (input plugin) and CloudWatch (output plugin). The input plugin collects information (using SNMP polling) from the monitored devices. These devices must be running SNMP daemons, listening on standard port 161 (Poll). After Logstash processes the information, the output plugin forwards the metrics to CloudWatch using standard AWS APIs. I use AWS Secrets Manager and Logstash keystore for SNMP credential management. You can expand this solution by using Logstash Snmptrap input plugin.

(SNMP monitoring using Amazon CloudWatch and Elastic Logstash, available at https://aws.amazon.com/blogs/mt/snmp-monitoring-using-amazon-cloudwatch-and-elastic-logstash/ (last accessed January 2026 and incorporated by reference).)



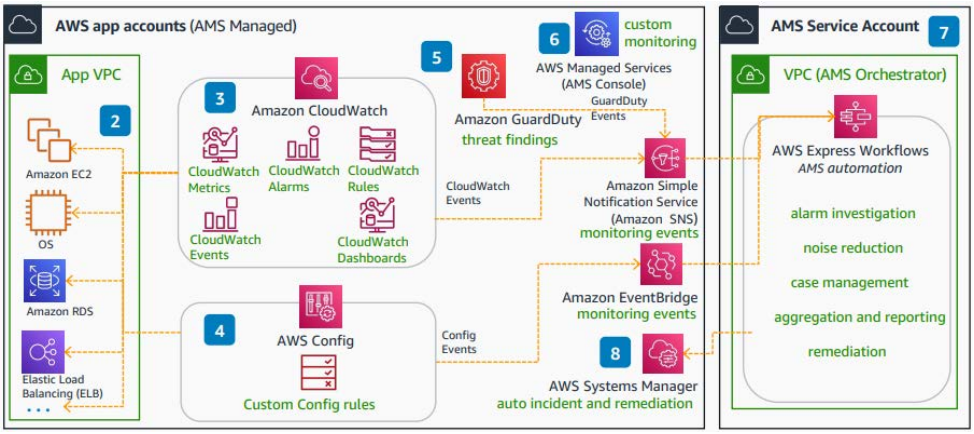
(*Id.*) CloudWatch then receives the metric information from Logstash.

172. Limitation B of Claim 1 requires "processing, by the first application server instance, the first adapter processed information based on an event management service to produce application processed information." CloudWatch performs this step. For example, CloudWatch creates ("produces") alarms/events ("application processed information") based on the metrics ("first adapter processed information").

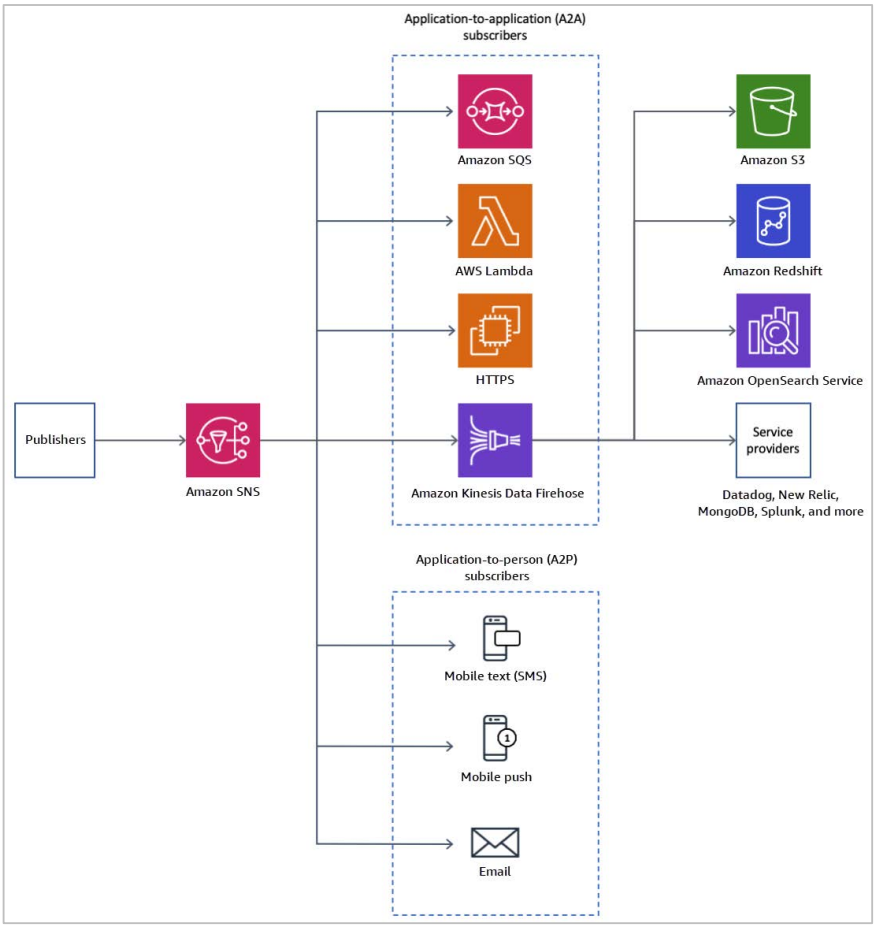
> You can use an *alarm* to automatically initiate actions on your behalf. An alarm watches a single metric over a specified time period, and performs one or more specified actions, based on the value of the metric relative to a threshold over time. The action is a notification sent to an Amazon SNS topic or an Auto Scaling policy. You can also add alarms to dashboards.

(Amazon CloudWatch User Guide at p. 125, available at https://docs.aws.amazon.com/pdfs/AmazonCloudWatch/latest/monitoring/acw-ug.pdf (last accessed January 2026 and incorporated by reference).)

173. Limitation C of Claim 1 requires "sending, by the first application server instance, the application processed information to a gateway device, wherein the gateway device is one of a plurality of gateway devices respectively associated with the plurality of application server instances and is configured to transfer the application processed information to a second adapter of a plurality of second adapters configured to process the application processed information based on a second communication protocol to produce second adapter processed information and transfer the second adapter processed information to an operation support system device." CloudWatch performs this step. For example, the alarms/events referenced above with respect to limitation B are sent to Amazon Data Firehose ("gateway device") either directly or via the Amazon SNS.



(AWS Managed Services (AMS) for Operational Excellence at p. 3, available at https://d1.awsstatic.com/architecture-diagrams/ArchitectureDiagrams/AWS-managed-services-for-operational-excellence-ra.pdf (last accessed January 2026 and incorporated by reference).)



(What is Amazon SNS?, available at https://docs.aws.amazon.com/sns/latest/dg/welcome.html (last accessed January 2026 and incorporated by reference).) Amazon Data Firehose forwards the alarms/events to Amazon OpenSearch Service ("operation support system") using, e.g., interfaces ("second adapter") that support the HTTPS protocol ("second communication protocol").

> Amazon Data Firehose is a fully managed service for delivering real-time streaming data to destinations such as Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon OpenSearch Service, Amazon OpenSearch Serverless, Splunk, Apache Iceberg Tables, and any custom HTTP endpoint or HTTP endpoints owned by supported third-party service providers, including Datadog, Dynatrace, LogicMonitor, MongoDB, New Relic, Coralogix, and Elastic. With Amazon Data Firehose, you don't need to write

(What is Amazon Data Firehose?, available at https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html (last accessed January 2026 and incorporated by reference).)

174. Limitation D of Claim 1 requires "in response to determining that the first application server instance has become disabled, facilitating establishing an association between the first adapter and a second application server instance of the plurality of application server instances and between the gateway device and the second application server instance." CloudWatch performs this step. For example, the AWS architecture implements Availability Zones, which support failover between zones, indicating that when one CloudWatch agent (deployed on an EC2 instance) goes down, another CloudWatch agent takes its place by, *e.g.*, establishing a connection between the new instance and the SNMP agents/Logstash and the Amazon Data Firehose ("gateway").

Resilience in Amazon CloudWatch

The AWS global infrastructure is built around AWS Regions and Availability Zones. Regions provide multiple physically separated and isolated Availability Zones, which are connected through low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones

Compliance validation 3636

Amazon CloudWatch User Guide

without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

(Amazon CloudWatch User Guide at pp. 3636-37, available at https://docs.aws.amazon.com/pdfs/AmazonCloudWatch/latest/monitoring/acw-ug.pdf (last accessed January 2026 and incorporated by reference).)

175. Additionally, and/or alternatively, Amazon has indirectly infringed and continues to indirectly infringe one or more of the claims of the '282 Patent, in violation of 35 U.S.C. § 271(b) by actively inducing users of Amazon CloudWatch to directly infringe one or more claims of the '282 Patent. For example, (a) Amazon had actual knowledge of or was willfully blind to the existence of the '282 Patent and its infringement thereof no later than the filing of this Complaint, and (b) Amazon intentionally causes, urges, or encourages users of Amazon CloudWatch to take action that, when taken, directly infringe one or more claims of the '282 Patent. Amazon's encouragement is accomplished by promoting, advertising, and instructing customers and potential customers to use Amazon CloudWatch, including infringing uses thereof. Amazon knows (based on the claim chart previously provided by K.Mizra and/or after reading this Complaint should know) that its actions will induce users of Amazon CloudWatch to directly infringe one or more

claims of the '282 Patent, and users thereof directly infringe one or more claims of the '282 Patent. For instance, at a minimum, Amazon has supplied and continues to supply Amazon CloudWatch to customers while knowing that use thereof will infringe one or more claims of the '282 Patent.

176. Due to Amazon's knowledge of the '282 Patent and of its infringement thereof no later than the filing of this Complaint, Amazon's infringement of the '282 Patent has been and continues to be willful since no later than that date.

177. Amazon's acts of infringement have occurred within this District and elsewhere throughout the United States.

178. As a result of Amazon's infringing conduct, K.Mizra has suffered damages. Amazon is liable to K.Mizra in an amount that adequately compensates K.Mizra for Amazon's infringement in an amount that is no less than a fully paid-up, lump-sum, reasonable royalty, together with interest and costs as fixed by this Court under 25 U.S.C. § 284.

REQUEST FOR RELIEF

WHEREFORE, K.Mizra respectfully requests the Court find in its favor and against Amazon, and that the Court grant K.Mizra at least the following relief:

A. Judgment that Amazon has directly infringed, literally and/or under the DoE, one or more claims of the Asserted Patents;

B. Awarding damages to K.Mizra in an amount to be proven at trial and in the form of a fully paid-up, lump sum, reasonable royalty that takes into account and runs through expiration of the Asserted Patents;

C. Awarding enhanced damages, as appropriate, under 35 U.S.C. § 284;

D. Awarding K.Mizra's costs (including disbursements) and declaring this an exceptional case and awarding K.Mizra its attorneys' fees in accordance with 35 U.S.C. § 285;

E. Pre-judgment and post-judgment interest at the maximum rate permitted by law on the damages caused to K.Mizra by reason of Amazon's infringing activities and other conduct complaint of herein; and

F. Awarding such other and further relief as this Court deems just and proper under the circumstances.

DEMAND FOR JURY TRIAL

Pursuant to Fed. R. Civ. P. 38(b), K.Mizra hereby demands a trial by jury on all issues so triable.

Dated: February 10, 2026

Respectfully submitted,

By: */s/ Michael C. Smith*

Michael C. Smith
Texas Bar No. 18650410
michael.smith@solidcounsel.com
Scheef & Stone, LLP
113 E. Austin Street
Marshall, TX 75670
(903) 938-8900

Robert R. Brunelli*
CO State Bar No. 20070
rbrunelli@sheridanross.com
Brian S. Boerman*
CO State Bar No. 50834
bboerman@sheridanross.com
Matthew C. Holohan*
CO State Bar No. 40996
mholohan@sheridanross.com
SHERIDAN ROSS P.C.
1560 Broadway, Suite 1200
Denver, CO 80202
Telephone: 303-863-9700

Facsimile: 303-863-0223
litigation@sheridanross.com

Of Counsel:
Claire A. Henry
Texas Bar No. 24053063
Miller Fair Henry PLLC
1507 Bill Owens Parkway
Longview, TX 75604
Telephone: (903) 757-6400
Facsimile: (903) 757-2323
claire@millerfairhenry.com

**To be admitted pro hac vice*
Attorneys for Plaintiff K.Mizra LLC

EXHIBIT 1

US008234705B1

(12) **United States Patent**
Roskind et al.

(10) **Patent No.: US 8,234,705 B1**
(45) **Date of Patent: Jul. 31, 2012**

(54) **CONTAGION ISOLATION AND INOCULATION**

(75) Inventors: **James A. Roskind**, Redwood City, CA (US); **Aaron T. Emigh**, Incline Village, NV (US)

(73) Assignee: **Radix Holdings, LLC**, Incline Village, NV (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1520 days.

(21) Appl. No.: **11/237,003**

(22) Filed: **Sep. 27, 2005**

Related U.S. Application Data

(60) Provisional application No. 60/613,909, filed on Sep. 27, 2004.

(51) **Int. Cl.**
G06F 11/00 (2006.01)
G06F 12/14 (2006.01)
G06F 12/16 (2006.01)
G08B 23/00 (2006.01)

(52) **U.S. Cl.** **726/22**; 726/23; 726/24; 726/25

(58) **Field of Classification Search** 726/11, 726/14, 22–25
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,823,479	B1 *	11/2004	McElhaney et al.	714/43
6,996,845	B1 *	2/2006	Hurst et al.	726/25
7,263,720	B2 *	8/2007	Silvester	726/22
7,287,278	B2 *	10/2007	Liang	726/22
7,424,610	B2 *	9/2008	Ranganathan	713/164
2002/0083328	A1 *	6/2002	Riordan	713/183
2003/0188173	A1 *	10/2003	Zimmer et al.	713/189
2003/0188194	A1 *	10/2003	Currie et al.	713/201
2005/0018618	A1 *	1/2005	Mualem et al.	370/252
2005/0033987	A1 *	2/2005	Yan et al.	713/201
2005/0131997	A1 *	6/2005	Lewis et al.	709/203
2006/0053426	A1 *	3/2006	Dive-Reclus et al.	719/328

OTHER PUBLICATIONS

Bohra, A.; Neamtiu, I.; Gallard, P.; Sultan, F.; Iftode, L.; Remote repair of operating system state using Backdoors; Autonomic Computing, 2004. Proceedings. International Conference on May 17-18, 2004 pp. 256-263.*

Mark V. Glover, Integrating a Trusted Computing Base Extension Server and Secure Session Server into the LINUX Operating System, Sep. 2001, Naval Postgraduate School.*

Pearson et al., Trusted Computing Platform: TCPA Technology in Context, 2003, Hewlett-Packard Books: Patricia Pekary.*

OLS: Linux and trusted computing, Jul. 22, 2005, http://lwn.net/Articles/144681/ (as lined on Jul. 15, 2010).*

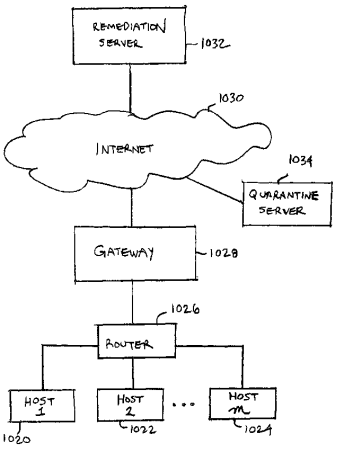
* cited by examiner

Primary Examiner — Eleni Shiferaw
Assistant Examiner — Jing Sims

(57) **ABSTRACT**

Contagion isolation and inoculation is disclosed. In some embodiments, a request is received from a host, e.g., via a network interface, to connect to a protected network. It is determined whether the host is required to be quarantined. If the host is required to be quarantined, the host is provided only limited access to the protected network. In some embodiments, a quarantined host is permitted to access the protected network only as required to remedy a condition that caused the quarantine to be imposed, such as to download a software patch, update, or definition; install, remove, and/or configure software and/or settings as required by a policy; and/or to have a scan or other diagnostic and/or remedial operation performed. In various embodiments attempts to communicate with hosts not involved in remediation are redirected to a quarantine system, such as a server, that provides information, notices, updates, and/or instructions to the user.

19 Claims, 25 Drawing Sheets



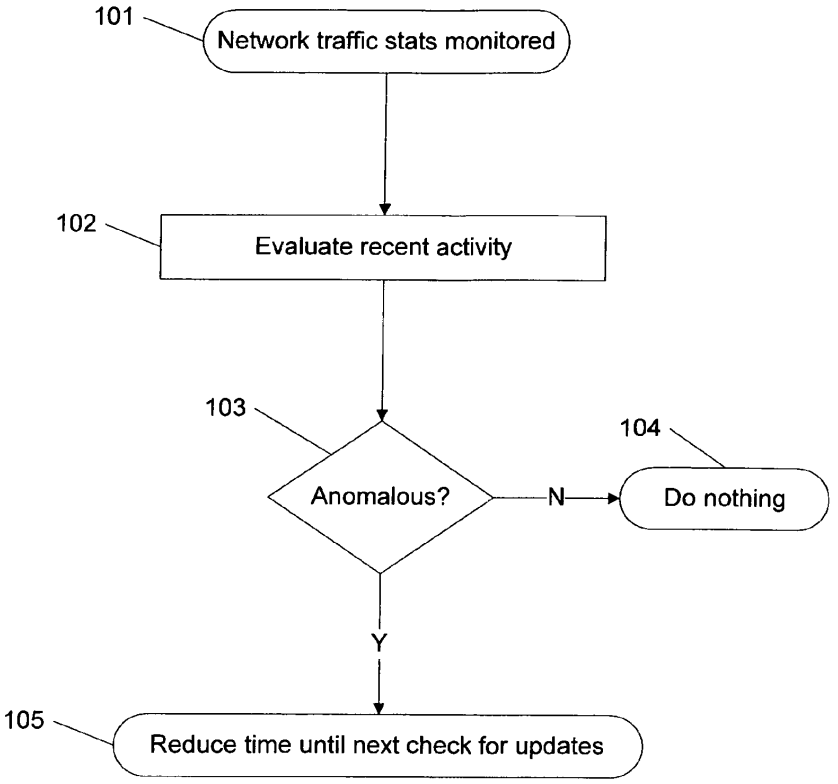

101
Network traffic stats monitored
102
Evaluate recent activity
103
Anomalous?
N
104
Do nothing
Y
105
Reduce time until next check for updates

Figure 1

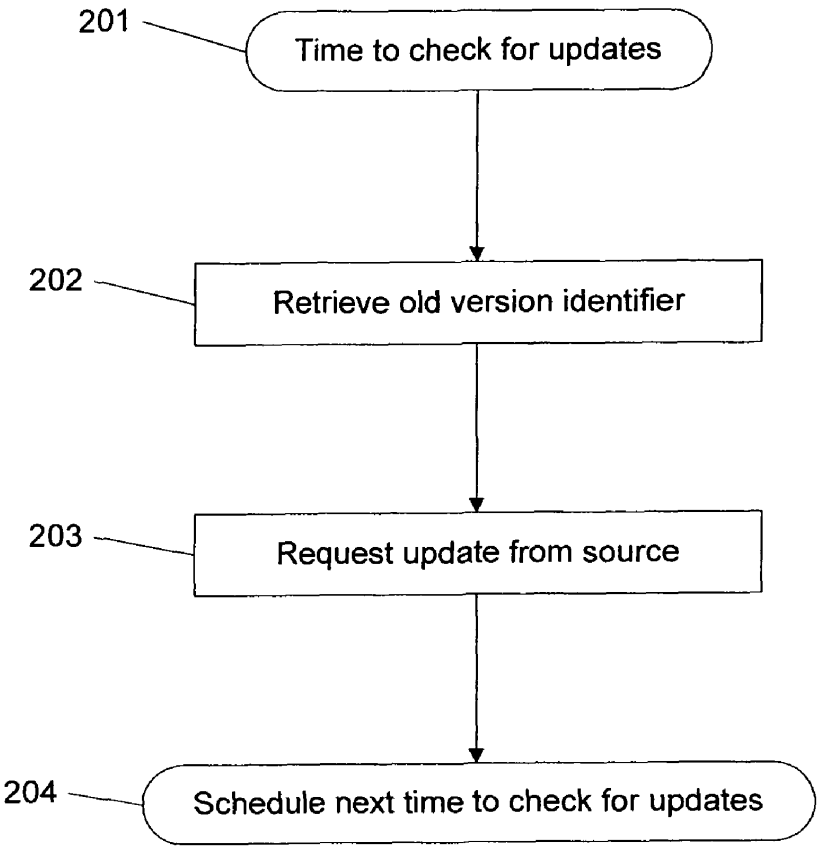

201
Time to check for updates
202
Retrieve old version identifier
203
Request update from source
204
Schedule next time to check for updates

Figure 2A

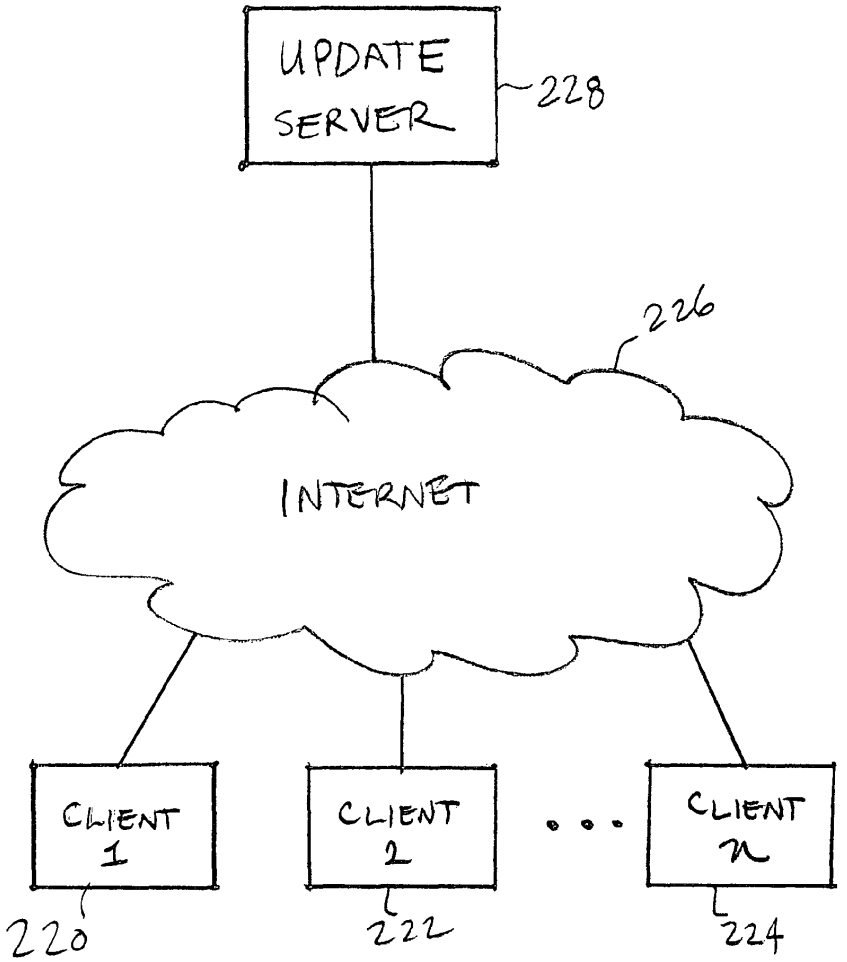

UPDATE SERVER
228
226
INTERNET
CLIENT 1
CLIENT 2
CLIENT n
220
222
224

FIG. 2B

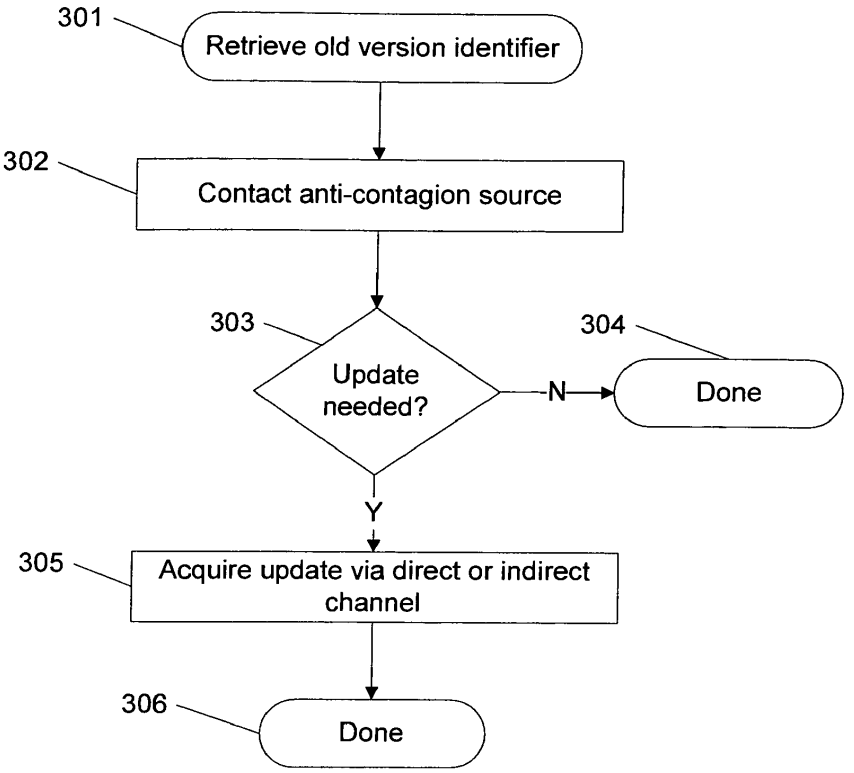

301
Retrieve old version identifier
302
Contact anti-contagion source
303
Update needed?
N
304
Done
Y
305
Acquire update via direct or indirect channel
306
Done

Figure 3A

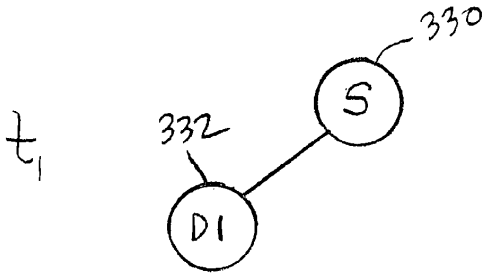


FIG. 3B

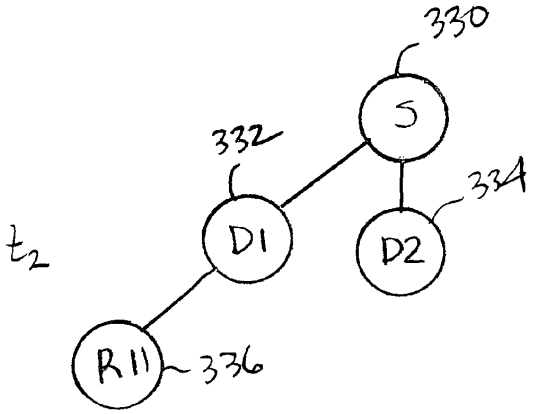


FIG. 3C

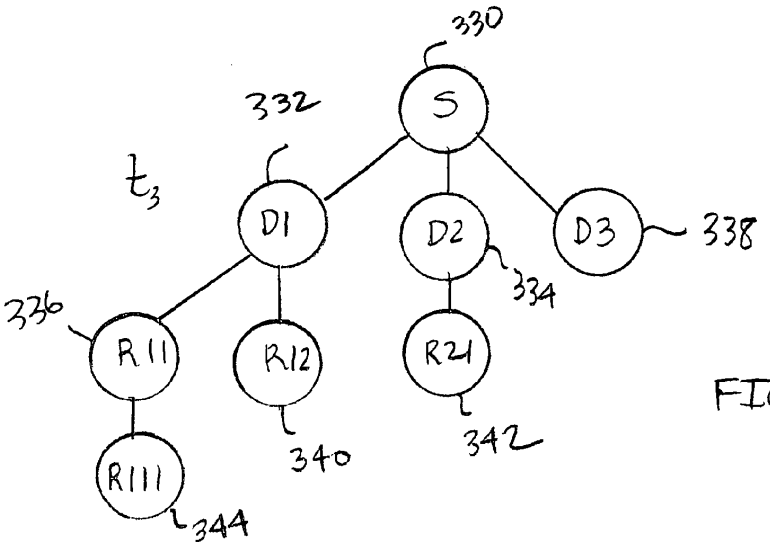


FIG. 3D

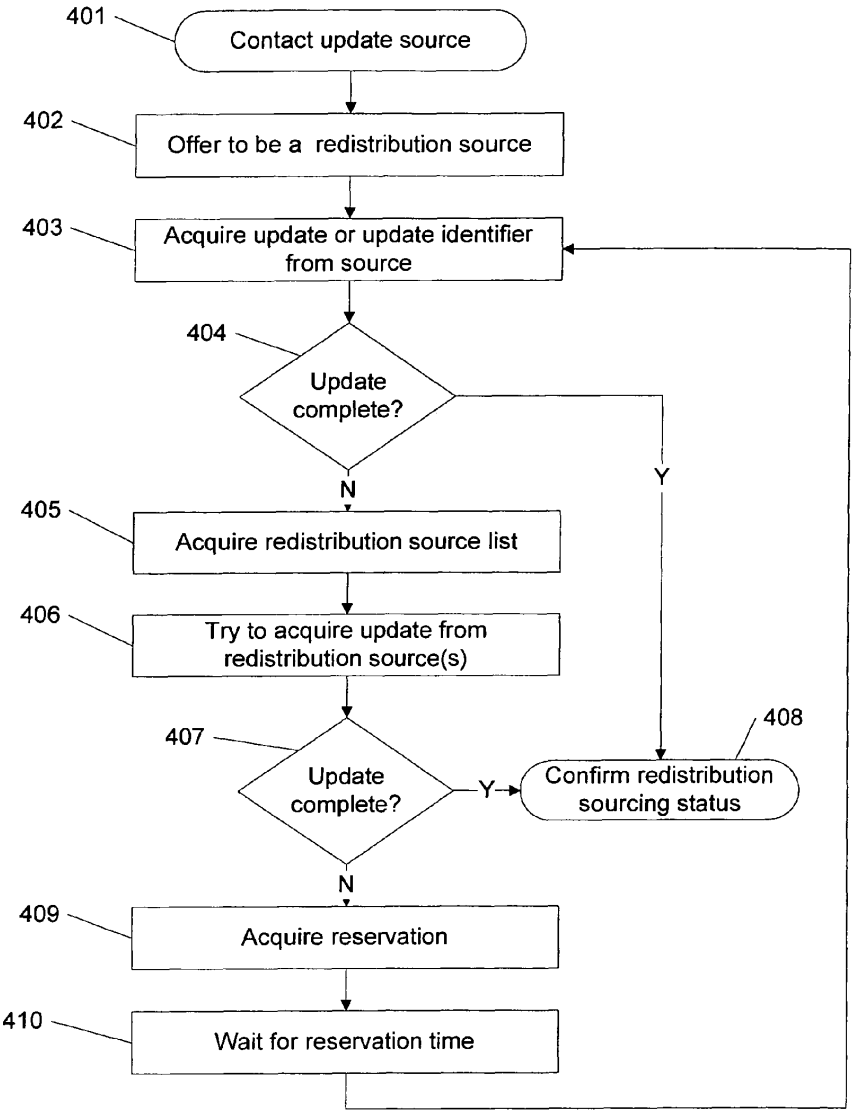

401
Contact update source
402
Offer to be a redistribution source
403
Acquire update or update identifier from source
404
Update complete?
N
Y
405
Acquire redistribution source list
406
Try to acquire update from redistribution source(s)
407
Update complete?
Y
408
Confirm redistribution sourcing status
N
409
Acquire reservation
410
Wait for reservation time

Figure 4A

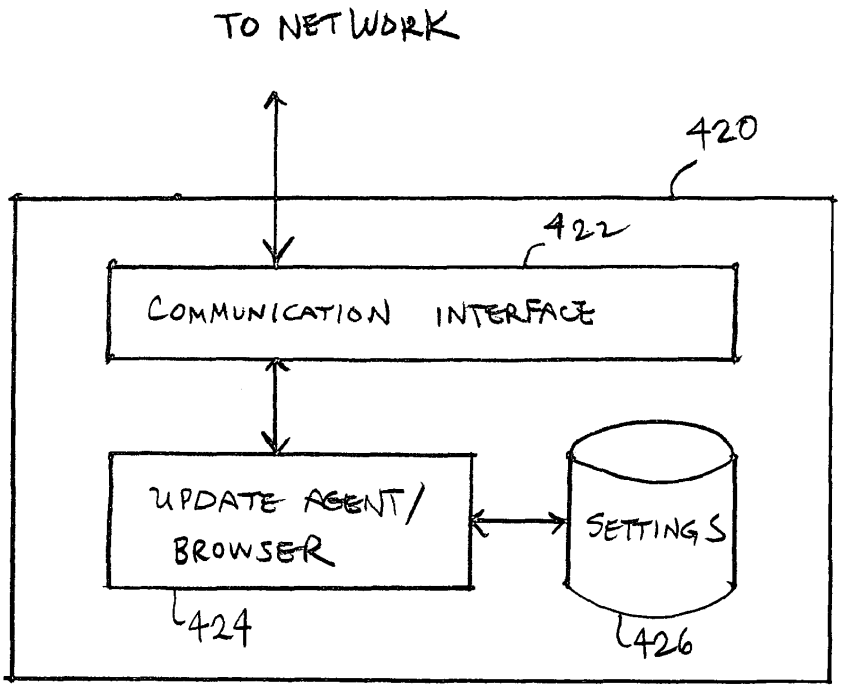

TO NETWORK
420
422
COMMUNICATION INTERFACE
UPDATE AGENT/
BROWSER
SETTINGS
424
426

FIG. 4B

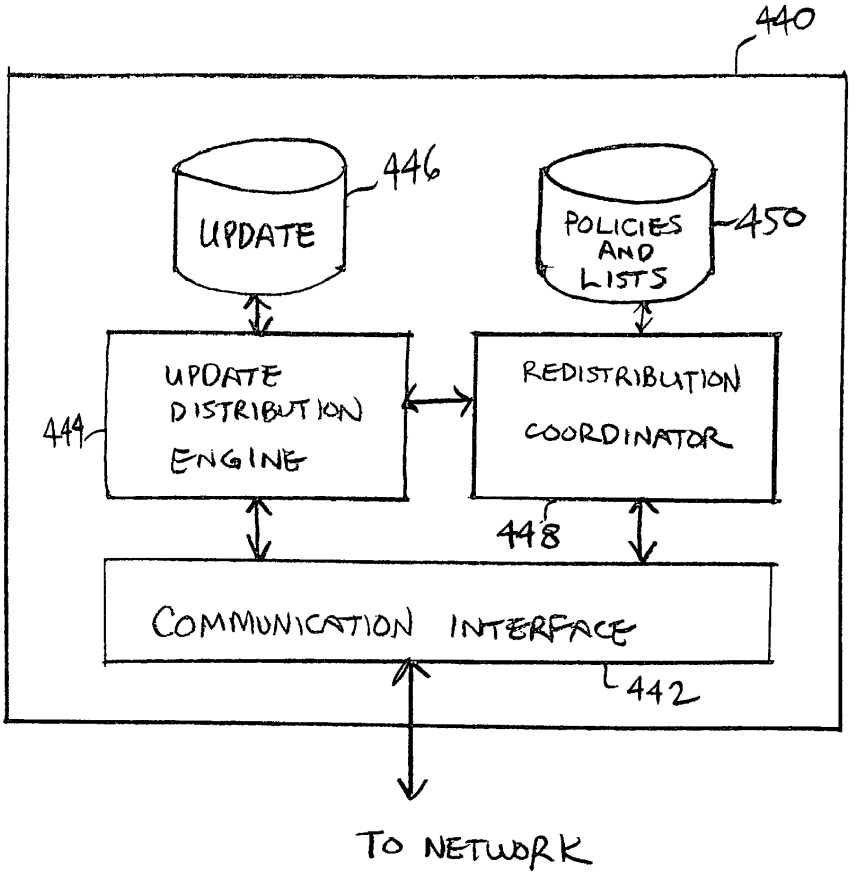

440
446
UPDATE
POLICIES AND LISTS
450
UPDATE DISTRIBUTION ENGINE
444
REDISTRIBUTION COORDINATOR
448
COMMUNICATION INTERFACE
442
TO NETWORK

FIG. 4C

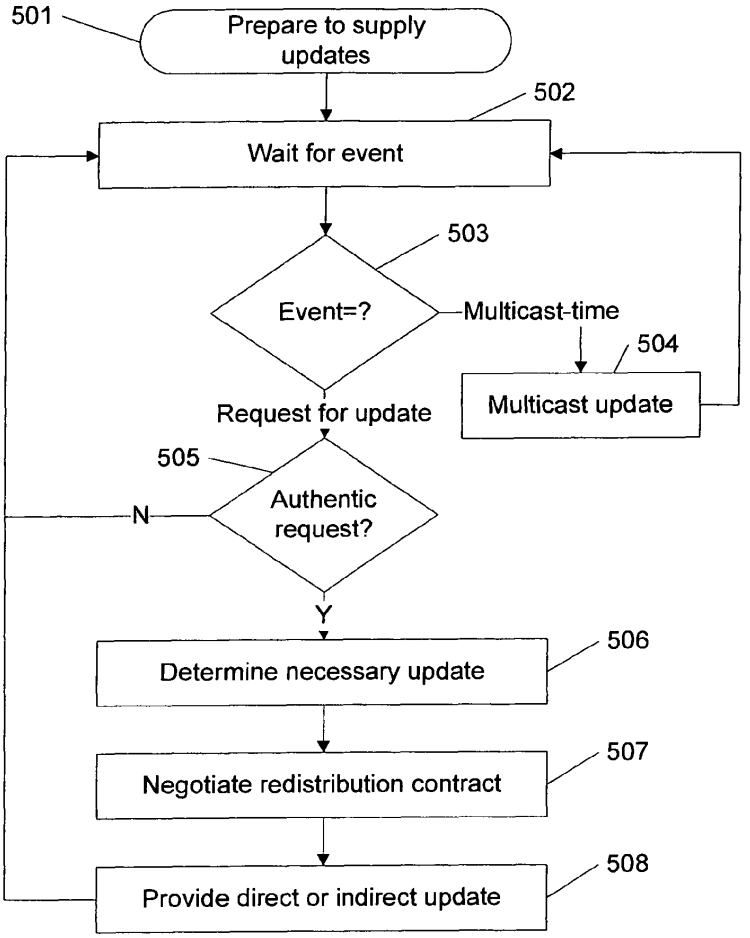

501
Prepare to supply updates
502
Wait for event
503
Event=?
Multicast-time
504
Multicast update
Request for update
505
Authentic request?
N
Y
506
Determine necessary update
507
Negotiate redistribution contract
508
Provide direct or indirect update

Figure 5

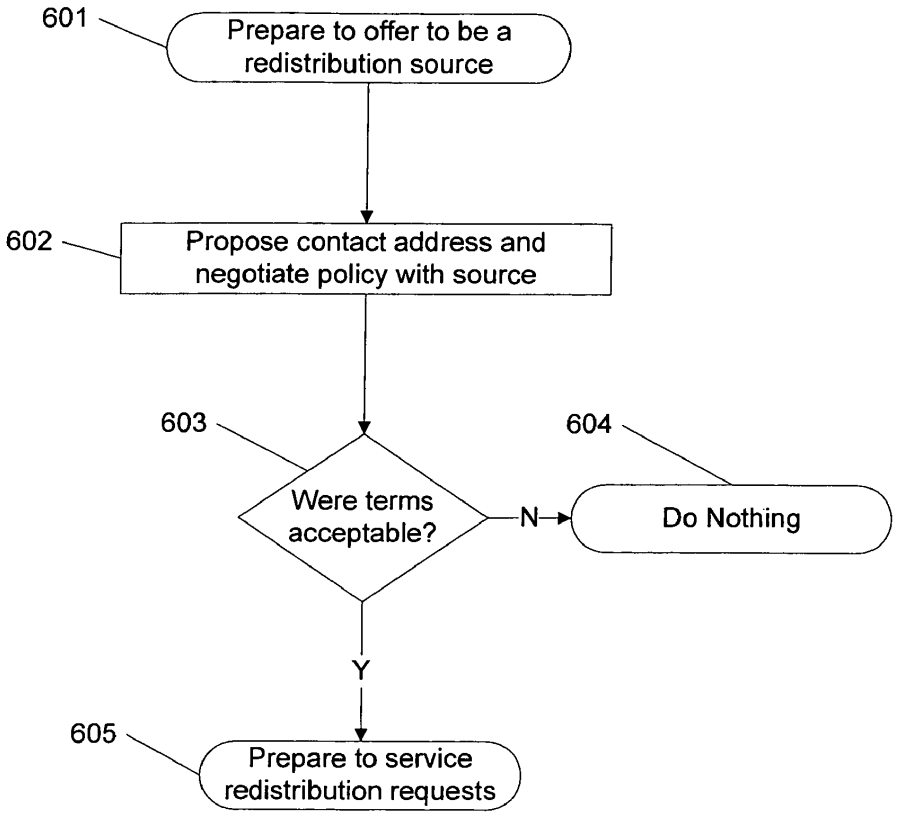

601
Prepare to offer to be a redistribution source
602
Propose contact address and negotiate policy with source
603
Were terms acceptable?
N
604
Do Nothing
Y
605
Prepare to service redistribution requests

Figure 6

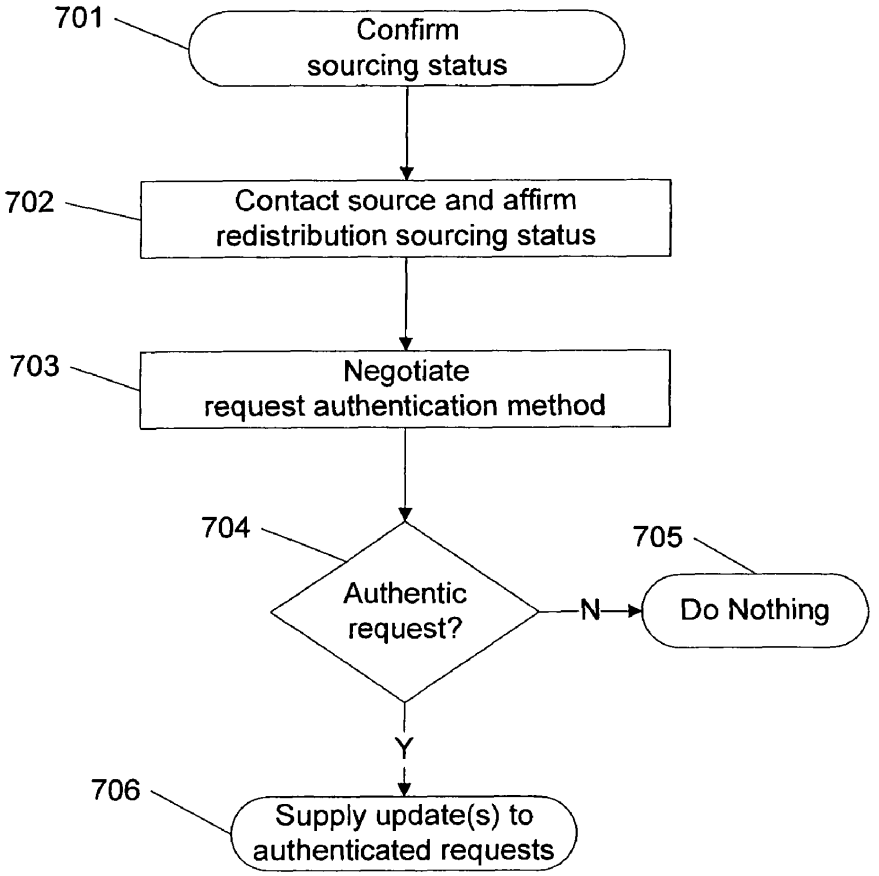

701
Confirm sourcing status
702
Contact source and affirm redistribution sourcing status
703
Negotiate request authentication method
704
Authentic request?
N
705
Do Nothing
Y
706
Supply update(s) to authenticated requests

Figure 7

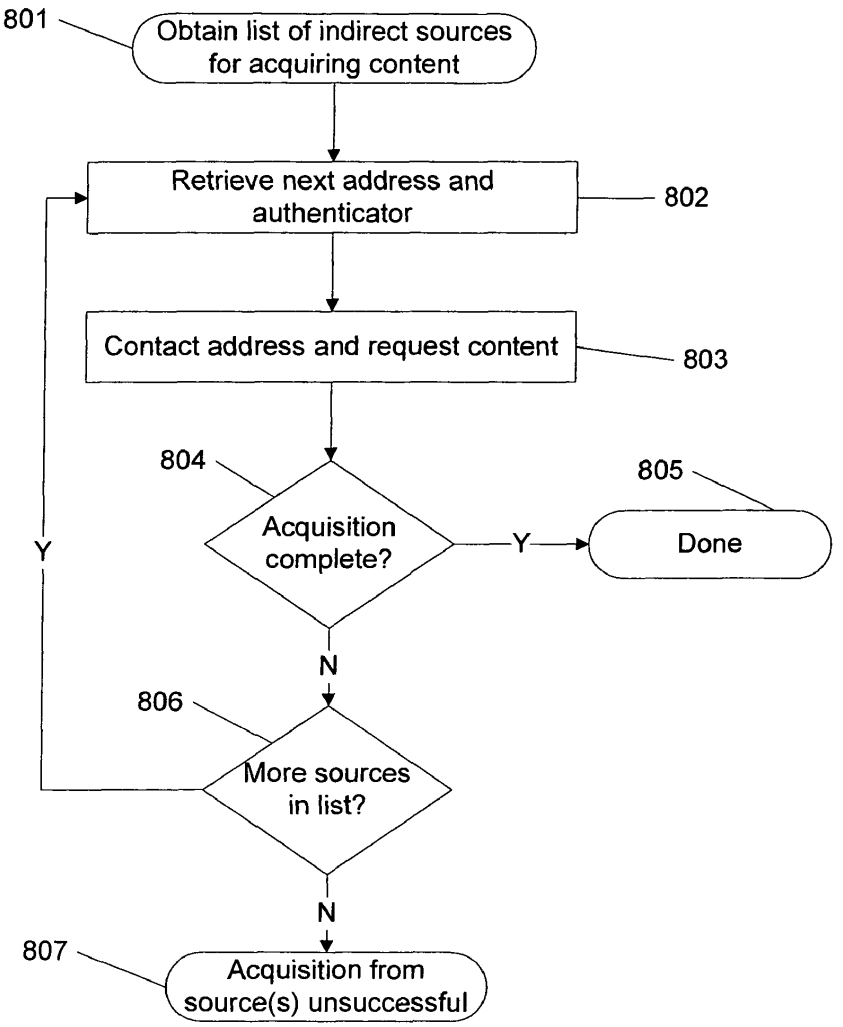

801
Obtain list of indirect sources for acquiring content
Retrieve next address and authenticator
802
Contact address and request content
803
804
Acquisition complete?
Y
805
Done
N
806
More sources in list?
Y
N
807
Acquisition from source(s) unsuccessful

Figure 8

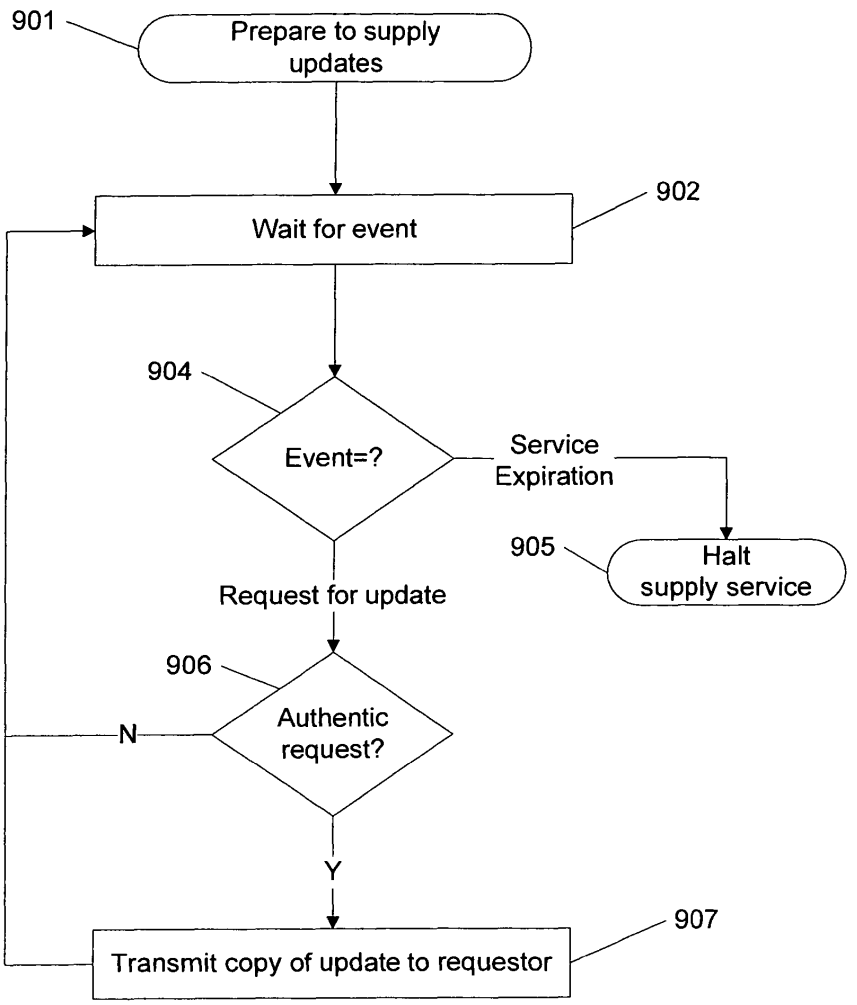

901
Prepare to supply updates
902
Wait for event
904
Event=?
Service Expiration
905
Halt supply service
Request for update
906
Authentic request?
N
Y
907
Transmit copy of update to requestor

Figure 9

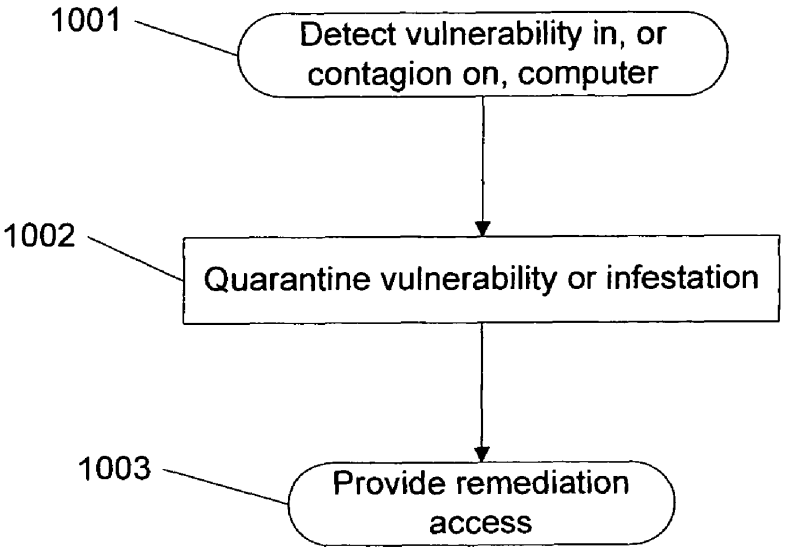

1001
Detect vulnerability in, or contagion on, computer
1002
Quarantine vulnerability or infestation
1003
Provide remediation access

Figure 10A

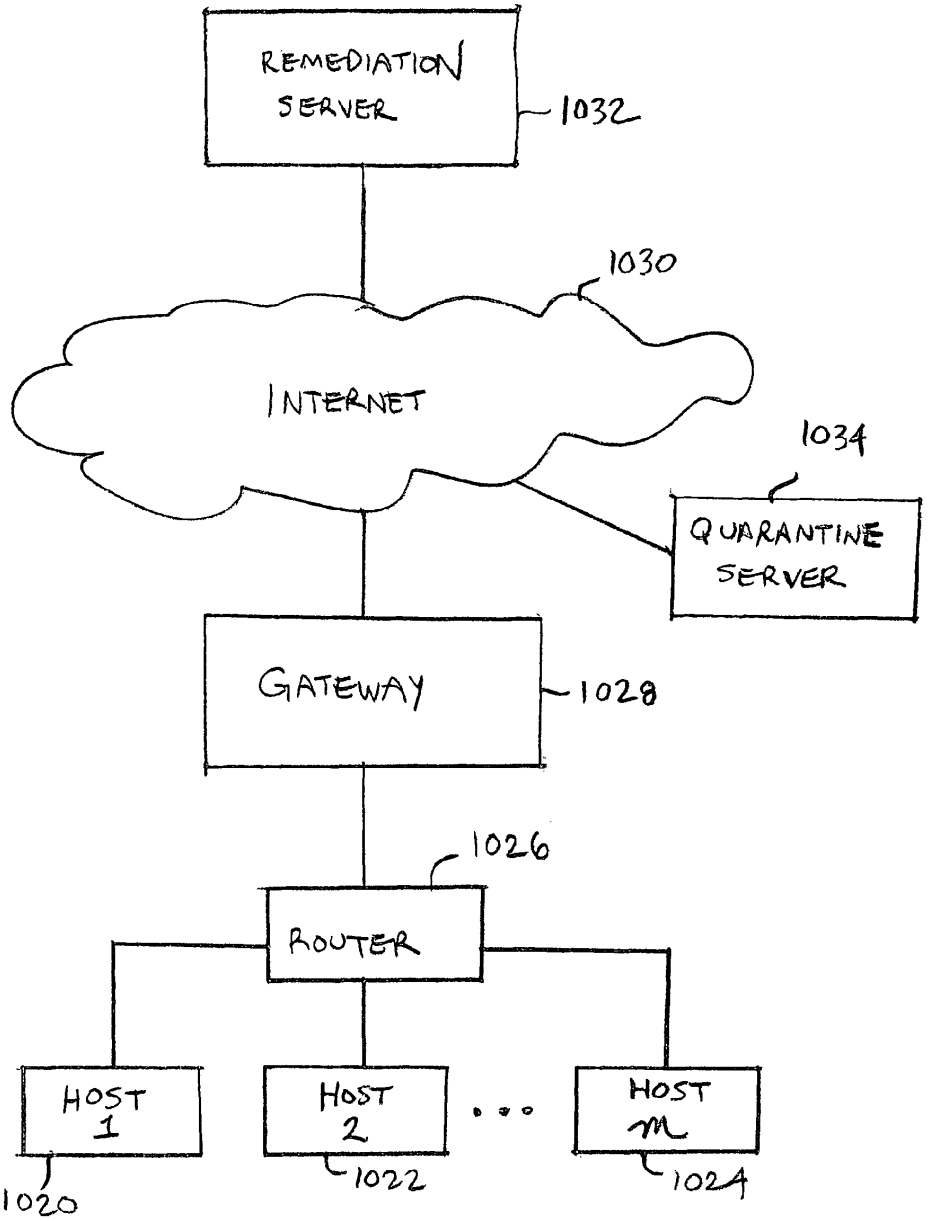

REMEDIATION SERVER
1032
1030
INTERNET
1034
QUARANTINE SERVER
GATEWAY
1028
1026
ROUTER
HOST 1
HOST 2
HOST m
1020
1022
1024

FIG. 10B

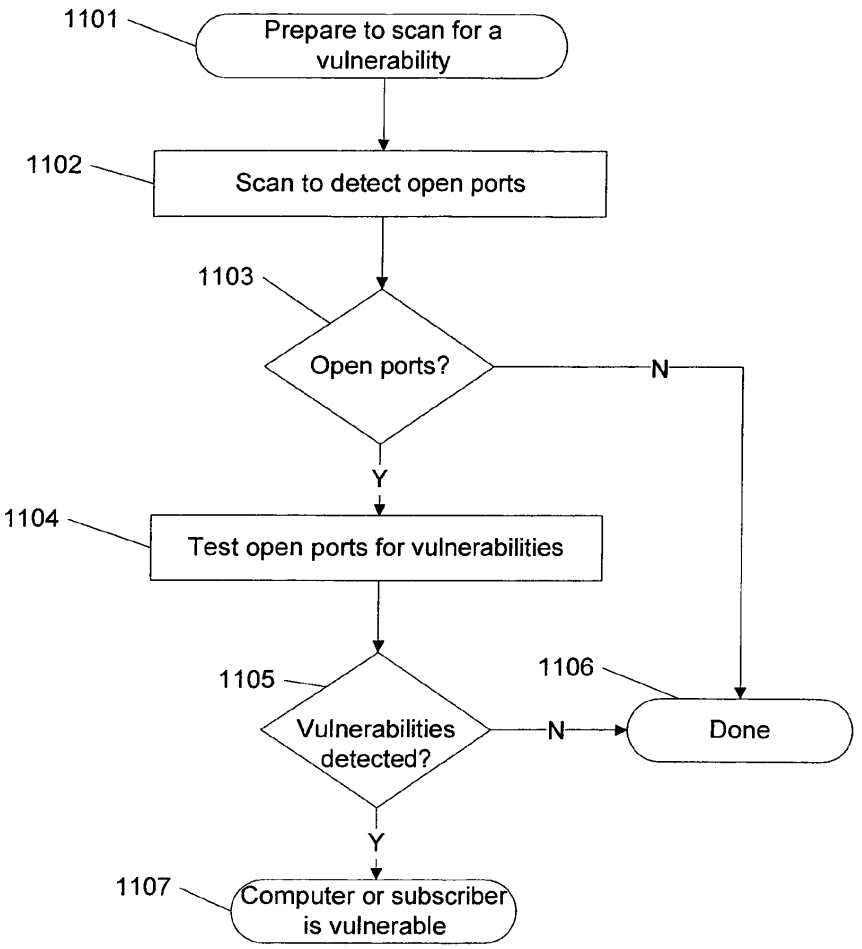

1101
Prepare to scan for a vulnerability
1102
Scan to detect open ports
1103
Open ports?
N
Y
1104
Test open ports for vulnerabilities
1105
Vulnerabilities detected?
N
1106
Done
Y
1107
Computer or subscriber is vulnerable

Figure 11

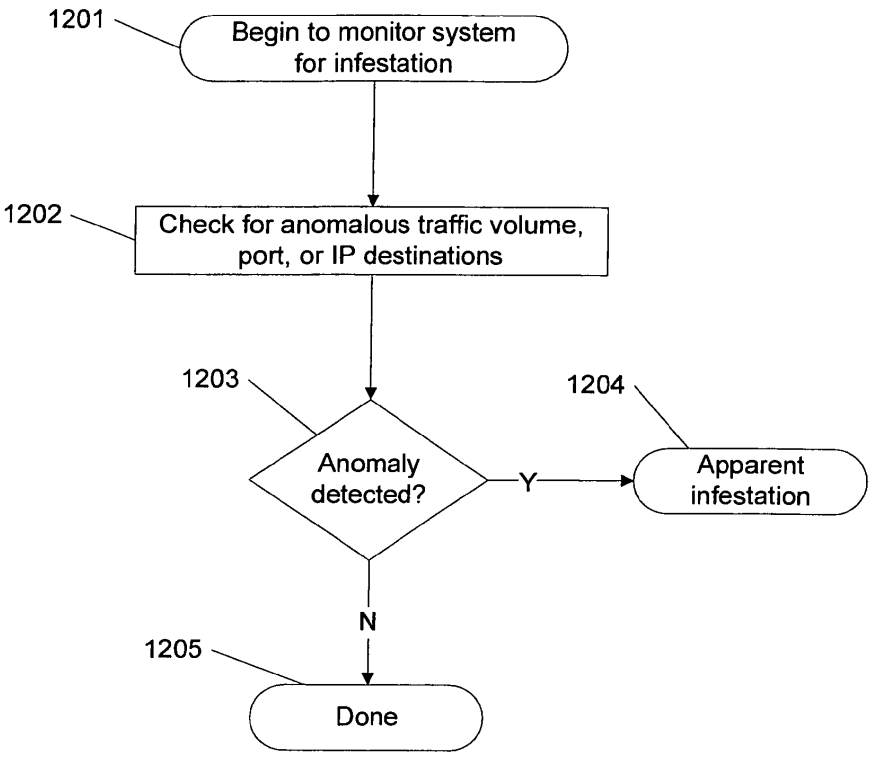

1201
Begin to monitor system for infestation
1202
Check for anomalous traffic volume, port, or IP destinations
1203
Anomaly detected?
Y
1204
Apparent infestation
N
1205
Done

Figure 12

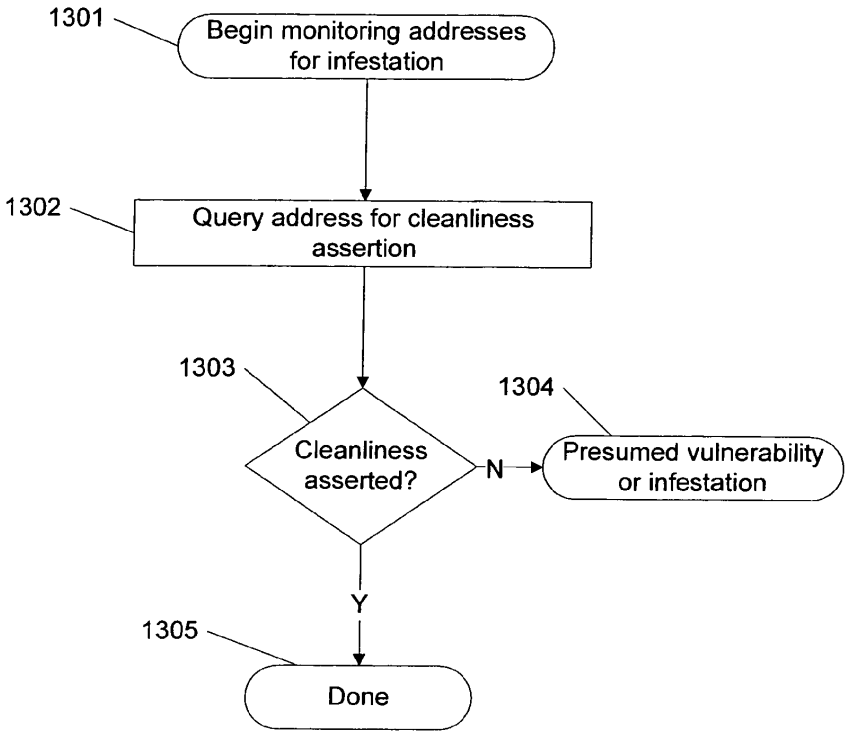

1301
Begin monitoring addresses for infestation
1302
Query address for cleanliness assertion
1303
Cleanliness asserted?
N
1304
Presumed vulnerability or infestation
Y
1305
Done

Figure 13

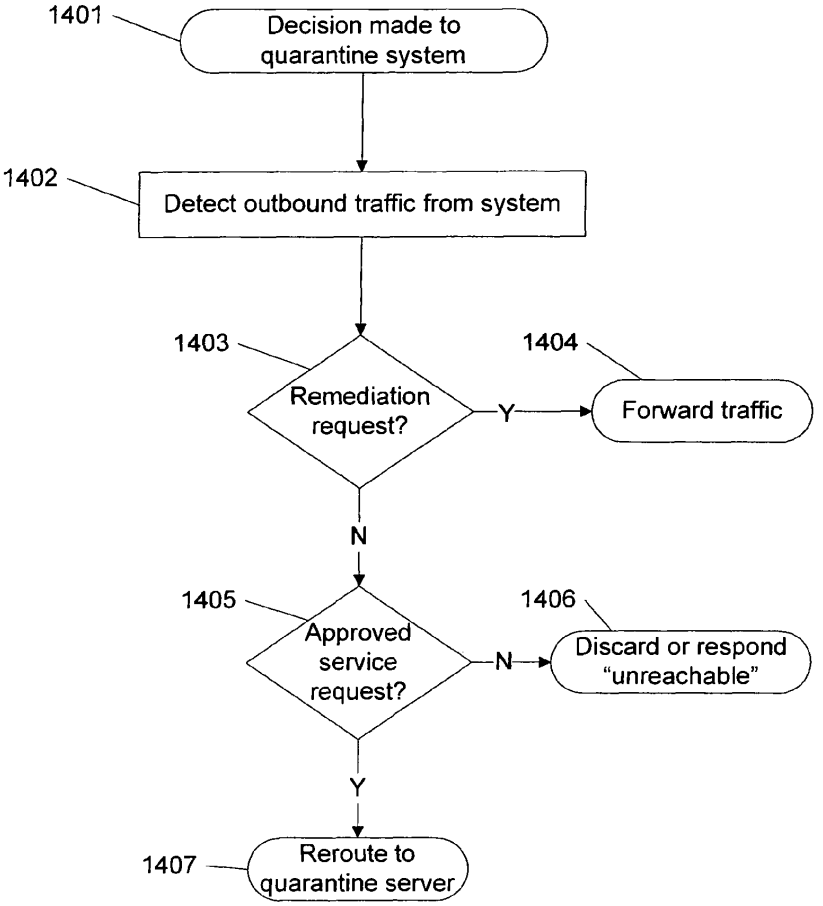

1401
Decision made to quarantine system
1402
Detect outbound traffic from system
1403
Remediation request?
Y
1404
Forward traffic
N
1405
Approved service request?
N
1406
Discard or respond "unreachable"
Y
1407
Reroute to quarantine server

Figure 14

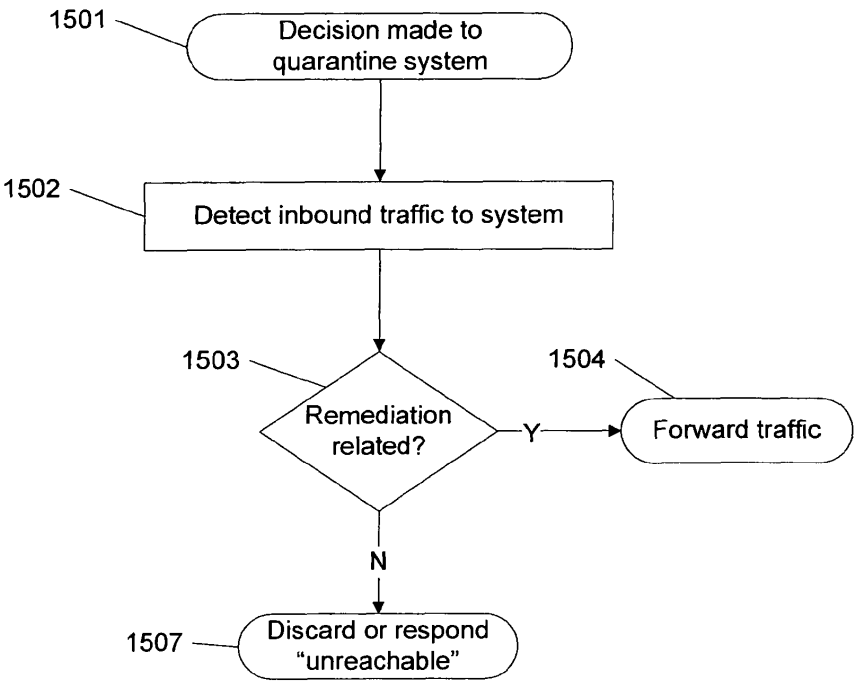

1501
Decision made to quarantine system
1502
Detect inbound traffic to system
1503
Remediation related?
Y
1504
Forward traffic
N
1507
Discard or respond "unreachable"

Figure 15

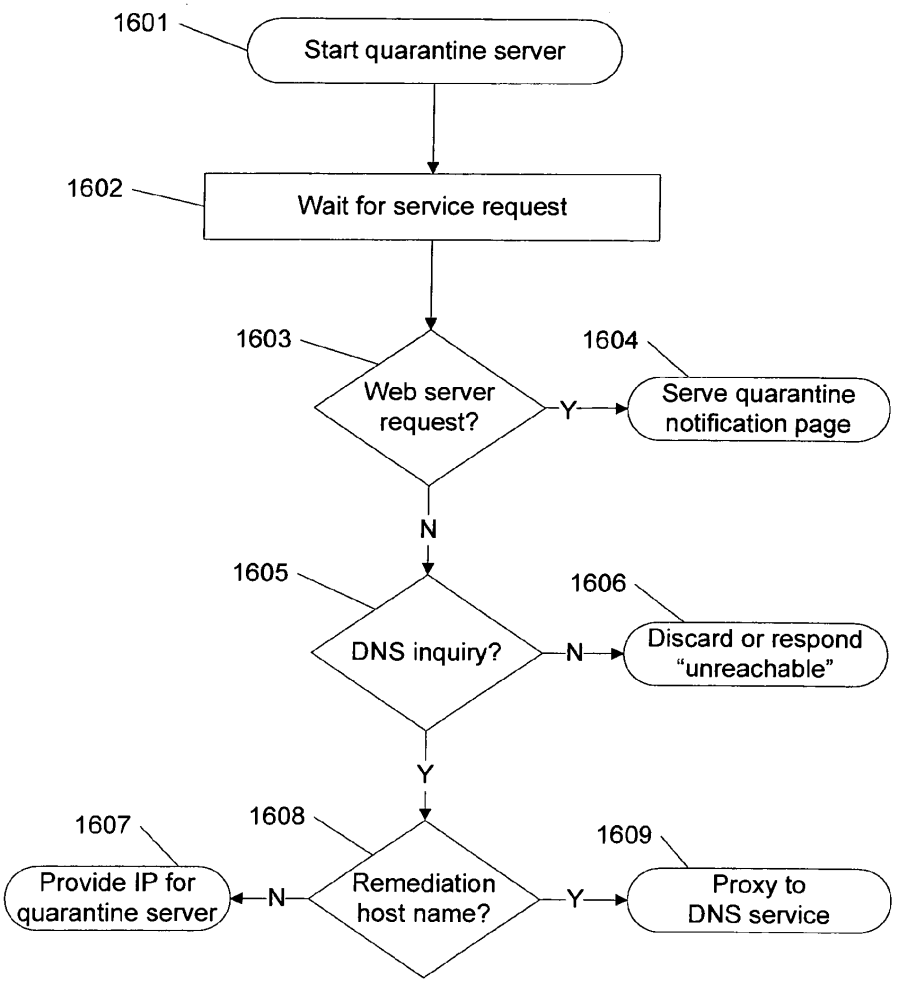

1601
Start quarantine server
1602
Wait for service request
1603
Web server request?
Y
1604
Serve quarantine notification page
N
1605
DNS inquiry?
N
1606
Discard or respond "unreachable"
Y
1607
Provide IP for quarantine server
N
1608
Remediation host name?
Y
1609
Proxy to DNS service

Figure 16

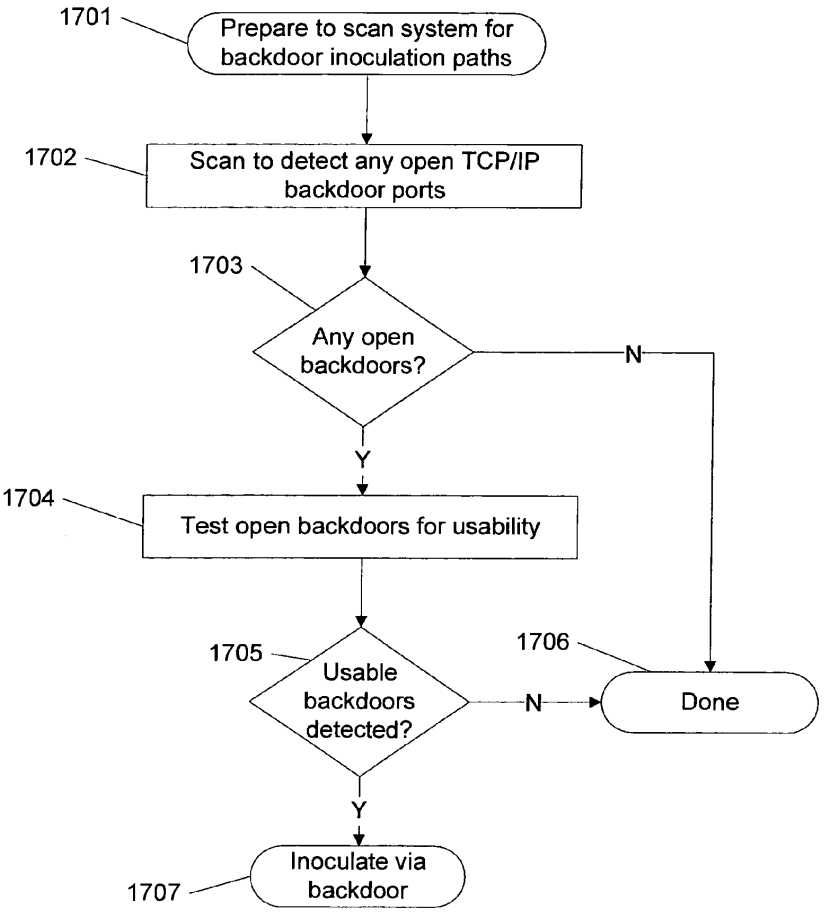

1701
Prepare to scan system for backdoor inoculation paths
1702
Scan to detect any open TCP/IP backdoor ports
1703
Any open backdoors?
N
Y
1704
Test open backdoors for usability
1705
Usable backdoors detected?
N
1706
Done
Y
1707
Inoculate via backdoor

Figure 17

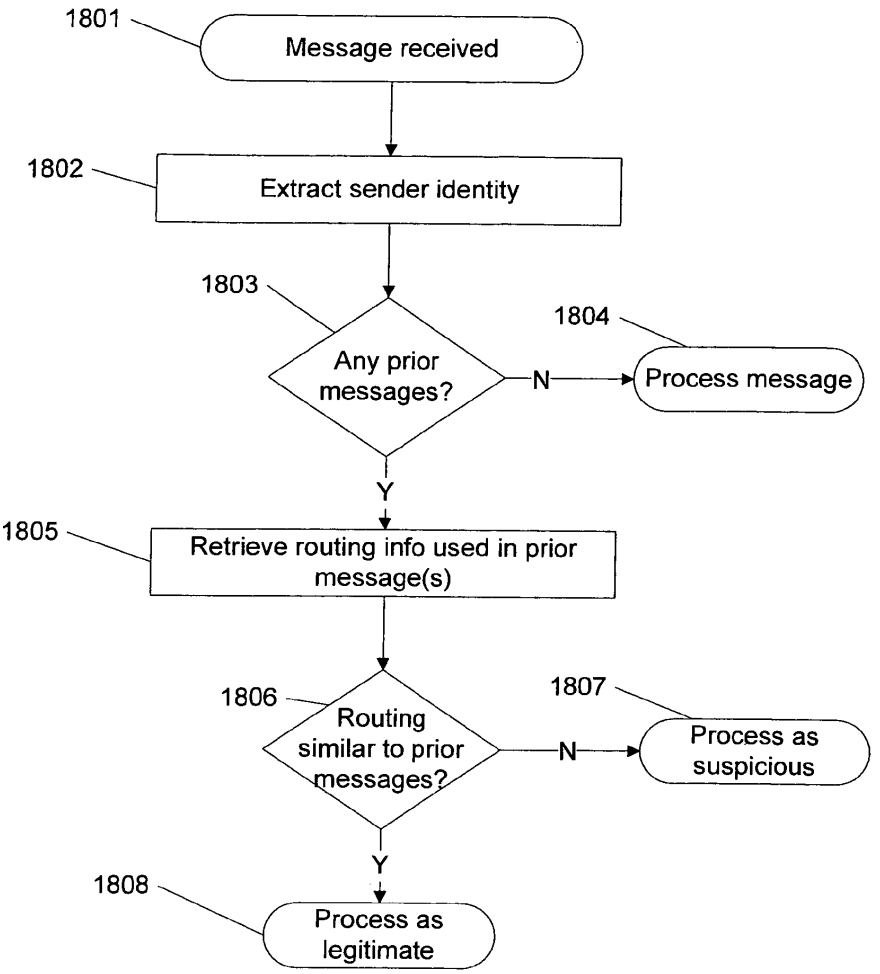

1801
Message received
1802
Extract sender identity
1803
Any prior messages?
N
1804
Process message
Y
1805
Retrieve routing info used in prior message(s)
1806
Routing similar to prior messages?
N
1807
Process as suspicious
Y
1808
Process as legitimate

Figure 18

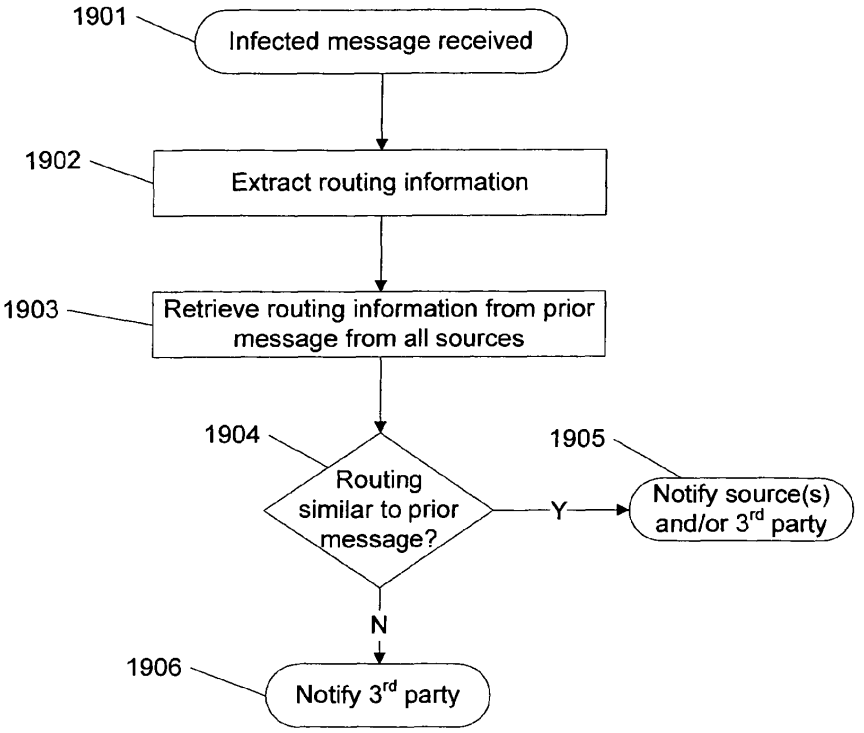

1901
Infected message received
1902
Extract routing information
1903
Retrieve routing information from prior message from all sources
1904
Routing similar to prior message?
Y
1905
Notify source(s) and/or 3rd party
N
1906
Notify 3rd party

Figure 19A

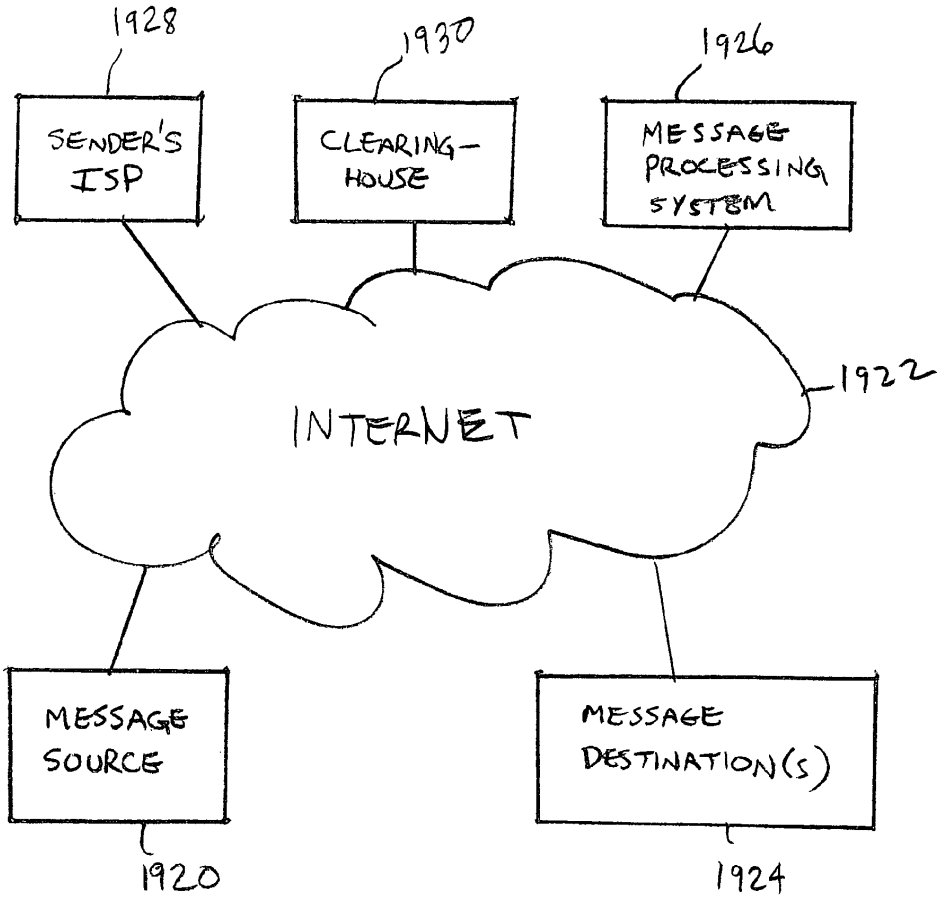

1928
SENDER'S ISP
1930
CLEARING-HOUSE
1926
MESSAGE PROCESSING SYSTEM
1922
INTERNET
MESSAGE SOURCE
1920
MESSAGE DESTINATION(S)
1924

FIG. 19B

CONTAGION ISOLATION AND INOCULATION

CROSS REFERENCE TO OTHER APPLICATIONS

This application claims priority to U.S. Provisional Patent Application No. 60/613,909 entitled CONTAGION ISOLATION AND INOCULATION filed Sep. 27, 2004, which is incorporated herein by reference for all purposes.

BACKGROUND OF THE INVENTION

Laptop and wireless computers and other mobile systems pose a threat to elements comprising and/or connected to a network service provider, enterprise, or other protected networks to which they reconnect after a period of connection to one or more networks and/or systems that are not part of the service provider, enterprise, or other protected network. By roaming to unknown domains, such as the Internet, and/or connecting to such domains through public, wireless, and/or otherwise less secure access nodes, such mobile systems may become infected by computer viruses, worms, backdoors, and/or countless other threats and/or exploits and/or have unauthorized software installed; have software installed on the mobile system by an operator of the protected network for the protection of the mobile system and/or the protected network removed or altered without authorization; and/or have configurations, settings, security data, and/or other data added, removed, and/or changed in unauthorized ways and/or by unauthorized person. Similarly, a stationary system may become infected, e.g., due to receipt and execution of malicious code via a network or other communication and/or a diskette and/or other removable media. Upon connecting to a protected network, a system may infect or otherwise harm resources associated with the protected network before measures can be taken to detect and prevent the spread of such infections or harm. Therefore, there is a need for a reliable way to ensure that a system does not infect or otherwise harm other network resources when connected to a protected network.

BRIEF DESCRIPTION OF THE DRAWINGS

Various embodiments of the invention are disclosed in the following detailed description and the accompanying drawings.

FIG. **1** is a flow diagram of a method for reducing an anti-contagion update interval during periods of suspicious network activity according to some embodiments.

FIG. **2**A is a flow diagram of a method for cooperatively scheduling an anti-contagion update interval according to some embodiments.

FIG. **2**B is a block diagram of a network environment in which anti-contagion updates are distributed according to some embodiments.

FIG. **3**A is a flow diagram of a method for contacting an anti-contagion source and optionally getting an update according to some embodiments.

FIGS. **3**B-**3**D illustrate distributed update distribution as implemented according to some embodiments.

FIG. **4**A is a flow diagram of a method for acquiring an update according to some embodiments.

FIG. **4**B is a block diagram of a client system configured to obtain an update according to some embodiments.

FIG. **4**C is a block diagram of a system used to distribute an update according to some embodiments.

FIG. **5** is a flow diagram of a method for distributing updates according to some embodiments.

FIG. **6** is a flow diagram of a method for establishing a redistribution policy according to some embodiments.

FIG. **7** is a flow diagram of a method for redistributing content such as updates according to some embodiments.

FIG. **8** is a flow diagram of a method for using peers or other indirect data sources to acquire content such as updates according to some embodiments.

FIG. **9** is a flow diagram of a method for redistributing content such as updates according to some embodiments.

FIG. **10**A is a flow diagram of a method for identifying and remedying problems according to some embodiments.

FIG. **10**B is a block diagram illustrating a network environment in which infected hosts and/or networks are quarantined according to some embodiments.

FIG. **11** is a flow diagram of a method for identifying vulnerabilities according to some embodiments.

FIG. **12** is a flow diagram of a method for identifying an infestation according to some embodiments.

FIG. **13** is a flow diagram of a method for monitoring one or more computers for infestation according to some embodiments.

FIG. **14** is a flow diagram of a method for quarantining a computer according to some embodiments.

FIG. **15** is a flow diagram of a method for quarantining a computer according to some embodiments.

FIG. **16** is a flow diagram of a method for a quarantine server to respond to requests according to some embodiments.

FIG. **17** is a flow diagram of a method for scanning backdoors and inoculating according to some embodiments.

FIG. **18** is a flow diagram of a method for categorizing messages according to some embodiments.

FIG. **19**A is a flow diagram of a method for notifying possible sources of infection according to some embodiments.

FIG. **19**B is a block diagram illustrating a network environment in which suspicious messages are identified and/or one or more notifications are sent when a malicious message is detected.

DETAILED DESCRIPTION

The invention can be implemented in numerous ways, including as a process, an apparatus, a system, a composition of matter, a computer readable medium such as a computer readable storage medium or a computer network wherein program instructions are sent over optical or electronic communication links. In this specification, these implementations, or any other form that the invention may take, may be referred to as techniques. A component such as a processor or a memory described as being configured to perform a task includes both a general component that is temporarily configured to perform the task at a given time and a specific component that is manufactured to perform the task. In general, the order of the steps of disclosed processes may be altered within the scope of the invention.

A detailed description of one or more embodiments of the invention is provided below along with accompanying figures that illustrate the principles of the invention. The invention is described in connection with such embodiments, but the invention is not limited to any embodiment. The scope of the invention is limited only by the claims and the invention encompasses numerous alternatives, modifications and equivalents. Numerous specific details are set forth in the following description in order to provide a thorough under-

standing of the invention. These details are provided for the purpose of example and the invention may be practiced according to the claims without some or all of these specific details. For the purpose of clarity, technical material that is known in the technical fields related to the invention has not been described in detail so that the invention is not unnecessarily obscured.

Contagion isolation and inoculation is disclosed. In some embodiments, a request is received from a host, e.g., via a network interface, to connect to a protected network. It is determined whether the host is required to be quarantined. If the host is required to be quarantined, the host is provided only limited access to the protected network. In some embodiments, a quarantined host is permitted to access the protected network only as required to remedy a condition that caused the quarantine to be imposed, such as to download a software patch, update, or definition; install, remove, and/or configure software and/or settings as required by a policy; and/or to have a scan or other diagnostic and/or remedial operation performed. In various embodiments attempts to communicate with hosts not involved in remediation are redirected to a quarantine system, such as a server, that provides information, notices, updates, and/or instructions to the user.

Contagion refers herein to software programs and/or code, data values, configurations, and settings, that expose a computer or other system to security threats, such as unauthorized access and/or use of the system and/or data stored thereon; corruption or loss of data; and/or damage, loss, and/or unauthorized reconfiguration of system software and/or hardware components. Examples of contagion include software programs and/or code designed to perform undesirable and/or unauthorized actions from the perspective of the (authorized) user of a computer, and at the same time perform desired actions from the perspective of the creator or controller of such contagion. Examples of contagion include computer worms, viruses, and Trojan horses. Anti-contagion software refers herein to software that prevents, impedes or remediates contagion, such as Norton Antivirus from Symantec, or VirusScan from McAfee, which identifies and/or removes contagion from a user's computer. Another example of anti-contagion software is software that repairs a security vulnerability through which contagion may spread. Many anti-contagion methods rely on contagion descriptions, such as virus signatures, to identify the contagion, and must be updated regularly to be effective.

FIG. **1** is a flow diagram of a method for reducing an anti-contagion update interval during periods of suspicious network activity according to some embodiments. An update interval refers herein to a period of time between acquisition of updates, such as security updates, virus signature file updates, or virus scanner updates. In this example, network traffic statistics may be monitored (**101**). An example of traffic statistics is latency experienced by email, such as average latency, or median latency. Another example of traffic statistics is the rate of unacceptable attempted TCP/IP traffic, such as attempts to connect to blocked or restricted ports, or insufficiently authenticated (or authorized) connections to ports, or connections that provided invalid or illegal content such as extraordinarily long strings, or specific content detected to be malicious. Recent activity may be evaluated (**102**) to see if it is anomalous. Examples of anomalous activity include median email latency in mail queues beyond some threshold, for example above a 4 hour median queuing delay, and rates of unacceptable attempted TCP/IP traffic beyond some threshold, for example greater than two attempts per minutes over a one hour period. Another example of a threshold for anomalous activity is a comparison with recent historical statistics, such as unacceptable attempted TCP/IP traffic at a rate greater than ten times the expected rate over a 30 minute period. The expected rate may for example be an average over a recent historic period, such as an average rate over the last week of connectivity. If activity is not deemed anomalous (**103**), then in this example, processing continues with no change (**104**) in the next update interval. If the activity is deemed anomalous (**103**), then in this example, the next update interval is reduced, for example reduced to some proportion of the planned update interval, for example reduced to 25% of the previous or default update interval. Another example of reducing the next update interval is to reduce the currently remaining time until an update, for example reducing by a proportion, such as 50%, the remaining time until an update is requested.

FIG. **2**A is a flow diagram of a method for cooperatively scheduling an anti-contagion update interval according to some embodiments. In this example, the planned time for contacting an anti-contagion source may arrive (**201**). An old version identifier may be retrieved (**202**), such as a checksum or hash for a file, or a version number for software stored in a file system or registry, or a version number for an anti-virus signature file stored in a file system or registry. An update may be requested from an anti-contagion source (**203**), for example as is discussed in conjunction with FIG. **3**A, which may include updating or concluding that an update is not needed. A time for next contacting an anti-contagion source is decided upon (**204**). In some embodiments, the scheduled time may be made sooner when the source of the update, such as a vendor of software to which the update pertains, anticipates the impending release of a fully tested update, such as scheduling a next contact time in a few hours, or a few days. If the current computer's version of anti-contagion software contains resolutions to all known problems and otherwise anomalous traffic patterns then, in this example, the update interval may be made longer, such as the full default interval for a computer, for example one week. In some embodiments, the selection of the next update interval may be dependent on the specific product. For example, a "pro version" may have a short default interval, such as one day, and a "home version" may have a relatively long default interval, such as two weeks or one month.

FIG. **2**B is a block diagram of a network environment in which anti-contagion updates are distributed according to some embodiments. A plurality of clients **1** to "n", represented in FIG. **2**B by clients **220**, **222**, and **224**, are connected to the Internet **226**. An update server **228** also is connected to the Internet **226**. In various embodiments, one or more of the clients **220**-**224** is configured to request and obtain updates, if any, from update server **228** via the Internet, e.g., as described above in connection with FIG. **2**A.

FIG. **3**A is a flow diagram of a method for contacting an anti-contagion source and optionally getting an update according to some embodiments. In this example, an old version identifier may be retrieved (**301**), for example as was discussed in **202** of FIG. **2**A. Contact may be made with an anti-contagion source (**302**) via a network such as the Internet. The old version identifier may be provided, for example using a transport protocol such as TCP/IP, and a data encapsulation format such as XML, or using a higher level protocol and format such as SOAP. A decision may be made as to whether an update is needed (**303**). In some embodiments, this update decision may be made by a server, upon review of the old version identifier. In some embodiments, this update decision may be made by the user's computer. In some embodiments, the update decision may be made by incorporating preferences held on a user's computer, such as a pref-

erence to always have the newest updates, or preferences to avoid updates that have been available for less than a threshold length of time (for example, 2 weeks), or preferences that accept an additional update when the source indicates urgency associated with the update. In some embodiments, the update decision may be made based on considerations such as the size of the potential update, for example by avoiding the overhead of small updates, and only performing an update when a threshold amount of change has taken place, or when a threshold length of time has elapsed since an actual update was made. An example of a threshold amount of change taking place is the addition of a prescribed number of viral signatures, such as 2 new signatures. An example of a threshold length of time is 3 weeks. If the update is not needed (**303**) then the process of contacting the source is complete (**304**) in this example. If an update is needed (**303**) then an update is acquired in this example via direct or indirect channels (**305**), for example as further described in conjunction with FIG. **4**A. After the update has been acquired (**305**), the contact may be complete (**306**).

Distributed update distribution is disclosed. In some embodiments, at least some users receive an update indirectly, e.g., from another user that received it previously, either directly from the original source or indirectly from a source that had itself earlier received it, again either directly or indirectly, and so on. FIGS. **3**B-**3**D illustrate distributed update distribution as implemented according to some embodiments. In the example shown in FIGS. **3**B-**3**D, it is assumed for expository purposes that each host that receives the update will serve as a redistribution node, each recipient has the ability to redistribute at the same rate as the original source, and there is continuous demand for the update from all sources. As shown in FIG. **3**B, at a time t**1** a first direct recipient of the update D**1** (**332**) downloads the update from the original source S (**330**). As shown in FIG. **3**C, at a time t**2** a second direct recipient D**2** (**334**) downloads the update from the original source and an indirect recipient R**11** (**336**) downloads the update from the first direct recipient D**1** (**332**), which in this case acts as a redistribution node. As shown in FIG. **3**D, at a time t**3** a third direct recipient D**3** (**338**) downloads the update from original source S (**330**), a second indirect recipient R**12** (**340**) downloads the update from the first direct recipient D**1** (**332**), a third indirect recipient R**21** (**342**) downloads the update from the second direct recipient D**2**, and a fourth indirect recipient R**111** (**344**) downloads the update from first indirect recipient R**11** (**336**). In this example, in the three download cycles shown in FIGS. **3**B-**3**D on its own the original source would only have been able to download the update to the three direct recipients, but by using previous recipients to redistribute, in the example shown in FIGS. **3**B-**3**D a total of seven recipients have been able to obtain the update. In a generalized case, under the assumptions used in this example in "n" download cycles/periods a total of $2n-1$ recipients are able to obtain the update, whereas only "n" recipients, i.e., one per cycle, would otherwise have been able to obtain the update if it were only available directly from the original source. In an even more generalized case, under an assumption that each server may download roughly k copies in a time interval, for example where more than one download is provided by a server at a time, then in "n" download cycles (or periods) a total of roughly $(k+1)n-1$ recipients are able to obtain the update, where as only $k*n$ recipients, i.e., k per cycle, would otherwise have been able to obtain the update if it were only available directly from the original source.

FIG. **4**A is a flow diagram of a method for acquiring an update according to some embodiments. In this example, an update source is contacted (**401**), such as via an Internet connection, for example via TCP/IP. An offer to serve as a redistribution source is made (**402**). In some embodiments, a redistribution source may be contacted by other computers to acquire an update. An example of an offer to be a redistribution server is described in conjunction with FIG. **6**. In various embodiments, **402** is optional, performed only if a criterion or test is satisfied, or omitted. An update or an update identifier is acquired (**403**). An example of an update identifier is a version number for an update. Another example of an update identifier is a checksum for an update. Another example of an update identifier is a set of checksums (such as MD5, SHA-1, or CRC-32) for a corresponding set of byte ranges that form the update. Another example of an update identifier is a combination two or more identifiers, such as the identifiers enumerated above. If the update was completed (**404**) then the user confirms redistribution sourcing status (**408**) in this example. In some embodiments, confirmation includes asserting that the client is prepared to redistribute copies of the received data to one or more recipients. If the update was not completed (**404**), then a list of peer or other sources is acquired (**405**) in this example. In some embodiments, the list is or may be explicit. In some embodiments, the list is or may be implicit, for example represented by an iterator or accessor, which may retrieve elements on an as needed basis. In some embodiments, implicit lists are dynamic, and calculated or created in part on an as needed or as requested basis. In some embodiments, the list of other sources contain addresses that may be contacted to acquire an update, such as a list of IP addresses and ports, as well as optionally some authentication information to be used when contacting a listed source. In some embodiments, a requesting host that agrees to serve as a redistribution source is provided with a list of sources from which the update is available immediately, or at least sooner than it will be made available to hosts that will not redistribute. One or more attempts may be made to acquire the update from a listed source (**406**), for example as exemplified in FIG. **8**. If the update is completed via a listed source (**407**), then redistribution status is confirmed (**408**) in this example. If the update is not complete (**407**) then a reservation is optionally negotiated and/or acquired (**409**) in this example. In some embodiments, a reservation may include a time and/or a specific source or IP address. In some embodiments, a reservation address may be selected from a collection of source addresses, for example in order to distribute the load across more than one server, or to provide a server that is closer to a recipient. An example of selecting a server close to a recipient is to select a server having relatively lower latency during a download, or relatively lower cost, for example by selecting a source within a recipient's ISP complex or within a recipient's intranet or home network. If a reservation time was acquired, then a wait for that reservation time (**410**) is performed in this example. At the reservation time, a further attempt to acquire an update or update identifier from the source (**403**) is made, and the flow proceeds as described earlier. This flow may continue until an acquisition is complete (**408**), as also described earlier. In various embodiments, one or more of confirming redistribution sourcing status (**408**), attempting to obtain the update from a redistribution source (**405-407**), and/or acquiring and/or waiting for a reservation time (**409-410**) is/are optional, conditional, or omitted.

FIG. **4**B is a block diagram of a client system configured to obtain an update according to some embodiments. In some embodiments, the client system shown in FIG. **4**B is used to implement one or more of clients **220-224** of FIG. **2**B. The client system **420** includes a communication interface **422**,

such as a network interface card, configured to send/receive communications via a network, such as the Internet. The client system **420** further includes an update agent and/or browser **424**. In some embodiments, the update agent/browser **424** comprises browser software used to navigate the World Wide Web, e.g., to send/receive data to/from a website from which an update is available for download. In some embodiments, update agent/browser **424** includes an agent associated with anti-contagion software installed on the client system **420**. In various embodiments, the update agent/browser **424** comprises one or more components and is implemented as one or more software programs running on a processor and/or one or more hardware components. A settings data store **426** stores one or more settings and/or other data associated with the software with which an update and/or software to which the update does or may pertain, such as periodicity or next scheduled (e.g., reservation) time to check for updates, a list of update identifiers, locators, and/or redistribution sources from which an update is or may be available, policies for whether and/or under what terms, conditions, circumstances, etc. the client system **420** may be used redistribute an update, a last update time, an old version identifier (see, e.g., **202** of FIG. **2**A), etc.

FIG. **4**C is a block diagram of a system used to distribute an update according to some embodiments. In some embodiments, the system of FIG. **4**C is used to provide an update server, such as update server **228** of FIG. **2**B. In this example, the distribution system **440** includes a communication interface **442** configured to send/receive data via a network, such as the Internet. An update distribution engine **444** is configured to receive and service requests for updates. For updates provided directly to a request source, the update distribution engine **444** obtains the update from an update storage location **446** and provides the update to the request source via the communication interface **442**. In some embodiments, if data included in the request indicates the source of the request is or may be willing to serve as a redistribution source, the update distribution engine **444** hands the request off to a redistribution coordinator **448**, which negotiates with the request source, e.g., based on policies, settings, rules, etc. stored in a policies and lists data store **450**, through communications sent/received via communication interface **442**, to determine if the requesting source will serve as a redistribution source and, if so, to determine the terms and conditions of such service and data, such as an address, locator, or identifier, that can be provided to other requesting systems and/or users to enable them to attempt to obtain the update from the redistribution source. In some embodiments, if update distribution engine **444** determines that it is not able to provide the update to the request source directly (e.g., due to capacity limits or because a policy applicable to the request and/or the request source indicates the request source should be directed to a redistribution source, e.g., because the request source has a less expensive version of the software or is not willing to serve as a redistribution source), the update distribution'engine **444** hands the request off to redistribution coordinator **448**, which provides to the request source an update identifier and a list of one or more redistribution sources from which the update is or may be available to the request source.

FIG. **5** is a flow diagram of a method for distributing updates according to some embodiments. In this example, a source prepares to supply updates to recipients (**501**), for example by starting up a server, for example an HTTP or FTP server, or a multicast server, such as an RTP multicast server. The server waits for an event (**502**), such as a contact on a predefined port, or a notification that the server should start a multicast. If a multicast-time event appears (**503**) then an update is multicast (**504**) in this example. In some embodiments, a multicast time event may be generated internally or externally to the server to start a multicast. In some embodiments, a multicast-time event may be automatically generated at predetermined times, such as hourly, weekly, or daily, or on individually specified dates and times. In some embodiments, a multicast-time event may include a specific identifier for the multicast. If an event of the form of an update request arrives (**503**) then a test for authenticity is optionally performed (**505**) in this example. An example of determining whether a request is authentic is to check a credential. Examples of credentials include a provided shared secret, such as a secret used by all requestors to this server, or a different secret used by each requestor such as a user serial number, or a cryptographically signed authentication, such as credentials or a certificate issued to users that had paid for an update service, for example, by inclusion in a purchased software package. In some embodiments, all requests are assumed to be authentic without the use of any authenticator. If the request was not authentic (**505**) then, in this example, the server continues to wait for an event (**502**). If the request was authentic (**505**) then the server may determine a necessary update (**506**) in this example. In some embodiments, determination of an update may be made based on the requestor's state, which in some embodiments may be indicated in a received request for update. A redistribution contract may be negotiated (**507**). In some embodiments, some requestors may agree to redistribute updates to other requestors. In some embodiments, some requestors may decline to redistribute updates. In some embodiments, a source may decline redistribution proposals from a requestor. An example of a redistribution contract is an agreement not to redistribute. Another example of a redistribution contract is an agreement to redistribute a fixed number of copies, such as five copies. Another example of a redistribution contract is an agreement to redistribute for a fixed period of time, such as one day. Another example of a redistribution contract is an agreement to redistribute within a specified geographic region within a fixed period of time. Another example of a redistribution contract is an agreement to redistribute indefinitely, for example until a replacement update is provided. A redistribution contract may be made with terms that are combinations of the above mentioned exemplary term limitations. An update may be provided directly or indirectly (**508**). A direct update may for example be provided by allowing an FTP download, or by encapsulating the transmission in an HTTP transfer, for example using XML and/or an appropriate coding, such as uuencode. An indirect update may be provided by directing the requestor to an alternate source, such as a redistribution server, or a server at a different address, or a server at a different time, such as a future time, or a combination of two or more such specifications. In some embodiments, a decision to provide direct vs. indirect updates may be based in part on the redistribution contract, for example a requestor that is willing to redistribute may be updated sooner or more directly. An example of a redistribution contract is a promise by a user to provide redistribution in exchange for purchase price concession, such as a reduced initial purchase price, or a quantity discount price. In some embodiments, a decision to provide direct vs. indirect updates may be based in part on previous agreements, such as prepayment for expedited service, such as may be included in a more expensive version of a product that utilizes the updates. The server may wait for an event (**502**). In some embodiments, events may be processed in parallel with previous events, or may be sequentially pro-

cessed, or some combination, for example up to a threshold number of events in parallel, such as a maximum of 32 events in parallel.

FIG. **6** is a flow diagram of a method for establishing a redistribution policy according to some embodiments. In this example, a computer may prepare to offer to be a redistribution source (**601**). In some embodiments, preparations may include retrieving preferences, such as the number of redistributions that may be done, or the duration of time that the computer is willing to act as a redistribution source. In some embodiments, preferences may be defaults, or may be configured by a user and stored persistently, such as in a registry or file system. A contact address may be proposed to the source and a policy may be negotiated (**602**). A contact address may for example be a globally routable IP address and a port, or a local IP address (reachable from a LAN behind a firewall) and port that can be accessed by other computers on the LAN, such as computers that shared a common NAT address in their contact with the original source. In some embodiments, negotiation may include proposing a policy. In some embodiments, negotiations may involve a counter proposal in response to a proposal. An example of a counter proposal criterion is to require a minimum duration for redistribution. If terms were not acceptable (**603**) then, in this example, the computer does nothing (**604**). If the terms were acceptable (**603**) then, in this example, the computer prepares to service redistribution requests (**605**), for example by establishing a server to respond at the agreed contact address, reserving storage to hold the content for redistribution, etc. In some embodiments, redistribution policy may be a broader contractual matter, where an update is provided more quickly or more directly in exchange for a promise to redistribute. In some embodiments, an auditing mechanism may be used to measure compliance and penalize contract breakers, such as failure to accept connections for redistribution, where penalties may for example include refusal to accept future similar contracts. As another example, a lower purchase or subscription price may be offered for operators of a software product, such as antivirus software, who agree to act as redistribution points. In some embodiments such purchase agreements may be configured into the software, for example by establishing agreements to act as a redistribution point for updates as defaults that a user cannot modify.

FIG. **7** is a flow diagram of a method for redistributing content such as updates according to some embodiments. In this example, a redistribution computer is contacted and confirms its sourcing status (**701**), for example by accepting a TCP/IP connection on a predefined port, and processing a request for content. The redistribution computer may optionally contact another computer, for example a more original source, and affirm its redistribution sourcing status (**702**). A more original source may be a source that has provided content in the past to the redistribution computer, directly or indirectly. In some embodiments, contacting a more original source demonstrates that the redistribution computer is performing redistribution. In some embodiments, contacting a more original source allows the redistribution computer to further authenticate the computer that requested the content, for example by getting confirmation from a more original source that a specific computer was authorized to request the content. An authentication to be used to authenticate requests is negotiated (**703**). In some embodiments, authentication includes providing a shared secret to the redistribution computer. In some embodiments, authentication includes presenting an authorization cryptographically signed by a more original source, such as a certificate. In some embodiments, all requests are presumed authentic. If the request is not authentic (**704**), then nothing further is done (**705**) in this example. If the request is authentic (**704**), then the requested update is supplied (**706**) in this example. The updates may be supplied for example via a transport protocol TCP/IP, or a higher level protocol such as FTP, HTTP, or HTTPS, etc.

FIG. **8** is a flow diagram of a method for using peers or other indirect data sources to acquire content such as updates according to some embodiments. In this example, a list of one or more peers or other indirect sources for acquiring updates is obtained (**801**), for example as was discussed in conjunction with **405** of FIG. **4**A. In one example, an indirect source may be a peer, such as another computer that has downloaded the content. In another example, an indirect source may be a computer configured to redistribute content without other use of the content, such as an ISP server established to redistribute content to subscribers. In another example, an indirect source may be a server created or authorized by the original source to perform additional distributions. In some embodiments, an obtained list of indirect sources may be implicit, for example by referring to a list using an iterator or accessor construct that can itemize entries in the list when requested, for example using a query to a service when an additional item in the list is required, which may for example dynamically complete the list on an as requested basis. A next address and (optional) authenticator in the list of sources is retrieved (**802**). Any order may be used. In some embodiments, an order of processing may correspond to the original list order. In some embodiments, an order of processing may be such that the earlier processed addresses would be expected to have lower latency and/or higher bandwidth, for example by first processing a local address on a LAN. In some embodiments, elements of the list are or may be processed in parallel, and portions (such as byte ranges) of the content may be acquired from different addresses in the list. An address on the list may be contacted to request content (**803**), which may in some embodiments be performed as described in conjunction with FIG. **7**. In one example, a source address may consist of an identifier for a broadcast with optional time(s) for a broadcast, and contact with the source may include receiving the broadcast. In another example, contact with the source may include making a connection such as a TCP/IP connection with the source. If the acquisition is complete (**804**), then the processing is complete (**805**) in this example. If the acquisition was not completed (**804**) then additional sources in the list are considered in this example. If there are more sources in the list (**806**) then, in this example, a next source is retrieved along with an optional authenticator (**802**) and the processing continues. If there are no more sources (**806**) then the attempted acquisition from source(s) is unsuccessful (**807**) in this example.

FIG. **9** is a flow diagram of a method for redistributing content such as updates according to some embodiments. In this example, a computer prepares to supply content (or updates) to a requestor/recipient (**901**), for example by starting up one or more servers. Examples of a server include an HTTP server, an FTP server, and a multicast server. A server waits for an event (**902**), such as a contact on a predefined port, or a notification that the server should be terminated, or a notification that a time for a multicast has arrived. If a service expiration event appears (**904**) then the supply service is terminated (**905**) in this example. A service expiration event may be automatically generated, for example after a predetermined duration of service, or after a predetermined number of copies of an update have been redistributed. If a request event arrives (**904**), then a test for authenticity is performed (**906**) in this example. Authenticity may be established in the request, for example by receiving and validating, or perform-

ing a mutual proof of possession of, a shared secret, such as a secret used by all requestors to this server, or a different secret used by each requestor of this service and held in a list of single-use secrets by this server, or by providing a cryptographically signed authentication. In some embodiments, all requests are considered authentic without the use of any authenticator. If the request was not authentic (**906**) then, in this example, the server continues to wait for an event (**902**) as described earlier. If the request was authentic (**906**) then, in this example, the server transmits a copy of the content to the requestor (**907**), for example by allowing an FTP download, or by encapsulating the transmission in an HTTP transfer, for example using XML and/or an appropriate coding, such as uuencode.

FIG. **10**A is a flow diagram of a method for identifying and remedying problems according to some embodiments. A problem may be identified as some event or status that may cause direct or potential harm. Exemplary problems include vulnerability or infestation. In this example, a problem such as a vulnerability or infestation is detected (**1001**). Infestation herein refers to the current presence and/or execution of contagion. In some embodiments, detection may be made by a router, firewall, or other network component, separately or in concert, using techniques such as those illustrated by FIGS. **11**, **12**, and **13**. In some embodiments, detection may be made by a computing device, for example a computing device operated by an ISP. After a detection of a problem, a computer or network containing the problem is quarantined (**1002**). In some embodiments, quarantining includes preventing unauthorized communications from being sent from the quarantined computer, such as efforts to infect, probe, or harass other computers, or efforts to communicate data without authorization, including for example file contents and keyboard entry logs. In some embodiments, no communications are authorized. In some embodiments, quarantining includes preventing undesired communications from reaching the quarantined computer, such as control messages or infectious messages. Examples of quarantining activities are illustrated in FIGS. **14** and **15**. In this example, the quarantined computer is provided access to remediation services (**1003**). In some embodiments, access to remediation includes or may include allowing connections to servers providing security patches or advisories, for example allowing connections to a vendor's server, or providing virus and worm disinfecting tools, such as focused removal tools, or data updates for previously installed repair tools. In some embodiments, access to remediation includes allowing direct access to vendor sites known to provide remediation assistance, such as to Microsoft's Windows Update service, where security patches may be obtained. In some embodiments, access to vendor sites is restricted by the quarantine to certain classes of traffic, such as HTTP requests, certain port access, such as port **80** or port **443**, certain content requests, such as specific URLs or specific file downloads. In some embodiments, remediation content may be substituted for other requested content, for example as discussed in conjunction with FIG. **14**.

FIG. **10**B is a block diagram illustrating a network environment in which infected hosts and/or networks are quarantined according to some embodiments. A plurality of hosts **1** to "m", represented in FIG. **10**B by hosts **1020**, **1022**, and **1024**, connect via a router **1026** and a gateway **1028** to the Internet **1030**. In various embodiments the router **1026** is omitted or included in the gateway **1028**. In various embodiments, gateway **1028** comprises a gateway, router, firewall, or other device configured to provide and control access between a protected network and the Internet and/or another public or private network. In various embodiments, the process of FIG. **10**A is implemented, in the event of quarantine of one or more of hosts **1020**-**1024**, for example, on the affected (quarantined) host, router **1026**, and/or gateway **1028**. In some embodiments, a quarantined host (or a host associated with a quarantined network or sub-network) is permitted to access a remediation server **1032**, e.g., to download a patch, more current threat definition, etc. In some embodiments, requests to connection to a host other than the remediation server **1032** are redirected to a quarantine server **1034** configured to provide a notice and/or other information and/or instructions to a user of the quarantined host. In various alternative embodiments the quarantine server **1034** is omitted or resides on the router **1026** and/or gateway **1028**.

FIG. **11** is a flow diagram of a method for identifying vulnerabilities according to some embodiments. In this example, preparation is made to scan for one or more vulnerabilities (**1101**), for example by retrieving one or more IP addresses of computers that need to be scanned. For example, an IP address may be provided by a DHCP sever hosted in an ISP network complex, to identify current subscriber IP addresses. As another example, a subnet range may be scanned, for example by taking an address of the form A.B.C.X, and using some or all values of X in the range 0 to 255 to enumerate distinct IP addresses. A scan of one or more IP addresses may optionally be performed to detect an open port (**1102**). An example of scanning an IP address is scanning to detect open TCP/IP ports (**1102**). An open port may be identified by attempting to form a connection, such as a TCP/IP connection to that port. In some embodiments, only open ports on a given computer will provide acknowledgment messages needed to establish a TCP/IP connections, which for example causes a success code to be returned by Unix "connect( )" and a closed port will provide no response to the attempt to connect, which for example would cause an error code to be returned by the Unix function call "connect( )". If no open ports are detected (**1103**), then the scanning is complete (**1106**) in this example. If open ports are detected (**1103**) then those ports are tested for vulnerabilities (**1104**) in this example. In some embodiments, an open port is assumed (**1103**) and a test for vulnerability is performed or attempted (**1104**), for example by communicating a vulnerability test without independently testing to see if one or more requisite ports are open. In some embodiments, a vulnerability test of a port may include sending data that is known to create a discernable response when a vulnerability is present, such as sending a long string known to induce a buffer overflow with a consequent errant response, where the errant response may for example include a characteristic communication, premature termination of the connection, or other erroneous response(s). In some embodiments, a test for a vulnerability is an attempt to exercise a service, such as a sendmail or an RPC service, in search of a misconfiguration of that service that creates a vulnerability. One example of a misconfiguration of a service is provision of any such service. Another example of a misconfigured service is one where a password should be required for security purposes, but no password is required by the misconfiguration. Another example of a misconfiguration is the use of an easily guessable password, such as the default installation password, when resulting access creates a security vulnerability. Another example of a misconfigured service is support for a service provided by an unauthorized agent, such as a worm or virus. If no vulnerabilities are detected (**1105**) then scanning is complete (**1106**) in this example. If vulnerabilities are detected (**1105**) then an associated IP address is flagged as vulnerable (**1107**) in this example. In the case of a subscriber to an ISP, an IP address may be translated to a particular subscriber identity, for

example by checking in DHCP or static IP allocation tables for the subscriber's identity, or for example by extracting a MAC address from traffic associated with the IP address, and such identity may be appropriately flagged or recorded. In some embodiments, the record of the vulnerability may additionally include the port, and the details of the vulnerability that was detected, such as which buffer overrun was vulnerable, or which service was misconfigured to create a vulnerability.

FIG. **12** is a flow diagram of a method for identifying an infestation according to some embodiments. In this example, a device near a computer, such as a firewall on a LAN or an ISP's router, is used in some embodiments to monitor a computer for infestation (**1201**). Network traffic is monitored for anomalous traffic volume, port, or IP destination (**1202**), for example by detecting large volumes of outbound traffic, such as email traffic or instant messaging traffic. Another example of anomalous traffic is a TCP/IP connection to a server that is known to control a collection of infested machines, in a way that suggests the connecting machine is controlled by the server, such as connecting and transmitting an assertion that it is controlled or available to comply. Another example of anomalous traffic is a connection to a computer that is a current target of a denial of service attack, with a pattern that is consistent with the attack, such as a connection to a targeted port, or the use of a falsified return IP address. In some embodiments, a falsified return address may be detected by a router, such as an ISP router, that is aware that the return address is not on the sourcing subnet, or not associated with the MAC address that has supplied the corresponding packets. If an anomaly is detected (**1203**) then an apparent infestation is noted (**1204**) in this example. In some embodiments, a response to infestation may include quarantining of the computer or network containing the computer, for example as discussed in conjunction with FIGS. **14**, **15**, and **16**. In some embodiments, a response to infestation may include an attempt to remediate the problem, for example by inoculation, such as discussed in conjunction with FIG. **17**. If no anomaly is detected (**1203**) then monitoring is complete (**1205**) in this example. In some embodiments, monitoring for anomalous traffic may be continuous. In some embodiments, monitoring for anomalous traffic may be periodic. For example, periodic monitoring may consist of monitoring some number of computers, such as 1 computer, for some period of time, such as 2 minutes, and then not monitoring those computers again for another period of time, such as 1 hour. As another example, periodic monitoring may be randomized, for example by changing the duration of monitoring, pattern of selecting computers to monitor, and/or the unmonitored duration, to be difficult or impossible to predict, for example by being driven by a pseudo random number generator, subject to restrictions of available monitoring computers.

FIG. **13** is a flow diagram of a method for monitoring one or more computers for infestation according to some embodiments. In this example monitoring of a computer for infestation begins (**1301**), for example by retrieving a list of one or more addresses of computers, such as addresses of participating subscribers. In some embodiments, this list may be a list of addresses for some or all subscribers for a given ISP. In some embodiments, the list may be a list of IP addresses issued by a DHCP server, such as on a LAN, or a list of IP addresses observed by a router, such as a router on a LAN, or a complete list of IP addresses in a given subnet. A computer associated with an address identified in a list is queried for a cleanliness assertion (**1302**), for example by contacting a trusted computing base within a computer, and requesting an authenticated infestation scan by trusted software. An example of a trusted computing base within a computer is the Paladium security initiative under development by Microsoft and supported by Intel and American Micro Devices. Another example of a trusted computing base is described in various TCG specifications, such as the TCG Architecture Overview, published by the Trusted Computing Group. Trusted code bases may for example execute antivirus scans of the remainder of the computer, including untrusted portions of the disk and/or operating system. In some embodiments, trusted code bases may digitally sign assertions about the cleanliness (e.g. infestation status) and/or state of their computers. In some embodiments, the query for cleanliness (**1302**) may be responded to by anti-contagion software, such as antivirus software, with assertions about the currency of a scan, such as the last time a scan was performed, or a version associated with a current anti-contagion software or definition file in use, wherein a sufficiently updated software and/or scan may act as a cleanliness assertion. In some embodiments, an operating system may respond with information associated with its patch level, wherein a sufficiently recent patch level may be interpreted as an assertion of cleanliness. If a computer asserts it is clean (**1303**) then monitoring is complete (**1305**) in this example. If a cleanliness assertion is not provided (**1303**) then an infestation or vulnerability is presumed (**1304**) in this example.

FIG. **14** is a flow diagram of a method for quarantining a computer according to some embodiments. In this example, a decision has been made to quarantine a computer (**1401**), such as an ISP deciding to quarantine a subscriber, or a firewall or router for a LAN deciding to quarantine a local computer. In some embodiments, a decision to quarantine may be made based on apparent infestations or apparent vulnerabilities, for example as illustrated in conjunction with FIG. **11**, **12**, or **13**. In some embodiments a decision to quarantine may be based on the accumulation of evidence above a threshold, such as a threshold of one such piece of evidence, or a threshold of two such pieces of evidence. In some embodiments, a decision to quarantine may be made based on configuration rules, for example all new computers must be quarantined until they have contacted their OS vendor and been updated, or by user specification, such as a user requesting a quarantine for a specific computer. In some embodiments, requests for a quarantine may be made on an individual computer via settings, such as control panel in Microsoft Windows, or through a user interface to a firewall or router, for example by specifying an IP address, name, or MAC address to quarantine, or a list of addresses that are not to be quarantined (where other addresses may be quarantined). During a quarantine period outbound traffic is detected (**1402**), for example by a device such as a router, or by a software network stack on the quarantined computer, or by an upstream device, such as a device at an ISP. Outbound traffic is tested in this example to see if it is associated with a remediation request (**1403**), such as contact with an appropriate anti-contagion software distributor, or contact with a security patch update provider, such as Microsoft Windows Update, or contact with a verified remediation site, such as an internal ISP site. If outbound traffic is deemed to be associated with a remediation request (**1403**), then the traffic is routed or forwarded to its destination (**1404**) in this example. If outbound traffic is not associated with a remediation request (**1403**), then, in this example, it is tested to see if it is an approved service request (**1405**), such as a request for a web page. If the request was for an approved service (**1405**) then, in this example, the request is re-routed to a special quarantine server (**1407**), such as an ISP web server that

provides information and links to assist in remediation. Rerouting of request traffic may be accomplished in numerous ways, for example by providing an alternate DNS service that consistently returns a fixed IP address of a quarantine server in response to all DNS requests. Another example of rerouting a web page request is to respond with a redirect message, which would direct a browser on a quarantined computer to contact a quarantine server that provides remediation information. If the request was not for an approved service (**1406**), then the outbound traffic is discarded or responded to with in indication that the desired traffic destination is unreachable (**1406**) in this example.

FIG. **15** is a flow diagram of a method for quarantining a computer according to some embodiments. In this example, a decision has been made to quarantine a computer (**1501**), for example as described in conjunction with **1401** of FIG. **14**. Inbound traffic originally destined for a quarantined computer is detected (**1502**), for example by a network device such as by a router. An example of detecting inbound traffic destined for a quarantined computer is to have a list of quarantined IP addresses, and filter or provide additional processing on traffic destined for an address on the list. Detected inbound traffic is tested in this example to see if it is remediation related (**1503**), such as a response to a remediation request. Direct responses to remediation requests may for example be identified using stateful packet inspection firewall techniques known to those skilled in the art. In various embodiments, at least certain types of traffic from approved sites such as anti-contagion vendors, security update providers, and/or internal ISP sites are deemed remediation related. If traffic is remediation related (**1503**) then it is forwarded to the quarantined computer (**1504**) in this example. If traffic is not remediation related (**1503**) then, in this example, it is discarded or responded to with a message indicating that the destination is unreachable (**1507**).

In some embodiments, a configuration based quarantine during an installation of an operating system, such as Microsoft Windows, may be automatically setup, for example by a default installation setting up restrictions in a firewall installed with the OS or by an installation setting defaults for use when network connectivity is enabled, that establishes a quarantine, for example restricting internet connections, for example allowing internet connections only to a security update site such as Microsoft Windows Update. In some embodiments, settings that may establish a quarantine, such as default settings in an operating system installation, may be altered programmatically to remove the quarantine in response to a remediation site, such as a security update service such as Microsoft Windows Update, asserting that there are no other security updates pending. In some embodiments, configuration settings that may establish a quarantine may be altered programmatically to enable a quarantine in response to the presence of one or more security updates, such as provided by an operating system update service such as Microsoft Windows Update, that are available for installation on a computer.

FIG. **16** is a flow diagram of a method for a quarantine server to respond to requests according to some embodiments. In this example, a quarantine server is started (**1601**), for example by starting a web server, or DNS server, or proxy server, or ftp server, or some combination server. The server waits for a request (**1602**), for example by listening on a port and waiting for a connection. In some embodiments, a device such as a router may forward outbound traffic from a quarantined computer to a quarantine server, for example as illustrated in conjunction with **1407** of FIG. **14**. If a received connection is a web server request (**1603**), such as an HTTP request, then the server responds with a quarantine notification page (**1604**) in this example. An example of a quarantine notification page is a web page that provides notification that the computer is quarantined, and/or provides links to remediation sites appropriate to the quarantine, such as a link to a site that provides anti-contagion software for removing a virus that the quarantined computer is believed to contain. The links on the quarantine notification page may be examples of HTTP addresses that are for use in remediation. Another example of a quarantine notification page is a page that redirects to a quarantine notification page. If the request is not a web server request (**1603**), then it is tested to see if it is a DNS inquiry (**1605**) in this example. A DNS inquiry may for example appear on port **53** using TCP/IP or UDP. If the request is not a DNS inquiry (**1605**) then it may be discarded or responded to as "unreachable" (**1606**) in this example. If the request is a DNS inquiry (**1605**) then, in this example, the inquiry is tested to see if it is a DNS request for a remediation host name (**1608**). A remediation host name may include a host name as appropriate to the reason for a quarantine, such as a host providing anti-contagion software, or a host providing security updates, or a host internal to (or partnered with) an ISP. If the DNS inquiry was for a remediation host name (**1608**) then the request is proxied to an external DNS service provider (**1609**) in this example. A proxy to a DNS service (**1609**) may return a stored or cached response, such as the actual IP address of the host name, or it may contact an external DNS service to relay back the proposed IP address. If the DNS inquiry was not for a remediation host name (**1608**) then an IP address for a quarantine server is provided (**1607**) in this example. In some embodiments, when a redirected IP address is provided, such as that of a quarantine server, the DNS response may include a short time to live, for example **0** seconds. A short time to live ensures that a further DNS query will be sent prior to subsequent communications being sent from the quarantined computer to the host that was the subject of the DNS query, aiding a return to normal routing of traffic once the computer is no longer under quarantine. In some embodiments, the server(s) in this example may continue to wait for a service request (**1602**) in parallel with the processing of the requests as described above, or may wait for a service request again after disposition of the previous request as described above.

FIG. **17** is a flow diagram of a method for scanning backdoors and inoculating according to some embodiments. In this example, preparations are made for scanning a computer for a backdoor inoculation path (**1701**), for example by identifying the computer to be scanned, and enumerating or retrieving a list of one or more backdoors. In one example, an IP address to be scanned may be selected by an ISP from a list of subscribers. In one example, a list of backdoors may include information associated with preloaded software approved by the computer owner for access and inoculation purposes, such as an ISP provided utility that is installed by the subscriber. In some embodiments, an ISP defined backdoor may have an authentication protocol to preclude control by unauthorized parties, such as restrictions on IP address(es) that may contact the backdoor, or shared secrets such as a password required by the sender, or cryptographic authentication such as client authentication on an SSL connection. In some embodiments, a user is requested and/or required to authorize in advance use of an authorized and/or any unauthorized backdoor for purposes of inoculation. As another example, a list of backdoors may include virus or worm installed backdoors, such as those installed by a SubSeven virus or Back Orifice. As another example, backdoors may include exploitable security vulnerabilities, such as a weak-

ness in a web server, or a weakness in an RPC service. A scan may optionally be conducted to detect open TCP/IP backdoor ports (**1702**) similarly to the scan discussed in conjunction with FIG. **11**. If none of the backdoors are open (**1703**), for example there is no response on any of the ports, then the scan is complete (**1706**) in this example. If one or more backdoor ports are open (**1703**) then they are tested for usability (**1704**) in this example. In some embodiments, a backdoor port may be assumed open (**1703**) and may be tested for usability (**1704**) without an independent test to see if the port is open. An example of a usability test is no test at all, with the conclusion that a specific port is usable any time it is open. Another example of a usability test is based on the receipt of confirmation of a functional installation of ISP provided backdoor software, such as a confirmation that may be provided by the installed software. Another example of a usability test involves the use of a less invasive test of vulnerabilities such a buffer overrun exploit that does not have a permanent impact but provides observable validation (such as a dropped connection). If a backdoor is usable (**1705**) then the computer is inoculated via a backdoor (**1707**) in this example. One example of such an inoculation is the remotely controlled download of one or more remedy files into one or more directories, such as an operating system directory, such as the "\windows\system" directory in Microsoft Windows 2000. Another example of an inoculation is the download of an executable file to the computer, followed by the execution of that file. Another example of an inoculation is the download of a data file, such as an anti-contagion signature file, that is interpreted or processed by another executable already on the computer. Another example of an inoculation is the deletion of one or more files from the computer. Another example of an inoculation is the editing of one or more registry entries, or of one or more files, such as deleting, appending or rewriting entries in a registry or file. Another example of inoculation includes a combination of one or more of above techniques, optionally in conjunction with one or more remotely instigated reboot operations. Another example of inoculation includes a combination of one or more of the above techniques, along with an optional dialog box requesting a user's permission to proceed with the inoculation. Another example of inoculation is the use of a dialog box in conjunction with a timer that causes the inoculation to proceed if no user response is given in a prescribed period of time, such as 1 hour.

FIG. **18** is a flow diagram of a method for categorizing messages according to some embodiments. In this example, a message is received (**1801**), such as an email message or instant message. The purported sender's identity is extracted (**1802**), for example by examining the "From:" header line in an email message, or by examining the envelope information conveying an email message, or by examining the sender's ID in an instant message. A check is performed to see if the sender has sent any prior messages (**1803**), for example by scanning an inbox of prior email, or by checking for the sender in a buddy list. If there were no prior messages from the sender (**1803**) then the message is processed (**1804**) in this example. Processing the message may for example include presenting the message to the user, and/or recording routing information for comparison and categorization of future messages. If there were prior messages from the sender (**1803**) then routing information used in prior messages is retrieved (**1805**) in this example. Routing information may for example be extracted from the header in email. Routing information may for example be obtained during an instant message session by retrieving the IP address of the party being communicated with, for example as is done when an instant message session coordinates a peer-to-peer activity such as a direct file transfer with a remote party. In some embodiments, routing information may be stored implicitly, such as in headers of a previously received message, or explicitly, for example by extracting routing information and storing it in a database or file system. Routing information for the current message is compared to prior messages (**1806**), for example by checking that the IP address for the party being communicated with is in a similar subnet, or allocated within the same geographical region, for example the same continent, as a previous IP address for the same party. IP address to geographical location may be estimated by various software applications and services, such as services provided by MaxMind and described on http://www.maxmind.com. In some embodiments, routing information for an email message may be deduced by examining header information related to mail transfer agents. In some embodiments, similarity in IP addresses for mail transfer agents may be interpreted as implying similar routing. If routing is similar to previous messages from the same sender (**1806**), then the message is processed as legitimate (**1808**) in this example. If the routing information for a message purporting to be from a sender is not similar to previous messages from the same sender (**1806**) then the message is processed as suspicious (**1807**) in this example. In some embodiments, processing as suspicious may include contributing to a scoring that labels the message as an undesirable message, such as spam, viral, or phishing. In some embodiments, higher probability suspicious messages may be scrutinized further, or may be restricted to some extent in their impact on the computer. An example of restricting impact of a message may include discarding the message. Another example of restricting impact or further scrutiny of a message may include placing a message in a spam inbox. Another example of a restricting impact of a message may include preventing execution of content contained in the message, and/or requiring additional user interactions to bypass the restriction, for example by having a user agree to ignore warnings which describe the suspicious circumstances. In various embodiments, the process of FIG. **18** is implemented on a mail, messaging, or other server associated with the sender and/or the sender's ISP; a mail, messaging, or other server associated with the recipient and/or the recipient's ISP; and/or one or more intermediate nodes.

FIG. **19**A is a flow diagram of a method for notifying possible sources of infection according to some embodiments. In this example, an infected message is received (**1901**). An example of an infected message is a message classified as such by anti-contagion software. Routing information is extracted from the message (**1902**). An example of extracting routing information in an email message is to extract header lines inserted by mail transfer agents that propagated the message. Another example of extracting routing information is to acquire an IP address of a party making contact via instant messaging, for example as is provided by AOL's instant message infrastructure for a peer-to-peer file transfer. Routing information from previously received messages is retrieved (**1903**). An example of retrieving email routing information from prior messages is scanning all received messages, such as messages in an email inbox, and extracting routing information. Another example of retrieving routing information is to query a database or file system where routing information from previous messages has been gathered. Routing information for the infected message is compared with the retrieved routing information (**1904**). If no similarity with previous messages is found (**1904**) then in this example a third party notification is made (**1906**). An example of a third party notification is sending notification to the ISP

that provides network access to the message source. Another example of notifying a third party is notifying a viral clearinghouse of the apparent source of a viral message. Another example of a notifying a third party is notifying a service that determines the credibility of a report, such as a collaborative filter service, that accumulates accusations of viral disseminations. In some embodiments, similarity of routing of an infected message with a previous message may include having a previous message from the IP address of the infected message source. If a similarity is found with previous messages (**1904**) then in this example a notification is sent (**1905**). In some embodiments, notification may be sent (**1905**) to one or more sources of the prior message(s) that best match the infected message, and/or to third parties. As an example, notification may be sent to an email source that had the same originating IP address as the infected message. In some embodiments, a user may be asked to approve notification(s), for example via a pop-up dialog. In some embodiments, notification may be automatically sent without user intervention.

FIG. **19**B is a block diagram illustrating a network environment in which suspicious messages are identified and/or one or more notifications are sent when a malicious message is detected. A message source **1920** sends a message via the Internet **1922** to a message processing system **1926** for delivery via the Internet to one or more message destinations **1924**. In various embodiments, the message processing system **1926** comprises a mail or messaging server associated with the message source **1920**, a mail or messaging server associated with the destination(s) **1924**, and/or an intermediate processing node. In some embodiments, the message source **1920** is connected directly and/or via a local network, and not via the Internet **1922** as shown in FIG. **19**B, to message processing system **1926**. In some embodiments, message processing system **1926** is configured to identify suspicious messages using current and historic routing information, e.g., as described above in connection with FIG. **18**. In some embodiments, the process of FIG. **19**A is implemented on message processing system **1926**. For example, in some embodiments if message processing system **1926** determines that a message sent by message source **1920** is infected, message processing system **1926** compares routing information associated with the malicious message with corresponding information for one or more messages received previously from the message source **1920**. If the routing information is similar, in some embodiments the message processing system **1926** notifies the message source **1920**. If the routing information is not similar, in various embodiments the message processing system **1926** notifies the message source **1920**, the sender's ISP **1928**, and/or a clearinghouse or similar applicable service **1930**, as described above.

Although the foregoing embodiments have been described in some detail for purposes of clarity of understanding, the invention is not limited to the details provided. There are many alternative ways of implementing the invention. The disclosed embodiments are illustrative and not restrictive.

What is claimed is:

1. A method for protecting a network, comprising:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition; and

permitting the first host to communicate with the remediation host.

2. A method as recited in claim **1**, wherein detecting an insecure condition further includes at least one of the following: scanning for a vulnerability, scanning for malicious data, determining whether a security software is installed, and detecting anomalous network traffic.

3. A method as recited in claim **1**, wherein detecting an insecure condition includes determining that the first host should be quarantined until an update to an operating system has been installed.

4. A method as recited in claim **1**, wherein detecting an insecure condition includes configuring an operating system to quarantine the first host upon initial startup after installation of the operating system.

5. A method as recited in claim **1**, wherein permitting the first host to communicate with the remediation host includes:

detecting an outbound communication from the first host; and

forwarding the outbound communication if it is addressed to the remediation host.

6. A method as recited in claim **1**, wherein preventing the first host from sending data to the one or more other hosts includes:

detecting an outbound communication from the first host; and

redirecting the outbound communication to a quarantine server if it comprises a request for an approved service and is not addressed to a remediation host.

7. A method as recited in claim **1**, wherein quarantining the first host further includes preventing the first host from receiving via the protected network data not related to remediation of the insecure condition.

8. A method as recited in claim **1**, performed at an Internet service provider.

9. A method as recited in claim **1**, wherein the software component on the first host is an operating system.

10. A method as recited in claim **1**, wherein determining that the response does not include a valid digitally signed attestation of cleanliness includes determining that the software component on the first host is not sufficiently updated.

11. A method as recited in claim **1**, wherein determining that the software component on the first host is not sufficiently updated includes determining that a patch level associated with the software component on the first host is not sufficiently recent.

12. A system for protecting a network, comprising:

a processor configured to:

detect an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantine the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition; and

permit the first host to communicate with the remediation host; and

a memory coupled to the processor and configured to provide instructions to the processor.

13. A system as recited in claim **12**, wherein the processor is configured to detect an insecure condition at least in part by performing one or more of the following: scanning for a vulnerability, scanning for malicious data, determining whether a security software is installed, and detecting anomalous network traffic.

14. A system as recited in claim **12**, wherein the processor is configured to detect an insecure condition at least in part by determining that an initial startup after installation of an operating system is being performed.

15. A system as recited in claim **12**, wherein the processor is configured to quarantine the first host at least in part by preventing the first host from receiving via the protected network data not related to remediation of the insecure condition.

16. A system as recited in claim **12**, wherein the software component on the first host is an operating system.

17. A system as recited in claim **12**, wherein determining that the response does not include a valid digitally signed attestation of cleanliness includes determining that the software component on the first host is not sufficiently updated.

18. A system as recited in claim **17**, wherein determining that the software component on the first host is not sufficiently updated includes determining that a patch level associated with the software component on the first host is not sufficiently recent.

19. A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition; and

permitting the first host to communicate with the remediation host.

* * * * *

EXHIBIT 2

US008144717B2

(12) **United States Patent**
Blange et al.

(10) **Patent No.: US 8,144,717 B2**
(45) **Date of Patent: Mar. 27, 2012**

(54) **INITIALIZATION OF A WIRELESS COMMUNICATION NETWORK**

(75) Inventors: **Marinus Johannes Blange**, Nootdorp (NL); **Miodrag Djurica**, Rotterdam (NL)

(73) Assignee: **Nederlandse Organisatie Voor Toegepast-natuurwetenschappelijk Onderzoek TNO**, Delft (NL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 784 days.

(21) Appl. No.: **12/159,753**

(22) PCT Filed: **Jan. 2, 2007**

(86) PCT No.: **PCT/NL2007/000001**

§ 371 (c)(1), (2), (4) Date: **Dec. 2, 2008**

(87) PCT Pub. No.: **WO2007/078193**

PCT Pub. Date: **Jul. 12, 2007**

(65) **Prior Publication Data**

US 2009/0215482 A1 Aug. 27, 2009

(30) **Foreign Application Priority Data**

Dec. 30, 2005 (EP) 05078044

(51) **Int. Cl.**
H04L 12/28 (2006.01)

(52) **U.S. Cl.** **370/402**; 370/395.2; 370/400; 370/401; 455/41.2; 455/445

(58) **Field of Classification Search** 455/41.2, 455/435.1, 445; 370/338, 328, 329, 341, 370/395.2, 395.21, 400–402, 406
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,548,523 B2 * 6/2009 Babbar et al. 370/331

(Continued)

FOREIGN PATENT DOCUMENTS

EP 1 471 693 A2 10/2004

OTHER PUBLICATIONS

Forno et al., "Design and implementation of a Bluetooth ad hoc network for indoor positioning" IEE Proceedings: Software, IEE, vol. 152, No. 5, Oct. 7, 2005 pp. 223-228.

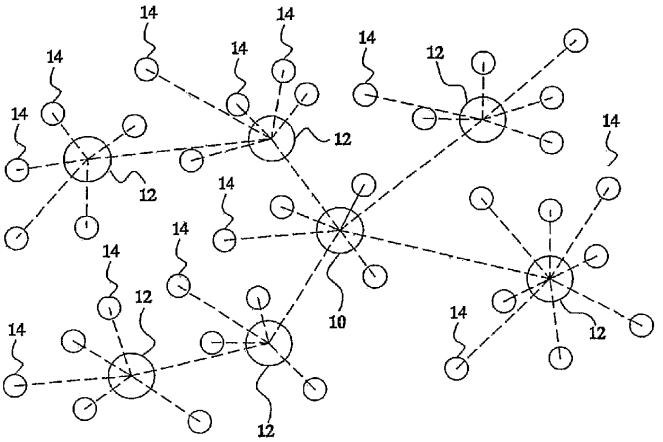
(Continued)

Primary Examiner — Dominic E Rego
(74) *Attorney, Agent, or Firm* — Leydig, Voit & Mayer Ltd.

(57) **ABSTRACT**

A wireless communication network comprises an association unit and a plurality of stations. At least part of the stations store association information indicative of an associated route for transmission of messages from the station to the association unit. The association information is generated by installing the stations in a not-associated state; transmitting association request messages from stations in the not-associated state; and transmitting association grant messages in response to the association request messages. Each association grant message is transmitted only from the association unit or, at least when the station that has transmitted association request message is outside a reception range of the association unit, from one of the stations that is in an associated state. Each station that has received the association grant message in response to its association request message is switched to the associated state. Association information is stored that indicates the station that has sent the association grant message as part of the associated route to the association unit.

20 Claims, 3 Drawing Sheets



U.S. PATENT DOCUMENTS

7,823,199	B1 *	10/2010	Rathi et al.	726/22
8,045,531	B2 *	10/2011	Cheng et al.	370/338
2001/0014868	A1 *	8/2001	Herz et al.	705/14
2003/0151513	A1	8/2003	Herrmann et al.	
2003/0174682	A1 *	9/2003	Barker	370/338
2004/0047307	A1 *	3/2004	Yoon et al.	370/324
2004/0223476	A1 *	11/2004	Jose et al.	370/338
2005/0088980	A1 *	4/2005	Olkkonen et al.	370/255
2005/0122927	A1	6/2005	Wentink	
2005/0195842	A1 *	9/2005	Dowling	370/401
2005/0221803	A1 *	10/2005	Hayashi	455/413
2006/0133266	A1 *	6/2006	Kim et al.	370/228
2006/0133328	A1 *	6/2006	Levendel et al.	370/338
2006/0166683	A1 *	7/2006	Sharma et al.	455/456.5
2006/0268756	A1 *	11/2006	Wang et al.	370/310
2007/0038346	A1 *	2/2007	Ehrlich et al.	701/33

OTHER PUBLICATIONS

Rubin et al., "TBONE: A Mobile-Backbone Protocol for Ad Hoc Wireless Networks" Aerospace Conference Proceedings, IEEE Mar 9-16, 2002. vol. 6, pp. 2727-2740.

International Search Report for PCT/NL/2007/0000001, dated Jun. 6, 2007.

* cited by examiner

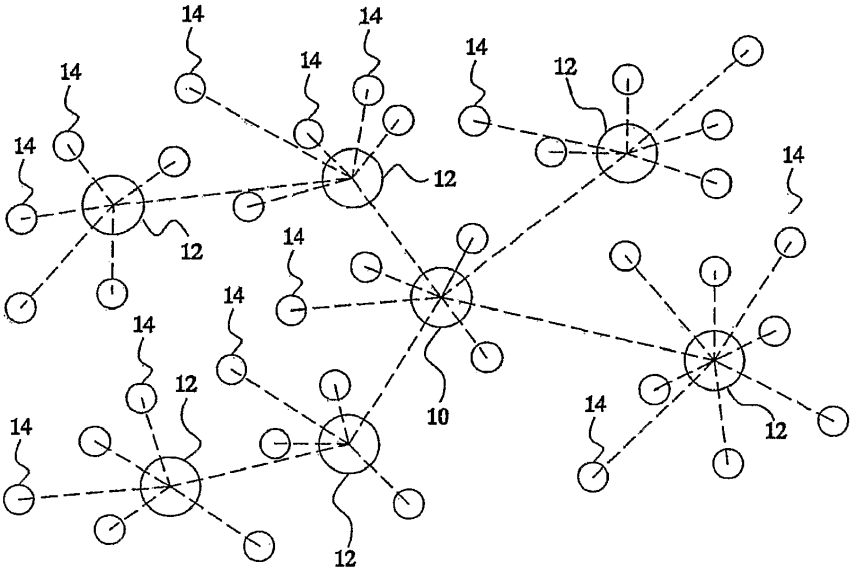


Fig.1

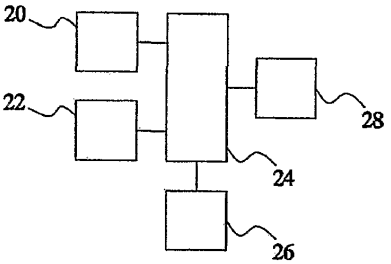


Fig.2

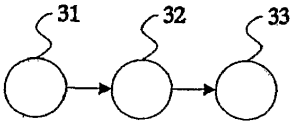


Fig.3

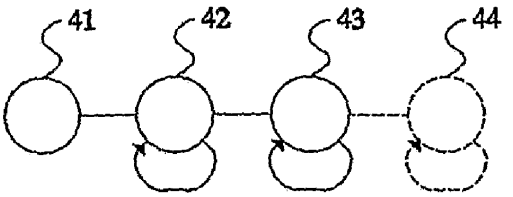


Fig4

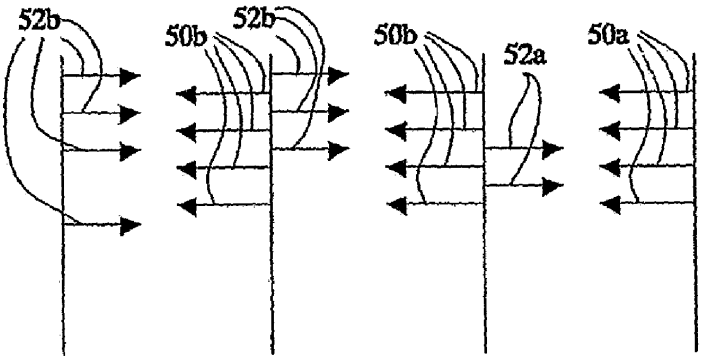


Fig.5a

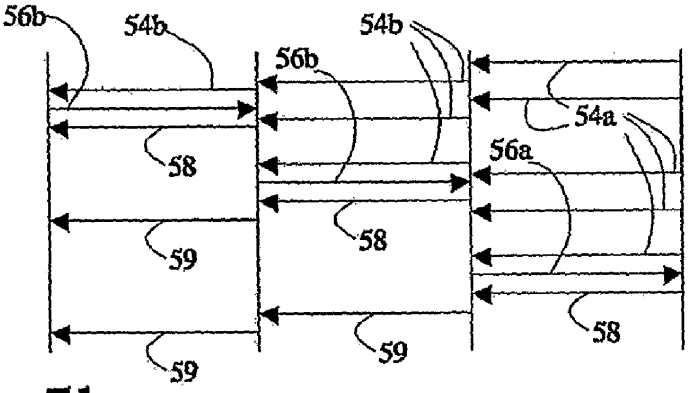


Fig.5b

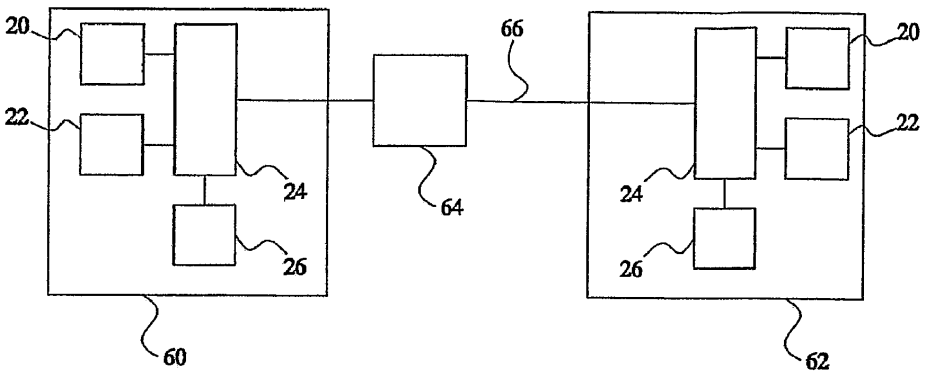


Fig.6

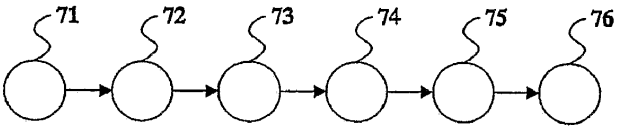


Fig.7

INITIALIZATION OF A WIRELESS COMMUNICATION NETWORK

FIELD OF THE INVENTION

The invention relates to a method of operating a wireless communication network, to a wireless communication system that uses such a method and to stations for use in such a network.

BACKGROUND

US patent application No. 2003/0151513 describes a wireless sensor station network. In operation data from the sensor stations is transmitted in multiple hops to a central panel and vice versa. This is because the sensor stations are not able to transmit directly to the central panel. Each sensor station is given its own, fixed route for transmitting and receiving data to and from the central panel. Typically, the data is first sent to another sensor station and from there to another sensor station and so on until a cluster head is reached. Then the data is sent through a network of cluster heads to the central panel.

After installation of the sensor stations at various locations in a sensing area a network topology is dynamically formed. That is, the routes for each of the sensor stations are automatically selected and programmed into the sensor stations. In US patent application No. 2003/0151513 describes a number of methods of dynamically forming the routes from the sensor stations to cluster head units that channel communication from groups of sensor stations. The initiative is taken by the cluster head unit, which transmits link discovery messages, which alert nearby sensor stations about the availability of the cluster head. These sensor stations forward the link discovery messages. Further sensor stations may receive and forward the forwarded link discovery messages and so on. The link discovery packets function to synchronize communication clocks of the cluster head and the sensor stations.

Next the routes for transmitting messages from the sensor stations to the association unit are defined. In one embodiment this is done by transmitting and forwarding route discovery packets from the association unit. The forwarded route discovery packets indicate the number of hops over which they have been forwarded. In this embodiment each sensor stations picks a route by selecting the association unit or sensor station from which it received a route discovery packet over least hops. The sensor stations record an indication of these routes for later use in normal operation. Subsequently, the sensor stations send route registration packets to the association unit to indicate the route that they have selected.

In another embodiment the sensor stations send link registration messages without waiting for route discovery messages. These link registration messages may reach the association unit via each available route. In this embodiment the association unit selects routes for the sensor stations from which it has received link registration messages. After making the selection the association unit sends route definition messages to the sensor stations for recording an indication of these routes for later use in normal operation.

A problem with these techniques is that they require transmission of information via sensor stations before the routing topology has been defined, i.e. when plural routes are still possible. This complicates the (sensor) stations, because they have to support different modes of communication while operating in low-power mode. If forwarding of information is only done using the definite routing topology, a central approach is slow (each station has to wait until its upward station has centrally been assigned a route) and a decentral

approach is inefficient (a routing topology with an unnecessary amount of branches has been created)

Another problem that may occur is that the wireless links between nodes are hindered by barriers and that using the same frequency for upward and downward link routes requires synchronization. This complicates the (sensor) station, because they have to support synchronization and powerful RF signals.

A mobile communications protocol is known from an article titled "TBONE: A Mobile-Backbone Protocol for Ad Hoc Wireless Networks" by Izhak Rubin et al. and published in the IEEE Aerospace Conference proceedings 2002 (Piscataway N.J., Mar. 9-16, 2002) page 2727-2740. The mobile protocol of this article uses two types of network: a high power network termed the BNET, or backbone, and a plurality of low power networks called ANETs. In each low power network (ANET) low power nodes communicate under control of a central node of that ANET. Each of the central nodes also functions as backbone node, i.e. as a node in the high power backbone network.

Devices are provided that are capable of operating as backbone nodes. Initially, each of these devices decides whether it will switch to operation as backbone node, dependent on whether it has a higher qualifications to do so than neighboring devices that are capable of operating as backbone nodes. Subsequently, those devices that have switched to operation as backbone nodes receive join requests from remaining nodes that want to become part of a low power network (ANET) centred on a particular backbone node. The low power networks (ANETs) are formed by granting these join requests. Join requests can come from both remaining backbone capable devices that have not decided to operate as backbone devices, and from low power nodes that are not capable of operating as backbone devices at all.

The article also describes the possibility that remaining backbone capable devices (that have not initially decided to operate as backbone devices) can be commanded to become backbone nodes by existing backbone nodes. This happens when a remaining backbone capable device that is part of a low power network (ANET) receives and relays a join request from a requesting node that cannot be accepted by any existing backbone node. When this happens the central backbone node of the low power network (ANET) commands the remaining backbone capable device to switch to operation as a backbone node and to grant the join request of the requesting node. In this case the remaining backbone capable device joins the high power network (BNET), sets up a low power network (ANET) of its own and ceases to use the paths in the low power network (ANET) of the commanding backbone node. The remaining backbone capable device uses the high power network to relay further messages to or from the requesting node. No provision is made for routing through successive previously established low power network (ANET) nodes and association for this form of routing.

This system addresses the use of two types of network links: links in the low power networks and links in the high power networks. Direct communication links (not via other associated nodes) are used in the low power networks. The selection of communication paths in the high power network is not described, but as high power communication is used longer range paths can be used in this network than in the low power networks. As the system is intended to provide communication in a dynamic environment rather than control of devices in the system there is no single central backbone

node. The use of two types of network without central unit makes this system complex and inefficient in bandwidth use.

SUMMARY OF THE INVENTION

Among others, it is desirable to provide for a wireless network wherein routes for communication can be determined without the need to forward information via stations for which the routes have not yet been defined.

Among others, it is desirable to provide for a method of operating a wireless network wherein routes for communication can be determined without the need to forward information via stations for which routes have not yet been defined.

Among others it is desirable to provide for stations for flexible use in wireless networks.

The invention provides for a wireless communication network according to claim **1**. This network allows routes between stations and an association station to be set up for further use, using routes that have already been defined in response to earlier associations. Each route is established when an association request from a station is granted. Establishment of a route is effected by the association unit for example by storing information that defines the route to the associated station in a memory that the association unit uses to select the route each time when a message has to be sent to the associated station. Routes are established upon an association grant from the association unit or from a station that has switched to the associated state directly or indirectly based on an association grant from the association unit, in a wave of association that radiates from nodes to nodes starting from the association unit. Only the association unit and stations that have an established route to the association unit, which has been established in this way in response to association requests and corresponding association grants by nodes along the route, are allowed to grant association requests. Therefore their routes can immediately be used for the station that is associated.

In an embodiment a station is configured to transmit association requests first for association via an active frequency channel and subsequently, if the first transmitted association request is not granted, for association via an as yet unused frequency channel. The latter request may be transmitted for example directly to the association unit. Because the route that is to be used to the station is established only upon association it is prevented that inconsistent multiple routes are assigned when such different association requests are transmitted.

More generally, in an embodiment the station that issues the association grant to a specific station is configured to make issue of the association grant dependent on whether the specific station indicates that a function that is active in that specific station or not. This means that a specific station may associate with different other stations, dependent on whether the function is active in the specific station or not. As a result the specific station has the opportunity to issue a plurality of association requests: without activating the function and after the function is activated, for example with a request directly to the association unit. Thus, for example, if the function consumes a scarce resource (such as function that uses a frequency channel for communication between the stations, thereby excluding use of the frequency channel by other stations), it can be prevented that the resource is consumed needlessly by granting the association request before activating the function. Because the routes to stations are established only upon association it is prevented that inconsistent station addresses are assigned.

In an embodiment association is preceded by a discovery stage, wherein no routes are yet established but station discover other stations, that can function to relay messages. Thus, stations may be enabled first to discover addresses of available other stations to which association requests can be sent, or stations that may send association requests before a route is established. In a further embodiment discovery advertisement messages are used that identify stations that are enabled to grant discovery requests. Thus, it is made possible to avoid waste of energy or bandwidth on association requests that cannot be granted.

In another embodiment part of the stations are dual function stations, which start to send association grant messages only after receiving approval from the association unit to do so. Thus, the association unit has control over the selection of stations that act as relay station, even after the dual function station has obtained a route for its "own" messages (e.g. sensor messages from a built in sensor station). In an embodiment the dual function station comprises units that function as a beacon frame transmitter and as a station with sensor station-like communication abilities respectively. The association unit may follow a strategy of enabling such dual function stations selectively to avoid enabling many stations that act as relay for only a few stations each, or to avoid selecting stations when these will become a relay for messages for too many stations.

In another embodiment dual function stations are used that comprises a mechanically flexible connection between separate units that each have an antenna or antennas for transmission and reception. Thus such stations can be used to connect different spaces "around the corner".

In another embodiment dual function stations are used that operate concurrently in different frequency channels that are periodically active in time slots that repeat with different asynchronous periods. This simplifies coordination between different network parts.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other objects and advantageous aspects will be described for exemplary examples, using the following figures.

FIG. **1** shows a wireless network
FIG. **2** shows a station
FIG. **3** shows a state diagram of a sensor station
FIG. **4** shows a state diagram of a beacon frame transmitter
FIG. **5***a* shows messages during a discovery phase
FIG. **5***b* shows messages during an association phase
FIG. **6** shows an embodiment of a beacon frame transmitter
FIG. **7** shows a state diagram of a beacon frame transmitter

DETAILED DESCRIPTION OF THE DRAWINGS

FIG. **1** shows a wireless network. The network comprises an association unit **10**, a plurality of beacon frame transmitters **12** and a plurality of sensor stations **14** (only some numbered explicitly). Fixed communication routes are defined between association unit **10** and sensor stations **14**, using wireless communication links (shown by dashed lines) between association unit **10**, beacon frame transmitters **12** and sensor stations **14**. The communication routes form a tree structure with association unit **10** at its root and beacon frame transmitters **12** at its nodes. Each beacon frame transmitter **12** is coupled in a star topology to a number of the sensor stations **14**.

FIG. **2** shows a station. At the level of detail shown, the figure applies to beacon frame transmitters **12** as well as

sensor stations **14**. The station contains a receiver circuit **20**, a transmitter circuit **22**, a processor circuit **24**, a memory **26** and, at least in case of sensor stations, a sensing circuit **28**. Processor circuit **24** is coupled to receiver circuit **20**, transmitter circuit **22**, memory **26** and optionally to sensing circuit **28**. Each of receiver circuit **20** and transmitter circuit **22** include an antenna (not shown). Preferably, the antennas of receiver circuit **20** and transmitter circuit **22** are formed by a shared antenna of receiver circuit **20** and transmitter circuit **22**.

Preferably the stations are battery operated, with on board batteries (not shown) or supplied by other on board power supplies such as solar cells. This means that it is desirable to reduce power consumption as much as possible.

In operation, data is exchanged between sensor stations **14** and association unit **10** via the links that are shown in FIG. **1**. Because the links form a tree structure, a single route to and from association unit **10** is defined for each sensor station **12**.

It should be appreciated that the communication routes are a result of selection from a greater number of possible routes. In principle, each sensor station **14** could communicate directly with association unit **10** and each beacon frame transmitter **12**. In practice, a number of these direct communication routes are not possible, because of insufficient signal strength. But direct wireless communication from a sensor station **14** to still more than one beacon frame transmitters **12** and/or to association unit **10** may be possible. Similarly, each beacon frame transmitter **12** may be able to communicate directly with association unit **10** and/or a number of beacon frame transmitters **12**.

The wireless connections shown in FIG. **1** are the result of a selection from these possible routes. A number of operable wireless communication links between sensor stations **14**, beacon frame transmitters **12** and association unit **10** is not used. These links are not shown in FIG. **1**. Sensor stations **14** transmit messages to association unit **10** via fixed routes and conversely association unit **10** transmits messages to sensor stations via the fixed routes. The routes are defined by information in memory **26**.

Each sensor station **14** stores at least information that is selective for the beacon frame transmitter **12** at the start of its route to association unit **10**; this address information is used to form messages that the sensor station **14** sends to association unit **10**. Typically, each beacon frame transmitter **12** stores at least information that is selective for the next beacon frame transmitter **12** in the route or for association unit **10**. This information is used to forward messages from the sensor station **14** sends to association unit **10**. Alternatively, each sensor station **14** may store information that is selective for the entire route to association unit **10** and the sensor station **14** may use this information to form messages, from which beacon frame transmitters **12** can determine how to forward the message.

Conversely, association unit **10** preferably stores information for each sensor station **14** that is selective for the beacon frame transmitter **12** at the start of the route to that sensor station **14**. Association unit **10** uses this information to form messages that association unit **10** transmits to the sensor station **14**. Typically, association unit **10** stores information that identifies the entire route to each sensor station **14**. The association unit **10** may use this information to form messages, from which beacon frame transmitters **12** can determine how to forward the message to the sensor station **14**. Alternatively, each beacon frame transmitter **12** stores, for each sensor station **14** for which it can receive messages, at least information that is selective for the next beacon frame transmitter **12** in the route to that sensor station **14**. This information is used to forward messages for the sensor station **14** from association unit **10** to the sensor station **14**.

An embodiment of the part of operation wherein the routes are established will now be described. Basically this embodiment includes two stages:

- a discovery stage wherein beacon frame transmitters **12** and sensor stations **14** become "aware" of stations (association unit **10** or beacon frame transmitters **12** in the case of sensor stations and association unit **10**, other beacon frame transmitters **12** and sensor stations **14** in the case of beacon frame transmitters **14**) that are within reception range and
- an association stage wherein sensor stations **14** and beacon frame transmitters **14** (or sensor stations that are fixedly coupled to these beacon frame transmitters) send requests to associate to association unit **10** or beacon frame transmitters **12**, i.e. to establish a route through the association unit **10** or beacon frame transmitter **12** for subsequent use.

The association stage for a sensor station **14** (or beacon frame transmitter **12**) is completed when an association request is granted by the association unit **10** or a beacon frame transmitter **12**. The route via the granting station is thereby established for subsequent use. In one embodiment each beacon frame transmitter **12** is enabled to grant association requests immediately after it has received a grant of an association request itself, but preferably the beacon frame transmitter **12** is enabled to grant association requests only after receiving secondary grant message, this time from the association unit **14** that allows the beacon frame transmitter **12** to grant association requests. Association unit **10** may for example make issuance of the secondary grant message dependent on information about the number of as yet unassociated stations that the beacon frame transmitter **12** has discovered. Thus, association unit **10** can control network topology.

Preferably, the association requests are addressed specifically to beacon frame transmitters **12** or the association unit **10**, using addresses (or more generally any identifier of these stations) that have been discovered in the discovery stage. Thus, it is excluded that conflicting association grants can be received in response to an association requests. However, addressing may be made unnecessary, for example when stations address all subsequent messages to one station that has sent a grant, or if some conflict resolution scheme is used.

In the embodiment the discovery stage comprises use of discovery messages from sensor stations **14** and beacon frame transmitters **12** (or sensor stations that are fixedly coupled to these beacon frame transmitters **12**) and responses in reaction to these discovery messages. Alternatively or in addition, stations may be discovered from unsolicited messages. The discovery stage and the association stage may overlap, the stations continuing to collect identifiers of discovered stations between transmission of association requests during the association stage. Alternatively non-overlapping consecutive stages may be used. In another embodiment the discovery stage may even be eliminated, for example if association requests are used that are not specifically addressed, or if association requests are used to discover stations.

When the discovery stage and the association stage may overlap, it is preferred that beacon frame transmitters **12** include information in their messages to indicate whether they have been enabled to grant association requests (preferably in general, i.e. not limited to requests from specific sensor stations **14**, but in a further embodiment specific sensor stations **14** or beacon frame transmitters **12** from which association requests may be accepted may be indicated).

Thus, sensor stations **14** and/or beacon frame transmitters **12** can avoid transmission of association requests to discovered beacon frame transmitters **12** that are not enabled to grant association requests.

FIG. **3** shows a state diagram of operation of a sensor station **14** before a route is established. FIGS. **5***a,b* shows messages exchanged with sensor station **14** during discovery and association respectively. Initially the sensor station **14** is in a first state **31**. In this first state **31** processor circuit **24** causes transmitter **22** to transmit discovery messages **50***a*, for example once every minute, every two minutes or every hour. When processor circuit **24** detects that receiver **20** has received a beacon frame **52***a* from a beacon frame transmitter **12** or association unit **10**, processor circuit **24** switches sensor station **14** to a second state **32**. In this way, the beacon frame **52***a* acts as a discovery confirm message.

Preferably, sensor station **14** switches to second state **32** in response indiscriminately to both beacon frames **52***a* that are transmitted in response to the discovery message or to any other beacon frame that is received spontaneously. In an embodiment sensor station **14** switches to second state **32** in response only to beacon frames **52***a* wherein it is indicated that the transmitting station is prepared to grant association requests and not to optional other beacon frames **52***a* that do not contain such an indication.

In second state **32** processor circuit **24** causes transmitter **22** to transmit association request messages **54***a* for example once every minute, every two minutes or every hour. Preferably each association request message **54***a* includes an identification of the sensor station **14** that allows messages specifically directed at the sensor station **14** to be sent in response. Also preferably, processor **24** reads an identification of the beacon frame transmitter **12** or association unit **10** from a previously received discovery confirm message and includes that identification in the association request message. This may be done for example when it is desirable to avoid collisions between association grant messages in response to the association request message.

When processor circuit **24** detects that receiver **20** has received an association grant message **56***a* from a beacon frame transmitter **12** or association unit **10** directed at the sensor station **14** of the processor circuit **24** in response to an association request message **54***a*, processor circuit **24** switches sensor station **14** to a third state **33** and records information for reaching this beacon frame transmitter **12** or the association unit **10** in memory **26** to define route to association unit **10** for use during normal operation. Optionally sensor station sends an acknowledgment **58** of the association grant message back to the beacon frame transmitter **12** or association unit **10**.

In third state **33** sensor station **14** stops transmitting association request messages **54***a*. In one embodiment sensor station **14** stops transmitting discovery messages **50***a* already when it switches to second state **32**. Alternatively sensor station **14** may be arranged to continue transmitting discovery messages **50***a* in second state **32**, only to stop after switching to third state **33** or after a predetermined time interval has elapsed after switching to second state **32**, if this occurs earlier than the switch to third state **33**.

In an embodiment a station is configured to transmit association requests first for association via an active frequency channel and subsequently, if the first transmitted association request is not granted, for association via an as yet unused frequency channel. The latter request may be transmitted for example directly to the association unit. Because the route that is to be used to the station is established only upon association it is prevented that inconsistent multiple routes are assigned when such different association requests are transmitted.

More generally, in an embodiment the station that issues the association grant to a specific station is configured to make issue of the association grant dependent on whether the specific station indicates that a function that is active in that specific station or not. This means that a specific station may associate with different other stations, dependent on whether the function is active in the specific station or not. As a result the specific station has the opportunity to issue a plurality of association requests: without activating the function and after the function is activated, for example with a request directly to the association unit. Thus, for example, if the function consumes a scarce resource (such as function that uses a frequency channel for communication between the stations, thereby excluding use of the frequency channel by other stations), it can be prevented that the resource is consumed needlessly by granting the association request before activating the function. Because the routes to stations are established only upon association it is prevented that inconsistent station addresses are assigned.

FIG. **4** shows a state diagram of operation of a beacon frame transmitter **12** before a route is established. Initially the beacon frame transmitter **12** is in a first state **41**. In this first state **41** processor circuit **24** causes transmitter **22** to transmit discovery messages **50***b*, for example once every minute, every two minutes or every hour. When processor circuit **24** detects that receiver **20** has received a beacon frame **52***b* from another beacon frame transmitter **12** or association unit **10**, processor circuit **24** switches beacon frame transmitter **12** to a second state **42**. In this way, the beacon frame **52***b* acts as a discovery confirm message. In an embodiment discovery confirm messages may also be transmitted unsolicited, i.e. not in response to a discovery message, in this sense discovery messages could also be called "hello" messages. The term "discovery advertisement message" is used herein both for messages in response to discovery requests and unsolicited (hello) messages. Examples of use of discovery advertisement messages will be given in terms of discovery confirm messages, but unless indicated otherwise these examples apply generally for discovery advertisement messages. In a further embodiment stations need not transmit discovery messages **50***a,b* at all. In this case the stations listen only to unsolicited messages. The discovery confirm message does not need to be addressed to any station in particular: any station may react to any discovery confirm message.

Preferably, beacon frame transmitter **12** switches to second state **42** in response indiscriminately to both a beacon frame **52***b* that is transmitted in response to the discovery message or to any other beacon frame that is received spontaneously. In an embodiment sensor station **14** switches to second state **42** in response only to beacon frames **52***b* wherein it is indicated that the transmitting station is prepared to grant association requests and not to optional other beacon frames **52***b* that do not contain such an indication.

In second state **42** processor circuit **24** causes transmitter **22** to transmit association request messages **54***b*, for example once every minute, every two minutes or every hour. Preferably each association request message **54***b* includes an identification of the beacon frame transmitter **12** that allows messages specifically directed at the beacon frame transmitter to be sent in response. Also preferably, processor **24** reads an identification of the beacon frame transmitter **12** or association unit **10** from a previously received discovery confirm message and includes that identification in the association request message. This may be done for example when it is

desirable to avoid collisions between association grant messages in response to the association request message.

When processor circuit **24** detects that receiver **20** has received an association grant message **56***b* from another beacon frame transmitter **12** or association unit **10** in response to an association request message **54***b*, processor circuit **24** switches beacon frame transmitter **12** to a third state **43** and records information for reaching this other beacon frame transmitter **12** or the association unit **10** in memory **26** to define a route to association unit **10** for use during normal operation. Optionally frame transmitter **12** sends an acknowledgement **59** of the association grant message back to the beacon frame transmitter **12** or association unit **10**.

In third state **43** frame transmitter **12** stops transmitting association request messages **54***b*. In one embodiment frame transmitter **12** stops transmitting discovery messages **50***b* when it switches to second state **42**. Alternatively frame transmitter **12** may stop after switching to third state **43**, or after a predetermined time interval has elapsed after switching to second state **42**, if this occurs earlier than the switch to third state **42**.

Preferably beacon frame transmitter **12** transmits discovery confirm messages in at least when in second and third state **42**, **43**. More preferably beacon frame transmitter **12** includes an indication in the discovery confirm messages indicating whether the beacon frame transmitter **12** is ready to respond to association grant messages with association requests. This allows stations to avoid using bandwidth for transmission of useless association requests. In principle beacon frame transmitter **12** may also be arranged to transmit discovery confirm messages in the first state **41**. Preferably, when beacon frame transmitter **12** receives a discovery request message from a sensor station **14** or another beacon frame transmitter **12** beacon frame transmitter **12** records an identification of the sensor station **14** or other beacon frame transmitter **12** from which the discovery request was received. Optionally beacon frame transmitter **12** also records identifications of stations that have transmitted association requests that the beacon frame transmitter **12** did not respond to, if any.

In first state **41** and second state **42** beacon frame transmitter **12** does not respond to association request messages **54***a*. However, in third state **43** beacon frame transmitter **12** does respond to association request messages **54***a*. In this state processor circuit **24** causes transmitter **22** to transmit an association grant message **56***a* in response to each received association request message **54***a*, directed at the station that has sent association request message **54***a*, using the station identification in the association request message **54***a*. Processor circuit **24** includes information for reaching the beacon frame transmitter **12** in the association grant message **56***a* and optionally also further information for reaching association unit **10**.

Furthermore, beacon frame transmitter **12** transmits an update message **59** with information that the sensor station **14** has been associated to association unit **10** after sending the association grant message **56***a* (and after receiving an optional acknowledgment **58** of that message).

This response of beacon frame transmitter **12** to association request messages **54***a* is not limited to such messages from sensor stations **14**, but applies also to association request messages **54***b* from other beacon frame transmitters **12**.

It should be noted that beacon frame transmitter **12** and sensor station **14** become stations of the network with well defined routes only when reaching their respective third states **33**, **43**. Furthermore it should be noted that beacon frame transmitters **12** respond to association request messages **54***a,b* only once they have entered their third state **43**, that is, only when the beacon frame transmitter **12** has a well defined route to association unit **10**. Thus, the information about association can be transmitted along the route that will be used during normal operation. This route may also be used as part of the route to the newly associated sensor station **14**. Association unit **10** may use information from its memory, about the route to the beacon frame transmitter **12** to define the route to the sensor station **14**. In another embodiment, the beacon frame transmitters **12** along the route may add their identity to inform association unit **10** of the route, or in yet another embodiment the beacon frame transmitter **12** that associates to the sensor station **14** may insert this information.

It may be noted that not all beacon frame transmitters **12** need to become part of the network as beacon frame transmitters. If a beacon frame transmitter **12** receives no association grant signal **56***b* or grants no association grant requests from other beacon frame transmitters **12** or sensor stations **14** the beacon frame transmitter **12** does not become a functional beacon frame transmitter in the network.

In an embodiment association unit **10** decides on whether a beacon frame transmitter **12** becomes a functional beacon frame transmitter. In this embodiment functional beacon frame transmitters **12** in third state **43** forward association requests **54***a* from other beacon frame transmitters **12** (and optionally also from sensor stations **14**) to association unit **10** before sending an association grant message **56***b*. (Alternatively a further message derived from this association request is forwarded).

In this embodiment association unit **10** is arranged to grant or deny the forwarded association request in response to forwarding (or to the further message). When the association request is granted association unit sends back a grant message, which is used to generate the association grant signal **56***b* to the original association request **54***b*. Thus association unit **10** (and/or a beacon frame transmitter **12**) is enabled to control generation of the network topology, for example by first granting association to selected preferred beacon frame transmitters and granting association requests to others later only if this is still necessary to complete connections to all sensor stations **14**.

In a further embodiment beacon frame transmitters **12** are arranged to supply information about the number and/or identity of the sensor stations **14** and or beacon frame transmitters **12** from which it has received beacon frame request messages and/or association request messages. The identity may be provided by including identity information in beacon frame request messages and/or association request messages. The beacon frame transmitters **12** may supply this information for example in their own beacon frame request messages and/or association request messages. In this further embodiment the association unit **10** (and/or a beacon frame transmitter) is provided with information which it may use to select beacon frame transmitters **12** for granting association. This may be used for example to distribute links over several beacon frame transmitters **12**, or on the contrary to avoid distribution by concentrating links on one beacon frame transmitter **12**.

In a further embodiment beacon frame transmitters **12** return grants of in response to association requests only after receiving a grant of a secondary association request, distinguished, at least in function, if not in format, from the previously described association request. For this purpose optionally an additional fourth state **44** may be added to the states of FIG. **4**. When in third state **43** (after receiving the previously described grant to one of the previously described association requests, which will be termed primary association requests beacon frame transmitter **12** transmits a secondary associa-

tion request to association unit **10**. Upon receiving a grant of this secondary association request beacon frame transmitter **12** switches to the fourth state **44**. In this embodiment the beacon frame transmitter **12** is enabled to begin to return association grant messages to requesting stations only in fourth state **44**. Optionally the beacon frame transmitter indicates this in discovery confirm messages.

Preferably beacon frame transmitter **12** is configured to include a list of stations from which beacon frame transmitter **12** has received discovery request messages (and/or not-granted association request) in the secondary association request. As an alternative beacon frame transmitter **12** may optionally be configured to transmit one or more accompanying messages to provide such a list to association unit **10**. This enables association unit **10** to select to which of the beacon frame transmitters **12** it will grant association requests of the second type. In this way association unit **10** can control the network topology, after admitting beacon frame transmitters **12** to the network by granting a primary association request.

In a preferred embodiment each functional beacon frame transmitter **12** (and association unit **10**, which acts as a beacon frame transmitter in this respect) operates in a time slot scheme during normal operation, periodically transmitting a beacon signal to which the sensor stations **14** that are associated with that beacon frame transmitter may synchronize. After the beacon signal the beacon frame transmitter **12** and its associated stations exchange messages for a limited period.

This allows the beacon frame transmitters **12** and the sensor stations **14** to switch between a low power sleep mode and a normal operating mode. After exchanging messages the beacon frame transmitters **12** and the sensor stations **14** switch to the sleep mode wherein power supply and/or clock signals to at least part of the beacon frame transmitter **12** and the sensor station **14** is switched off. After counting off a predetermined time-interval the beacon frame transmitters **12** and the sensor stations **14** switch back to the normal operating mode. With a predetermined delay after switching to the normal operating mode the beacon frame transmitter **12** transmits the beacon frame and the sensor stations **14** receive this beacon frame.

In this embodiment the association unit **10** and/or the beacon frame transmitters preferably select the timing of the first of the periodic beacon frames so that the time slots of different beacon frame transmitters **12** that are within range to receive each others messages do not overlap. In this embodiment the information in the sensor stations **14** after association defines the time slot in which the sensor station **14** should switch to the normal operating mode.

In one embodiment each beacon frame transmitter **12** also switches to a forwarding mode in the time slot defined by the further beacon frame transmitter **12** through which the original beacon frame transmitter **12** transmits messages to the association unit **10** (or in the time slot of the association unit **10**). When in the forwarding mode the original beacon frame transmitter **12** exchanges messages with the further beacon frame transmitter **12**.

This embodiment has the disadvantage that overlap of the time slots of neighboring beacon frame transmitters **12** must be avoided. This means that beacon frame transmitters **12** must monitor time slots of other beacon frame transmitter **12**. In turn this means that the beacon frame transmitters **12** cannot operate in the sleep mode as much as desirable.

In an alternative embodiment the beacon frame transmitter **12** are designed to transmit and receive in a selectable one of a plurality of frequency channels. In this embodiment the frequency channels for different functional beacon frame transmitters **12** are selected during initialization, for example by the beacon frame transmitters themselves after detecting that no messages are received in the frequency channel during a predetermined time period, or by the association unit **10** so that beacon frame transmitters **12** that have been detected to receive each other get different frequency channels to avoid collisions. In an embodiment the beacon frame transmitters **12** and their associated sensor stations **14** switch to the selected frequency channel when they start normal operation. In this embodiment the information in the sensor stations **14** after association defines the frequency channel for communicating with the beacon frame transmitter **12**. Use of different frequency channels has the advantage that no extensive synchronization between time slots of beacon frame transmitters **12** is needed. Therefore beacon frame transmitters **12** can switch to the sleep mode more often.

In this further embodiment one frequency channel may be used during initialization. Alternatively a plurality of the frequency channels may be used in parallel. As an alternative the beacon frame transmitters **12** themselves may select a frequency channel during initialization after detecting that no messages are received in the frequency channel during a predetermined time period. Subsequently each beacon frame transmitter **12** performs all of its operations towards sensor stations **14** using the selected frequency channel, and uses another frequency channel to make upstream connections toward association unit **10**.

FIG. **6** shows a further embodiment of a beacon frame transmitter. The beacon frame transmitter contains two unit **60**, **62** as shown in FIG. **2**, plus a message exchange memory **64** coupled between the processor circuits **24** of the two units. In one embodiment mutually different units **60**, **62** are used, a first unit **60** arranged to perform the functions of a beacon frame transmitter (having a higher power transmitter **22** and/or a differently programmed processor circuit for example than the other unit) and a second unit **62** of a type used in sensor stations **14**. A mechanically flexible connection **66** is provided between the units **60**, **62** (e.g. between message exchange memory **64**, which may be part of the first unit **60** and second unit **62**). Flexible connection **66** is for example at least 5 centimetre long.

In this embodiment a first unit **60** acts as beacon frame transmitter, except for the transmission of discovery requests **50***b*, association requests **54***b* and any communication towards association unit (e.g. update message **59**). The second unit **62** performs the remaining functions of the beacon frame transmitter. It may be noted that this second unit performs much like a sensor station **14**, except that it does not generate or receive its own messages. Instead it forwards messages to or from the other unit. Messages to and from the association unit **10** or other beacon frame transmitters **12** (or data from such messages) are placed in message exchange memory **64**.

In normal operation units **60**, **62** switch between the sleep mode and normal active mode in time slots defined by respective beacon frames. The unit that acts as beacon frame transmitter transmits its own beacon signal, preferably in the select frequency channel for the beacon frame transmitter. The unit that acts as sensor station switches to the normal operation mode in the time slot defined by the beacon frames of the further beacon frame transmitter to which the beacon frame transmitter **60**, **62**, **64** is associated. Thus, no power expensive synchronization is needed.

FIG. **7** shows a state diagram of operation of the paired units of FIG. **6** during discovery and association. The paired units **60**, **62** start in a first state **71** wherein a second unit **62** of

the pair **60**, **62** that acts like a sensor station transmits discovery messages. When the second unit **62** detects that a beacon frame from a beacon frame transmitter **12** or association unit **10** has been received, the paired units switch to second state **72**. In second state **72** the first unit **60** of the pair that acts as a beacon frame transmitter starts transmitting beacon frames in response to received discovery messages by transmitting a discovery confirm message. In these beacon frames the first unit **60** includes information to indicate that the first unit **60** is not yet associated.

Also in second state **72** the second unit **62** starts monitoring whether it receives beacon frames with information that indicates that the station that transmits the beacon frame has been associated (or is association unit).

When the second unit **62** detects a beacon frame with information that indicates that the unit that transmits the beacon frame has been associated, the paired units **60**, **62** switch from second state **72** to a third state **73**. Herein the second unit **62** starts transmitting association requests. When the second unit **62** receives an association confirmation in response, the paired units **60**, **62** switch to a fourth state **74**. In second, third and fourth state **72**, **73** and **74** first unit **60** starts and restarts counting a time out interval. The time out interval is restarted each time when first unit **60** receives a discovery message from a new source (sensor station or other beacon frame transmitter). The time out interval expires after a predetermined duration (e.g. a minute or more) wherein first unit **60** has received no discovery message from any new source. After expiration of the time out interval unit **60** stops transmitting beacon frames in response to received discovery messages.

From fourth state **74**, after expiry of the time out interval the paired units **60**, **62** switch to a fifth state **75** wherein second unit **60** transmits a beacon frame transmitter association request for the first unit **62**, listing the sources from which the first unit has received a discovery message. This beacon frame transmitter association request is forwarded to association unit **10**, which grants or denies the request, dependent on the list (granting the request for example if more than a threshold number of sources are listed that have not yet been associated). When the beacon frame transmitter association request is denied, new beacon frame transmitter association requests may be sent. Association unit **10** may be configured to lower the threshold with time so that the same request may be granted later if a source that is mentioned in the request cannot be reached in any other way even after an extended period of time.

Association unit **10** sends a grant message with a grant signal to the first unit **60** via the second unit **62**. When the grant message is received the paired units **60**, **62** switch to a sixth state **76**. Wherein first unit **62** starts transmitting beacon frames indicating that it has been associated, after which it can cause other units to switch from the first state **72** to the third state **73**.

Although completely separate units **60**, **62** are shown, it should be appreciated that part or all of these units may use shared circuits. Use of separate circuits has the advantage that a standard sensor station unit may be used. The use of a flexible connection has the advantage that the different units may be located "around a corner" from each other, so that one unit has contact with a space to which the other has no contact. Thus, a system in a building can be realized very easily with this type of beacon frame transmitter.

Although the network has been described with beacon frame transmitters **12**, it should be appreciated that instead of beacon signal synchronized transmission other forms of message exchange may be used. In this case beacon frame transmitters **12** act simply as stations. In that case beacon frame requests are only discovery requests and beacon frame transmissions responses to discovery requests.

Furthermore, although an example has been discussed wherein the beacon frame transmitters **12** merely act to relay sensor data, it should be appreciated that the beacon frame transmitters **12** themselves may also be equipped with sensors, so that the beacon frame transmitters **12** can also transmit their own sensor data.

Although a network of sensor stations has been described (each provided with a temperature sensor, and/or a smoke sensor and/or a light intensity sensor and/or a moisture sensor and/or a gas sensor etc.) has been described it should be appreciated that the described techniques can also be applied to non-sensor or part-sensor networks wherein sensor stations **14** are simply information handling stations.

Although an association process with a discovery phase (FIG. **5***a*) and an association phase has been described, it should be appreciated that the two phases can be integrated partly or wholly. In one embodiment beacon frame requests **50***a* double as association requests **54***a*. In another embodiment association grant messages **56***a* double as beacon transmissions **52***a*. Both embodiments may be combined. However, the use of separate phases has the advantage that the whole process runs more quickly.

The invention claimed is:

1. A wireless communication network that comprises an association unit and a plurality of stations, each station being configured to start up in a not-associated state and to transmit association request messages in the not-associated state, each station being configured to switch to an associated state upon reception of an association grant in response to one of its association request messages, the association request message and/or the association grant in response thereto establishing an operating route associated with the station for exchanging operating messages with the association unit during subsequent operation, the operating route associated with the station running through a source of the association grant in response to said one of the association request messages and, if the source of the association grant is not the association unit, the operating route between the source of the association grant and the association unit previously established for the source of the association grant in response to association request messages and/or the association grants for intermediate stations, in which network

the association unit is configured to transmit association grants as responses to received ones of the association request messages;

at least part of the stations is configured to become active to transmit association grants as responses to received ones of the association request messages, but only after switching to the associated state upon an association grant from the association unit or one of the at least part of the stations switched to the associated state directly or indirectly based on an association grant from the association unit.

2. A wireless communication network according to claim **1**, wherein said part of the stations comprises at least one dual function station, said dual function station being configured to transmit association grants only after receiving a secondary association grant, the dual function station being configured to transmit one or more secondary association request messages to the association unit along its operating route when in the associated state, only the association unit being configured to originate the secondary association grant in response to the secondary association request message.

3. A wireless communication network according to claim **2**, wherein the dual function station is configured to transmit information to the association unit indicating from which of the stations said dual function station has received transmissions, the association unit being configured to select whether to grant the secondary association request dependent on said information.

4. A wireless communication network according to claim **3**, wherein the stations are configured to transmit discovery messages before switching to the associated state, said transmissions comprising at least the discovery messages.

5. A wireless communication network according to claim **2**, wherein the dual function station comprises a first and second unit, each with at least a respective antenna, coupled via a mechanically flexible connection, the first unit being configured to transmit first ones of the association request messages and to receive first ones of the association grants, the second unit being configured to receive second ones of the association request messages and to the transmit second ones of the association grants in response to the first ones of the association request messages.

6. A wireless communication network according to claim **2**, wherein the dual function station is configured to operate using respective, mutually distinct first and second frequency channels, using the first frequency channel for transmitting association request messages and receiving association grants, and using the second frequency channel for receiving association request messages and transmitting association grants.

7. A wireless communication network according to claim **1**, wherein each station from said at least part of the stations is configured to transmit discovery advertisement messages including an identification of the station from said at least part of the stations, the stations that transmit association request messages being configured to address the association request messages using identifications from the discovery advertisement messages.

8. A wireless communication network according to claim **7**, wherein each station in said at least part of the stations is configured to indicate, in the discovery advertisement messages, whether the station is enabled to transmit association grants, the stations that transmit association requests being configured to transmit association requests addressed to stations from said at least part of the stations only if the addressed station has indicated in at least one of the discovery advertisement messages that it is enabled to transmit association grants.

9. A wireless communication network according to claim **1**, wherein said at least part of the stations are configured to transmit route information toward the association unit after transmitting an association grant, the route information defining the associated route for the station to which the association grant has been sent.

10. A wireless communication network according to claim **9**, wherein said at least part of the stations is configured to forward at least part of the route information towards the association unit, to store at least a further part of the route information and to route subsequent messages from the association unit to the station for which the route information was sent according to said stored at least further part.

11. A station for use in a wireless communication network, the station comprising a transmitter arrangement, a receiver arrangement and a processor circuit coupled to the transmitter arrangement and the receiver arrangement, the processor circuit being configured to start up in a not-associated state and to cause the transmitter arrangement to transmit association request messages in the not-associated state, the processor circuit being configured to switch to an associated state upon reception at the receiver arrangement of an association grant in response to one of the association request messages, the processor circuit being configured to establish an associated route for communication with an association unit during subsequent communication of operating messages to an association unit, the associated route including a source of the association grant, and, if the source of the association grant is not the association unit, the operating route between the source of the association grant and the association unit previously established for the source of the association grant in response to association request messages and/or the association grants for intermediate stations, the processor circuit being configured to cause the transmitter arrangement to initiate and transmit association grants in response to association request messages, but only after switching to the associated state.

12. A station according to claim **11**, wherein the processor circuit is configured, when switched to the associated state, to cause the transmitter arrangement to transmit a secondary association request message to the association unit, and to cause the transmitter arrangement to transmit the association grants only after receiving a secondary association grant in response to said secondary association request message.

13. A station according to claim **11**, wherein the processor circuit is configured to send information to the association unit indicating from which one of the stations said dual function station has received messages prior to receiving the secondary association grant.

14. A method of operating a wireless communication network that comprises an association unit and a plurality of stations, the method comprising

installing the stations in a not-associated state;

transmitting association request messages from stations in the not-associated state;

transmitting the association grants in response to the association request messages,

switching each station to an associated state when the station has received an association grant in response to one of its association request messages;

storing route information defining a route between the station that has switched to the associated state and the association unit, the route including the station that has sent the associated grant and, if the source of the association grant is not the association unit, the operating route between the source of the association grant and the association unit previously established for the source of the association grant in response to association request messages and/or the association grants for intermediate stations; wherein

each association grant is transmitted only from the association unit or from an intermediate station, the latter only after said intermediate station has switched to the associated state upon an association grant from the association unit or one of the at least part of the stations switched to the associated state directly or indirectly based on an association grant from the association unit.

15. A method according to claim **14**, wherein the intermediate station is a dual function station, the method comprising

transmitting a secondary association request message from the dual function station when the dual function is in the associated state;

transmitting a secondary association grant originating from the association unit in response to the secondary association request message;

enabling the dual function station to transmit the association grants only upon receiving the secondary association grant.

16. A method according to claim **15**, comprising

transmitting information from the intermediate station to the association unit about from which one of the stations the intermediated station has received transmissions;

selecting, in the association unit, whether to grant the secondary association request dependent on said information.

17. A method according to claim **16**, further comprising transmitting discovery messages from the stations before switching to the associated state, said transmissions comprising the discovery messages.

18. A method according to claim **15**, further comprising the steps of:

transmitting discovery advertisement messages from at least part of the stations, including an identification of the station that has transmitted the discovery advertisement message;

addressing the association request messages using identifications from the discovery advertisement messages.

19. A method according to claim **18**, further comprising the steps of:

including an indication in the discovery advertisement messages, whether the station that transmits the discovery advertisement message is enabled to transmit association grants,

transmitting association requests addressed to stations only if the addressed station has indicated that it is enabled to transmit association grants.

20. A method according to claim **15**, further comprising switching the intermediate station to operate at a first and second, mutually different transmission frequency after switching to the associated state, for directly or indirectly communicating with the association unit and for communicating with its associated stations respectively.

* * * * *

EXHIBIT 3

US008438120B2

(12) **United States Patent**
Raaijmakers

(10) **Patent No.: US 8,438,120 B2**
(45) **Date of Patent: May 7, 2013**

(54) **MACHINE LEARNING HYPERPARAMETER ESTIMATION**

(75) Inventor: **Stephan Alexander Raaijmakers**, Amsterdam (NL)

(73) Assignee: **Nederlandse Organisatie voor toegepast-natuurwetenschappelijk Onderzoek TNO**, Delft (NL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 607 days.

(21) Appl. No.: **12/597,257**

(22) PCT Filed: **Apr. 25, 2008**

(86) PCT No.: **PCT/NL2008/050247**
§ 371 (c)(1),
(2), (4) Date: **Jul. 8, 2010**

(87) PCT Pub. No.: **WO2008/133509**
PCT Pub. Date: **Nov. 6, 2008**

(65) **Prior Publication Data**

US 2010/0280979 A1 Nov. 4, 2010

(30) **Foreign Application Priority Data**

Apr. 25, 2007 (EP) 07106963
Jul. 9, 2007 (EP) 07112037

(51) **Int. Cl.**
G06N 5/00 (2006.01)

(52) **U.S. Cl.**
USPC **706/12**; 706/45

(58) **Field of Classification Search** 706/12, 706/45
See application file for complete search history.

(56) **References Cited**

PUBLICATIONS

Duckstein, et al., Bayesian forecasting of hydrologie variables under changing climatology, the Influence of Climate Change and Climatic Variability on the Hydrologie Regime and Water Resources (Proceedings of the Vancouver Symposium, Aug. 1987). IAHSPubl. No. 168, 1987, pp. 301-311.*
Raaijmakers, "Sentiment classification with interpolated information diffusion kernels," Proceedings of the 1st International Workshop on Data Mining and Audience Intelligence for Advertising, pp. 34-39, (Aug. 12, 2007).
AMIDA Project: "Chapter 3: subjectivity recognition" Deliverable 5.2: Multimodal Content Abstraction, 2007, pp. 28-47.
De Boer et al., "A Tutorial on the Cross-Entropy Method," Annals of Operating Research, vol. 134, No. 1, (Feb. 2005), pp. 19-67.
Mannor et al., "The cross entropy method for classification," Proceedings of the 22nd International Conference on Machine Learning (Aug. 9, 2005) (the whole document).
International Search Report dated Oct. 9, 2008 for PCT/NL2008/050247.

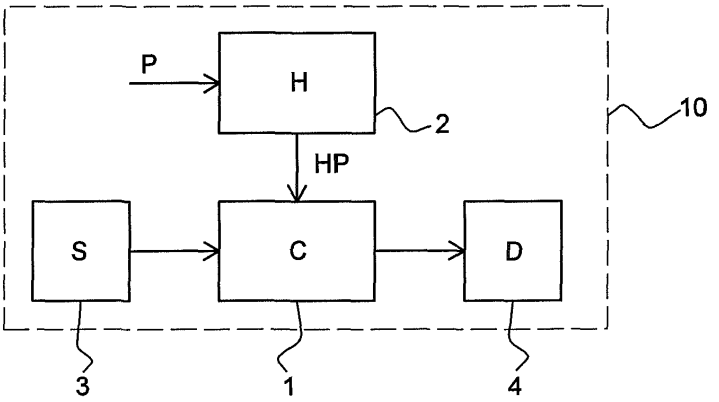
* cited by examiner

Primary Examiner — Wilbert L Starks
(74) *Attorney, Agent, or Firm* — Leydig, Voit & Mayer, Ltd.

(57) **ABSTRACT**

A method of determining hyperparameters (HP) of a classifier (**1**) in a machine learning system (**10**) iteratively produces an estimate of a target hyperparameter vector. The method comprises the steps of selecting from the random sample the hyperparameter vector producing the best result in the present and any previous iterations, and updating the estimate of the target hyperparameter vector by using said selected hyperparameter vector. The random sample may be restricted by using the hyperparameter vector producing the best result in the present and any previous iterations.

16 Claims, 1 Drawing Sheet



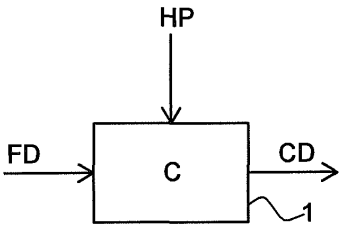


FIG. 1

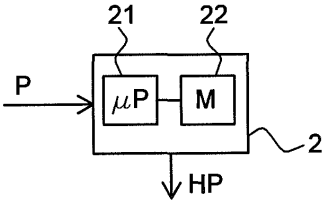


FIG. 2

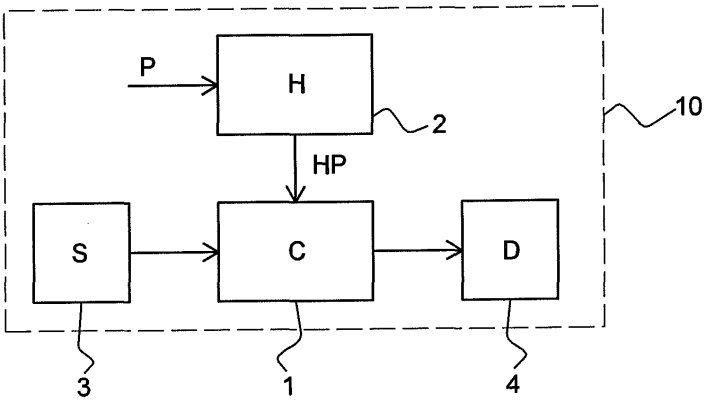


FIG. 3

MACHINE LEARNING HYPERPARAMETER ESTIMATION

FIELD OF THE INVENTION

The present invention relates to hyperparameter estimation. More in particular, the present invention relates to a method and device for determining hyperparameters of classifiers in machine learning systems and applications.

BACKGROUND

Classifiers are used in machine learning systems to classify physical objects and/or their (typically digital) representations. Machine learning systems may, for example, be used to assist fruit picking robots. The classifier of the machine learning system is trained to distinguish ripe fruit (e.g. tomatoes) from unripe fruit. For each fruit item, the classifier determines a set of parameters which are compared with stored parameters in order to classify the items as "ripe" or "unripe". The classification process is, in turn, controlled by hyperparameters which determine, for example, decision criteria such as threshold values.

Classifiers are typically trained using a training set prepared on the basis of human input: an operator indicates the correct classification for the items of the training set. On the basis of these training data, the correct hyperparameters of the classifier can be estimated.

Hyperparameters may be determined using various methods, for example heuristic or statistical optimisation methods. However, many heuristic methods are ad hoc methods, while not all statistical optimisation methods are suitable for determining hyperparameters.

A particularly efficient yet relatively simple method for estimating parameters in general is the cross-entropy (CE) method. This iterative method comprises the repeated steps of drawing, in a parameterised way, a random sample of candidate solutions, and updating the parameters on the basis of the random sample. At the time of writing, the paper "A Tutorial on the Cross-Entropy Method" by P. T. de Boer et al. could be found at http://iew3.technion.ac.il/CE/tutor.php, said paper is herewith incorporated in this document in its entirety.

The paper "The cross-entropy method for classification" by S. Mannor, D. Peleg and R. Rubinstein, Proceedings of the 22nd International Conference on Machine Learning, 2005, discloses an application of the CE algorithm for searching the space of support vectors. Hyperparameter values are determined by using a simple grid search, not by using the CE algorithm, for the reasons discussed below.

The cross-entropy method as described in the above-mentioned papers is unfortunately not suitable for determining hyperparameters. Determining optimal sets of hyperparameter values is a difficult problem due to the extremely large size of the search space: the optimal solution typically is a rare event. While the cross-entropy method is geared towards sampling rare event spaces, it is not a priori clear how the process of drawing hyperparameter values can be parameterised, as the hyperparameter samples are not classical parameterised probability density functions.

SUMMARY OF THE INVENTION

It is an object of the present invention to overcome these and other problems of the Prior Art and to provide a method and device for determining hyperparameters which allow efficient methods similar to the cross-entropy method to be utilized.

Accordingly, the present invention provides a method of determining hyperparameters of a classifier in a machine learning system by iteratively producing an estimate of a target hyperparameter vector, each iteration comprising the steps of:

drawing a random sample of hyperparameter vectors from a set of possible hyperparameter vectors, and

updating the estimate of the target hyperparameter vector by using the random sample,

the method being characterised in that each iteration comprises the step of:

selecting from the random sample a hyperparameter vector producing the best result in the present and any previous iterations,

and in that the step of updating the estimate of the target hyperparameter vector involves said selected hyperparameter vector.

By selecting from the random sample a or the hyperparameter vector producing the best result in the present and any previous iterations, and using said selected hyperparameter vector to update the estimate of the target hyperparameter vector, the properties of the best performing hyperparameters are used to guide the next iteration. In this way, a method similar to the cross-entropy method can be used effectively for determining hyperparameters, even when the hyperparameters are not continuous and their effects are not transparent. By using the method according to the present invention, an improved classification accuracy is achieved.

In the method of the present invention, hyperparameter vectors obtained through randomly sampling a set of hyperparameter vectors are used to estimate a desired (that is, target) hyperparameter vector to be used in the classifier. Each hyperparameter vector preferably comprises two or more elements, that is hyperparameters, but it is also possible for a hyperparameter vector to contain only a single hyperparameter. Although the samples are preferably completely random, pseudo-random samples may be used instead.

In a preferred embodiment, the method according to the present invention further comprises the steps of:

selecting from the random sample a further hyperparameter vector producing the best result in any previous iterations, and

restricting the random sample of hyperparameter vectors by using the further selected hyperparameter vector.

The step of restricting the random sample preferably involves the use of an interval surrounding the further selected hyperparameter vector which produces the best result in any previous iteration.

It is preferred that the step of restricting the random sample is carried out prior to the step of determining the hyperparameter vector producing the best result in the present and any previous iterations. However, in alternative embodiments, the step of restricting the random sample is carried out during the step of updating the estimate of the target hyperparameter vector.

In a preferred embodiment, E^t is the hyperparameter vector X^t_i at iteration t producing the best result $S(X^t_i)$ in the present iteration t and all previous iterations (if any). The random sample X^t_i has elements X^t_{ij}, while the step of updating the

hyperparameters comprises the step of determining the (target) hyperparameter v^t_j, where

$$v^t_j = \frac{\sum_{i=1}^{n} I\{S(X^t_i) \geq \gamma^t\} W(X^t_i; E^t) X^t_{ij}}{\sum_{i=1}^{n} I\{S(X^t_i) \geq \gamma^t\} W(X^t_i; E^t)},$$

wherein γ^t is a threshold value, W is a weighting function and $I\{S(X_i^t) \geqq \gamma^t\}$ is an indicator function which is equal to 1 if $S(X_i^t)$ greater than or equal to the threshold value γ^t, and else is equal to 0. The weighting function W is preferably given by

$$W(X^t_i; E^t) = 1 - \frac{\sqrt{\sum_{j=1}^{m} (X^t_{ij} - E^t_j)^2}}{\sqrt{\sum_{j=1}^{m} (X^t_{ij})^2} \sqrt{\sum_{j=1}^{m} (E^t_j)^2}}.$$

Accordingly, the step of updating the hyperparameters may involve a weighting function based upon a distance function, preferably a Euclidean distance function. It is noted that when the Euclidean distance between E^t and X^t_{ij} is zero, the weighting function W is equal to one.

The method according to the present invention is typically carried out by computer apparatus, for example a general purpose computer system comprising a processor and an associated memory, one part of the memory storing a software program for instructing the processor to carry out the method steps of the present invention, and another part of the memory storing data, said data comprising the hyperparameter values referred to above.

The present invention also provides a computer program product for carrying out the method as defined above. A computer program product may comprise a set of computer executable instructions stored on a data carrier, such as a CD or a DVD. The set of computer executable instructions, which allows a programmable computer to carry out the method as defined above, may also be available for downloading from a remote server, for example via the Internet.

The present invention additionally provides a classifier for use in a machine learning system, the classifier being arranged for using hyperparameters as control parameters, wherein the hyperparameters have been determined by the method according to any of the preceding claims. The classifier is preferably embodied in software, but may also be embodied in hardware, or in a combination of hardware and software.

The present invention still further provides a device for determining hyperparameters of a classifier in a machine learning system, the device comprising a processor arranged for iteratively producing estimates of a target hyperparameter vector, each iteration comprising the steps of:

drawing a random sample of hyperparameter vectors from a set of possible hyperparameter vectors, and

updating the estimate of the target hyperparameter vector by using the random sample,

the device being characterised in that the processor is arranged such that each iteration comprises the step of:

selecting from the random sample a hyperparameter vector producing the best result in the present iteration and any previous iterations, and

in that the step of updating the estimate of the target hyperparameter vector involves said selected hyperparameter vector.

The processor may be arranged for carrying out the steps mentioned above by providing a suitable software program in the memory associated with the processor. Alternatively, or additionally, the processor may be designed specifically to carry out the steps mentioned above.

A machine learning system comprising a device as defined above and/or arranged for carrying out the method as defined above is also provided by the present invention. The machine learning system may comprise hardware and/or software components.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will further be explained below with reference to exemplary embodiments illustrated in the accompanying drawings, in which:

FIG. **1** schematically shows a classifier having hyperparameters as input control variables.

FIG. **2** schematically shows a device for determining hyperparameters according to the present invention.

FIG. **3** schematically shows a machine learning system according to the present invention.

DETAILED DESCRIPTION OF THE DRAWINGS

The classifier (C) **1** shown schematically in FIG. **1** receives feature data FD and hyperparameters HP, and produces classification data CD. The feature data FD may include image data produced by a CCD (Charge Coupled Device) array or other light-sensitive device. Additionally, the feature data FD may include sound data or data concerning database items, such as library data (book numbers and titles). The classifier may be constituted by an apparatus (hardware embodiment), but is preferably constituted by a software program running on a computer apparatus, for example a general purpose computer ("personal computer"), a dedicated computer, or other suitable apparatus.

The classifier **1** is capable of classifying the items represented by the input feature data FD. The feature data FD typically comprise parameters of the items, for example their size and/or colour in the case of fruit or machine parts. Under control of the hyperparameters HP the items are each assigned to a class, for example "ripe" in the case of fruit. At least two different classes are used, and typically an item is assigned to one class only.

The hyperparameters HP determine the decision function of the classifier that classifies items. For example, when this function is based upon thresholding, items producing an aggregate value (derived from the feature values) equal to or greater than the threshold are assigned to class A, while items producing a value smaller than the threshold are assigned to class B. It is noted that other classification techniques may be used as well.

Applications of the classifier **1** include picking fruit, sorting parts (e.g. machine parts), classifying books, selecting songs, recognising adult video material, detecting damage, etc.

The present invention allows a method similar to the cross-entropy (CE) method to be used for estimating and determining hyperparameters. To this end, the present invention proposes to include a memory facility into the algorithm by preserving samples which produce a good result (the result being defined by a suitable function, such as a loss function). Additionally, or alternatively, the sampling of candidate

hyperparameter settings is made dependent on the (hyper) parameter vector updated by the algorithm, and/or a suitably weighting function (also known as "change of measure" or "likelihood ratio" in the CE algorithm) guiding the search process should be provided.

The method of the present invention uses parameters N, ρ and μ which are typically predetermined. It is noted that these parameters may be thought of as hyperparameters of the method but are distinct from the hyperparameters to be determined by the method. The parameter N represents the number of random draws; the parameter N may for example be equal to 10, 100 or 1000.

In the iterations a parameter n is used which represents the size of each data vector, that is, the number of elements X_{ij} of each hyperparameter vector X_i of the set of possible hyperparameter vectors. The parameter n may for example be equal to 5, 10 or 100, although other numbers are also possible. The parameter ρ represents a fraction and may have a value ranging from 0.2 to 0.01, although other values may also be used. The width parameter μ, which will be explained in more detail below, indicates a preferred sampling interval.

The initial set of target hyperparameters, the hyperparameter vector u, can typically be chosen arbitrarily. Each estimated (target) hyperparameter vector v^t at point in time (or iteration number) t consists of hyperparameters v^t_j.

In the present invention, the "best" hyperparameter vector v of iterations 1 . . . t is kept for use in the next iteration and is denoted E^t. This "best" (or "elitist") hyperparameter vector is found using a function $S(X^t)$ which may be a loss function or any other suitable function producing a result value based upon X^t. The value of $S(X^t)$ may also be viewed as an importance measure. The value of X^t corresponding with the maximum value of $S(X^t)$ is denoted argmax $S(X^t)$. In addition, the values of $S(X^t)$ are ranked and are used to determine a threshold γ^t, involving ρ and N to determine the $((1-\rho)N)^{th}$ value of the ranked values of $S(X^t)$. It is noted that the number $((1-\rho)N)$ is not necessarily an integer and is therefore rounded up, expressed mathematically as $\lceil((1-\rho)N\rceil$.

Subsequently, the (target) hyperparameters are determined using an expression involving indicator functions $I\{S(X^t_i)\geqq\gamma^t\}$ and weighting functions $W(X_i; E^t)$. The indication functions are equal to 1 when $S(X^t_i)\geqq\gamma^t$ and equal to 0 when $S(X^t_i)<\gamma^t$. The weighting functions $W(X_i; E^t)$, also known as "change of measure" functions, have the data vector elements X_i (actually X^t_i) and E^t as input variables. This implies that the best data vector E^t of the present iteration is used to determine the (target) hyperparameters.

The weighting function $W(X^t_i; E^t)$ is preferably defined as:

$$W(X_i^t; E^t) = 1 - \frac{\sqrt{\sum_{j=1}^{m}(X_{ij}^t - E_j^t)^2}}{\sqrt{\sum_{j=1}^{m}(X_{ij}^t)^2}\sqrt{\sum_{j=1}^{m}(E_j^t)^2}} \tag{1}$$

This function produces a value of 1 when the Euclidean distance between a sample and the "best" sample is 0.

Accordingly, a preferred embodiment of the method according to the present invention comprises the following steps:

1. Choose initial parameters N, ρ and μ and generate an initial (target) hyperparameter vector u.
2. Let $v^0=E^0=u$ and set iteration number t=1.
3. Generate a random sample $X^t_1, \ldots, X^t_n$ of hyperparameters drawn from a (given) set of possible hyperparameters. The random sample preferably is restricted by v^{t-1}. Compute the performance $S(X^t_i)$ for every value of i and rank the results: $S_{(1)}\leqq \ldots \leqq S_{(n)}$. Compute $\gamma^t=S_{percentile}$ and compute E^t=argmax $v_{i=1 \ldots t}\gamma^t$.
4. For every (target) hyperparameter v^t_j let

$$v_j^t = \frac{\sum_{i=1}^{n} I\{S(X_i^t) \geq \gamma^t\} W(X_i^t; E^t) X_{ij}^t}{\sum_{i=1}^{n} I\{S(X_i^t) \geq \gamma^t\} W(X_i^t; E^t)} \tag{2}$$

5. Let t:=t+1, repeat from step 3 until the stopping condition is met.

The final set v of j (target) hyperparameters v_j is the desired optimal set.

By using the best sample E^t_j in hyperparameter estimation/update formula (2), the better results are reinforced, leading to a more efficient convergence of the method. In addition, discontinuity problems associated with hyperparameters are overcome by the method of the present invention.

The stopping criterion (or condition) is preferably a non-changing value of the parameter γ^t during a number of iterations, but could also be a threshold value. The value $S_{percentile}$ is determined using the $((1-\rho)N)^{th}$ value of S in the ranking $S_{(1)}\leqq \ldots \leqq S_{(n)}$, where rounding up may be used if this value is not an integer. The weighting function $W(X^t_i; E^t)$ used in formula (2) is preferably defined by formula (1) above.

The method of the present invention may be summarized as deriving a target hyperparameter vector from a random sample of hyperparameter vectors while taking the performance of the hyperparameters of the random sample into account.

The restriction which may be applied when generating a random sample at time t in step 3 may be defined by the interval:

$$[v^t_j*(1-\mu), v^t_j*(1+\mu)] \tag{3}$$

where μ is a width parameter and "*" denoted multiplication. If the sample X^t_i lies within this interval, its distance relative to the best sample E^t_j of the previous iteration is limited to the value of μ. It is noted that the use of this interval is preferred but not essential as the method of the present invention could be carried out without using this interval.

The device **2** for determining hyperparameters comprises a processor (μP) **21** and an associated memory (M) **22**. The memory stores both instructions for the processor and data. The device **2** receives parameters P (more in general: feature data), such as the parameters ρ and μ mentioned above, and outputs hyperparameters HP. The stored instructions allow the processor to carry out the method according to the present invention.

The machine learning system **10** shown merely by way of non-limiting example in FIG. **3** comprises a classifier (C) **1** and a device (H) **2** for determining hyperparameters. The machine learning system **10** may further comprise at least one sensor (S) **3** and a display screen (D) **4**. The sensor **3** serves to collect data while the display screen **4** serves to display classification result and other data.

The present invention is based upon the insight that methods similar to the cross-entropy method can be adapted for hyperparameter estimation by adding a best performance preservation feature. The present invention benefits from the additional insight that the best performance in a particular iteration may be preserved by restricting the sample using the best performance of the previous iterations.

It is noted that any terms used in this document should not be construed so as to limit the scope of the present invention. In particular, the words "comprise(s)" and "comprising" are not meant to exclude any elements not specifically stated. Single (circuit) elements may be substituted with multiple (circuit) elements or with their equivalents.

It will be understood by those skilled in the art that the present invention is not limited to the embodiments illustrated above and that many modifications and additions may be made without departing from the scope of the invention as defined in the appending claims.

The invention claimed is:

1. A method of determining hyperparameters of a classifier in a machine learning system by iteratively producing an estimate of a target hyperparameter vector, each iteration comprising the steps of:

drawing a random sample of hyperparameter vectors from a set of possible hyperparameter vectors,

updating the estimate of the target hyperparameter vector by using the random sample, and

selecting, from the random sample of hyperparameter vectors, a hyperparameter vector producing a best result in the present and any previous iterations, and wherein the step of updating the estimate of the target hyperparameter vector uses said hyperparameter vector producing the best result.

2. The method according to claim **1**, further comprising the steps of:

selecting a further hyperparameter vector producing the best result in any previous iterations, and

restricting the random sample of hyperparameter vectors by using the further selected hyperparameter vector.

3. The method according to claim **2**, wherein the step of restricting the random sample of hyperparameter vectors involves using an interval surrounding the further selected hyperparameter vector.

4. The method according to claim **3**, wherein the step of restricting the random sample of hyperparameter vectors is carried out prior to the step of selecting the hyperparameter vector producing the best result in the present and any previous iterations.

5. The method according to claim **3**, wherein the step of restricting the random sample of hyperparameter vectors is carried out during the step of updating the estimate of the target hyperparameter vector.

6. The method according to claim **2**, wherein the step of restricting the random sample of hyperparameter vectors is carried out prior to the step of selecting the hyperparameter vector producing the best result in the present and any previous iterations.

7. The method according to claim **2**, wherein the step of restricting the random sample of hyperparameter vectors is carried out during the step of updating the estimate of the target hyperparameter vector.

8. The method according to claim **1**, wherein the step of updating the estimate of the hyperparameter vector uses a weighting function.

9. The method according to claim **1**, wherein E^t is the selected hyperparameter vector X^t_i at iteration t producing the best result $S(X^t_i)$, the selected hyperparameter vector X^t_i having elements X^t_{ij}, wherein the step of updating the estimate of the target hyperparameter vector comprises the step of determining the hyperparameter v^t_j, where

$$v^t_j = \frac{\sum_{i=1}^{n} I\{S(X^t_i) \geq \gamma^t\} W(X^t_i; E^t) X^t_{ij}}{\sum_{i=1}^{n} I\{S(X^t_i) \geq \gamma^t\} W(X^t_i; E^t)},$$

wherein γ^t is a threshold value and W is a weighting function.

10. The method according to claim **9**, wherein the weighting function W is given by

$$W(X^t_i; E^t) = 1 - \frac{\sqrt{\sum_{j=1}^{m} (X^t_{ij} - E^T_j)^2}}{\sqrt{\sum_{j=1}^{m} (X^t_{ij})^2} \sqrt{\sum_{j=1}^{m} (E^t_j)^2}}.$$

11. The method according to claim **1**, wherein the steps are carried out by a programmed computer apparatus including a processor and a computer readable medium including computer executable instructions.

12. A classifier for use in a machine learning system, the classifier using hyperparameters as control parameters, wherein the hyperparameters are determined according to the set of steps recited in claim **1**.

13. A non-transitory computer readable medium product including computer-executable instructions for carrying out a method of determining hyperparameters of a classifier in a machine learning system by iteratively producing an estimate of a target hyperparameter vector, each iteration comprising the steps of:

drawing a random sample of hyperparameter vectors from a set of possible hyperparameter vectors,

updating the estimate of the target hyperparameter vector by using the random sample, and

selecting, from the random sample of hyperparameter vectors, a hyperparameter vector producing a best result in the present and any previous iterations, and wherein the step of updating the estimate of the target hyperparameter vector uses said hyperparameter vector producing the best result.

14. A device for determining hyperparameters of a classifier in a machine learning system, the device comprising a processor arranged for iteratively producing an estimate of a target hyperparameter vector, each iteration comprising the steps of:

drawing a random sample of hyperparameter vectors from a set of possible hyperparameter vectors,

updating the estimate of the hyperparameter vector by using the random sample, and

selecting, from the random sample of hyperparameter vectors, a hyperparameter vector producing a best result in the present and any previous iterations, and wherein the step of updating the estimate of the target hyperparameter vector uses said hyperparameter vector producing the best result.

15. The device according to claim **14**, wherein the processor is further arranged for:

selecting a further hyperparameter vector producing the best result in any previous iterations, and

restricting the random sample by using the hyperparameter vector producing the best result in any previous iteration.

16. A machine learning system, comprising the device for determining hyperparameters of a classifier defined in claim **14**.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 8,438,120 B2
APPLICATION NO. : 12/597257
DATED : May 7, 2013
INVENTOR(S) : Stephan Alexander Raaijmakers

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the Title Page:

The first or sole Notice should read --

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 864 days.

Signed and Sealed this
Eighth Day of September, 2015

Michelle K. Lee

Michelle K. Lee
Director of the United States Patent and Trademark Office

EXHIBIT 4

US009235259B2

(12) **United States Patent**
van Staalduinen et al.

(10) **Patent No.: US 9,235,259 B2**
(45) **Date of Patent: Jan. 12, 2016**

(54) **METHOD FOR DETECTING AUDIO TICKS IN A NOISY ENVIRONMENT**

(75) Inventors: **Mark van Staalduinen**, 's-Gravenzande (NL); **Victor Bastiaan Klos**, The Hague (NL); **Peter Jan Otto Doets**, Delft (NL)

(73) Assignee: **NEDERLANDSE ORGANISATIE VOOR TOEGEPAST-NATUURWETENS CHAPPELIJK ONDERZOEK TNO**, Delft (NL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 735 days.

(21) Appl. No.: **13/512,139**

(22) PCT Filed: **Nov. 26, 2010**

(86) PCT No.: **PCT/NL2010/050795**

§ 371 (c)(1), (2), (4) Date: **Jul. 3, 2012**

(87) PCT Pub. No.: **WO2011/065828**

PCT Pub. Date: **Jun. 3, 2011**

(65) **Prior Publication Data**

US 2012/0288103 A1 Nov. 15, 2012

(30) **Foreign Application Priority Data**

Nov. 27, 2009 (EP) 09177411

(51) **Int. Cl.**

H04R 29/00	(2006.01)
G06F 3/00	(2006.01)
G06F 1/16	(2006.01)
G06F 3/01	(2006.01)
G10L 25/48	(2013.01)

(52) **U.S. Cl.**

CPC ***G06F 3/002*** (2013.01); ***G06F 1/1626*** (2013.01); ***G06F 3/017*** (2013.01); ***G10L 25/48*** (2013.01); *G06F 2200/1636* (2013.01)

(58) **Field of Classification Search**

CPC combination set(s) only.
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2003/0132950 A1* 7/2003 Surucu et al. 345/700
2005/0204904 A1 9/2005 Lengeling et al.

(Continued)

FOREIGN PATENT DOCUMENTS

EP 1 978 508 10/2008
EP 2 018 032 1/2009

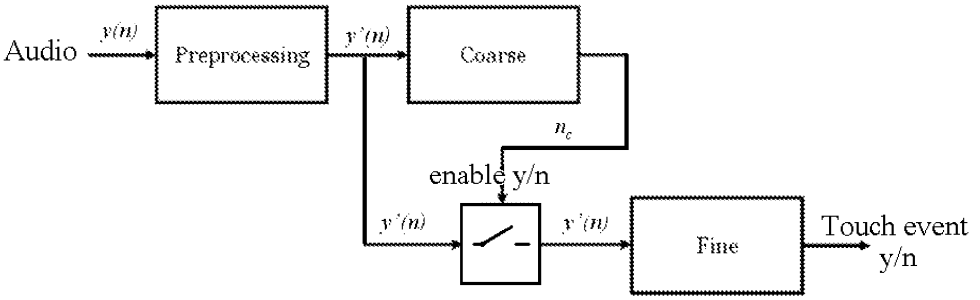
(Continued)

Primary Examiner — Duc Nguyen
Assistant Examiner — Assad Mohammed
(74) *Attorney, Agent, or Firm* — Preti Flaherty Beliveau & Pachios LLP

(57) **ABSTRACT**

A method for detecting, in an audio signal including a stream of audio samples (y(n), (y'(n)), ticks in a noisy environment, comprises the steps of applying the signal to a coarse tick detection processor arranged to decide whether it is likely that the signal includes a tick and, only then, enabling a fine tick detection processor to decide, by more thorough processing the audio signal, whether the audio signal represents an audio tick. The coarse processing step preferably includes buffering audio samples (y(n), (y'(n)), determining the maximum and minimum values of each sample, forming their local range, delaying the local range with one or more samples and comparing the difference between the actual local range and the delayed local range with a threshold value (T_c), and, if the difference exceeds the threshold value, outputting a trigger (n_c) for enabling the fine detection processor The fine processing step preferably includes buffering audio samples (y(n), (y'(n)), computing, for each buffer content (B_p(n,m)), a Fast Fourier Transformed buffer content (E_c(ω,m)), determining the difference (F_c(ω,m)) between the resulting frequency components, comparing the difference (F_c(ω,m)) to the content of a previously trained fingerprint set (F*(ω,m)) and calculating a correlation coefficient (p(F_c,F*)) between them, comparing the correlation coefficient to a threshold value (T_P) and, if the correlation coefficient exceeds the threshold value, outputting a signal indication the presence of a tick in the audio signal.

23 Claims, 3 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

2009/0153342 A1* 6/2009 Thorn G06F 1/1626 340/669
2009/0176505 A1* 7/2009 Van Deventer G06F 1/1626 455/456.1
2009/0215397 A1* 8/2009 Thorn et al. 455/41.2

FOREIGN PATENT DOCUMENTS

EP 2 073 515 6/2009
WO WO 2006/094739 9/2006

* cited by examiner

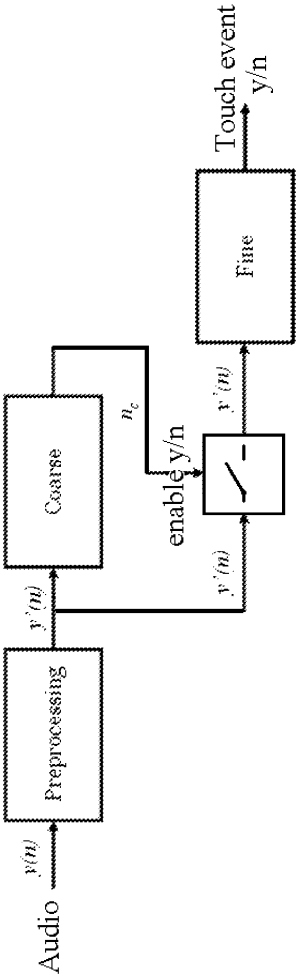

Audio
y(n)
Preprocessing
y'(n)
Coarse
n_c
enable y/n
y'(n)
y'(n)
Fine
Touch event
y/n

FIG. 1

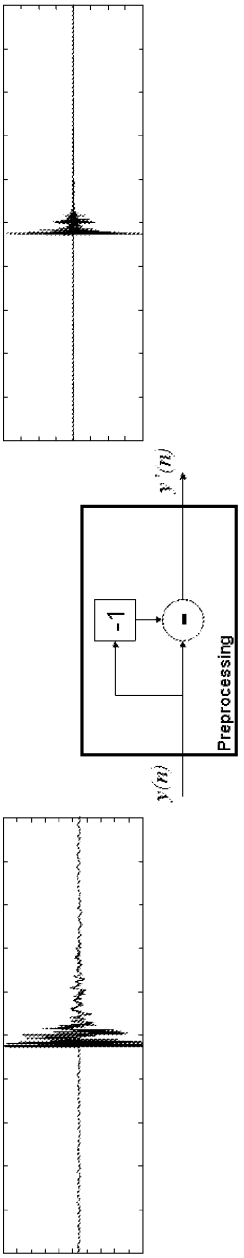

y(n)
-1
-
Preprocessing
y'(n)

FIG. 2

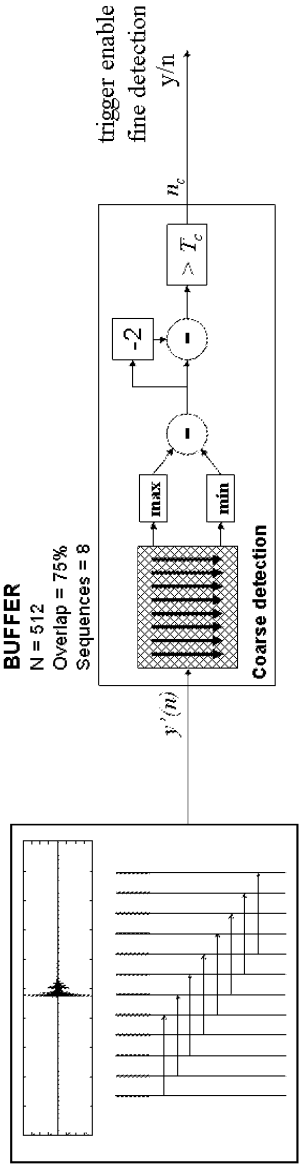

y'(n)
BUFFER
N = 512
Overlap = 75%
Sequences = 8
max
min
-
-2
-
> T_c
Coarse detection
n_c
trigger enable
fine detection
y/n

FIG. 3

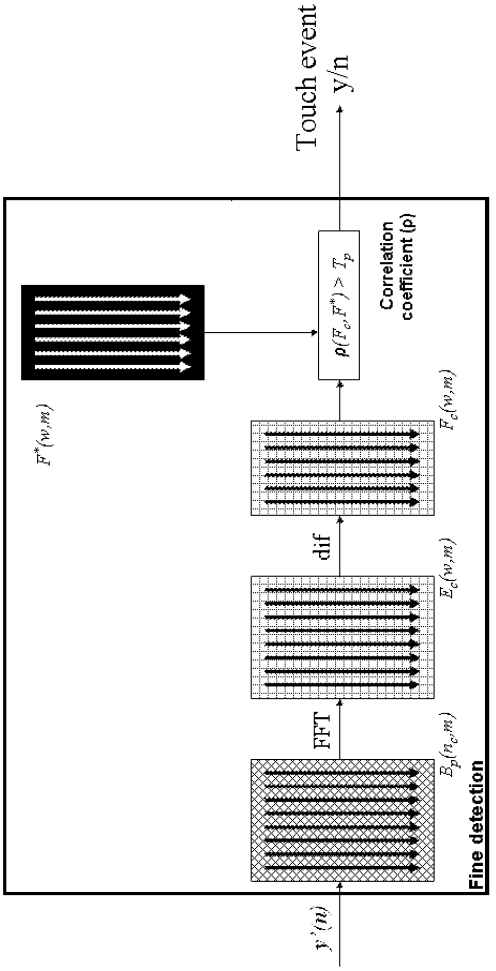

$y'(n)$
Fine detection
$E_p(n_c,m)$
FFT
$F_c(w,m)$
dif
$F_c(w,m)$
$F^*(w,m)$
$\rho(F_c,F^*) > T_p$
Correlation coefficient (ρ)
Touch event y/n

FIG. 4

METHOD FOR DETECTING AUDIO TICKS IN A NOISY ENVIRONMENT

FIELD OF THE INVENTION

The invention refers to a method and means for detecting audio ticks in a noisy environment. In particular said tick detection is envisaged to be used in mobile terminals in which tapping the terminals together—causing a tapping tick sound—is used as (part of a pairing protocol.

BACKGROUND

A pairing protocol is disclosed in International Patent Application WO 2009/014438. This document describes a method to enable two mobile terminals to establish each others identity. One mobile terminal is physically tapped (softly hit) against the other. Both terminals detect the tapping. The terminals send messages identifying themselves and providing information about the detected tapping. When it is determined that the information about the detected tapping of two terminals matches, acknowledgement messages are sent to the mobile terminals to establish the mutual identification.

This method has a problem in that not each relatively loud and short noise is a tick caused by tapping terminals together, where too many "false ticks" affect the ease and usefulness of such a "tap-to-pair" system.

Another problem is that discrimination of ticks from non-ticks, especially in noisy environments needs much processing capacity when implemented in conventional ways, where the processing capacity of (small) mobile terminals is restricted and, besides, the terminal's battery has a limited capacity.

EP 1978508 discloses a method of musical beat detection for Karaoke purposes, among others. A musical tune is composed on the basis of a measure of time, such as a bar and a beat. The bar and beat determine the musical rhythm. The document describes system for determining beat position time points in from audio wave files. It is proposed to detect beat positions from large instantaneous peaks in the time-series waveform. But since beat components cannot be extracted highly accurately from a time-series waveform, an alternative is disclosed wherein a spectrogram is computed using an FFT (Fast Fourier Transform) on the time-series waveform. The system considers portions of the spectrogram at which the power spectrum instantaneously changes significantly. These are assumed to be beat components of the rhythm, due to hitting a drum for example.

EP 1978508 notes that such a beat detection does not provide in accurate beat timing extraction. To improve the accuracy an alignment process is performed on the beat position information of a musical tune or musical tune portions expected to have an identical tempo in the extracted beat position information. Such musical tune portions may be selected based on the variance of beat timing. By realigning the time points according to the tempo more accurate positioning of the beat is realized.

Thus, in EP 1978508 initial coarse detection of beat position from audio data in the audio file is followed by further processing of the detected positions. EP 1978508 does not propose to process the audio data itself more thoroughly. EP 1978508 would not be considered for a pairing protocol, because one cannot expect users of mobile terminals against

each other according to a musical rhythm that lasts long enough to provide for a musical tune portion expected to have an identical tempo.

SUMMARY

One aim is to provide a method and apparatus for the reliable detection of audio ticks in noisy environments.

Another aim is to provide a method and apparatus for the detection of audio ticks in noisy environments which are suitable for execution by (small) mobile terminals having a limited computing capacity.

Accordingly, a method for detecting ticks in a noisy environment is provided, from an audio signal which includes a stream of audio samples. The method comprises the steps of:

- applying the audio signal to a coarse tick detection processor arranged to decide whether it is likely that the audio signal includes a tick and, only then,
- enabling a fine tick detection processor to decide, by more thorough processing said audio signal, whether the audio signal represents a tick.

In an embodiment, the method is used to detect ticks arising from touches against a specific device. In this embodiment, the fine tick detection may be performed dependent on comparison with predetermined properties of an audio response to the touches that are characteristic for the specific device, whereas coarse tick detection may be performed in a more simple generic way that does not depend a comparison with properties of the specific device. A computation of a correlation between the detected sound signal and a reference sound signal may be used to perform the comparison. The reference properties or reference signal may be determined in advance by training using the specific device, but if the reference properties or reference signal are substantially the same for all devices in a class, the reference properties or reference signal for another device or devices in the class may be used.

It will be understood that the ticks are typically sounds of a relatively short duration which may be contained in the audio signal, which audio signal will also contain noise. It will further be understood that the coarse tick detection processor (arranged for coarse processing) and the fine tick detection processor (arranged for fine processing) may be implemented in either hardware or software, or in a combination of hardware and software. Accordingly, the first step may be referred to as coarse processing the audio signal so as to decide whether the audio signal contains at least one tick sound, while the second step may be referred to as fine processing the audio signal so as to confirm whether the audio signal contains a tick sound. It will be understood that the terms "coarse processing" and "fine processing" are relative.

According to an aspect, therefore, the method for detecting, in an audio signal, ticks in a noisy environment, is preferred to comprise the steps of:

- coarse processing the signal, including normalizing it;
- deciding, based on the value of the normalized signal whether the signal should be subjected to fine processing or not;
- fine processing the signal, including converting the signal from the time domain into the frequency domain;
- comparing, in the frequency domain, the signal with an exemplary sample of an audio tick;
- deciding whether the audio signal represents an audio tick or not.

This is based on the understanding that, for "tick detection" in a "tap-to-pair" system using mobile terminals which have only restricted computing capacity, a two-step computing

approach is most suitable, i.e. a coarse processing step which does not need intensive computing, and a fine processing step which does require more computing power in order to perform well, but which is only executed if the coarse step is positive, i.e. indicates that the signal part probably represents a sound tick. Moreover, the fine processing step is only executed using the (restricted) signal part selected by means of the coarse processing step, causing that always the only a relative small signal parts will be intensively processed.

Preferably, the coarse processing step includes buffering audio samples, determining the maximum and minimum values of each sample, forming their local range, delaying said local range with one or more samples, and comparing the difference between the actual local range and the delayed local range with a threshold value, and, if the difference exceeds the threshold value, outputting a trigger for enabling said fine detection processor.

More preferably, the fine processing step includes buffering audio samples, computing, for each buffer content, a Fast Fourier Transformed buffer content, determining the difference between the resulting frequency components, comparing said difference to the content of a previously trained reference set and calculating a correlation coefficient between them, comparing said correlation coefficient to a threshold value and, if the correlation coefficient exceeds the threshold value, outputting a signal indication the presence of a tick in the audio signal. The reference set may be called a fingerprint set.

Hereinafter the complete processing will be outlined in detail. As already stated before, the detection should be:

- robust in terms of small false positive and negative rates. This is achieved by forming a reference of a touch that is specific for a mobile device by means of a training method. This reference exploits characteristics of the audio signal of a touch;
- efficient in terms of battery use, what requires to minimize the use of complex operators. This is achieved by a coarse-to-fine approach, where using simple operations most touches are detected plus some noisy touch. In the fine stage, is test whether the touch matches with touch characteristics.

Fine detection may be used to cause mutual identification op a pair of mobile communication that were tapped together to generate the tick, as described for WO 2009/014438, or to open a communication channel between the mobile communication devices.

A system is provided for detecting, in an audio signal which includes a stream of audio samples, ticks in a noisy environment, the system comprising a coarse tick detection processor and a fine tick processor, wherein the course tick processor is arranged to decide whether it is likely that the signal includes a tick and, only then, to enable the fine tick detection processor to decide, by more thorough processing said audio signal, whether the audio signal represents an audio tick. The coarse tick detection processor and a fine tick processor may be a same processor, operating under control of respective software modules for coarse and fine tick detection respectively.

Further a computer program product is provided for carrying out the method as defined above. A computer program product may comprise a set of computer executable instructions stored on a data carrier, such as a computer readable disk like a CD or a DVD or a magnetic disk or a semi-conductor memory. The set of computer executable instructions that, when executed, make a programmable computer carry out the method as defined above, may also be available for downloading from a remote server, for example via the Internet.

Still further provides a consumer product is provided, comprising a system as defined above, which consumer product may be a mobile telephone device. Additionally provides a server is provided for identifying proximate devices, which server is arranged for carrying out the method as defined above and/or comprises a system as defined above.

BRIEF DESCRIPTION OF THE DRAWING

These and other objects and advantageous aspects will become apparent from a description of exemplary embodiments.

FIG. **1** shows an overview of the preferred embodiment of the system, showing its main modules;

FIG. **2** shows a preprocessing module;

FIG. **3** shows a coarse processing module and illustrates loading of its input buffer;

FIG. **4** shows a fine processing module;

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

FIG. **1** shows that an audio signal y(n) which is investigated whether it includes a portion representing a "tick", is fed to an (optional) preprocessing module (shown in FIG. **2** more in detail) in which the signal is normalized in order to facilitate further processing. The preprocessed signal y'(n) is fed to a coarse tick detection processor (shown in FIG. **3** more in detail) which decides whether it is likely that the signal (part) y'(n) represents a tick or not. In the first case a trigger value n_c is set which enables—represented by a switch—to feed the preprocessed signal y'(n) to a fine detection processor (shown in figure FIG. **4** more in detail) in order to have it processed there. Absence of the trigger disables the fine processing. Due to this enabling/disabling control output by the coarse processor the fine detection processor (which requests relatively much processing capacity) is solely enabled when the coarse processor (which does not need much processing capacity) "suspects" the presence of a "tick" in an audio signal portion and converts this "suspection" into enabling trigger.

It is noted that, although the enabling/disabling function has been represented by a switch (symbol), this enabling/disabling function can be implemented in several other ways. Important is that the fine processor is solely activated or involved (anywise), by means an "enable" trigger (signal) from the coarse processor.

FIG. **2** shows the preprocessing module in which the difference y'(n) is generated between each audio sample y(n) and its previous audio sample y(n−1). This action is performed by the shown digital delay circuit.

FIG. **3** illustrates, at the left side of the figure, the construction of an input buffer for the coarse processor which is always loaded with 75% overlapping audio samples. In that way the buffer collects the last 16 normalized samples whereafter the maximum and minimum values of the collected samples can be selected. The difference between these values determine the local range. In a next step the local range is delayed with two samples. The difference between the current range and the delayed range is tested by a threshold: if the difference is larger than a threshold value T_c then a trigger n_c is produced, causing enablement of the fine detection processor; otherwise no such trigger is given.

FIG. **4** presents an embodiment of a fine detection processing module, where a fast Fourier transform is computed for each buffer, and the difference is determined between the resulting frequency components. This difference is compared then to a (previously stored) trained reference (top right)

using a correlation coefficient. This configuration is scale invariant, causing that both soft and heavy touches (ticks) are detected correct. The trained reference may be obtained for example by signaling when a touch is generated, determining properties of the audio signal at that time and storing the generated properties, or storing averages of the properties obtained with respective touches. Training may also comprise determining relative weights that may be assigned to different properties, based on the observed spread in the values obtained using respective touches during training. Training may have the advantage that detection accounts for the particular mechanical vibration modes of the device that is touched and used to receive the audio signal (e.g. the vibration modes of its housing).

Summarizing the complete processing when the fine processor, under control of the coarse processor output n_c, has been enabled, the content y(n), recorded by e.g. the microphone of a mobile device is normalized to a zero-mean signal y'(n) which is buffered with (e.g. 75%) overlap in the time domain, resulting in a matrix $B_p(n,m)$. Fourier transformation (FFT) is performed on that buffer content $B_p(n,m)$ resulting in a matrix $E_c(w,m)$ which is also buffered with overlap in the time domain. Finally the frequency wise difference between the (FFT) buffers is calculated, resulting in a matrix $F_c(w,m)$, also buffered with overlap in the time domain, which matrix $F_c(w,m)$ is compared to a reference matrix $F^*(\omega,m)$. When the resulting correlation coefficient exceeds a predetermined threshold value, the microphone signal is deemed to include a "tick", which can be used as (part of) a pairing protocol as e.g. disclosed in WO 2009/014438.

A more detailed discussion of the coarse and fine audio signal processing steps will be given below.

Coarse Detection

A sample of the audio signal recorded by a mobile device is given by y(n) ($y(n)\epsilon[-1,1]$), where n ($n\epsilon N$) is its sample number. The audio signal is sampled with a frequency f with unit samples per second. The touches in this example touches were sampled with a sample frequency of 11025 Hz. The audio signal could be biased with a constant, which suppressed by taking the sample difference:

$$y'(n)=y(n)-y(n-1). \tag{1}$$

Actually, this is a basic high pass filter. In the coarse approach a touch is detected, if the local difference between maximum and minimum signal range increases significantly. This point of change represents the start of the touch. This sample is detected as follows.

The signal is buffered using a first-in-first-out (FIFO) buffer that contains N_c ($N_c\epsilon N$) samples. This buffer is given by

$$B_c(n)=B_c[y'](n)=\{y'(n-N_c+1), \ldots, y'(n-1), y'(n)\}. \tag{2}$$

From this buffer the difference between maximum and minimum is computed, which represents the signal range R in the last N_c samples:

$$R(n)=\max B_c(n)-\min B_c(n). \tag{3}$$

If this range increases with more than T_c, then a coarse touch δ' is detected. As soon as a coarse touch is detected, another touch cannot be detected within the next N_c samples.

$$\delta'(n)=\begin{cases} 1 & \text{if } R(n)-R(n-2)>T_c, \\ & \delta'(n-N_c), \ldots, \delta'(n-2), \delta'(n-1)=0, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

FIG. **1** presents the detection of the coarse touch sample for an example. The sample n_c is the start sample of a touch:

$$n_c=\{n|\delta'(n)=1\}. \tag{5}$$

This approach detects most touches, also within noisy environments. Therefore, the false negative rate is rather high. On the other hand it detects more than touches, each event that causes a change in the range of the audio signal is identified as a touch. Therefore, the next stage explores another signal property of a touch to reduce the false negative rate.

Other types of course detection may be used, such as detection of output pulses based on a comparison of pulse amplitude with signal amplitude in a time interval that surrounds the pulse.

Fine Detection

Precise touch detection exploits the signal characteristic of a touch. A primitive model of a touch is a pulse, and an important property of a pulse is that its Fourier Transform (FT) is a uniform energy spectrum for all frequency components according to [2]. This property is exploited using the FT applied to FIFO buffers before and after a coarse touch. In total N_b ($N_b\epsilon N$) buffers are filled containing N_p ($N_p\epsilon N$) samples with p ($p\epsilon[0,1]$) percentage of overlap. The last sample in buffer m ($m\epsilon\{0, \ldots, N_b-1\}$) for coarse touch n_c is given by

$$n_c(m)=n_c+mN_p(1-p), \tag{6}$$

such that buffer m is given by

$$B_p(n_c,m)=\{y'(n_c(m)-N_p), \ldots, y'(n_c(m)-2), y'(n_c(m)-1)\}. \tag{7}$$

Note that the first buffer (m=0) does not contain sample $n=n_c$, which was the first sample of a touch. For each buffer is the FT computed, such that the energy of each frequency component (ω) is given by

$$E_c(\omega,m)=\log_{10}|[B_p(n_c,m)](\omega)|. \tag{8}$$

The fast FT is used to compute the energy spectrum fast. For this reason the number samples in a buffer is a power of two: $N_p\epsilon 2^k$, $k\epsilon 1, 2, \ldots$.

The energy of the frequency components is used to represent a reference, which will be called a fingerprint of a touch, which is given by

$$F_c(\omega,m')=E_c(\omega,m')-E(\omega,m'-1). \tag{9}$$

FIG. **2** presents this fingerprint as an image. The difference is only computed if the energy components exist, such that $m'\epsilon\{0, \ldots, N_b-3\}$. The difference between the energy components is taken for three reasons:

1. The difference between buffers with and without touch properties capture the touch properties.

2. The recorded signal y is a superposition of a touch plus background signals, which we assume stationary within the buffers. Influence of these background signals is suppressed by this difference.

3. Due to this difference the average is by definition equal to zero, such that it is allowed to compute a correlation coefficient to compare fingerprints according to [1].

TABLE 1

Overview of parameter used for touch detection

Parameter	Range	Description
N_c	16	Number of samples in coarse buffer
T_c	0.6 . . . 1.0	Threshold applied to range change difference
N_p	512	Number of samples in precise buffer

TABLE 1-continued

Overview of parameter used for touch detection

Parameter	Range	Description
N_b	5 . . . 8	Number of precise buffers
p	0.75	Percentage of overlap between buffers
T_p	0.6 . . . 0.8	Threshold applied to range change difference

From multiple touches the optimal fingerprint F* is learned as its average. This fingerprint is device specific on which the touch is detected. This fingerprint is compared to the measured fingerprint (F_c) using the correlation coefficient ($\rho\epsilon[-1,1]$):

$$\rho[F_c, F^*] = \frac{\sum_{\omega,m} F_c(\omega, m)F^*(\omega, m)}{\sqrt{\left(\sum_{\omega,m} F_c(\omega, m)^2\right)\left(\sum_{\omega,m} F^*(\omega, m)^2\right)}} \tag{10}$$

Most important for robust fingerprint computation is that the start sample of a touch (n_c) is detected correctly, such that the fingerprint exploits always the same touch characteristics. This solution is innovative.

Using this comparison the precise touch is detected as

$$\delta(n) = \begin{cases} 1 & \text{if } \rho[F_c, F^*] > T_p, \\ & \delta'(n) = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Touch detection is based on quite some parameters. Table 1 presents an overview of these parameters with their range of use.

Although an advantageous example of a comparison of properties of the audio signal with the fingerprint has been illustrated by means of correlation of spectral properties, it should be appreciated that other ways of comparing may be used. For example, the values of the spectral properties may be compared directly, the tick being detected when no difference exceeds a threshold, or no more than a predetermined number of differences exceeds a threshold, or an aggregate difference does not exceed a threshold. Instead of spectral values other properties may be compared, for example the results of filtering the audio signal with different predetermined filters, or a time correlation may be used.

In summary, coarse touch detection presents an efficient method to detect touches, because the used operators are simple: buffer, maximum, minimum, difference and a comparison. Precise touch detection is not efficient, because a Fourier transform is computed. Since it is only computed, if a coarse touch is detected, it achieves a minimum processor and power use.

Although coarse and fine touch detection from an audio signal have been described, the skilled person will appreciate that the detection method does not depend on whether the sound arises from a touch. Nor does it depend on the type of objects that are touched. However, almost all tick sounds are produced by touching objects. According to one aspect the method is applied to mobile communication devices (commonly referred to as mobile devices in the telecommunication art), in order to form pairs of such devices as explained in WO 2009/014438. As noted in this document the detected touch may be a touch of the mobile devices themselves, or external touches such as clapping hands within detection range from the mobile devices.

Each mobile device has a microphone and a processing unit. An audio identifier may be produced by tapping mobile devices together. The tapping sound is received by the microphones of both mobile devices and the audio signals registered by the microphones of the respective devices is supplied to the processing units of the respective devices. The processing unit of the devices may comprise the preprocessing module, the coarse processing module and/or the fine processing module. These modules may be realized for example using a programmable processor and program modules with programs of instructions for the programmable processor that make the programmable processor perform the functions of the preprocessing module, the coarse processing module and/or the fine processing module.

The embodiments may be used in methods and systems for matching proximate devices, and also in devices, such as servers, for use in such systems. An example of a system for matching proximate devices is disclosed in WO 2009/014438 mentioned above. A method of identifying proximate mobile devices, may be used wherein the method comprises a first mobile device detecting a tick, a second mobile device detecting the tick, the first mobile device sending a request message associated with the tick to the second mobile device, and the second mobile device, upon receipt of the request message, sending an acknowledgement message to the first mobile device so as to establish mutual identification. The request message and/or the acknowledgement message may comprise information about the detected tick, so that at least one device is enabled to verify that the devices have detected the same tick by comparing the information about the detected tick from both devices. Alternatively, a server may perform the verification and send confirmation messages to one or both mobile devices, or establishes a communication channel between the mobile devices if the information about the detected tick from both devices matches. Alternatively the devices may send request messages with their information about the tick to the server and the server may send acknowledgement messages to a pair of devices when it is verified that these devices have both detected the same tick.

Advantageously, the embodiments may be utilized in consumer devices, such as mobile telephone devices, PDAs (Personal Digital Assistants), computers, in particular portable (e.g. laptop and notebook) computers, electronic book viewers, and other consumer devices. In an embodiment the coarse tick detection processor is part of a mobile device and the fine tick detection processor is part of a server. In this case the mobile device may be configured to perform coarse tick detection and to transmit selected audio signal data to the server in response to detection of a tick by said coarse tick detection, the selected audio signal data comprising audio signal data that resulted in the coarse tick detection. Audio signal data from a time window of predetermined length may be selected that comprises a time point at which the coarse tick was detected. In this embodiment the server receives the selected audio signal data and applies the fine tick detection to the audio signal data.

A system is provided for detecting ticks in a noisy environment, from an audio signal which includes a stream of audio samples (y(n), (y'(n)), the system comprising a coarse tick detector and a fine tick detector, wherein the course tick detector is arranged to perform an initial detection of ticks from the audio signal and to trigger the fine tick detector to perform fine tick detection only when the course tick detector has made an initial detection of a tick, the fine tick detection

comprising more thorough processing of said audio signal than course tick detection. In an embodiment the system comprises a mobile communication device with a microphone, coupled to the coarse tick detector and a fine tick detector to supply the audio signal, the fine tick detector being configured to compare properties of the audios signal with reference properties that are specific to the mobile device, the coarse tick detector only performing generic processing without a comparison with reference properties that are specific to the mobile device. The system may be configured to provide information to the mobile communication device that establishes the identity of another mobile communication device that was used to generate the tick by touching the mobile communication device, or to open a communication channel between the devices, triggered by fine detection.

It is noted that any terms used in this document should not be construed so as to limit the scope of the invention. In particular, the words "comprise(s)" and "comprising" are not meant to exclude any elements not specifically stated. Single (circuit) elements may be substituted with multiple (circuit) elements or with their equivalents.

It will be understood by those skilled in the art that the invention is not limited to the embodiments illustrated above and that many modifications and additions may be made without departing from the scope of the invention as defined in the appending claims.

REFERENCES

[1] A. Leon-Garcia: Probability and Random Processes for Electrical Engineering. Number 0-201-50037-X. Addison-Wesley Publishing Company, Inc., second edition, 1994.

[2] B. Porat: A course in digital signal processing. John Wiley & Sons, Inc., New York, N.Y., USA, 1996.

The invention claimed is:

1. Method for detecting ticks on a device with which an audio signal is received in a noisy environment, from the audio signal which includes a stream of audio samples (y(n), (y'(n)), the method comprising the steps of:

applying the audio signal to a coarse tick detection processor arranged to decide whether the audio signal likely includes a tick; and

only if the course tick processor decides that the audio signal likely includes a tick, enabling a fine tick detection processor to decide, by more thorough processing of said audio signal, whether the audio signal represents a tick on the device, wherein the fine tick detection is performed using a comparison with a previously trained set of reference properties that are characteristic for the device with which the audio signal is received.

2. Method according claim **1**, wherein the stream of audio samples is derived from a microphone of a mobile communication device that captures sound generated by tapping the mobile communication device against other objects and noise from an environment of the mobile communication device.

3. Method according claim **1**, wherein said comparison comprises computing a correlation coefficient between the reference properties and information derived from the stream of audio samples.

4. Method according claim **1**, wherein the coarse processing step includes buffering a series of audio samples (y(n), (y'(n)) in a buffer, determining maximum and minimum values from this buffer, computing a local range between the maximum and minimum values, delaying said local range with one or more samples, and comparing a difference between an actual local range and the delayed local range with a threshold value (T_c), and, if the difference exceeds the threshold value, outputting a trigger (n_c) for enabling said fine detection processor.

5. Method according to claim **1**, wherein the fine processing step includes buffering the audio samples (y(n), (y'(n)) from a time interval that has been decided to likely include the tick by the coarse tick detection into buffer contents that include a buffer content with said tick and a buffer content that precedes said tick, computing, for each buffer content (B_p(n,m)), a Fast Fourier Transformed buffer content (E_c(w,m)), obtaining frequency components from the Fast Fourier Transformed buffer content (E_c(w,m)), determining a difference (F_c(w,m)) between frequency components for the buffer contents with said tick and preceding said tick, comparing said difference (F_c(w,m)) to contents of a previously trained reference set (F*(w,m)) and calculating a correlation coefficient ($\rho(F_c,F^*)$) between them, comparing said correlation coefficient to a threshold value (T_p) and, if the correlation coefficient exceeds the threshold value, outputting a signal indicating the presence of a tick in the audio signal.

6. Method according to claim **5**, comprising a training step, wherein a reference set of a touch is formed that is specific for a mobile device from which the stream of audio samples is derived using a microphone of a mobile communication device.

7. Method according to claim **1**, wherein the audio signal is preprocessed before being processed in the coarse and/or fine processing steps respectively, including determining and outputting a difference (y'(n)) between each audio sample (y(n)) and a previous audio sample (y(n−x)).

8. A non-tangible computer readable storage medium comprising a program of instructions for making a programmable computer to carry out one or more of the method steps according to claim **1**.

9. System for detecting, in an audio signal which includes a stream of audio samples (y(n), (y'(n)), ticks on a device with which the audio signal is received in a noisy environment, the system comprising a coarse tick detection processor and a fine tick processor, wherein the coarse tick processor is arranged to decide whether the signal likely includes a tick; and, only if the course tick processor decides that the audio signal likely includes a tick, to enable the fine tick processor to decide, by more thorough processing of said audio signal, whether the audio signal represents a tick on the device, wherein the fine tick detection is performed using a comparison with a previously trained set of reference properties that are characteristic for the device with which the audio signal is received.

10. System according claim **9**, comprising a mobile telecommunication device comprising a microphone, coupled to the coarse tick detection processor and the fine tick processor to supply the stream of audio samples.

11. System according to claim **9**, wherein the coarse tick processor is adapted for buffering a series of the audio samples (y(n), (y'(n)) in a buffer, for determining maximum and minimum values from the buffer, computing a local range between the maximum and the minimum, for delaying said local range with one or more samples and for comparing a difference between the actual local range and the delayed local range with a threshold value (T_c), and for, if the difference exceeds the threshold value, outputting a trigger (n_c) for enabling said fine tick processor.

12. System according to claim **9**, wherein the fine tick processor is adapted for buffering the audio samples (y(n), (y'(n)) from a time interval that includes the tick that has been decided by the coarse tick detection into buffer contents that include a buffer content with said tick and a buffer content

that precedes said tick, for computing, for each buffer content ($B_p(n,m)$), a Fast Fourier Transformed buffer content ($E_c(w,m)$), for obtaining frequency components from the Fast Fourier Transformed buffer content ($E_c(w,m)$), for determining a difference ($F_c(w,m)$) between the frequency components for the buffer contents with said tick and preceding said tick, for comparing said difference ($F_c(w,m)$) to contents of a previously trained reference set ($F^*(w,m)$) and calculating a correlation coefficient ($\rho(F_c,F^*)$) between them, for comparing said correlation coefficient to a threshold value (T_p) and for, if the correlation coefficient exceeds the threshold value, outputting a signal indicating a presence of the tick in the audio signal.

13. System according to claim **9**, including a preprocessor adapted for preprocessing the audio signal before being processed in the coarse and/or fine processors respectively, the preprocessor being arranged for determining and outputting a difference ($y'(n)$) between each audio sample ($y(n)$) and a previous audio sample ($y(n-x)$).

14. System according to claim **9**, comprising a mobile communication device with a microphone coupled to the coarse tick detection processor and the fine tick processor, the fine tick processor being configured to trigger mutual identification of, or opening of a communication channel between, a pair of mobile communication devices that includes the mobile communication device, in response to fine tick detection.

15. Consumer product, adapted for carrying out the method according to claim **1**, and/or comprising the system according to claim **9**.

16. Consumer product according to claim **15**, which is a mobile telephone device or a PDA.

17. Server for identifying proximate devices, adapted for carrying out the method according to claim **1** and/or comprising the system according to claim **9**.

18. The method according to claim **1**, comprising executing, before said applying the audio signal to the coarse tick detection processor and enabling the fine tick detection processor, the preceding steps of:

signalling when a reference touch or respective reference touches are generated on the device,

determining reference properties of a reference audio signal obtained from the device at a time of the signaled touch or respective touches;

storing the reference properties, or averages of the reference properties obtained with the respective touches, as the reference properties that are characteristic for the device; and

wherein the fine tick detection is performed using a comparison with the reference properties that are characteristic for the device.

19. The system according to claim **9**, the system being configured to receive signals when a reference touch or respective reference touches are generated on the device, determine reference properties of a reference audio signal obtained from the device at a time of a signaled touch or respective touches and store the reference properties, or averages of the reference properties obtained with the respective touches, as the reference properties that are characteristic for the device, the fine tick processor being configured to perform the comparison with the reference properties that are characteristic for the device.

20. A method for detecting ticks in a noisy environment, from an audio signal received by a device, the audio signal including a stream of audio samples ($y(n)$, ($y'(n)$), the method comprising the steps of:

applying the audio signal to a coarse tick detection processor adapted to decide whether the audio signal likely includes a tick; and

only if the course tick processor decides that the audio signal likely includes a tick, enabling a fine tick detection processor to decide, by more thorough processing of said audio signal, whether the audio signal represents a tick, wherein the fine tick detection is performed using a comparison with predetermined reference properties that are characteristic for the device with which the audio signal is received, suppressing influence of background signals, by

computing Fourier transforms of the audio samples in a plurality of buffers;

determining differences between energies of respective frequency components of the Fourier transforms for successive buffers; and

comparing the differences to contents of a previously trained reference set.

21. The method according to claim **20**, wherein the Fourier transforms of the audio samples are computed in buffers $B_p(n_c,m)$, wherein $B_p(n_c,m)=\{y'(n_c(m)-N_p), \ldots, y'(n_c(m)-2), y'(n_c(m)-1)\}$,

wherein n_c is a sample number corresponding to a coarse touch, m is an index of a buffer, N_p a number of samples in the buffer and $n_c(m)=n_c+mN_p(1-p)$, with p in the interval [0,1]

said determining the difference comprising determining a difference $F_c(\omega,m')=E_c(\omega,m')-E(\omega,m'-1)$ between the energies $E_c(\omega,m')$ and $E_c(\omega,m'-1)$ of respective frequency components at frequencies ω of the Fourier transforms for successive buffer numbers m', m'−1; and

said comparing comprising comparing the differences ($F_c(\omega,m)$) to the content of the previously trained reference set ($F^*(\omega,m)$).

22. A system for detecting ticks from a noisy environment in an audio signal received by a device, wherein the audio signal includes a stream of audio samples ($y(n)$, ($y'(n)$) of the noisy environment, the system comprising a coarse tick detection processor and a fine tick detection processor, wherein the coarse tick detection processor is adapted to decide whether the audio signal likely includes a tick; and, only if the course tick processor decides that the audio signal likely includes a tick, to enable the fine tick detection processor to decide, by more thorough processing of said audio signal, whether the audio signal represents a tick, wherein the fine tick detection processor is configured to perform a comparison with predetermined reference properties that are characteristic for the device with which the audio signal is received, suppressing influence of background signals, by

computing Fourier transforms of the audio samples in a plurality of buffers;

determining differences between energies of respective frequency components of the Fourier transforms for successive buffers; and

comparing the differences to contents of a previously trained reference set.

23. The system according to claim **22**, wherein the fine tick detection processor is configured to compute said Fourier transforms of the audio samples in buffers $B_p(n_c,m)$ $B_p(n_c,m)=\{y'(n_c(m)-N_p), \ldots, y'(n_c(m)-2), y'(n_c(m)-1)\}$, wherein n_c is a sample number corresponding to a coarse touch, m is an index of a buffer $B_p(n_c,m)$, N_p is a number of samples in the buffer and $n_c(m)=n_c+mN_p(1-p)$, with p in the interval [0,1]; and wherein the fine tick detection processor is configured to:

determine a difference $F_c(\omega,m')=E_c(\omega,m')-E(\omega,m'-1)$ between the energies $E_c(w,m')$ and $E_c(w,m'-1)$ of respective frequency components at frequencies w of the Fourier transforms for successive buffer numbers m', m'−1; and

comparing the differences ($F_c(w,m)$) to the content of the previously trained reference set ($F^*(w,m)$).

* * * * *

EXHIBIT 5

US009602649B2

(12) **United States Patent**
Laarakkers et al.

(10) **Patent No.: US 9,602,649 B2**
(45) **Date of Patent: Mar. 21, 2017**

(54) **EVENT DISAMBIGUATION**

(75) Inventors: **Jeroen Laarakkers**, Nootdorp (NL); **Mattijs Oskar van Deventer**, Leidschendam (NL); **Victor Bastiaan Klos**, The Hague (NL)

(73) Assignee: **Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek TNO** (NL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1283 days.

(21) Appl. No.: **13/386,461**

(22) PCT Filed: **Jul. 23, 2010**

(86) PCT No.: **PCT/NL2010/050478**

§ 371 (c)(1),
(2), (4) Date: **Apr. 9, 2012**

(87) PCT Pub. No.: **WO2011/010925**

PCT Pub. Date: **Jan. 27, 2011**

(65) **Prior Publication Data**

US 2012/0184372 A1 Jul. 19, 2012

(30) **Foreign Application Priority Data**

Jul. 23, 2009 (EP) 09166191

(51) **Int. Cl.**
A63F 13/31 (2014.01)
G08B 3/10 (2006.01)
(Continued)

(52) **U.S. Cl.**
CPC ***H04M 1/7253*** (2013.01); ***H04M 1/72544*** (2013.01); *A63F 13/31* (2014.09); *G08B 2001/085* (2013.01); *H04L 63/08* (2013.01)

(58) **Field of Classification Search**
CPC H04M 1/72544; H04M 1/7253; H04M 2207/18; H04M 2242/30; H04M 2242/40;
(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,819,220 B1 * 11/2004 Lee H04M 3/42382 340/573.1
2006/0282649 A1 12/2006 Malamud et al.

FOREIGN PATENT DOCUMENTS

WO WO 01/81885 11/2001
WO 2004/062131 7/2004
WO WO 2009/014438 1/2009

OTHER PUBLICATIONS

Extended European Search Report for Application No. 09166191.8, mailed Dec. 17, 2009.

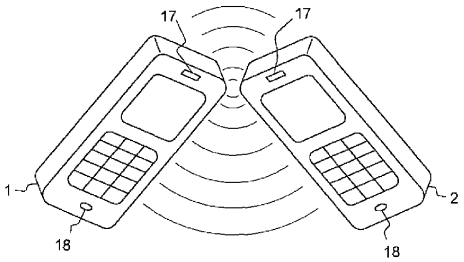
(Continued)

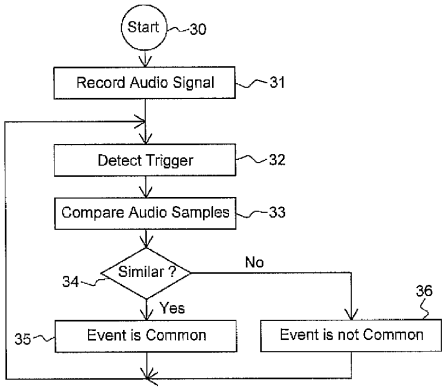
Primary Examiner — Damon Pierce
(74) *Attorney, Agent, or Firm* — Patterson Intellectual Property Law PC

(57) **ABSTRACT**

A method of identifying at least two proximate devices (**1**, **2**) comprises the steps of:—recording, in each device, an audio signal (AS) so as to produce respective audio samples, —detecting, in each device, a sensory identifier serving as a trigger (TR), —comparing the audio samples during a common interval (I) defined relative to the trigger (TR), and —deciding that the event is common to the devices only if the respective audio samples are sufficiently similar. The common interval (I) starts prior to the trigger (TR) and preferably also ends prior to the trigger.

26 Claims, 2 Drawing Sheets





(51) **Int. Cl.**
H04M 1/725 (2006.01)
G08B 1/08 (2006.01)
H04L 29/06 (2006.01)

(58) **Field of Classification Search**
CPC .. H04M 2250/02; H04L 63/08; A63F 13/217; A63F 13/31; A63F 13/327; A63F 13/34; A63F 13/215; A63F 13/424; A63F 13/44; A63F 2300/572; A63F 2300/6072; G08B 1/08; G08B 3/00; G08B 2001/085
See application file for complete search history.

(56) **References Cited**

OTHER PUBLICATIONS

PCT/NL2010/050478 International Search Report, mailing date Oct. 12, 2010.

* cited by examiner

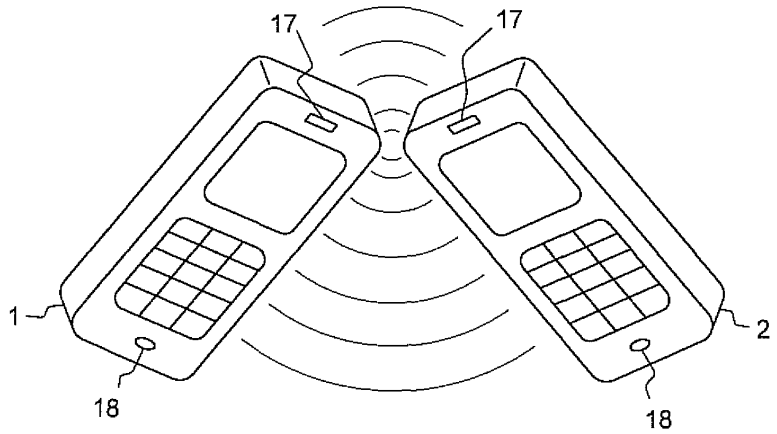


FIG. 1

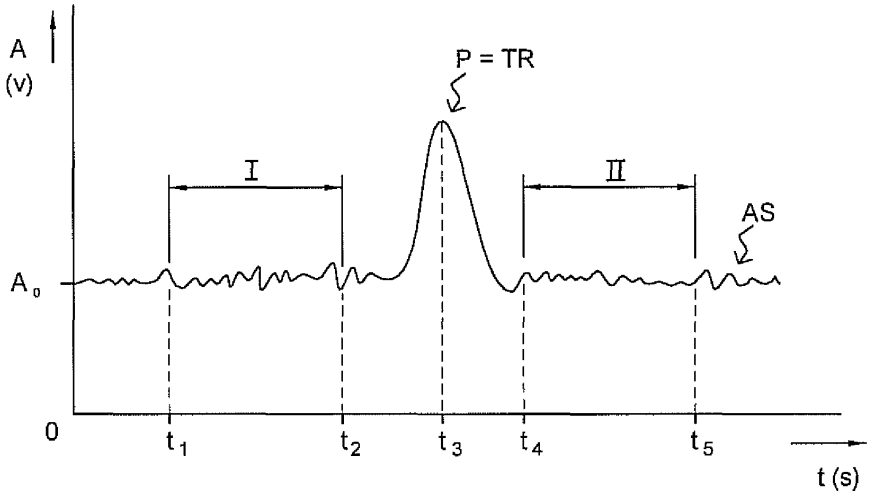


FIG. 2

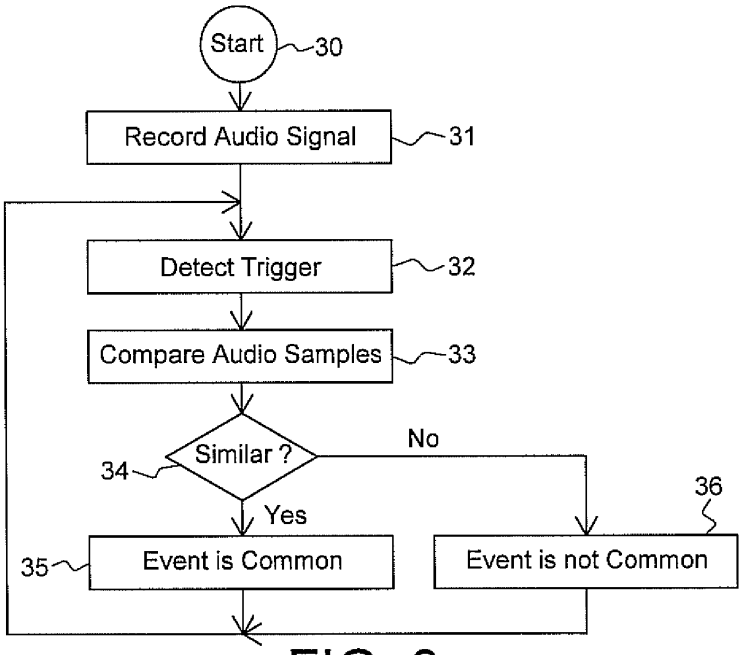


FIG. 3

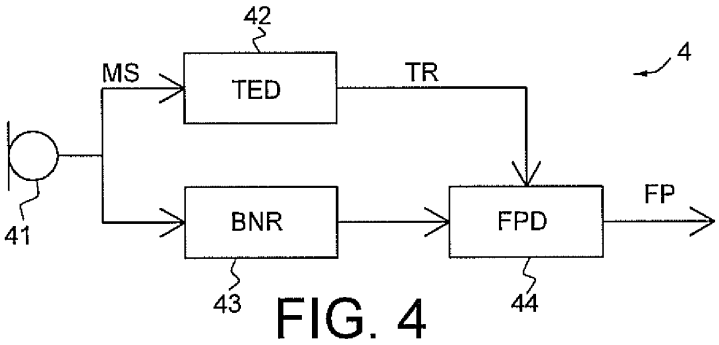


FIG. 4

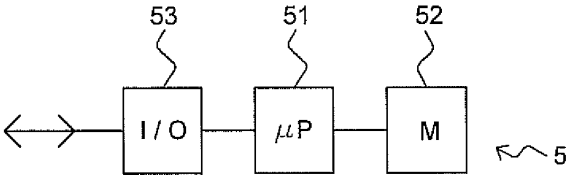


FIG. 5

EVENT DISAMBIGUATION

The present invention relates to event disambiguation. More in particular, the present invention relates to a method of and a device for identifying proximate devices by using a sensory identifier, that is, to a method of and a device for pairing at least two devices if they are involved in a common event.

International Patent Application WO 2009/014438 (TNO/KPN) discloses a method of identifying proximate mobile devices by using sensory identifiers, that is, identifiers which can be perceived with the senses. Such a sensory identifier may for example be an audio identifier, such as a particular sound, or a video identifier, such as a particular image. The identifier serves to (uniquely) identify a common event, such as two mobile devices touching or two mobile devices registering the same symbol. However, even if an identifier is unique, it may be ambiguous when detected. The sound of two mobile devices being tapped together may be unique to those devices, but may be reflected and be detected again as an echo, or may be picked up by other mobile devices which were not involved in the tapping event. This may lead to identification errors, as some devices may detect a sensory identifier they are not proximate to, for example an echo of a touching event.

International Patent Application WO 01/81885 (Metravib) discloses a method and installation for detecting and locating a source of noises and vibrations in a pipe system. This known method relates to an entirely different field of technology: locating defects in pipes. In this known method, at least two sensors mounted on a pipe are separated by a known distance, which allows the location of a noise source to be determined. A triggering time may be followed by a period of noise signal analysis. The noise signal captured during this period may be transmitted to neighbouring sensors for intercorrelation in order to increase the sensitivity of the sensors. WO 01/81885 fails to suggest a solution for the problem of event disambiguation, in particular in the context of device pairing.

It is an object of the present invention to overcome these and other problems of the Prior Art and to provide a method of identifying proximate devices using a sensory identifier, in which method the chance of erroneous identifications is significantly reduced.

It is a further object of the present invention to provide a method of identifying proximate devices using a sensory identifier which can even be carried out without requiring the content of the sensory identifiers.

It is a still further object of the present invention to provide a device, preferably a mobile consumer device, as well as a system for carrying out the method of the present invention.

Accordingly, the present invention provides a method of identifying proximate devices, the method comprising the steps of:

- recording, in each device, an audio signal so as to produce respective audio samples,
- detecting, in each device, a sensory identifier,
- using said detecting as a trigger and using said sensory identifier for matching the devices,
- comparing the audio samples during a common interval defined relative to the trigger, and
- deciding that the devices are proximate only if the respective audio samples are sufficiently similar.

By detecting a sensory identifier, such as a sound or image, by each device concerned, a matching process can be started: all devices that detected the same sensory identifier

are assumed to be proximate to each other. In addition, the detection of the sensory identifier is used as a trigger, creating a reference point in time. By comparing audio samples of substantially the same time interval it can be determined whether those audio samples match. If the audio samples match, that is, if they are sufficiently similar, it is decided that the event providing the sensory identifier is common to the devices: their "audio background" is (nearly) the same and proximity may be assumed. If the audio samples fail to match, that is, if they are too dissimilar, it is decided that the event providing the sensory identifier is not common to the devices and that the sensory identifier has led to a false detection. In case of an audio identifier, for example, erroneously detecting the audio identifier (that is, the sound) may be due to reflections off walls or ceilings.

It can therefore be seen that the invention substantially reduces or even eliminates false common proximity detections by taking the "audio background" of the devices into account. It will be understood that deciding that the devices are proximate, that is, that the devices are in the vicinity of each other, is only possible if the devices concerned have all detected matching representations of a sensory identifier. In other words, each device which detected a sensory identifier made a representation (i.e. a numerical code) of the detected sensory identifier, and the representations of the devices concerned were found to match, that is, to be identical or to be sufficiently similar. The representations may be exchanged between the devices, or between the devices and a server, to establish the match.

To determine whether audio samples are sufficiently similar, the audio samples (that is, the audio recordings) may be aligned (that is, they may be synchronised in time) and then their (dis)similarity may be determined using a suitable measure, for example the least squares method, and subsequently be compared with a threshold value. The threshold value may be predetermined but may also be dynamic, that is, adjusted according to the circumstances.

The method of the present invention can be used with two devices, but also with three or more devices, for example, four, five, eight, ten or even more. It is noted that the step of detecting a sensory identifier may comprise detecting the providing of a sensory identifier, and that a prior step of providing a sensory identifier may be included in some embodiments of the present invention.

It is possible to start recording the audio samples when the trigger is detected. In a preferred embodiment, however, the common interval starts prior to the trigger and preferably also ends prior to the trigger. That is, the audio sample is preferably pre-recorded, before the trigger is detected. This offers the advantage that the determination whether the audio samples match can be made sooner, as there is no need to wait for the recording to finish: the recording has already started and has preferably also finished before the trigger was detected.

The common interval may also overlap with the trigger, however, it is preferred that the common interval ends before the point in time at which the trigger occurs or, at least, is detected.

Advantageously, the method of the present invention may comprise the further step of comparing the audio samples during an additional common interval also defined relative to the trigger, wherein the additional common interval starts after the trigger. By using two time intervals, one before and one after the trigger, a more reliable determination of the "audio background" can be made, at the expense of a slightly longer recording and/or processing time.

In a preferred embodiment, the sensory identifier is part of the audio signal. That is, the sensory identifier is detected by analysing the audio signal and searching for a peak, trough, pattern, or other detectably signal property, which detection is then used as a trigger.

The audio samples may consist of series of digitised values of the audio signals. However, the amount of data to be stored and/or processed may be significantly reduced if the audio samples are represented by characteristic parameters. It is well known to reduce audio samples to sets of parameters, such as average amplitude, average frequency, bandwidth, and other parameters.

In a first embodiment, the step of comparing is carried out in a server. This requires the transmission of the audio samples to the server and the transmission of the matching results back to the mobile devices, but reduces the required amount of processing in the mobile devices. In a second embodiment, the step of comparing is carried out in at least one of the devices. It is possible for the comparing step to be carried out in both devices and then to exchange the results, which provides an extra check. However, the comparing step may also be carried out in a single device.

The step of using the sensory identifier for matching the devices may advantageously comprise exchanging messages between the devices, at least some of said messages containing a representation of the sensory identifier. In addition, the step of using the sensory identifier for matching the devices may advantageously comprise comparing representations of the sensory identifier and deciding that a match exists if the representations are sufficiently similar.

Accordingly, a first device may send a message containing a representation of the sensory identifier to a second device, which then compares its representation with the representation contained in the received message and sends an acknowledgement is a match is made. Alternatively, a first and a second device may each send a message containing their respective representation of the sensory identifier to a server, which compares the representations and sends an acknowledgement to each device if a match is made.

In a particularly preferred embodiment, the trigger is produced by tapping the two devices together. The sound of the two devices touching serves as a sensory (i.e. audio) identifier for identifying the tapping event.

The devices may be mobile consumer devices, for example mobile telephone devices, which are preferably arranged for playing games. However, the present invention is not limited to mobile devices and at least one of the devices may be stationary.

The present invention also provides a computer program product for carrying out the method as defined above. A computer program product may comprise a set of computer executable instructions stored on a data carrier, such as a CD or a DVD. The set of computer executable instructions, which allow a programmable computer to carry out the method as defined above, may also be available for downloading from a remote server, for example via the Internet.

The present invention additionally provides a device for identifying proximate devices, the device comprising:

- a recording unit for recording an audio signal so as to produce respective audio samples,
- a detection unit for detecting a sensory identifier and for producing a trigger in response thereto,
- a matching unit for matching representations of sensory identifiers,
- a comparison unit for comparing the audio samples during a common interval defined relative to the trigger, and
- a decision unit for deciding that the devices are proximate only if the respective audio samples are sufficiently similar.

The device of the present invention has the same advantages as the method of the present invention. The device for identifying proximate devices may itself be one of the proximate devices, or may be a remote device such as a remote server.

In a preferred embodiment, the common interval starts prior to the trigger and preferably also ends prior to the trigger. Advantageously, the comparison unit may be further arranged for comparing the audio samples during an additional common interval also defined relative to the trigger, wherein the additional common interval starts after the trigger.

The device of the present invention may be a mobile consumer device, for example a mobile telephone device, and is preferably arranged for playing games. A suitable game is Living Stratego®, although the invention is not limited to this particular game. The present invention further provides a system for gaming comprising at least one device as defined above.

The present invention still further provides a server for carrying out the method of the present invention. Such a server is preferably arranged for receiving audio samples from devices and comprises a comparison unit for comparing the audio samples during a common interval defined relative to the trigger, and a decision unit for deciding that the event is common to the devices if the respective audio samples are sufficiently similar.

The present invention will further be explained below with reference to exemplary embodiments illustrated in the accompanying drawings, in which:

FIG. **1** schematically shows exemplary embodiments of portable consumer devices for use in the method of the present invention.

FIG. **2** schematically shows an exemplary audio signal and suitable time intervals as used in the present invention.

FIG. **3** schematically shows a flow diagram representing an embodiment of the method according to the present invention.

FIG. **4** schematically shows an embodiment of a audio fingerprint arrangement according to the present invention.

FIG. **5** schematically shows a microprocessor arrangement for use in the present invention.

The mobile telephone devices **1** and **2** shown merely by way of non-limiting example in FIG. **1** each comprise a loudspeaker **17** and a microphone **18**. The mobile telephone devices **1** and **2** are examples of portable consumer devices which can be "matched" when they are in each other's proximity. This "matching" or "pairing", which may for example be achieved by tapping the devices together, may be used to exchange information between the devices. This information may comprise telephone numbers or other identification information, which in turn can be used to play games and/or engage in other activities. Thus, tapping the mobile telephone devices together may constitute an invitation to play a game or to share information. Reference is made to International Patent Application WO 2009/014438 mentioned above, which explains such pairing processes in more detail.

When the mobile telephone devices **1** and **2** are tapped together, the resulting tapping sound is received substantially simultaneously by the microphones **18** of both mobile devices. This tapping sound can be used as an identifier for identifying the pairing attempt: it is very likely that only devices which detect the tapping sound are involved in the

encounter. It can thus be said that the tapping sound is an audio identifier, a special case of a sensory identifier: an identifier that can be perceived by the (human) senses. However, other identifiers may also be used for pairing, for example pressing buttons more or less simultaneously. The present invention is, however, primarily concerned with (but not limited to) audio identifiers.

The audio identifier may alternatively be constituted by a sound signal produced by a loudspeaker (e.g. a loudspeaker **17**), the sound signal preferably consisting of multiple tones and/or pulses so as to be able to distinguish a plurality of different identifiers and/or time stamps. Such sound signals may in turn be produced in response to pressing a button.

When using sensory identifiers, in particular audio identifiers, for pairing or similar actions the problem of false matches arises. The microphones of the mobile devices may pick up reflections of the (tapping or other) sound off walls, and also sounds (or their reflections) made by other devices. In practice, therefore, the sound (or a similar identifier) is often not sufficient to properly identify the actual devices involved.

For this reason, the present invention proposes to use additional information for identifying the devices involved, and in particular, to use the "audio background" of the devices. To this end, the present invention proposes to receive and record an audio signal in both (or all) devices, and to compare the audio signals (or their representations). If the audio signals are sufficiently similar, the audio backgrounds (or acoustic backgrounds) of the devices match and it can be concluded that the devices were involved in the same encounter, in particular if the sensory identifiers also match. Accordingly, the matching of the audio signals can be used as a check of the matching of the sensory identifiers: if both the sensory identifiers and the audio signals match, the proximity of the devices is established with much greater certainty.

The (detection of the) audio identifier, such as the tapping sound, is in the present invention used as a trigger for triggering the comparison process. It is preferably also used as an identifier, for example by determining its properties (e.g. time of occurrence, amplitude and/or power, duration, etc.), for identifying the event which produced the audio identifier.

The inventive process of determining the audio background of a trigger is illustrated in FIG. **2**, where the amplitude A of an audio signal AS is shown as a function of the time t. The audio signal AS may be received by the microphones **18** of the mobile telephone devices **1** and **2** of FIG. **1**, but may also be received by a third device. The audio signal AS is recorded, that is, is received, stored and/or converted into a suitable representation (such as a digital representation after analogue/digital conversion, optionally followed by parameter extraction). The amplitude A, which may be measured in volts (V), is shown to have an initial average value A_0, while deviations from this value A_0 are mainly due to background noise. At t_3, the audio signal amplitude reaches a peak P, indicative of an audio identifier, such as the tapping together of two mobile telephone devices. The detection of the peak P triggers a comparison of the background noise during certain time intervals. For this reason, the peak P may be considered a trigger TR.

It is noted that a similar type of background matching may be carried out for video images, in particular when the sensory identifier is a visual identifier. A "video background" or "image background" may be matched to increase the reliability of matching on the basis of a visual identifier. It is noted that audio backgrounds may also be used to enhance the reliability of visual identifiers.

In FIG. **2**, two background noise measuring intervals are shown by way of non-limiting example. A first interval I extends from t_1 to t_2, and therefore both begins and ends before the occurrence of the trigger P, while a second interval II extends from t_4 to t_5 and therefore both begins and ends after the occurrence of the trigger P. Other intervals may be used (not shown) which may include the peak P. The audio signals AS, or their representations (that is, samples), recorded by both mobile telephone devices (**1** and **2** in FIG. **1**) are compared during at least one interval. If the audio signals of the two devices during the interval(s) are sufficiently similar, then it may be concluded that the devices share an acoustic background and that they are in each other's proximity. In other words, it may be concluded that the event causing the trigger (here: the peak P in the audio signal) was common to the devices.

It will be understood that the number of audio signals that may be compared is not limited to two, and that the audio signals from three, four or even more devices may be compared.

The audio signals may be compared in the first interval I, in the second interval II or in both. In a preferred embodiment of the present invention, the audio signals are only compared in the first interval I, prior to the trigger TR. This offers the advantage of being able to start comparing the signals immediately after the trigger, without having to wait for a suitable recording interval to finish. This embodiment requires the audio signal to be recorded prior to the trigger, a continuous recording is therefore preferred. It will be understood that continuous recording is not required when only interval II is used, since then the recording may be started by the trigger.

The length of the interval(s) may depend on the particular application and may be dynamically adjustable. In practice, the length of an interval may range from a few ms (milliseconds) to approximately one s (second). The preferred gap between t_2 (end of the first interval) and t_3 (trigger) may range from a few ms to a few hundred ms.

A preferred embodiment of the present invention is summarised in the flow diagram of FIG. **3**. In step **30**, the method starts. In step **31**, the audio signal (AS in FIG. **2**) is recorded, resulting in audio samples. As noted above, this recording may take place continuously as long as it may be necessary to identify events.

In step **32**, a trigger is detected, such as an audio identifier (peak P in FIG. **2**). This leads to step **33**, in which audio samples are compared. The audio samples (or audio fragments) are stored audio signals or their representations. The audio samples may originate from two or more devices. In step **34**, it is decided whether the audio samples are sufficiently similar. If they are, the event is common to the devices and a signal indicating this is produced in step **35**, after which the routine returns to step **32** (or **31** if the audio signals are not recorded continuously). If the audio samples are not sufficiently similar, the event is not is common to the devices and a signal indicating this is produced in step **36**, after which the routine also returns to step **32** (or **31** if the audio signals are not recorded continuously).

The extent of the similarity of the audio samples may be determined using a known method and criterion. For example, the audio samples may be aligned (in time), their mean values may be reduced to zero, and their absolute amplitudes may be established. Then the mean square error (MSE) may be determined using the difference of the absolute amplitudes. Subsequently, the resulting mean

square error may be compared with a threshold value. This threshold value may be predetermined or may be adjusted dynamically.

With regard to the time intervals during which audio samples are compared, there are several possibilities, which each offer specific advantages.

In a first embodiment, the audio signals are only recorded during the time interval. In the example of FIG. **2**, the recording may thus take place from t_4 to t_5, during time interval II. The start of this time interval, at t_4, may take place as soon as possible after the detection of the trigger. In this embodiment, therefore, the recording only takes place during the interval and then substantially the entire time interval is used in the comparison. This saves memory space but only allows time intervals after the trigger.

In a second embodiment, the audio signals are recorded virtually continuously and a section (that is, audio sample) of these recordings is used for the comparison (in some embodiments, more than one section may be used). In this embodiment, the time intervals correspond with the selected sections. This embodiment requires more memory space but allows audio samples recorded prior to the trigger to be used.

As stated above, the audio signals are represented by audio samples, which may be digitised audio fragments, or their representations, for example by characteristic parameters. Suitable parameters are average amplitude, time of occurrence, average frequency, bandwidth, and other parameters. By producing suitable characteristic parameters, a "fingerprint" of each audio signal can be made. This will further be explained with reference to FIG. **4**.

The merely exemplary arrangement **4** of FIG. **4** comprises a microphone **41**, a touch event detector (TED) **42**, a background noise recorder (BNR) **43**, and a fingerprint determination unit (FPD) **44**. The arrangement **4** is designed for detecting "touch events", that is, events in which two (mobile) devices touch, for example when being tapped together so as to initiate a matching process.

The microphone **41** receives sound and produces a microphone signal MS (equivalent to the audio signal in FIG. **2**), which is fed to both the touch event detector **42** and the background noise recorder **43**. The touch event detector **42** detects a peak (P in FIG. **2**) in the microphone signal, which peak is indicative of a touch event. The detection may be a simple threshold detection, or any other suitable detection. In response to the detection of the peak, the touch event detector **42** produces a trigger signal TR, which is fed to the fingerprint determination unit **44**. Signal characteristics of the peak may additionally be used as an audio identifier for matching proximate devices.

The background noise recorder **43** records the microphone signal for the purpose of registering the background noise (or, more in general, background sound) and passes it on to the fingerprint determination unit **44**. When triggered by the trigger signal TR, the unit **44** produces a fingerprint of the background noise signal by extracting suitable parameters of the signal. This fingerprint FP is then passed on to an event server or other suitable apparatus, such as the other of the two devices shown to be touching in FIG. **1**. The communication with the server is preferably wireless, using an appropriate communication link (Bluetooth®, Ethernet, GMS, UMTS, GPRS, HSDPA, . . .).

The arrangement of FIG. **4** may be implemented in hardware, in software, or in a combination of hardware and software.

The background noise or background sound constituting the "audio background" of the devices may, in part, be generated by one or all of the devices involved. At least one of the devices could, using its loudspeaker (**17** in FIG. **1**), produce an (audible and/or ultrasonic) audio signal in order to contribute to a common audio background. The audio signal could consist of a single tone, but could also comprise a certain sound pattern. The production of an audio signal by the devices could be substantially continuous, but could also be initiated by the trigger. When two or more devices produce an audio signal, they could do so either simultaneously or in turns. In the latter case the devices could be arranged for "listening" to other, similar devices and producing an audio signal in response to a detected audio signal.

An exemplary embodiment of a device for carrying out the method of the present invention is schematically illustrated in FIG. **5**, where the device **5** is shown to comprise a microprocessor unit **51**, a memory unit **52** and an input/output (I/O) unit **53**. The memory unit **52** may comprise suitable software programs for making the microprocessor unit **51** carry out the steps of the present invention, such as detecting a trigger, comparing audio samples and/or deciding whether an event is common to the devices.

The device **5** of FIG. **5** may be a server, arranged for receiving the audio signals from a plurality of portable consumer devices, such as the mobile telephone devices shown in FIG. **1**, a laptop computer, a PDA (personal digital assistant), or an MP4 player. Alternatively, the device **5** may be incorporated in a portable consumer device.

Although the present invention has been explained with reference to pairs (or triplets, or quadruplets) of mobile devices, it is not essential that all devices involved are mobile. The present invention can also be carried out with a mobile (or at least portable) device and a stationary device.

The present invention is based upon the insight that the acoustic background of an acoustic identifier may assist in reducing the number of false or erroneous detections of proximate devices, in particular by adding a check to the identification process. The present invention benefits from the further insight that determining the acoustic background in a time interval prior to the acoustic identifier shortens the detection time.

The sensory identifier (an identifier that is perceptible by the human senses) is detected from an audio sensing signal or a sensing signal for another type of sense such as touch (contact) sensing, image sensing, sensing that a button is pushed etc. The detections of the sensory identifiers of different devices may be performed independently for each of the devices, for example in each device, by detecting signal peaks, signal troughs or signal patterns. These detections are used for two purposes: to perform matching with sensory identifiers of other devices and as a trigger to select a time interval of audio samples at predetermined relative temporal position with respect to a time point of the detected sensory identifier.

When the matching of the detected sensory identifiers produces the identification of a pair or group of devices with matching sensory identifiers, a comparison between the audio samples of the devices is used, for the time intervals that were selected by the triggers of those devices. If the audio in these time intervals also matches, it is decided that the devices are proximate and actions are enabled that depend on this decision. As a result, different forms of comparison may be used for matching sensory identifiers and for comparing audio in the selected time intervals respectively. The comparison of audio in the selected time intervals can be performed without first detecting selected identifiers in the audio. A more time consuming type of comparison, using significantly more data, may be used for comparison of audio in the selected time intervals than for

finding matching identifiers. A more robust type of comparison may be used that involves suppression of incidental differences such differences between the devices in amplitude sensitivity or distribution of spectral sensitivity.

It is noted that any terms used in this document should not be construed so as to limit the scope of the present invention. In particular, the words "comprise(s)" and "comprising" are not meant to exclude any elements not specifically stated. Single (circuit) elements may be substituted with multiple (circuit) elements or with their equivalents.

It will be understood by those skilled in the art that the present invention is not limited to the embodiments illustrated above and that many modifications and additions may be made without departing from the scope of the invention as defined in the appending claims.

The invention claimed is:

1. A method of identifying proximate devices, the method comprising the steps of:

recording, in each device, an audio signal so as to produce respective audio samples,

detecting, in each device, a detected sensory identifier, and

transmitting the respective audio samples from one or more of the devices to a server or another of the devices, wherein the server or device receiving the respective audio samples comprises a processor, a memory and software configured to perform the remaining steps of

using said detected sensory identifier for matching the devices, using said detecting as a trigger to select a common time interval of audio samples at a predetermined relative temporal position offset with respect to a time point of the detected sensory identifier;

comparing respective audio samples recorded during the common time interval by different ones of the devices with each other, and

deciding that the devices are proximate only if the respective audio samples are sufficiently similar.

2. The method according to claim **1**, wherein the common time interval starts prior to the time point of the detected sensory identifier.

3. The method according to claim **2**, wherein the common time interval ends prior to the time point of the detected sensory identifier.

4. The method according to claim **1**, comprising the further step of comparing the audio samples during an additional common time interval also defined relative to the time point of the detected sensory identifier, wherein the additional common time interval starts after the time point of the detected sensory identifier.

5. The method according to claim**1**, wherein the sensory identifier is part of the audio signal.

6. The method according to claim **1**, wherein the audio samples are represented by characteristic parameters.

7. The method according to claim **1**, wherein the step of comparing the audio samples is carried out in a server.

8. The method according to claim **1**, wherein the step of comparing the audio samples is carried out in at least one of the devices.

9. The method according to claim **1**, wherein the step of using said sensory identifier for matching the devices comprises exchanging messages between the devices, at least some of said messages containing a representation of the sensory identifier.

10. The method according to claim **9**, wherein the step of using said sensory identifier for matching the devices comprises comparing representations of the sensory identifier and deciding that a match exists if the representations are sufficiently similar.

11. The method according to claim **1**, wherein the sensory identifier is produced by tapping the devices together.

12. The method according to claim **1**, wherein the devices are mobile consumer devices, for example mobile telephone devices.

13. The method according to claim **12**, wherein the devices are arranged for playing games.

14. The method of claim **1**, comprising

using the detected sensory identifiers detected in the devices to pair the devices;

using said comparing of the respective audio samples recorded by different ones of the devices as a check of a match of the sensory identifiers.

15. The method of claim **1**, comprising

each device which detected the sensory identifier making a representation of the detected sensory identifier,

determining whether the representations of the detected sensory identifier made by the devices match;

using said comparing of the respective audio samples recorded by different ones of the devices as a check of the match of the sensory identifiers.

16. The method of claim **1**, comprising using said deciding to enable an exchange of information between the devices.

17. A tangible computer readable medium comprising a computer program for identifying proximate devices, by means of the steps of

recording, in each device, an audio signal so as to produce respective audio samples,

detecting, in each device, a detected sensory identifier,

using said detected sensory identifier for matching the devices, using said detecting as a trigger to select a common time interval of audio samples at a predetermined relative temporal position offset with respect to a time point of the detected sensory identifier;

comparing respective audio samples recorded during the common time interval by different ones of the devices with each other, and

deciding that the devices are proximate only if the respective audio samples are sufficiently similar,

wherein a processor, a memory and a computer software are configured in one or more of the devices or in a server to perform the matching, comparing and deciding steps.

18. A device for identifying proximate devices, the device comprising:

a recording unit for recording an audio signal so as to produce respective audio samples,

a detection unit for detecting a detected sensory identifier and for using said detecting as a trigger to select a common time interval of audio samples at a predetermined relative temporal position offset with respect to a time point of the detected sensory identifier,

a matching unit for matching representations of the detected sensory identifiers detected by the devices,

a comparison unit for comparing the audio samples recorded during the a common time interval by different ones of the devices with each other, and

a decision unit for deciding that the devices are proximate only if the respective audio samples are sufficiently similar.

19. The device according to claim **18**, wherein the comparison unit is arranged such that the common time interval starts prior to the time point of the detected sensory identifier.

20. The device according to claim **18**, wherein the comparison unit is further arranged for comparing the audio samples during an additional common time interval which starts after the time point of the detected sensory identifier.

21. The device according to claim **18**, further arranged for producing an audio signal, preferably in response to detecting of the sensory identifier.

22. The device according to claim **21**, wherein the audio signal produced is at least in part ultrasonic.

23. The device according to claim **18**, which is a mobile consumer device, for example a mobile telephone device.

24. The device according to claim **23**, which device is arranged for playing games.

25. The device of claim **18**, wherein the matching unit is configured to use said deciding as a check of a match of the sensory identifiers.

26. The device of claim **18**, wherein said decision unit is configured to enable an exchange of information between the devices if the respective audio samples are sufficiently similar.

* * * * *

EXHIBIT 6

US008782282B1

(12) **United States Patent**
Zhou et al.

(10) **Patent No.: US 8,782,282 B1**
(45) **Date of Patent: Jul. 15, 2014**

(54) **NETWORK MANAGEMENT SYSTEM**

(75) Inventors: **Eileen Zhou**, San Jose, CA (US); **Roger Liu**, San Jose, CA (US); **Vijoy Choyi**, Santa Clara, CA (US); **Moshe Itah**, Palo Alto, CA (US); **John Z. Yu**, Santa Clara, CA (US)

(73) Assignee: **Brixham Solutions Ltd.**, Tortola (VG)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 2172 days.

(21) Appl. No.: **10/742,573**

(22) Filed: **Dec. 19, 2003**

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.**
USPC **709/246**; 709/203; 719/320

(58) **Field of Classification Search**
USPC 709/203, 246; 719/320
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,812,768	A *	9/1998	Page et al.	709/228
7,111,077	B1 *	9/2006	Starkovich et al.	709/246
7,325,062	B2 *	1/2008	Hamilton et al.	709/226
2003/0093403	A1 *	5/2003	Upton	707/1
2003/0177240	A1 *	9/2003	Gulko et al.	709/226
2004/0040030	A1 *	2/2004	Fenton et al.	719/320
2004/0107277	A1 *	6/2004	Levesque et al.	709/223
2004/0249644	A1 *	12/2004	Schiefer et al.	705/1
2005/0108387	A1 *	5/2005	Li et al.	709/224

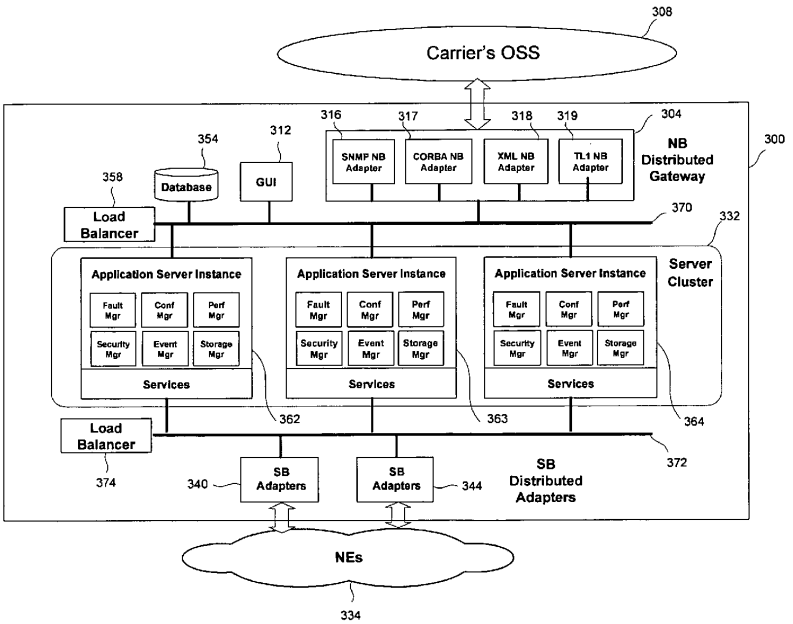
* cited by examiner

Primary Examiner — Glenford Madamba
(74) *Attorney, Agent, or Firm* — Amin, Turocy & Watson, LLP

(57) **ABSTRACT**

A technique for communicating in a network management system is disclosed. The technique includes transferring information from an application server to a gateway that processes the information, transferring the gateway processed information from the gateway to a plurality of adapters associated with the gateway that process the gateway processed information, and transferring the adapter processed information from one of the adapters to a client.

22 Claims, 6 Drawing Sheets



NB distributed gateway, SB distributed adapters, server cluster, & load balancers

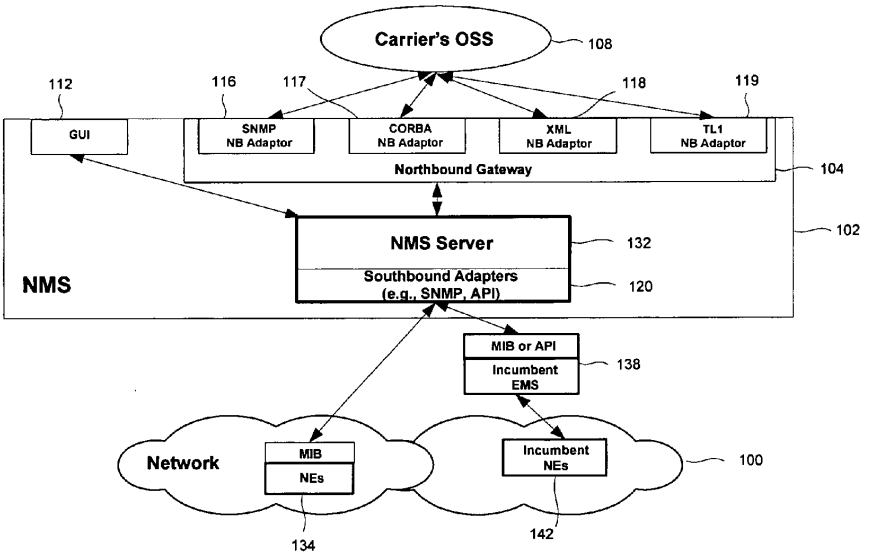


FIG. 1: An overall network management view

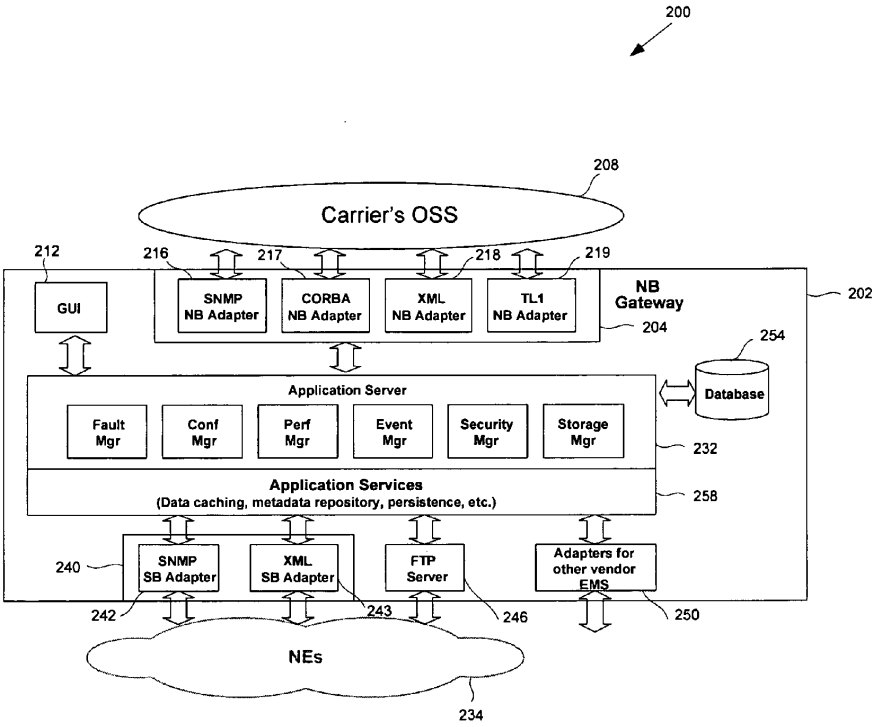


FIG. 2: NMS functional block diagram

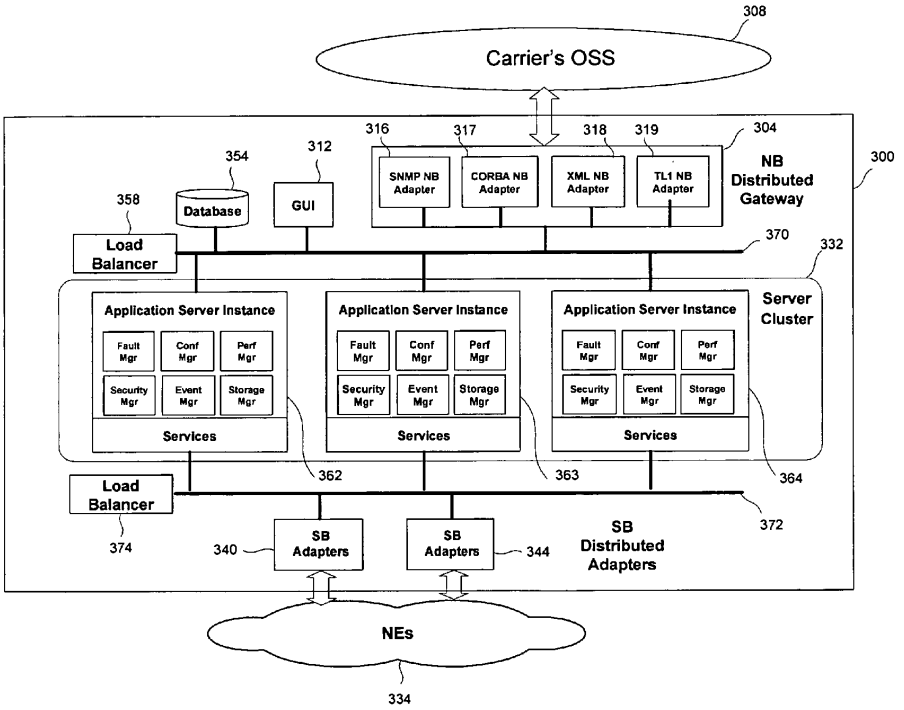


FIG. 3A: NB distributed gateway, SB distributed adapters, server cluster, & load balancers

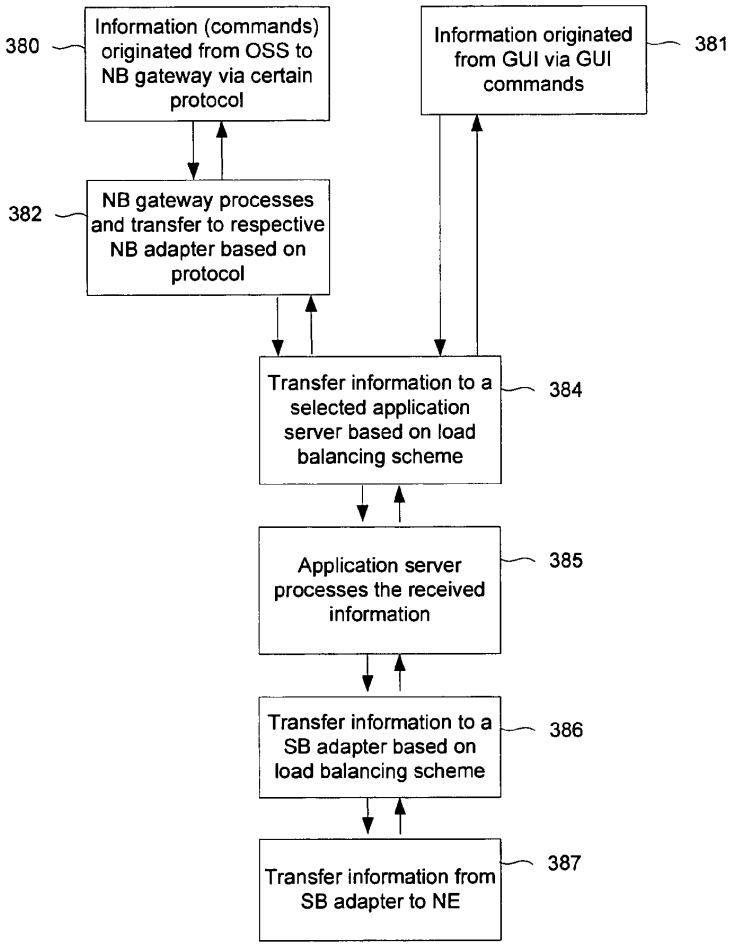


FIG. 3B: Downlink information flow (OSS or GUI as originator)

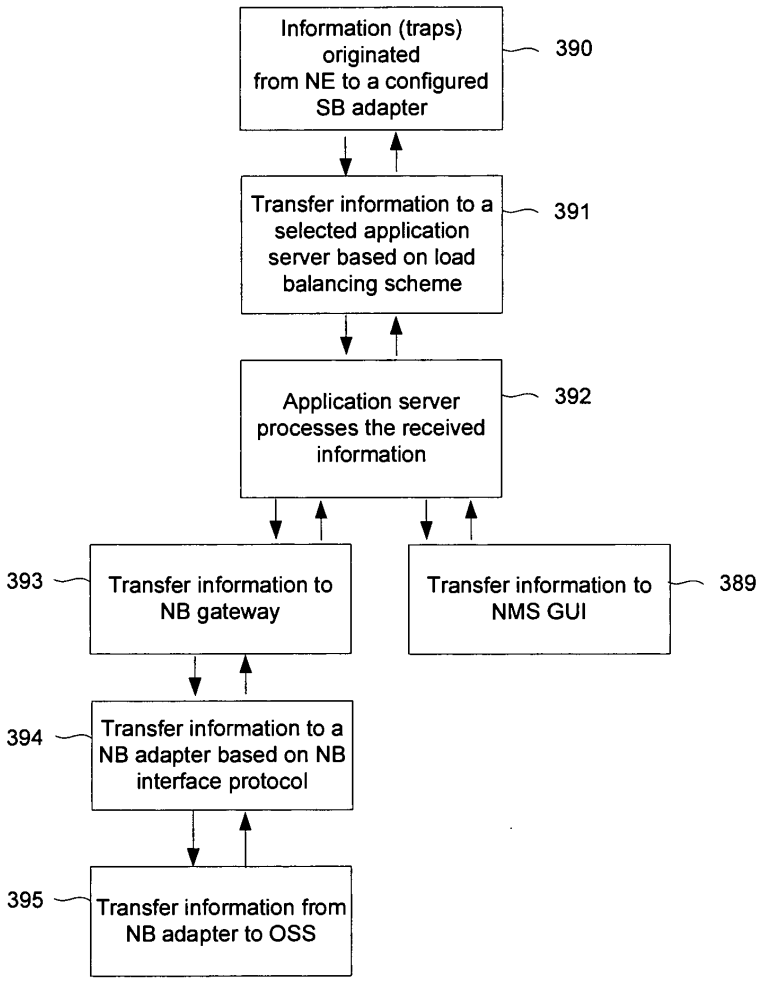


FIG. 3C: Uplink information flow (NE as originator)

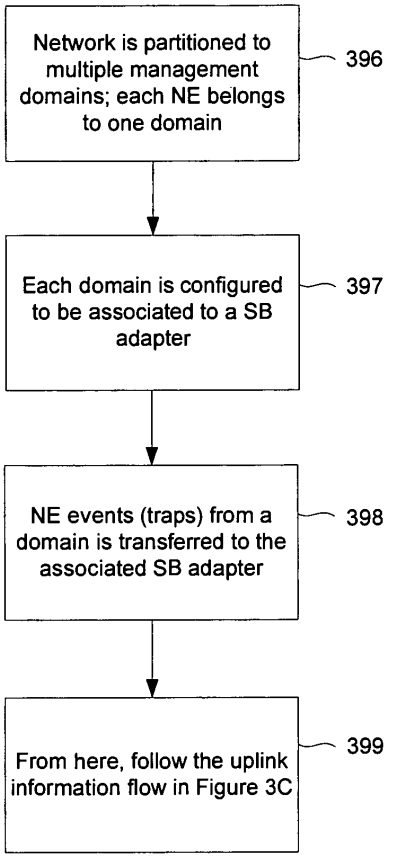


FIG. 3D: Uplink information flow with network partitioning

NETWORK MANAGEMENT SYSTEM

FIELD OF THE INVENTION

The present invention relates generally to communications networks. More specifically, a network management system is disclosed.

BACKGROUND OF THE INVENTION

Network management systems (NMS) facilitate the exchange of management information between Network Elements (NEs) and network administrators' Operations Support Systems (OSS). Network administrators can use an NMS to install and configure NEs, provision services, manage network performance, find and solve network problems, and plan for network growth. Network management systems today face many challenges as networks grow in size. Network management systems need to be scalable. For example, thousands of events can be generated when a fault is discovered at a network element. In a large network, a network management system can easily be overwhelmed. It would be desirable to develop a network management system that is robust, scalable, and capable of efficiently handling a growing network.

BRIEF DESCRIPTION OF THE DRAWINGS

Various embodiments of the invention are disclosed in the following detailed description and the accompanying drawings.

FIG. **1** is an overall view of a carrier's OSS **108**, an NMS **102**, and a network **100** including NEs **134** and **142**.

FIG. **2** is a functional block diagram illustrating a network management system **202** in a network **200**.

FIG. **3**A is a diagram illustrating NMS **300** using application server cluster and load balancing between NB gateway (adapters) and NMS server instances, and between SB adapters and NMS server instances.

FIG. **3**B is a flowchart illustrating a downlink information flow in which OSS **308** or NMS GUI **312** is the originator of the information while NE **334** or application server (**362**, or **363**, or **364**) is the recipient of the information.

FIG. **3**C is a flowchart illustrating an information flow in which NE **334** is the originator of the information while OSS **308** or NMS GUI **312** is the recipient of the information.

FIG. **3**D is a flowchart illustrating a technique for transferring information (e.g., traps) from an NE in a partitioned network to server cluster **332**.

DETAILED DESCRIPTION

The invention can be implemented in numerous ways, including as a process, an apparatus, a system, a composition of matter, a computer readable medium such as a computer readable storage medium or a computer network wherein program instructions are sent over optical or electronic communication links. In this specification, these implementations, or any other form that the invention may take, may be referred to as techniques. In general, the order of the steps of disclosed processes may be altered within the scope of the invention.

A detailed description of one or more embodiments of the invention is provided below along with accompanying figures that illustrate the principles of the invention. The invention is described in connection with such embodiments, but the invention is not limited to any embodiment. The scope of the

invention is limited only by the claims and the invention encompasses numerous alternatives, modifications and equivalents. Numerous specific details are set forth in the following description in order to provide a thorough understanding of the invention. These details are provided for the purpose of example and invention may be practiced according to the claims without some or all of these specific details. For the purpose of clarity, technical material that is known in the technical fields related to the invention has not been described in detail so that the invention is not unnecessarily obscured.

A scalable network management system (NMS) is disclosed. In some embodiments, the NMS includes a so-called northbound (NB) gateway, which includes various NB adapters, to facilitate transferring and translating information between the NMS server, or servers in a server cluster and an OSS from the network administrator, or carrier. The NB gateway design may be distributed so that multiple NB gateways, each of which includes a certain number of NB adapters, can be logically or physically distributed over different workstations. Information transferred from the gateway to the server cluster can be distributed among servers within the server cluster based on a load balancing scheme, to best utilize the NMS server's processing power. In the direction from NE to NMS (uplink), the managed network can be partitioned by configuring for each Network Element (NE) a so-called southbound (SB) adapter to handle information (or events) originating from that NE. SB adapters may be distributed so that SB adapters can be logically or physically distributed over different workstations. A load balancing scheme may be used to associate a SB adapter to a particular server or servers for processing power efficiency. In the direction from NMS to NE, the load balancing scheme may be used to distribute information from the NMS to a selected SB adapter to reach the destined NE.

FIG. **1** is an overall view of a carrier's OSS **108**, an NMS **102**, and a network **100** including NEs **134** and **142**. An NMS refers to a system responsible for managing a network, and facilitates the communication between a carrier's OSS and NEs in the network. In the embodiment illustrated, incumbent element management system (EMS) **138**, and incumbent NEs **142** refer to the existing management system and NEs, while NEs **134** refer to the newly introduced NEs for existing network growth. NMS **102** may be used to manage the existing NEs (via existing EMS **138**) and newly introduced NEs. A carrier may operate an OSS over the NMS to manage the entire network to provide services through the growing network. An NE, as used herein, includes any device or node on the network, such as a network switch. GUI **112** and NB gateway **104** are connected to NMS server **132** as clients. NB gateway **104** can be physically and logically separated from the NMS server. The NMS server may include a service cluster which includes multiple service instances. NB gateway **104** includes multiple NB adapters to process information to facilitate communications between NMS and OSS using different types of protocols. More specifically, NB gateway may include Simple Network Management Protocol (SNMP) NB adapter **116**, Common Object Request Broker Architecture (CORBA) NB adapter **117**, Extensible Markup Language (XML) NB adapter **118**, and Transaction Language 1 (TL1) NB adapter **119**.

NB devices, such as northbound adapters, include client side devices or network elements or devices that process information between an OSS and an NMS server. SB devices, such as SB adapter and API **120**, process information to facilitate communication between the NMS server and NEs, or between the NMS server and incumbent EMS **138**. North-

bound information includes information that travels in the direction from a network element to an OSS. Southbound information includes information that travels in the direction from an OSS to a network element.

NMS server **132** is connected to an MIB (Management Information Base) running on NEs **134** via SB SNMP/MIB adapter **120**. NMS server **132** is also connected to a MIB associated with incumbent EMS **138** via SB SNMP/MIB adapter **120**. NMS server **132** is also connected to an application program interface (API) associated with incumbent EMS **138** via API adapter **120**. Incumbent EMS **138** is connected to incumbent NEs **142**.

NMS **102** is shown to manage both NEs **134** and incumbent NEs **142**. NMS **102** can manage incumbent NEs **142** by communicating with incumbent EMS **138**. In general, NMS **102** can assume management of any incumbent network by connecting to an EMS associated with the incumbent network.

NMS server **132** communicates with adapters **116-119** via NB gateway **104**. Adapters **116-119** are software modules on NB gateway **104**. Accordingly, processing of common functions between adapters **116-119** can be shared. For example, an event sent from an NE to more than one NB adapter can be sent as one event to the NB gateway. Gateway **104** then processes the event and distributes the information appropriately to each NB adapter.

NMS **102** communicates in various protocols in different embodiments. For example:

- The SNMP protocol is used on its downlink (southbound) to communicate with switches or NEs. Command line interface (CLI) access to NEs through Telnet is also provided through the NMS in a cut-through mode;
- The SNMP protocol or a proprietary API is used on its downlink (southbound) to communicate with existing EMS managing incumbent NEs in installed networks;
- XML, CORBA, TL1, and SNMP protocols are used on its uplink (northbound) to communicate with external carriers' OSS's and NMS's.

In some embodiments, most of the time OSS **108** accesses NMS **102** for information and operation support. At other times, OSS **108** directly accesses NEs **134** or NEs **142** using the SNMP protocol. In some embodiments, hundreds or thousands of NEs are managed by an NMS. At the same time, an NE can be managed by multiple NMS's.

FIG. **2** is a functional block diagram illustrating a network management system **202** in a network **200**. Network **200** includes OSS **208**, NMS **202** and NEs **234**. NMS **202** includes NB gateway **204**, GUI **212**, application server **232**, NMS database **254**, SB adapters **240**, file transfer protocol (FTP) server **246**, and adapter **250** to communicate with external (incumbent) EMS's. NB gateway **204** includes SNMP adapter **216**, CORBA adapter **217**, XML adapter **218**, and TL1 adapter **219**.

OSS **208** is connected to adapters **216-219**. GUI **212**, NB gateway **204**, and database **254** are connected to application server **232**. NB gateway **204**, application server **232**, GUI **212**, database **254**, SB adapters **240**, FTP server **246**, and adapter **250** can be configured as physically or logically separate elements. For example, SB adapters **240**, FTP server **246**, or adapter **250** may be included on the same physical machine as server **232**. Application server **232** is connected to NEs **234** via SB adapters **240**. Application server **232** is also connected to adapter **250** and FTP server **246**, which is connected to NEs **234**. Application server **232** may include multiple independent application servers to form a server cluster. A load balancing scheme may be designed to associate NB adapters with application servers, and to associate SB adapters with application servers, an embodiment discussed further in conjunction with FIG. **3**.

NB gateway **204** relays management information and operations between server **232** and an external OSS **208**. Information includes data, messages, events, or other information that can be communicated in NMS **202**. For example, NB gateway **204** can provide network topology and configuration retrieval and provisioning services, and facilitate the integration with OSS **208** for fault management and flow through provisioning. External OSS **208** can communicate with NB gateway **204** via the SNMP, TL1, CORBA or XML protocol as NB interfaces. In some embodiments, NB gateway **204** includes a protocol adapter for each type of protocol, to convert messages of a specific protocol into a common message type, and vice versa. The common message type may be a proprietary message type. NB gateway **204** can communicate with server **232** through RMI interfaces. With NB gateway **232**, many functions, such as trap detector/listener, can be efficiently designed to support all NB interfaces with various protocols.

Information is transferred from application server **232** to NB gateway **204** which processes and distributes the information appropriately to adapters **216-219**. Processes common to adapters **216-219** can be performed prior to transferring the information to adapters **216-219**. NB gateway **204** may include multiple NB gateways, each of which includes a set of NB adapters. The information is then transferred from adapters **216-219** to OSS **208** (**292**).

Adapters include software modules which can be standalone Java services that run on separate Java Virtual Machines (JVM). The modules can connect to an external system using various protocols and connect back to NMS server **232** via an RMI interface. NMS **202** is shown to include SNMP, CORBA, XML, and TL1 NB adapters. NMS **202** can include any number of NB adapters each associated with any protocol. The following are examples of NB adapter communication within network **200**.

SNMP NB adapter **216** is the SNMP interface between external OSS **208** and NMS **202**. NMS **202** acts as an SNMP agent. In some embodiments, OSS **208** sends SNMP Set, Get, and GetBulk commands to SNMP NB adapter **216**. SNMP NB adapter **216** translates the messages into native Java messages and sends them to application server **232**. Application server **232** executes the commands and returns the responses to SNMP NB adapter **216**. SNMP NB adapter **216** then converts the responses into SNMP messages and sends the messages back to external OSS **208**.

An OSS that wishes to receive traps from NMS **202** can register itself through SNMP NB adapter **216** via the SNMP Set command. SNMP NB adapter **216** will then forward trap messages from application server **232** to the registered OSS. SNMP NB adapter **216** can also translate traps from SNMP v1 format to SNMP v2 format, and vice versa.

XML NB adapter **218** communicates with an external system using XML data. XML NB adapter **218** converts an XML message to a native Java message, and vice versa. In some embodiments, XML NB adapter **218** includes two adapters: XML command adapter and XML event adapter.

The XML command adapter receives a command from application server **232** and converts it to XML data. Then it creates a TCP/IP socket connection with the corresponding external system, and sends the XML data to the external system over the output stream of the connection. The XML command adapter then waits for the response on the input

stream of the connection. It converts XML data received in response to java.util.Map and returns this map to the invoker of the command.

Application managers that wish to receive events from external systems can register with the XML event adapter. The XML event adapter receives XML messages from external systems over a TCP/IP socket connection and translates the messages into name value maps. The XML event adapter then sends the name value maps to each registered application manager.

TL1 NB adapter **219** communicates with an external system using the TL1 protocol. The TL1 NB adapter converts TL1 commands and autonomous messages to a native Java message, and vice versa.

CORBA NB adapter **217** provides a CORBA interface to external OSS's. CORBA NB adapter **217** receives CORBA requests from external OSS's, translates the requests into a Java native message format and sends the messages to application server **232**. Application server **232** performs the requested operations and sends the response back to CORBA NB adapter **217**. CORBA NB adapter **217** then translates the response into CORBA messages and sends them to the invoking external OSS.

Application server **232** includes support for various management services, including the typical carrier-class fault-management, configuration, accounting, performance, and security (FCAPS) functions via Configuration Manager, Fault Manager, Performance Manager, Event Log Manager and Security Manager. For example:

Configuration Manager: A configuration manager is responsible for providing configuration services, including discovering, configuring, and provisioning managed NEs in the network. It provides support for defining and manipulating configuration of network physical inventory, facilities, and connections. It also supports the diagnostic testing of the network elements.

Fault Manager: A fault manager is responsible for monitoring fault conditions and notifying the OSS and other management systems. It collects fault notifications from NEs, analyzes the notifications, and communicates the appropriate information to the OSS and other management systems.

Performance Manager: A performance manager allows users to proactively manage the network's health and performance. At a high level, the Performance Manager can be divided into two functions: collecting performance data from NEs, and presenting performance information to a GUI client or other NB performance monitoring (PM) systems. PM data collectors can collect performance data from NEs using various collection schemes and protocols, then either store the data in an NMS database or directly pass the data to requesting parties, such as a GUI client or an NB third party PM system.

Event Log Manager: An event log manager collects and records various events from the network. Some examples of an event include an SNMP trap from an NE, periodic performance monitoring (PM) data from an NE, an alarm from an NE or other network device, a security audit log from a server, an SNMP command sent from the NMS to an NE, and the NMS shutting down. The event log manager is both an event recorder and an event publisher.

Security Manager: A security manager is a service manager that maintains information about users and their access rights to operations and system resources. It is also responsible for security auditing.

Storage Manager: A storage manager is responsible for managing NMS storage database tables and alarms/events files.

In some embodiments, NMS **202** follows a typical Java 2 Platform Enterprise Edition (J2EE)-based three-tier distributed model: a Java GUI front end tier, a J2EE Application Server tier, and a database tier which can be any Java Database Connectivity (JDBC) compliant database. Other distribution models and protocols are used in various other embodiments. Specifically, the server tier may include a set of backend server modules running in a J2EE Application Server environment, which provide the actual execution of the management functionality. In this embodiment, a complete server configuration includes a J2EE application server **232** with all the Enterprise Java Beans (EJB) running on it, a set of SB protocol adapters **240**, an integrated FTP server **246**, and a data services layer **258** which provides caching, metadata repository, and data persistency services.

All NMS management services, such as configuration management, fault management, performance management, security management, event management, and other operation support functionality, may be implemented as EJB or as remote method invocation (RMI) Objects working in conjunction with EJBs. The adapters can run on a separate JVM and communicate back to the EJBs or RMI servers via RMI interfaces. The application server can communicate with external systems using various protocols, including SNMP, CORBA, XML, TL1, and other proprietary interfaces. The FTP server can facilitate large data transfers between the NMS and other external systems.

NMS applications can be implemented as J2EE applications. Bean managed data persistency can be used to overcome the limitation of the default J2EE container managed persistency design, which might be more suitable for e-commerce applications than for OSS/NMS types of application.

NMS **202** is shown to include SNMP and XML SB adapters. NMS **202** can include any number of southbound adapters each associated with any protocol. The following are examples of SB adapters used for communicating with the network **234**.

SNMP SB adapter **242** supports communication between application server **232** and SNMP agents running on NEs in network **234**. For example, a trap may be sent from an NE to application server **232**. In some embodiments, communication between SNMP SB adapter **242** and application server **232** is implemented using Java RMI. SNMP SB adapter **242** actually can include two adapters: SNMP command adapter and SNMP trap adapter. SNMP command adapter is an adapter that communicates using SNMP. It can be used to send SNMP commands such as SNMP Get, Set, GetNext, GetBulk from application server **232** to the agent running on each NE and return responses to application server **232**. The SNMP trap adapter receives traps from SNMP agents, converts them into native Java messages and sends the messages to application server **232**. The SNMP trap adapter may also be responsible for synchronizing the traps between NMS **202** and all the managed NEs. The trap synchronization process can either be initiated by the configuration manager or by the trap adapter itself. In some embodiments, each trap from a node contains a trap sequence number. The trap adapter uses this sequence number to ensure that it receives all the traps from all the managed NEs. If the trap adapter detects that traps are missing from a node, it may perform appropriate trap synchronization operations depending on the amount of missing traps.

In some embodiments, if a few traps are missed, the trap adapter requests the node to replay the missing traps. If a large

number of traps are missed, it requests the node to save the missing traps into an event log file and ftp the file over. If too many traps are detected missing and it is infeasible to retrieve all the missing traps, the trap adapter gives up retrieving missing traps, and asks the node to save currently active alarms into an active alarm log file and FTP the file over. In the meantime, the trap adapter will notify the configuration manager to start the configuration check process.

In some embodiments, the trap adapter parses and processes active alarm log files and event log files from NEs in network **234**. The active alarm log files and event log files contain the protocol description units (PDU) of the traps from the NE. The SNMP trap adapter extracts each PDU, converts it into a name value map, and then sends the name value map to all the registered application managers.

XML SB adapter **243** communicates with NEs in network **234** using XML data. XML SB adapter **243** converts an XML message to a native Java message, and vice versa.

NMS **202** may also include SB adapter **250** to communicate with another vendor's EMS through standard SNMP or any mutually agreed proprietary API. For example, a mutually agreed proprietary API may be based on proprietary information the other vendor uses for service provisioning, such as proprietary MIB tables or proprietary API functional calls. Accordingly, a carrier can perform end-to-end service provisioning across different vendors' NEs for network expansion. A new vendor's NEs can be inserted to a carrier's installed network with minimal interruption for end-to-end service deployment.

In some embodiments, NMS **202** includes an internal FTP server **246**. FTP server **246** provides file transfer functionality between NMS **202** and external systems. In one embodiment, application managers or adapters that expect to transfer files to or from an external system register with FTP server **246** and provide FTP server **246** with a callback interface. Upon successful registration, FTP server **246** returns a unique, one-time-use username and password to the requester. The requester then sends the username and password to the external system. The external system can then use the received username and password to connect to FTP server **246** and start a file transfer. After the file transfer is completed, FTP server **246** invokes a corresponding callback function.

FIG. **3**A is a diagram illustrating NMS **300** using application server cluster and load balancing between NB gateway (adapters) and NMS server instances, and between SB adapters and NMS server instances. NMS **300** includes NB gateway **304**, GUI **312**, server cluster **332**, database **354**, SB adapters **340** and **344**, load balancer **358** (with shared bus **370**) between NB gateway and NMS server cluster, and load balancer **374** (with shared bus **372**) between SB adapters and NMS server cluster. The carrier's OSS may be connected to NB adapters via NB gateway **304**. NB gateway **304** includes SNMP adapter **316**, CORBA adapter **317**, XML adapter **318**, and TL1 adapter **319**. NMS **300** communicates with NEs **334** via SB adapters **340** and **344**.

GUI **312**, NB gateway **304**, and database **254** are connected to server cluster **332** via bus **370** controlled by the load balancer **358**. Server cluster **332** is connected to SB adapters **340** and **344** via bus **372** controlled by the load balancer **374**.

Service cluster **332** and load balancers **358** and **374** are introduced for managing large or expanding networks. Server cluster **332** includes NMS servers **362-364**. Server cluster **332** may include any number of servers, or server modules (or instances). NMS **300** is dynamic and scalable as the network grows. High scalability is facilitated by using server clustering and distributed adapter architecture, with the load balancing scheme to connect the server cluster and NB and SB adapters in northbound and southbound directions respectively. Note that the clustering of server modules and distribution of adapter modules can be completely transparent to the GUI client. A GUI client can still view the whole network topology without knowledge of the existence of multiple servers and adapters.

Servers **362-364** may be separate physical or logical server modules, or a combination of physically or logically separated server modules. In some embodiments, NMS **300** uses J2EE application server architecture in which each server may be an independent server instance in a J2EE application server. Accordingly, multiple instances of such application servers can run simultaneously to distribute a workload and protect each other. Each server instance by itself is a complete functioning NMS server that runs independently, and can be run on separate physical machines. Each server instance can be run on the same physical machine or distributed among several physical machines. These NMS server instances work together, forming the "server cluster".

Specifically, each server instance can be associated with the following software modules: GUI **312** and NB gateway **304** via load balancer **358** in the northbound direction, and SB adapters **340** and **344** via load balancer **374** in southbound direction. In some embodiments, there are multiple NB gateways, each of which includes a set of NB adapters performing a specific set of functions. Additional physically or logically separate NB gateways can be added as the network grows. When a first request comes from a functional entity, such as NB gateway, or GUI, or database, load balancer **358** is consulted to locate a server instance according to an algorithm based on certain criterions (e.g., selecting the lightest loaded server at that time), and then communication is established between the functional entity and the selected server instance. Similarly, when a first request comes from a service instance going southbound, the load balancer **374** is consulted to locate a SB adapter according to a method based on certain criteria, and then the communication is established between the server instance and the selected SB adapter.

FIG. **3**B is a flowchart illustrating a downlink information flow in which OSS **308** or NMS GUI **312** is the originator of the information while NE **334** or application server (**362**, or **363**, or **364**) is the recipient of the information. When the information is originated from OSS (**380**), it is transferred via a certain protocol (e.g., SNMP, CORBA, XML, or TL1) to NB gateway **304**. The NB gateway processes the information and then transfers it to an associated NB adapter based on the received protocol type (**382**). Alternatively, the information could be originated from GUI **312** where, for example, a user could enter a command on GUI **312** (**381**). The arriving information is then transferred to a selected server or servers according to load balancer **358** (**384**). Load balancer **358** may be a software module physically located on any device in NMS **300**. Load balancer **358** may use any load balancing algorithm that is appropriate. For example, load balancer **358** may use a round robin scheme, or distribute the load based on CPU usage on each server or servers. In some embodiments, from then on, all information from NB gateway **304** (or GUI **312**) is passed to the selected server for processing (**385**). Alternatively, the selected server may convert the format of the received information to a new format at particular times. The information is then transferred from the server to a selected SB adapter according to load balancer **374** (**386**). Load balancer **374** may use any load balancing algorithm that is appropriate. The information is then transferred from the SB adapter to the destined NE in network **334** (**387**). The response from the NE or NMS server follows the reverse information flow to the OSS or GUI.

FIG. **3**C is a flowchart illustrating an information flow in which NE **334** is the originator of the information while OSS **308** or NMS GUI **312** is the recipient of the information. Management information (e.g., traps/alarms) is originated from an NE in network **334** (**390**). The information is transferred to a pre-configured SB adapter. The arriving information at the SB adapter is then transferred to a selected server or servers according to load balancer **374** (**391**). Load balancer **374** may be a software module physically located on any device in NMS **300**. Load balancer **374** may use any load balancing algorithm that is appropriate. For example, load balancer **374** may use a round robin scheme, or distribute the load based on CPU usage on each server or servers. In some embodiments, from then on, all information from the SB adapter is passed to the selected server for processing (**392**). Alternatively, the selected server may convert the format of the received information to new format at particular times. Depending on the destination, the information is then transferred from the server to a selected NB gateway (**393**), or the NMS GUI (**394**). When the information arrives at the NB gateway, it is transferred to a registered NB adapter based on the NB interface (I/F) protocol for communicating with the OSS (**395**). The information is then transferred from the NB adapter to the OSS (**396**).

The NMS architecture presented can provide multiple, significant benefits with the design of NB distributed gateway (NB adapters), distributed SB adapters, NMS application server clusters, and load balancing schemes to inter-connect between NB gateway and application server(s), and between SB adapters and application server(s). Three such examples follow.

First, the NB gateway can serve as a single point for receiving different types of information (e.g., network events such as traps) from NEs or the NMS server for distributing to registered NB adapters, and then to the OSS. A NB gateway-based architecture allows the information common to different NB adapters to be sent to the NB gateway just once, instead of multiple copies, and then the NB gateway multicasts the information to multiple destined NB adapters. With this architecture of an NB gateway decoupled from NMS application servers, multiple NB gateways, each of which includes a certain number of NB adapters, can be designed and distributed on different workstations for scalability and performance optimization. In one embodiment, the NB gateways running on different workstations communicate with each other via RMI transporting over TCP/IP.

Secondly, as a network grows to an increasing large scale, more processing power in the NMS servers may be needed for managing the network. With this server cluster-based NMS architecture, new NMS server instances can be incrementally started and added into server cluster **332** without affecting existing operations. The server cluster can be logically and physically distributed, so servers in a cluster can run on different workstations for increasing processing power. The load balancer allows the management work load to be distributed among all server instances in any desired fashion, which results in maximal efficiency and flexibility for any management policy to be implemented as carriers wish. An instance of the server can also be shut down for any reason without interrupting the server functionality as a whole. When a server is shutdown, its associated software modules (NB gateway **304**, GUI **312**, or SB adapters **340** or **344**) automatically re-establish the association with another server instance (one of existing servers **362**-**364** or a standby server) based on certain criterion, such as selecting the lightest loaded server.

Thirdly, the distributed SB adapter-based NMS architecture can support a flexible network partitioning to manage different management domains based on the amount of management traffic in each domain in a large scale network. All NEs in a management domain are associated with a SB adapter to communicate with the NMS servers. That is, the events in each partitioned domain are monitored by a designated SB adapter. With the server cluster, server instances can work with multiple SB adapters simultaneously, allowing a distributed SB adapter infrastructure to support a large number of NEs or a geographically separated network topology. Each SB adapter can select an instance of service using the load balancer based on certain criterion (e.g., selecting the lightest loaded server) to establish the association between the SB adapter and server instance. SB adapter modules can also be separated from NMS server **332** and run on separate workstations. Thus the network can be logically or physically partitioned in different management domains by configuring for each NE a SB adapter to handle events from that NE.

FIG. **3**D is a flowchart illustrating a technique for transferring information (e.g., traps) from an NE in a partitioned network to server cluster **332**. More specifically, the network is partitioned to different management domains in which each NE belongs to one management domain (**396**). Each domain is associated with one SB adapter (**397**). The information that originated from the NEs in a management domain is transferred to a configured SB adapter (**398**). That is, a configured SB adapter is pre-selected to handle events from the NEs in a management domain. For example, an adapter can register its IP address in all NEs in a management domain so that all NEs send traps to that SB adapter. From then on, the information follows the uplink information flow in FIG. **3**C to be passed to a specific server instance and then to the OSS for processing (**399**). In some embodiments, the information is transferred to a designated application server or servers within cluster **332** wherein the designation was predetermined. That is, the SB adapter can be configured to communicate with certain registered server modules (instances). Because all the server instances are writing to the same database, the overall network view can be accessed from any partitioned region. In one embodiment, the SB adapters running on different workstations communicate with each other via RMI transporting over TCP/IP.

The NMS architecture presented can provide significant benefits of scalability and performance optimization. It can enable incremental addition of network management resources (such as server processing power, database memory) on an as needed basis without network operation interruption as the network grows. The NMS based on this architecture can truly serve carriers' large scale network management requirements in a most optimized way.

Although the foregoing embodiments have been described in some detail for purposes of clarity of understanding, the invention is not limited to the details provided. There are many alternative ways of implementing the invention. The disclosed embodiments are illustrative and not restrictive.

What is claimed is:

1. A method, comprising:

receiving, at a first application server instance selected from a plurality of application server instances based on a load balancing process, first adapter processed information from a first adapter, wherein the first adapter processed information comprises event information received by the first adapter from a network element and processed by the first adapter based on a first communication protocol;

processing, by the first application server instance, the first adapter processed information based on an event management service to produce application processed information;

sending, by the first application server instance, the application processed information to a gateway device, wherein the gateway device is one of a plurality of gateway devices respectively associated with the plurality of application server instances and is configured to transfer the application processed information to a second adapter of a plurality of second adapters configured to process the application processed information based on a second communication protocol to produce second adapter processed information and transfer the second adapter processed information to an operation support system device; and

in response to determining that the first application server instance has become disabled, facilitating establishing an association between the first adapter and a second application server instance of the plurality of application server instances and between the gateway device and the second application server instance.

2. The method of claim **1**, wherein the first application server instance is configured to execute on a separate physical machine from the first adapter.

3. A system, comprising:

a first application server instance configured to receive first adapter processed information from a first adapter process the first adapter processed information based on an event management service to yield application processed information, and send the application server processed information to a gateway device, wherein the first adapter processed information comprises event information from a network element that has been processed by the first adapter based on a first communication protocol; and

a load balancing component configured to select the first application server instance from a plurality of application server instances based on a load balancing process;

wherein the gateway device is one of a plurality of gateway devices respectively associated with the plurality of application server instances and is configured to transfer the application server processed information to a second adapter of a plurality of second adapters configured to process the application server processed information based on a second communication protocol to yield second adapter processed information, and send the second adapter processed information to an operation support system device, and

wherein, the first adapter and the gateway device are further configured to, in response to disablement of the first application server instance, establish an association with a second application server instance of the plurality of application server instances.

4. The method of claim **1**, further comprising converting, by the second adapter, a protocol specific message associated with the first application server instance to a formatted message associated with the second communication protocol.

5. The method of claim **1**, further comprising selecting the first application server instance based on a determination that the first application server instance has a lowest processing load of the plurality of application servers instances.

6. The method of claim **1**, wherein the first communication protocol and the second communication protocol comprise one or more communication protocols associated with at least one of extensible markup language, simple network management protocol, common object request broker architecture, or transaction language 1.

7. The method of claim **1**, further comprising at least one of collecting, recording, or publishing the event information in accordance with the event management service for access by the operation support system device.

8. The method of claim **1** wherein the plurality of gateway devices comprise a plurality of physically or logically separated gateway devices.

9. The method of claim **1**, wherein the gateway device is a physically and logically separate machine from the first application server instance.

10. The method of claim **1**, further comprising converting, by the first adapter, a message associated with the first communication protocol to a protocol specific message associated with the first application server instance.

11. The method of claim **1**, further comprising processing, by the gateway device, a function associated with the application server processed information.

12. The method of claim **1**, wherein the event information relates to a configuration of the network element.

13. The method of claim **1**, further comprising executing the first adapter and the second adapter on one or more virtual machines.

14. The method of claim **1**, wherein the first adapter and the second adapter are configured to communicate with the first application server instance over a remote method invocation interface.

15. The system of claim **3**, wherein the event information is associated with a configuration of the network element.

16. The system of claim **3**, wherein the first application server instance comprises a performance manager component configured to collect performance data from a plurality of network elements respectively associated with a plurality of first adapters including the first adapter.

17. The system of claim **16**, wherein the performance manager component is further configured to deliver at least a subset of the performance data to one or more of the plurality of network elements.

18. The system of claim **3**, wherein the first communication protocol and the second communication protocol comprise one or more communication protocols associated with at least one of extensible markup language, simple network management protocol, common object request broker architecture, or transaction language 1.

19. The system of claim **3**, wherein at least one of the first adapter or the second adapter are executed on one or more virtual machines.

20. The system of claim **3**, wherein the first adapter and the second adapter are configured to communicate with the first application server instance over a remote method invocation interface.

21. The system of claim **3**, wherein the load balancing component is configured to select the first application server instance based on a determination that the first application server instance has a lowest processing load of the plurality of application server instances.

22. The system of claim **3**, wherein the plurality of gateway devices comprises a plurality of physically or logically separated gateway devices.

* * * * *

EXHIBIT 7

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/742,573	12/19/2003	Eileen Zhou	002.P056	4061

23623 7590 03/17/2014
AMIN, TUROCY & WATSON, LLP
127 Public Square
57th Floor, Key Tower
CLEVELAND, OH 44114

EXAMINER
MADAMBA, GLENFORD J

ART UNIT	PAPER NUMBER
2451	

NOTIFICATION DATE	DELIVERY MODE
03/17/2014	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

docket1@thepatentattorneys.com
hholmes@thepatentattorneys.com
swati@thepatentattorneys.com

PTOL-90A (Rev. 04/07)

UNITED STATES DEPARTMENT OF COMMERCE
U.S. Patent and Trademark Office
Address : COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450

APPLICATION NO./ CONTROL NO.	FILING DATE	FIRST NAMED INVENTOR / PATENT IN REEXAMINATION	ATTORNEY DOCKET NO.
10/742,573	19 December, 2003	ZHOU ET AL.	002.P056

		EXAMINER	
AMIN, TUROCY & WATSON, LLP 127 Public Square 57th Floor, Key Tower CLEVELAND, OH 44114		GLENFORD MADAMBA	
		ART UNIT	PAPER
		2451	20140310

DATE MAILED:

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner for Patents

This communication is being sent out to correct the Allowance Notice document dated 2/2/2014, and to correctly indicate the proper dependency of renumbered claim 10 (original claim 47) as being dependent upon renumbered claim 1 (original claim 18).

/GLENFORD MADAMBA/
Examiner, Art Unit 2451

PTO-90C (Rev.04-03)

Allowable Subject Matter

1. Original claims 1-32 are renumbered to 1-22. Claims 1-17, 19-25, 27, 29-32, 36-42, 44, 52 and 56-59 have been canceled, and new claims 60-64 have been added.

2. The Office deems Applicant's latest claim amendments persuasive to overcome the rejection of the claims over the applied prior art references and/or any other candidate prior art, and the claims are accordingly considered in condition for allowance **(MPEP § 1302.14)**.

In addition, Applicants remark and/or argument that the current prior art references do not sufficiently teach or disclose *all* of the recited limitations of the amended independent claims, and claim 1 in particular, including the recited feature of "in response to determining that the first application server instance has become disabled, facilitating establishing an association between the first adapter and a second application server instance of the plurality of application server instances and between the gateway device and the second application server instance…" as recited [Remarks: par 2, pg. 9 & par 6, pg. 13], is also considered persuasive by the Office.

EXAMINER'S AMENDMENT

(*Corrected*)

1. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To

ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

2. Authorization for this examiner's amendment was given in a telephone interview with Applicant's Representative (Brian Steed, Reg. No. 64,095) on 1/8/2014. The Office also notes that as part of the interview, Applicant officially declared and stated on the record that amended claims 18, 26, 28, 62 and 63 - which respectively recite in part "wherein the gateway device is one of a plurality of gateway devices …and *is configured to* transfer...", " wherein the first application server instance *is configured to* execute...", "a first application server instance *configured to* receive…", "wherein the first adapter and the second adapter are *configured to* communicate…", and/or "wherein the load balancing component is *configured to* select…" - are *not* intended to recite a 'means (or step) plus function' limitation that invokes 35 U.S.C. 112, sixth paragraph.

3. The application has been amended as follows:

<u>In the Claims:</u>

Please AMEND claim 18 as follows:

18. (Currently Amended) A method, comprising:

receiving, at a first application server instance [[of]] <u>selected from</u> a plurality of application server instances <u>based on a load balancing process</u>, first adapter processed

information from a first adapter, wherein the first adapter processed information comprises event information received by the first adapter from a network element and processed by the first adapter based on a first communication protocol;

processing, by the first application server instance, ~~is configured to process~~ the first adapter processed information based on an event management service to produce application processed information;

sending, by the first application server instance, the application processed information to a gateway device, wherein the gateway device is one of a plurality of gateway devices respectively associated with the plurality of application server instances and is configured to transfer the application processed information to a second adapter of a plurality of second adapters configured to process the application processed information based on a second communication protocol to produce second adapter processed information and transfer the second adapter processed information to an operation support system device; and

in response to determining that the first application server instance has become disabled, facilitating establishing an association between the first adapter and a second application server instance of the plurality of application server instances and between the gateway device and the second application server instance.

Please AMEND claim 24 as follows:

24. (Canceled) ~~(Currently Amended) The method of claim 18, further comprising selecting the first application server instance is selected for receipt of the first adapter processed information based on a load balancing scheme process~~.

Please AMEND claim 26 as follows:

26. (Currently Amended) The method of claim 18, wherein the first ~~adapter~~ application server instance is configured to execute on a separate physical machine from the first ~~application server instance~~ adapter.

Please AMEND claim 28 as follows:

28. (Currently Amended) A system, comprising:

a first application server instance~~, of a plurality of application server instances,~~ configured to receive first adapter processed information from a first adapter process the first adapter processed information based on an event management service to yield application processed information, and send the application server processed information to a gateway device, wherein the first adapter processed information comprises event information from a network element that has been processed by the first adapter based on a first communication protocol; and

a load balancing component configured to select the first application server instance from a plurality of application server instances based on a load balancing process;

wherein the gateway device is one of a plurality of gateway devices respectively associated with the plurality of application server instances and is configured to transfer the application server processed information to a second adapter of a plurality of second adapters configured to process the application server processed information based on a second communication protocol to yield second adapter processed information, and send the second adapter processed information to an operation support system device, and

wherein, the first adapter and the gateway device are further configured to, in response to disablement of the first application server instance, establish an association with a second application server instance of the plurality of application server instances.

Please AMEND claim 33 as follows:

33. (Currently Amended) The method of claim [[59]] 18, further comprising converting, by the second adapter, a protocol specific message associated with the first application server instance to a formatted message associated with the second communication protocol.

Please AMEND claim 34 as follows:

34. (Currently Amended) The method of claim [[24]] 18, ~~wherein the~~ further comprising selecting the first application server instance ~~comprises selecting~~ based on a determination that

the first application server instance has a lowest processing load of the plurality of application servers instances.

Please AMEND claim 35 as follows:

35. (Currently Amended) The method of claim [[59]] 18, wherein the first communication protocol and the second communication protocol comprise one or more communication protocols associated with at least one of extensible markup language, simple network management protocol, common object request broker architecture, or transaction language 1.

Please AMEND claim 43 as follows:

43. (Currently Amended) The method of claim [[58]] 18, further comprising at least one of collecting, recording, or publishing the event information in accordance with the event management service for access by [[an]] the operation support system device.

Please AMEND claim 47 as follows:

47. (Currently Amended) ~~(Previously Presented)~~ The method of claim [[58]] 18, further comprising converting, by the first adapter, a message associated with the first communication protocol to a protocol specific message associated with the first application server instance.

Please AMEND claim 49 as follows:

49. (Currently Amended) The method of claim [[58]] 18, wherein the event information relates to a configuration of the network element.

Please AMEND claim 53 as follows:

53. (Currently Amended) The system of claim 28, ~~wherein the first adapter processed information comprises event information received at the first adapter from a network element, and~~ wherein the event information is associated with a configuration of the network element.

Please AMEND claim 56 as follows:

56. (Canceled) ~~(New) The system of claim 28, wherein the first adapter processed information comprises event information from a first network element that has been processed by the first adapter based on a first communication protocol.~~

Please AMEND claim 57 as follows:

57. (Canceled) ~~(New) The system of claim 56, wherein the plurality of second adapters are configured to process the application server processed information based on a second communication protocol to yield second adapter processed information, and send the second adapter processed information to an operation support system device.~~

Please AMEND claim 58 as follows:

58. (Canceled) ~~(New) The method of claim 18, wherein the first adapter processed information comprises event information received at the first adapter from a network element and processed by the first adapter based on a first communication protocol.~~

Please AMEND claim 59 as follows:

59. (Canceled) (New) ~~The method of claim 58, wherein the plurality of second adapters are configured to process the application processed information based on a second communication protocol to produce second adapter processed information, and transfer the second adapter processed information to an operation support system device.~~

Please ADD new claim 60 as follows:

60. (New) The system of claim 28, wherein the first communication protocol and the second communication protocol comprise one or more communication protocols associated with at least one of extensible markup language, simple network management protocol, common object request broker architecture, or transaction language 1.

Please ADD new claim 61 as follows:

61. (New) The system of claim 28, wherein at least one of the first adapter or the second adapter are executed on one or more virtual machines.

Please ADD new claim 62 as follows:

62. (New) The system of claim 28, wherein the first adapter and the second adapter are configured to communicate with the first application server instance over a remote method invocation interface.

Please ADD new claim 63 as follows:

63. (New) The system of claim 28, wherein the load balancing component is configured to select the first application server instance based on a determination that the first application server instance has a lowest processing load of the plurality of application server instances.

Please ADD new claim 64 as follows:

64. (New) The system of claim 28, wherein the plurality of gateway devices comprises a plurality of physically or logically separated gateway devices.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Glenford Madamba whose telephone number is 571-272- 7989. The examiner can normally be reached on Monday-Friday 7:00AM-4: 30PM, first Fridays OFF.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John Follansbee can be reached on 571-272-3964. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from

either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only.

For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, .contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Glenford Madamba/

Examiner, Art Unit 2451

EXHIBIT 8

SHERIDAN ROSS pc attorneys at innovation

January 6, 2025

Mr. David A. Zapolsky
SVP, General Counsel & Secretary
Amazon.com, Inc.
P.O. Box 81226
Seattle, WA 98108-1226

Robert R. Brunelli
Shareholder
Direct: 303.863.2980
rbrunelli@sheridanross.com

Via email
davidz@amazon.com

Re: K.Mizra LLC

Dear Mr. Zapolsky:

As you may be aware, K.Mizra LLC ("K.Mizra") is a patent licensing company run by experienced management. The Company focuses on high value, high quality patents with a global reach. It owns patent portfolios originating with a wide array of inventors, including portfolios developed by well-known multinationals such as IBM, Intel, Rambus and others, as well as from research institutes such as Nederlandse Organisatie voor Toegespast Natuurwetenschappelijk Onderzoek. By focusing on high quality patents, K.Mizra provides a secondary market for inventors to recoup their research and development investments and to continue their innovations. K.Mizra offers licenses to its patents on reasonable terms and in this way plays an important part in the development of the technologies that improve all our lives.

K.Mizra is the owner by assignment of United States Patent Nos. 8,234,705 ("the '705 Patent") (Ex. 1) and 9,516,048 ("the '048 Patent") (Ex. 2). The '705 Patent, titled "Contagion Isolation and Inoculation," was duly and legally issued by the United States Patent and Trademark Office ("USPTO") and expires on November 25, 2029. The '048 Patent, titled "Contagion Isolation and Inoculation Via Quarantine," was issued by the USPTO December 6, 2016 and expires on September 27, 2025.

The technology described and claimed in these Patents was invented by Dr. James Roskind and Mr. Aaron Emigh, two prolific inventors and colleagues living in the same area who had similar interests in innovating computer-related technologies. Dr. Roskind has bachelors, masters, and doctorate degrees from MIT in both electrical engineering and computer science and is the named inventor of over 300 U.S. patents. He has worked for Netscape as the Chief Architect and as the Netcenter Security Architect, and was a co-founder for Infoseek, a company that was eventually acquired by Disney for $770 million. He was also a key developer of Google's "transport protocol" that provides the tech giant billions of dollars in value every year. Mr. Emigh graduated from the University of California, Santa Cruz with degrees in linguistics and computer and information sciences and is the named inventor of over 140 patents. Prior to working with Dr. Roskind, Mr. Emigh worked in various positions developing software, including working as a software manager, architect, and engineer for Unicom and working as a manager for the software development and technical marketing groups for Philips TriMedia. He has founded or co-founded many companies, including CommerceFlow, Inc., which was acquired by eBay for its technology that Mr. Emigh helped to develop.

In 2003, the inventors decided to create a business together, Radix Labs, LLC, which focused on developing intellectual property related to various computer technologies, including network

security technologies. The inventors focused on conceiving inventions they knew were needed then or soon would be needed in the computer networking industry and then on drafting patent applications to capture and protect their innovations. In September of 2004, the inventors filed the Provisional Application to which both Patents claim priority. The Provisional Application described technology that focused on securing a network against the threats to which it was exposed by the connections of additional end points to that network. The Provisional Application, and by natural extension the Patents, also focus on remedying identified threats and quarantining those threats to mitigate any damage to the secured network.

More specifically, '705 Patent explains that "[l]aptop and wireless computers and other mobile systems pose a threat to elements comprising and/or connected to a network service provider, enterprise, or other protected network to which they reconnect after a period of connection to one or more networks and/or systems that are not part of the service provider, enterprise, or other protected network. By roaming to unknown domains, such as the Internet, and/or connecting to such domains through public, wireless, and/or otherwise less secure access nodes, such mobile systems may become infected by computer viruses, worms, backdoors, and/or countless other threats and/or exploits and/or have unauthorized software installed; have software installed on the mobile system by an operator of the protected network for the protection of the mobile system and/or the protected network removed or altered without authorization and/or have configurations, settings, security data, and/or other data added, removed, and/or changed in authorized ways and/or by unauthorized person[s]." *See, e.g.*, Ex. 1 at 1:14–31. The solution provided by and embodied in the Asserted Patents to these problems was a complete break from the conventional network access control solutions then in use, required unique hardware to be properly implemented and was then, as today, patent-subject eligible, highly valuable, novel and non-obvious technology.

After the Patents issued, Dr. Roskind and Mr. Emigh recouped their research and development investment by selling their rights thereto, continuing in their individual technology development pursuits. K.Mizra ultimately acquired these Patents and licensed them to many of the who's-who of the tech world. Some of the accused infringers chose to test the validity of the Patents prior to settling with K.Mizra. For instance, a few accused infringers previously sought *inter partes* review ("IPR") by the Patent Trial and Appeal Board ("PTAB") of the Patents. A Final Written Decision ("Decision") in the '705 Patent IPR found that the petitioners had not shown, by a preponderance of the evidence, that the asserted claims were unpatentable. Based on the '705 Patent IPR Decision, the similar '048 Patent IPR was not even instituted. The '705 Patent IPR Decision was appealed to the Court of Appeals for the Federal Circuit ("CAFC"), resulting in a procedurally focused remand back to the PTAB. That appeal is now concluded and the parties to the IPR have recently filed with the Board a joint motion to dismiss.

Amazon.com, Inc. is in the business of providing VPN-focused security products and services, among others, to its customers in the US and elsewhere. Included within these offerings is Amazon AWS Verified Access ("Involved Product"). As can be seen through review of the attached Notice of Technology Use chart (Ex. 3), the Involved Product meets all the limitations of several of the Patents' claims. Amazon.com, Inc.'s activities now need to be regularized, with K.Mizra remaining ready, willing and able to provide Amazon.com, Inc. a commercially reasonable

license to its various patented technologies, including the Patents. If you are interested in discussing purchasing such a license, please contact me directly.

Thank you for your attention to this matter.

Very truly yours,

SHERIDAN ROSS P.C.

Robert R. Brunelli
RRB/lrb
Enclosures
C: K.Mizra LLC

EXHIBIT 1

US008234705B1

(12) **United States Patent**
Roskind et al.

(10) **Patent No.: US 8,234,705 B1**
(45) **Date of Patent: Jul. 31, 2012**

(54) **CONTAGION ISOLATION AND INOCULATION**

(75) Inventors: **James A. Roskind**, Redwood City, CA (US); **Aaron T. Emigh**, Incline Village, NV (US)

(73) Assignee: **Radix Holdings, LLC**, Incline Village, NV (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1520 days.

(21) Appl. No.: **11/237,003**

(22) Filed: **Sep. 27, 2005**

Related U.S. Application Data

(60) Provisional application No. 60/613,909, filed on Sep. 27, 2004.

(51) **Int. Cl.**
G06F 11/00 (2006.01)
G06F 12/14 (2006.01)
G06F 12/16 (2006.01)
G08B 23/00 (2006.01)

(52) **U.S. Cl.** **726/22**; 726/23; 726/24; 726/25

(58) **Field of Classification Search** 726/11, 726/14, 22–25
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,823,479	B1 *	11/2004	McElhaney et al.	714/43
6,996,845	B1 *	2/2006	Hurst et al.	726/25
7,263,720	B2 *	8/2007	Silvester	726/22
7,287,278	B2 *	10/2007	Liang	726/22
7,424,610	B2 *	9/2008	Ranganathan	713/164
2002/0083328	A1 *	6/2002	Riordan	713/183
2003/0188173	A1 *	10/2003	Zimmer et al.	713/189
2003/0188194	A1 *	10/2003	Currie et al.	713/201
2005/0018618	A1 *	1/2005	Mualem et al.	370/252
2005/0033987	A1 *	2/2005	Yan et al.	713/201
2005/0131997	A1 *	6/2005	Lewis et al.	709/203
2006/0053426	A1 *	3/2006	Dive-Reclus et al.	719/328

OTHER PUBLICATIONS

Bohra, A.; Neamtiu, I.; Gallard, P.; Sultan, F.; Iftode, L.; Remote repair of operating system state using Backdoors; Autonomic Computing, 2004. Proceedings. International Conference on May 17-18, 2004 pp. 256-263.*

Mark V. Glover, Integrating a Trusted Computing Base Extension Server and Secure Session Server into the LINUX Operating System, Sep. 2001, Naval Postgraduate School.*

Pearson et al., Trusted Computing Platform: TCPA Technology in Context, 2003, Hewlett-Packard Books: Patricia Pekary.*

OLS: Linux and trusted computing, Jul. 22, 2005, http://lwn.net/Articles/144681/ (as lined on Jul. 15, 2010).*

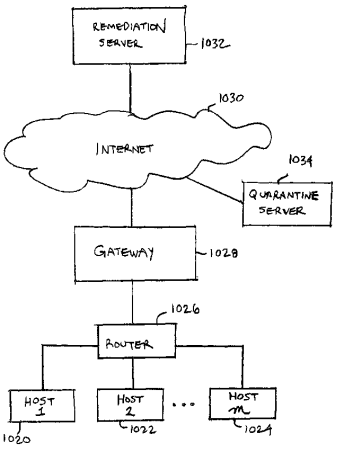
* cited by examiner

Primary Examiner — Eleni Shiferaw
Assistant Examiner — Jing Sims

(57) **ABSTRACT**

Contagion isolation and inoculation is disclosed. In some embodiments, a request is received from a host, e.g., via a network interface, to connect to a protected network. It is determined whether the host is required to be quarantined. If the host is required to be quarantined, the host is provided only limited access to the protected network. In some embodiments, a quarantined host is permitted to access the protected network only as required to remedy a condition that caused the quarantine to be imposed, such as to download a software patch, update, or definition; install, remove, and/or configure software and/or settings as required by a policy; and/or to have a scan or other diagnostic and/or remedial operation performed. In various embodiments attempts to communicate with hosts not involved in remediation are redirected to a quarantine system, such as a server, that provides information, notices, updates, and/or instructions to the user.

19 Claims, 25 Drawing Sheets



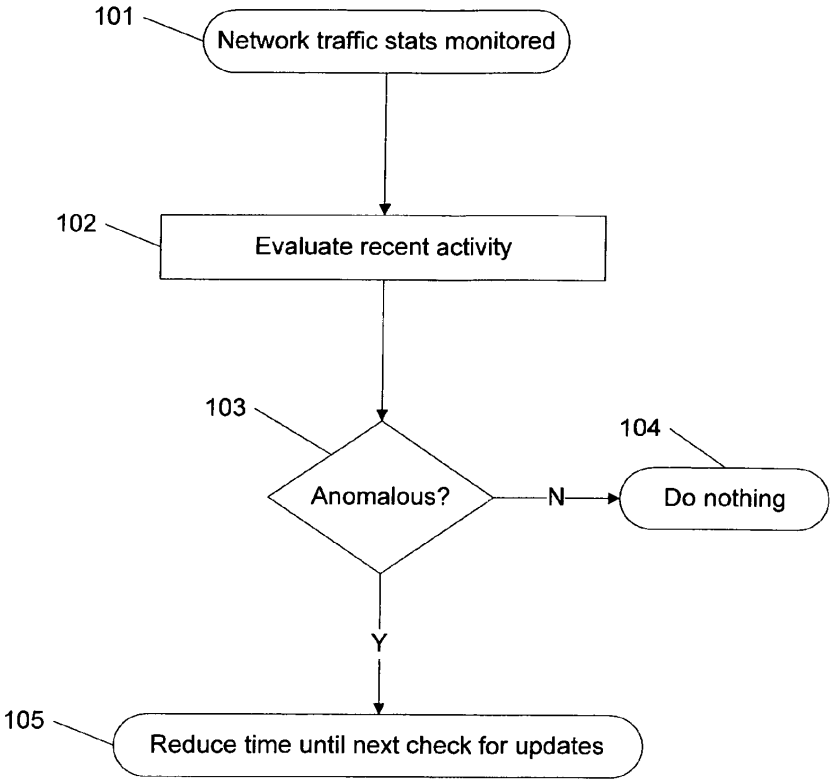

101
Network traffic stats monitored
102
Evaluate recent activity
103
Anomalous?
N
104
Do nothing
Y
105
Reduce time until next check for updates

Figure 1

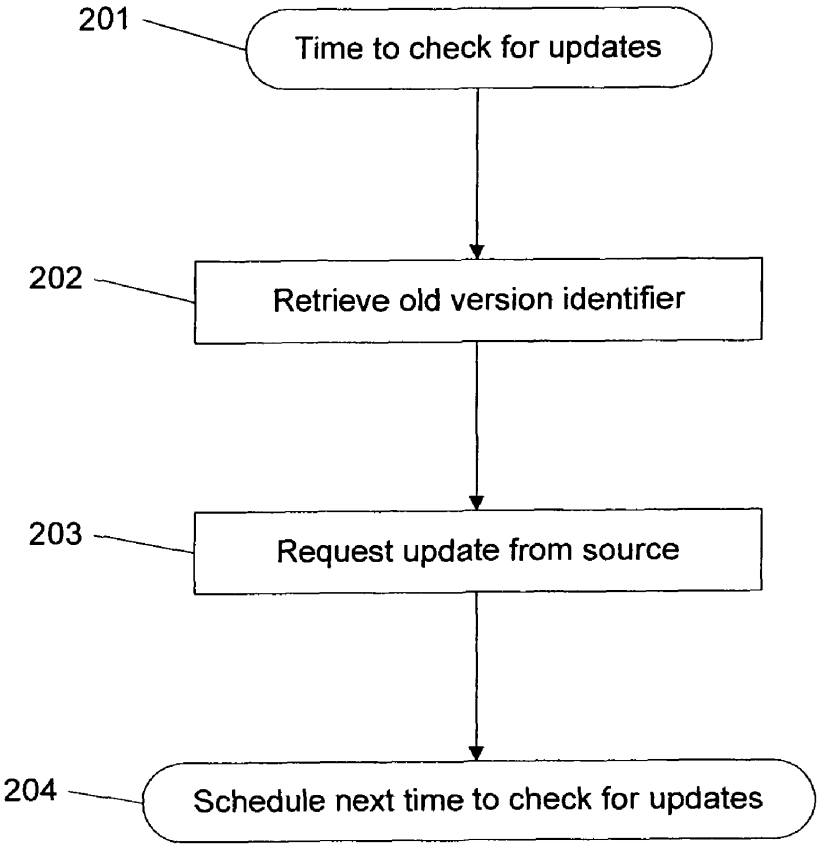

201
Time to check for updates
202
Retrieve old version identifier
203
Request update from source
204
Schedule next time to check for updates

Figure 2A

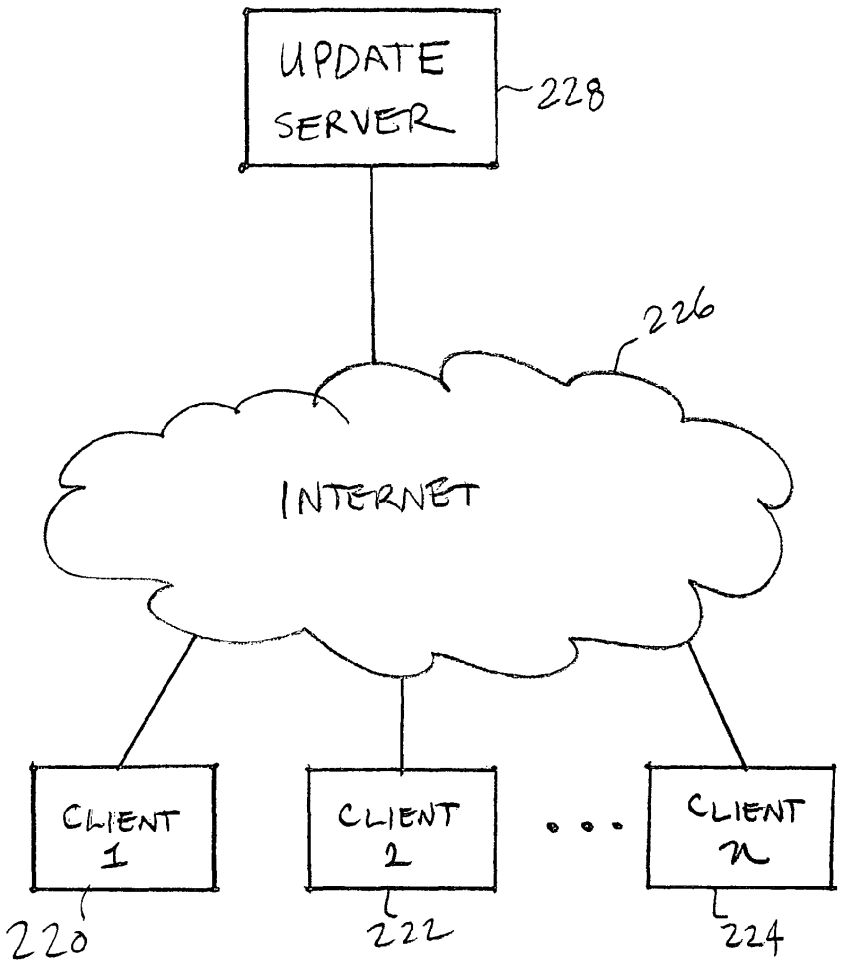

UPDATE SERVER
228
226
INTERNET
CLIENT 1
CLIENT 2
CLIENT n
220
222
224

FIG. 2B

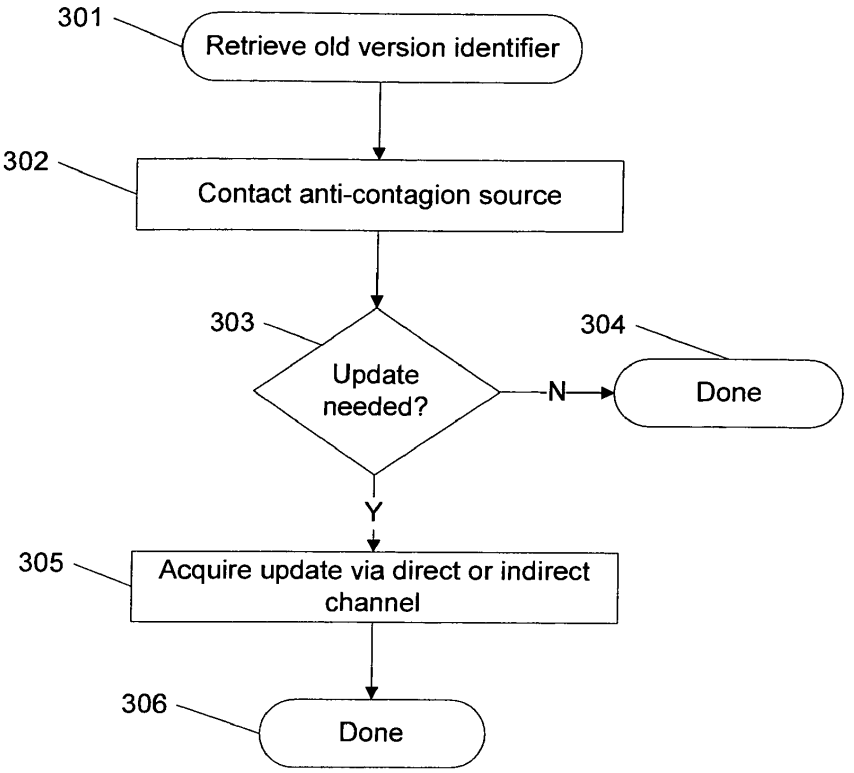

301
Retrieve old version identifier
302
Contact anti-contagion source
303
Update needed?
N
304
Done
Y
305
Acquire update via direct or indirect channel
306
Done

Figure 3A

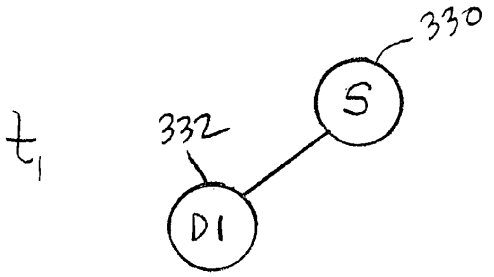


FIG. 3B

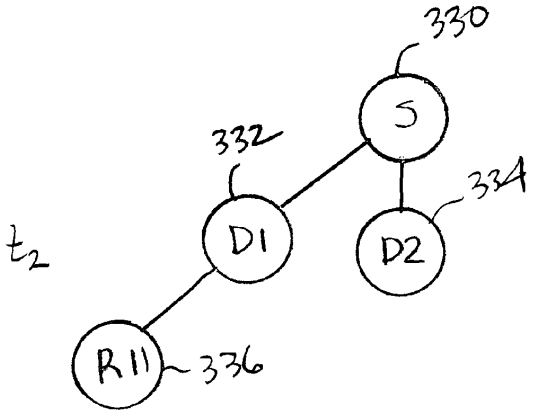


FIG. 3C

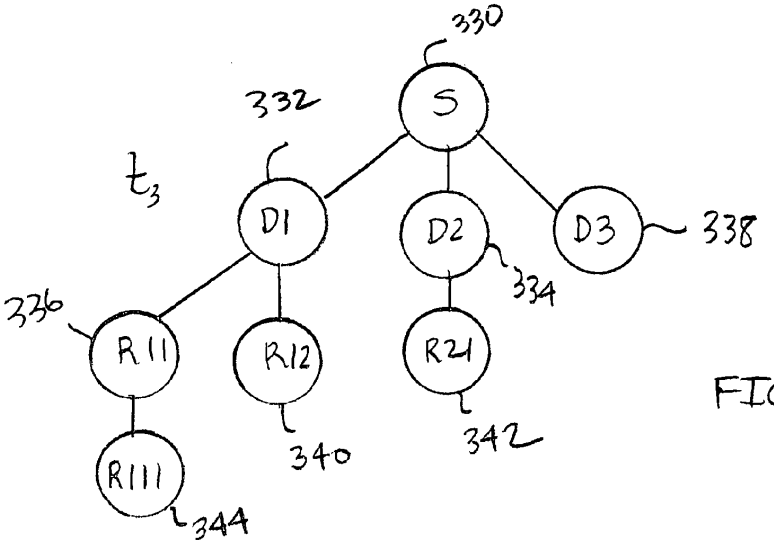


FIG. 3D

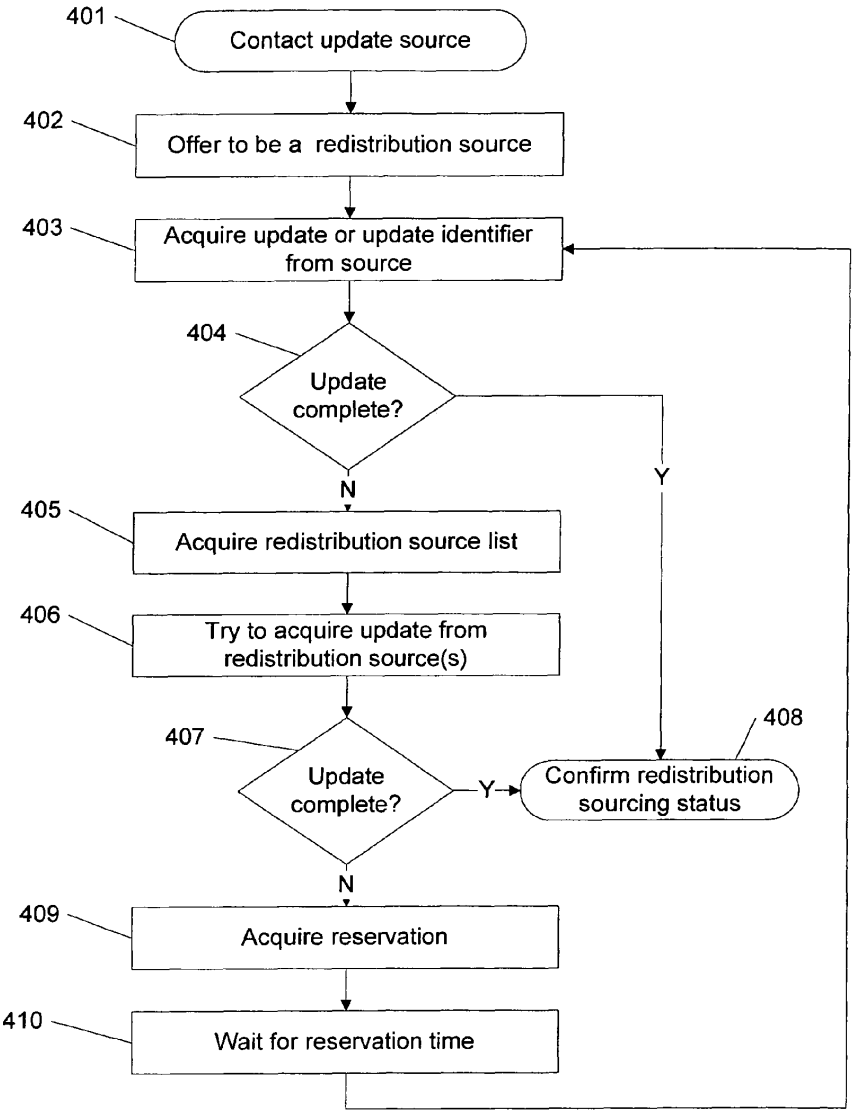

401
Contact update source
402
Offer to be a redistribution source
403
Acquire update or update identifier from source
404
Update complete?
N
Y
405
Acquire redistribution source list
406
Try to acquire update from redistribution source(s)
407
Update complete?
Y
408
Confirm redistribution sourcing status
N
409
Acquire reservation
410
Wait for reservation time

Figure 4A

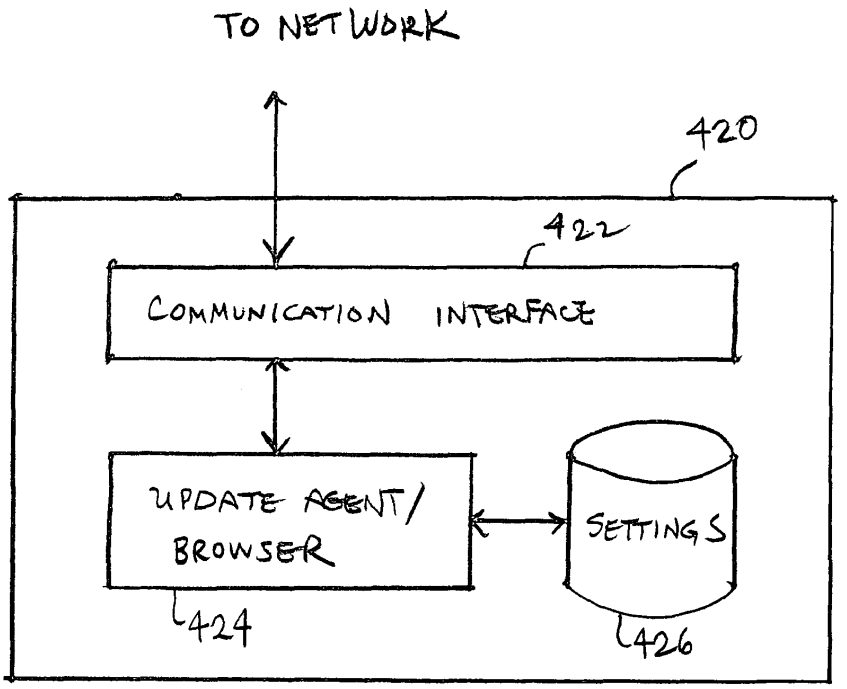

TO NETWORK
420
422
COMMUNICATION INTERFACE
UPDATE AGENT/
BROWSER
SETTINGS
424
426

FIG. 4B

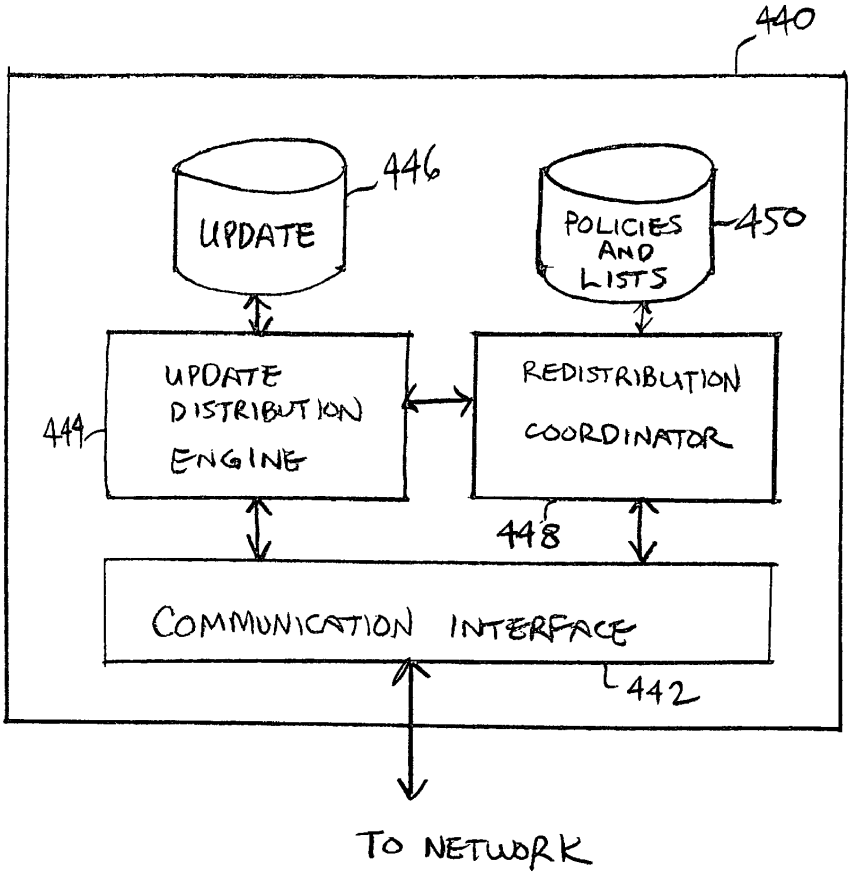

440
446
UPDATE
POLICIES AND LISTS
450
UPDATE DISTRIBUTION ENGINE
444
REDISTRIBUTION COORDINATOR
448
COMMUNICATION INTERFACE
442
TO NETWORK

FIG. 4C

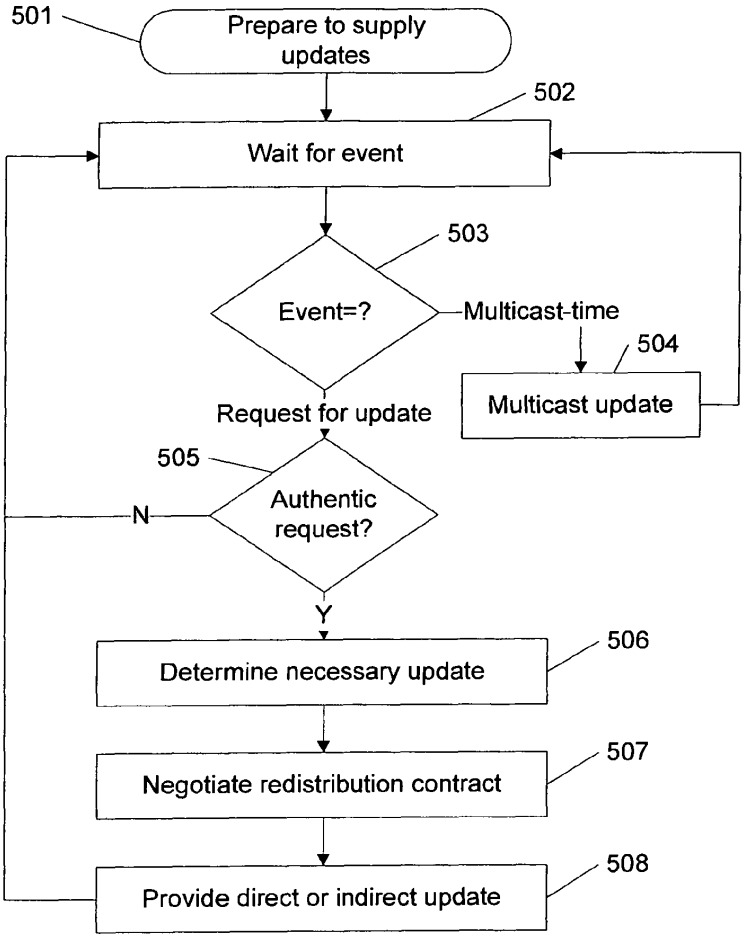

501
Prepare to supply updates
502
Wait for event
503
Event=?
Multicast-time
504
Multicast update
Request for update
505
Authentic request?
N
Y
506
Determine necessary update
507
Negotiate redistribution contract
508
Provide direct or indirect update

Figure 5

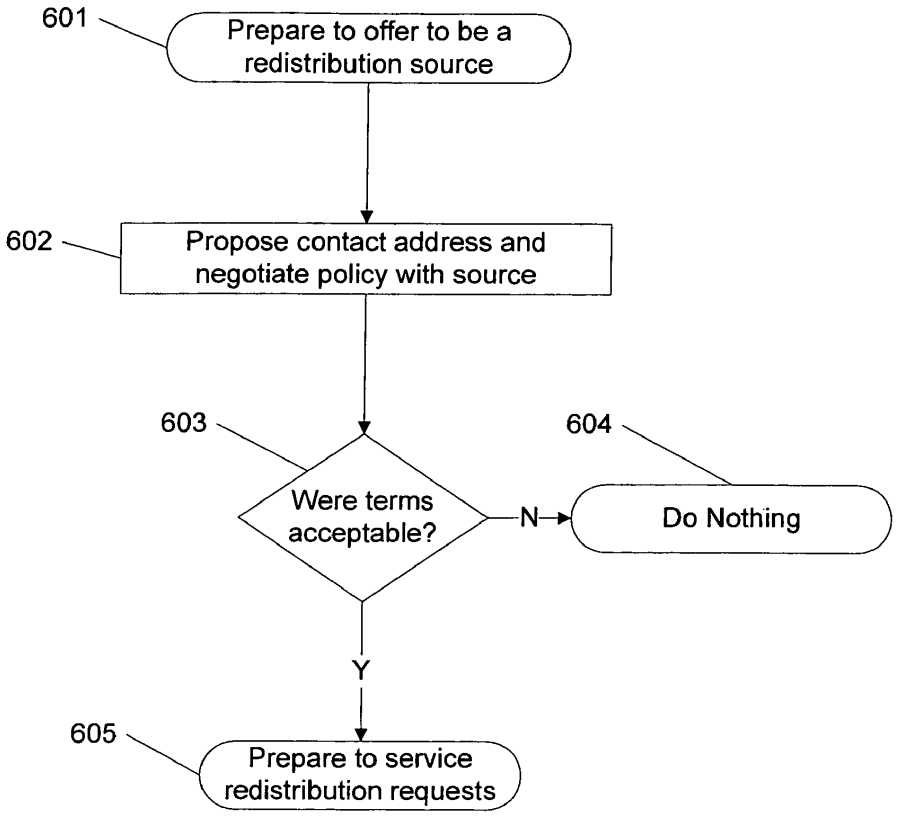

601
Prepare to offer to be a redistribution source
602
Propose contact address and negotiate policy with source
603
Were terms acceptable?
N
604
Do Nothing
Y
605
Prepare to service redistribution requests

Figure 6

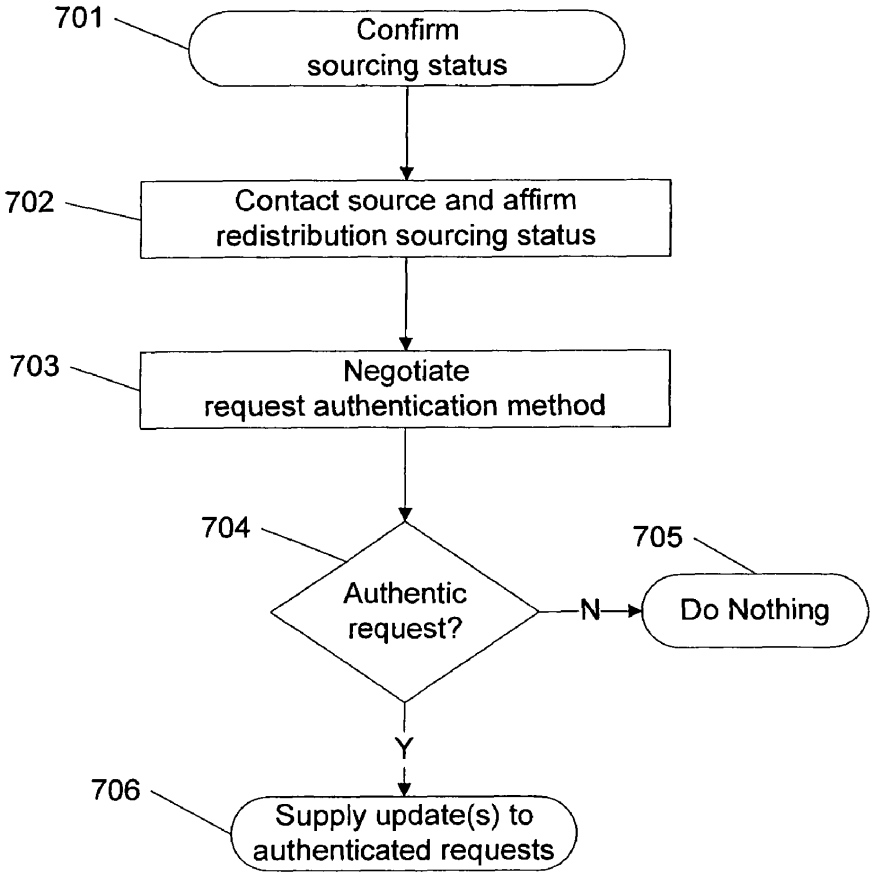

701
Confirm sourcing status
702
Contact source and affirm redistribution sourcing status
703
Negotiate request authentication method
704
Authentic request?
N
705
Do Nothing
Y
706
Supply update(s) to authenticated requests

Figure 7

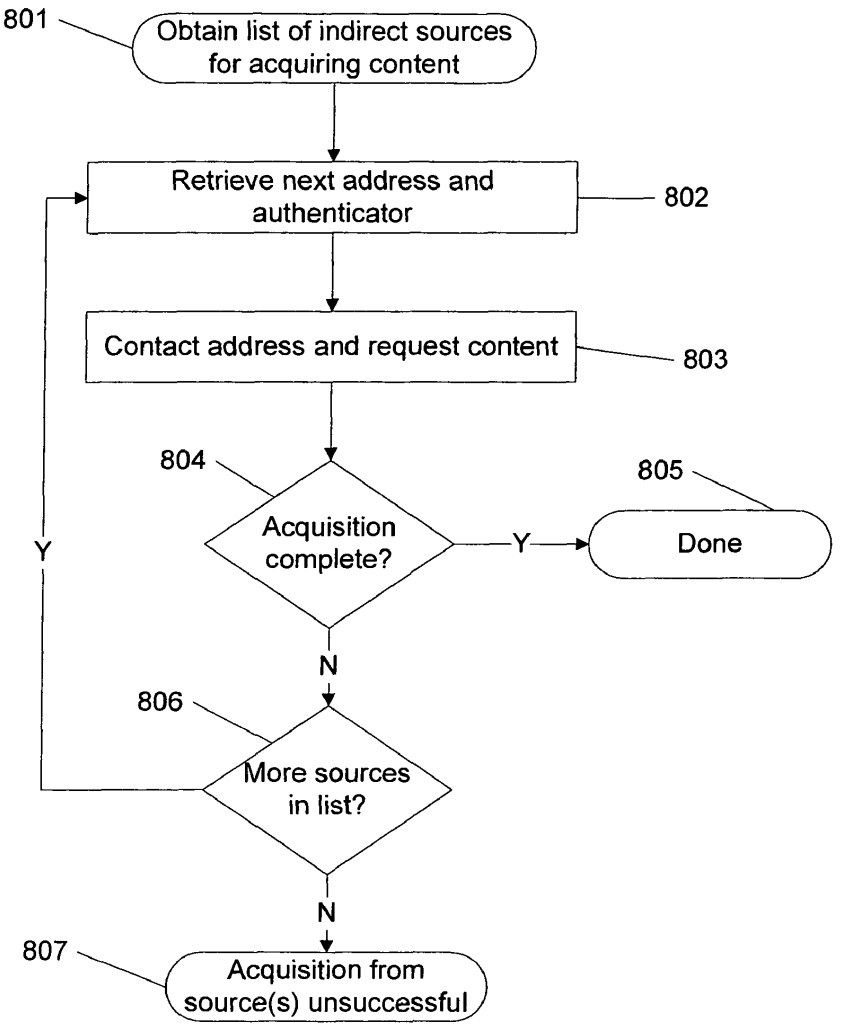

801
Obtain list of indirect sources for acquiring content
Retrieve next address and authenticator
802
Contact address and request content
803
804
Acquisition complete?
Y
805
Done
N
806
More sources in list?
Y
N
807
Acquisition from source(s) unsuccessful

Figure 8

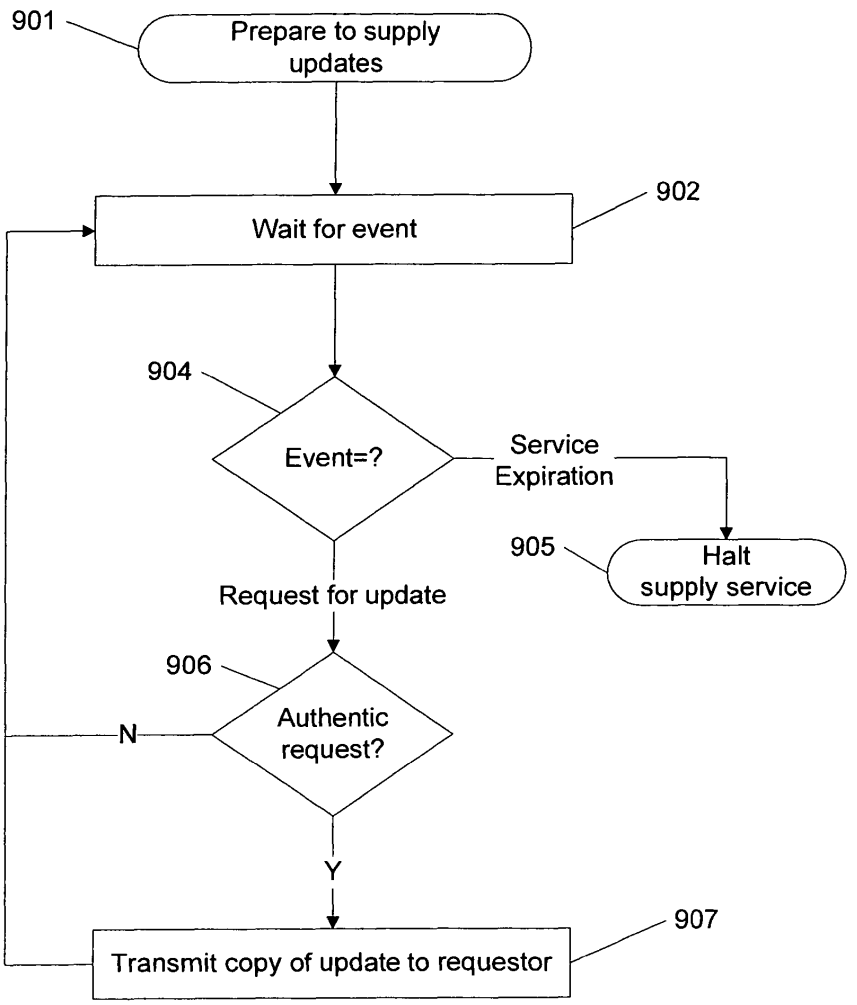

901
Prepare to supply updates
902
Wait for event
904
Event=?
Service Expiration
905
Halt supply service
Request for update
906
Authentic request?
N
Y
907
Transmit copy of update to requestor

Figure 9

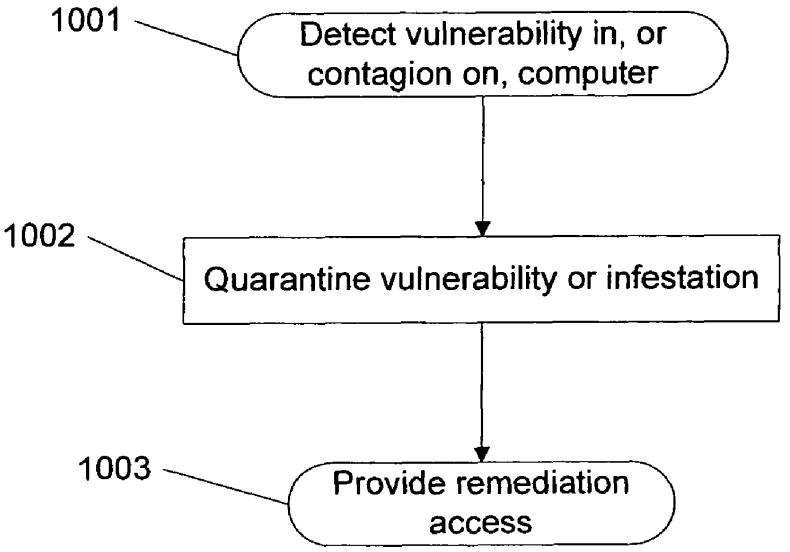

1001
Detect vulnerability in, or contagion on, computer
1002
Quarantine vulnerability or infestation
1003
Provide remediation access

Figure 10A

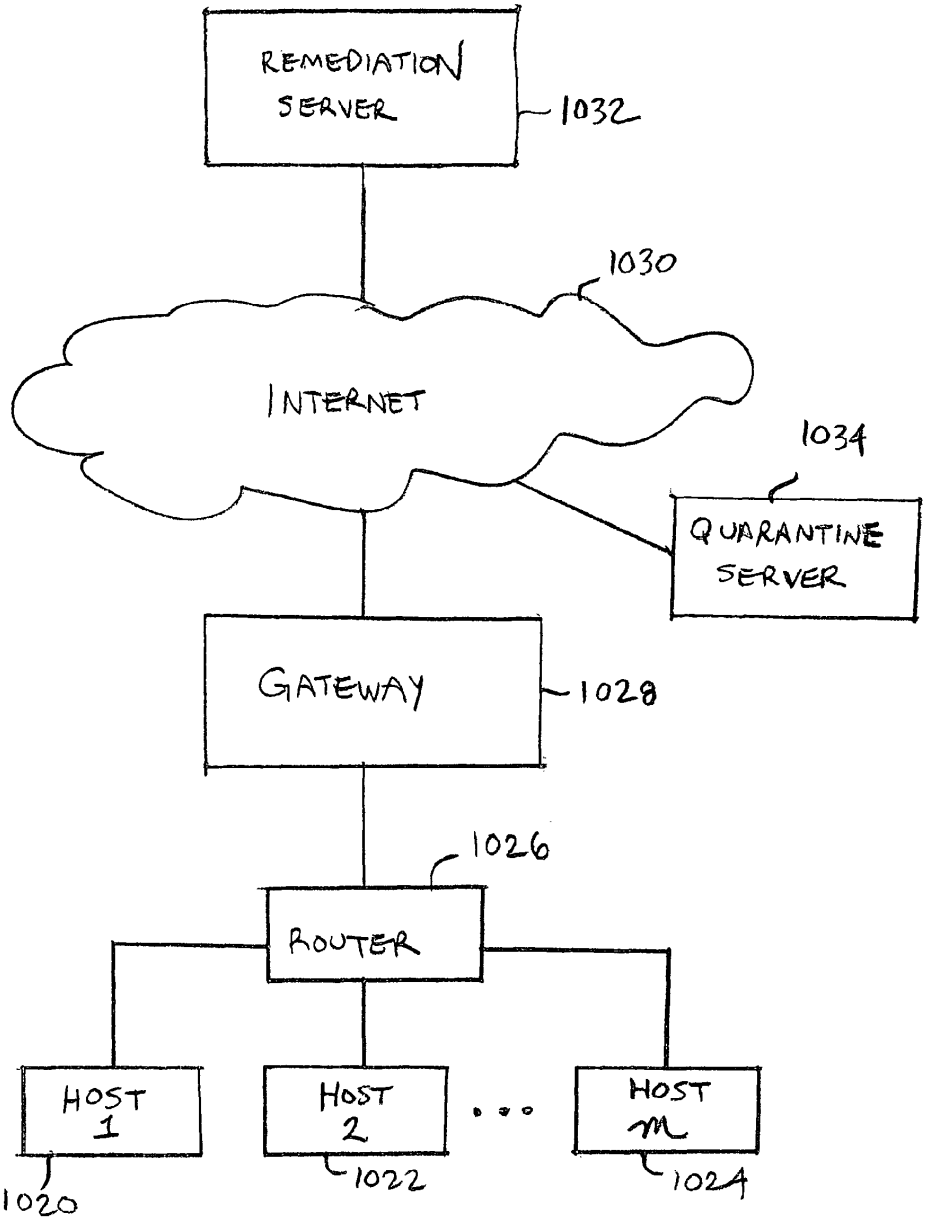

REMEDIATION SERVER
1032
1030
INTERNET
1034
QUARANTINE SERVER
GATEWAY
1028
1026
ROUTER
HOST 1
HOST 2
HOST m
1020
1022
1024

FIG. 10B

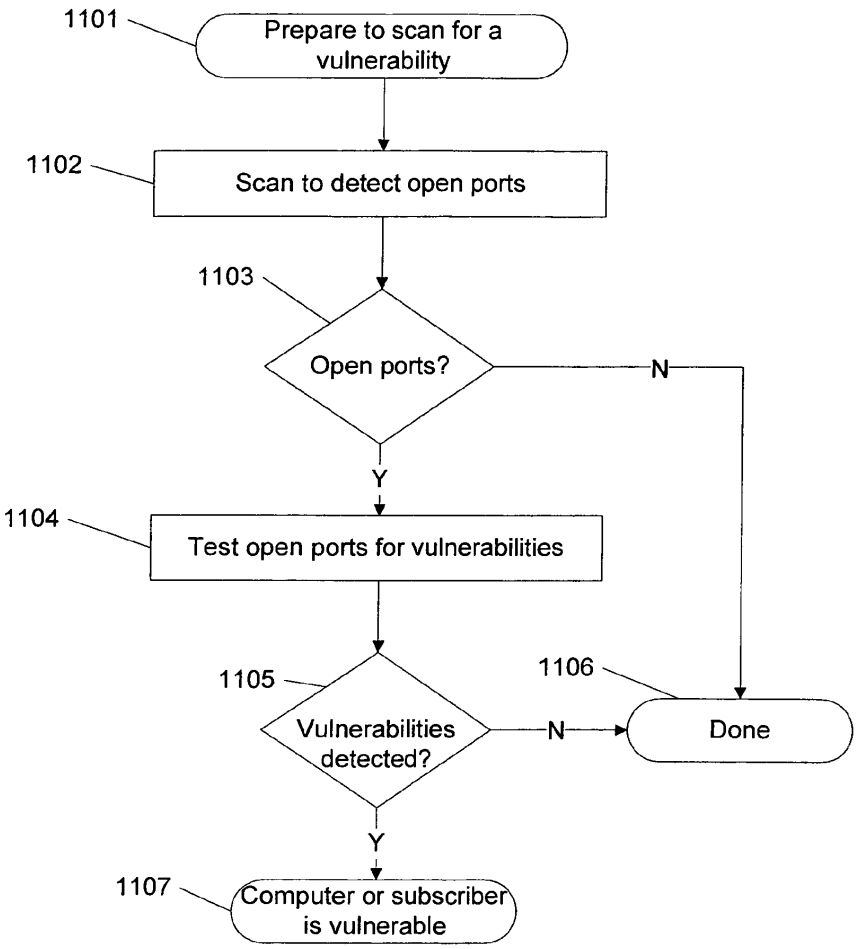

1101
Prepare to scan for a vulnerability
1102
Scan to detect open ports
1103
Open ports?
N
Y
1104
Test open ports for vulnerabilities
1105
Vulnerabilities detected?
N
1106
Done
Y
1107
Computer or subscriber is vulnerable

Figure 11

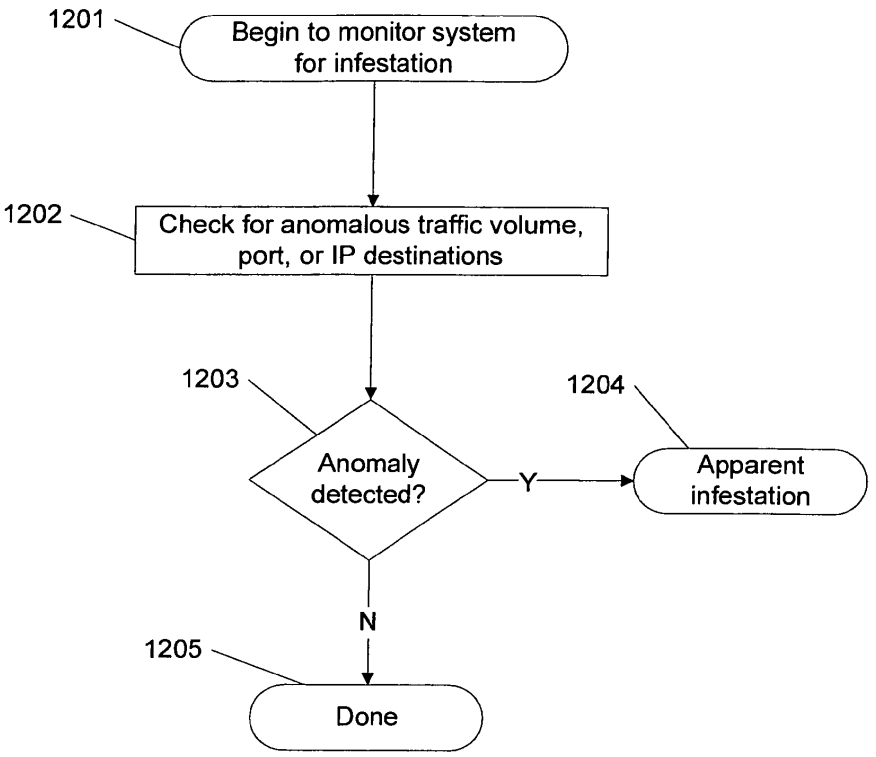

1201
Begin to monitor system for infestation
1202
Check for anomalous traffic volume, port, or IP destinations
1203
Anomaly detected?
Y
1204
Apparent infestation
N
1205
Done

Figure 12

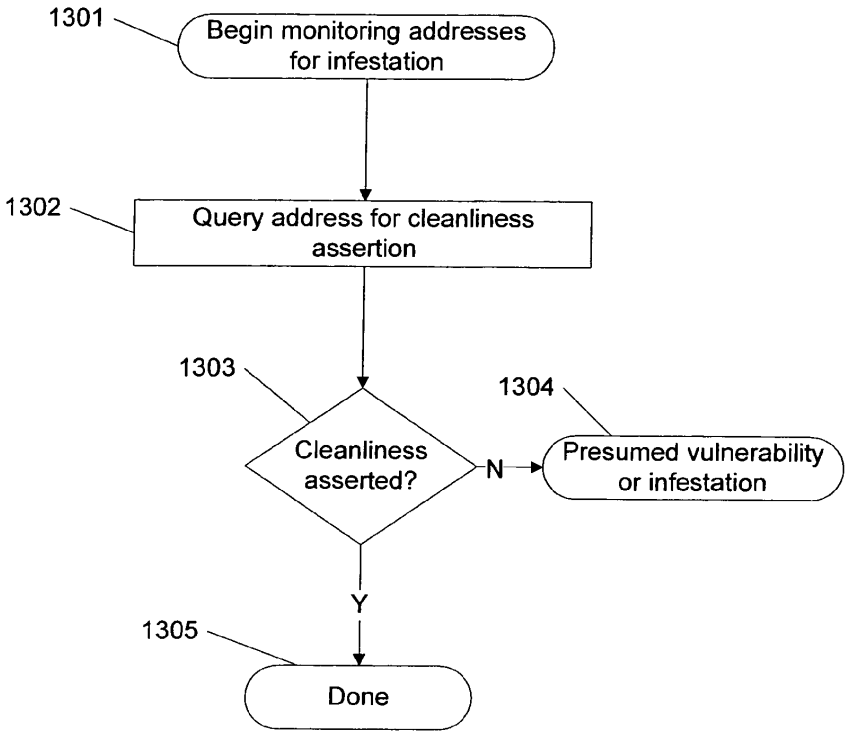

1301
Begin monitoring addresses for infestation
1302
Query address for cleanliness assertion
1303
Cleanliness asserted?
N
1304
Presumed vulnerability or infestation
Y
1305
Done

Figure 13

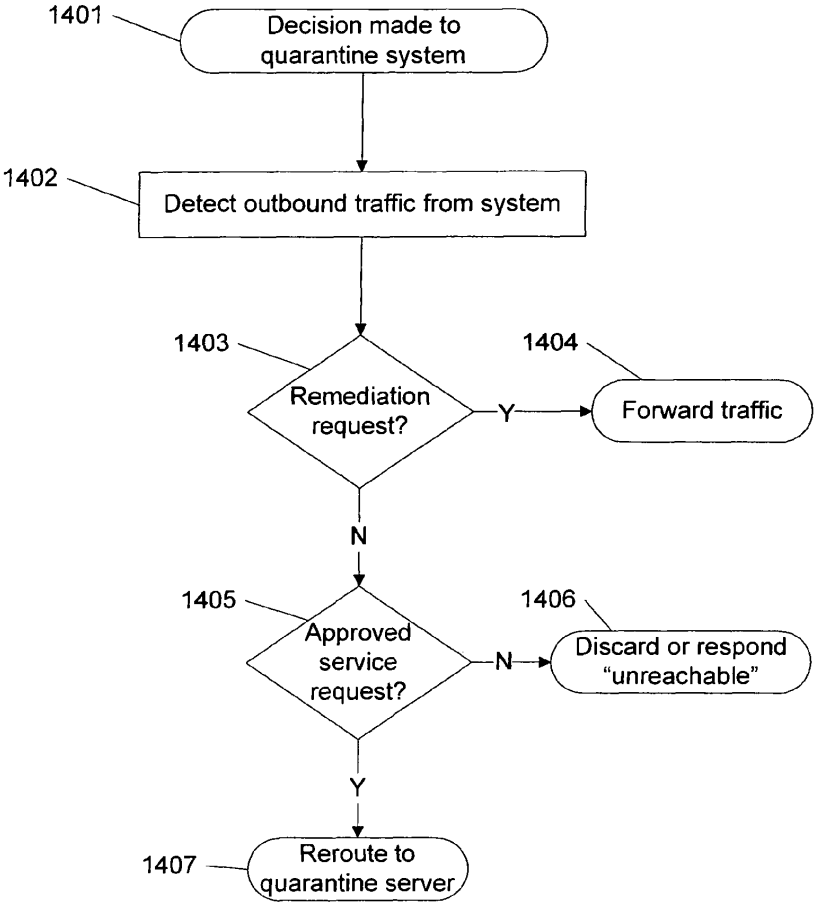

1401
Decision made to quarantine system
1402
Detect outbound traffic from system
1403
Remediation request?
Y
1404
Forward traffic
N
1405
Approved service request?
N
1406
Discard or respond "unreachable"
Y
1407
Reroute to quarantine server

Figure 14

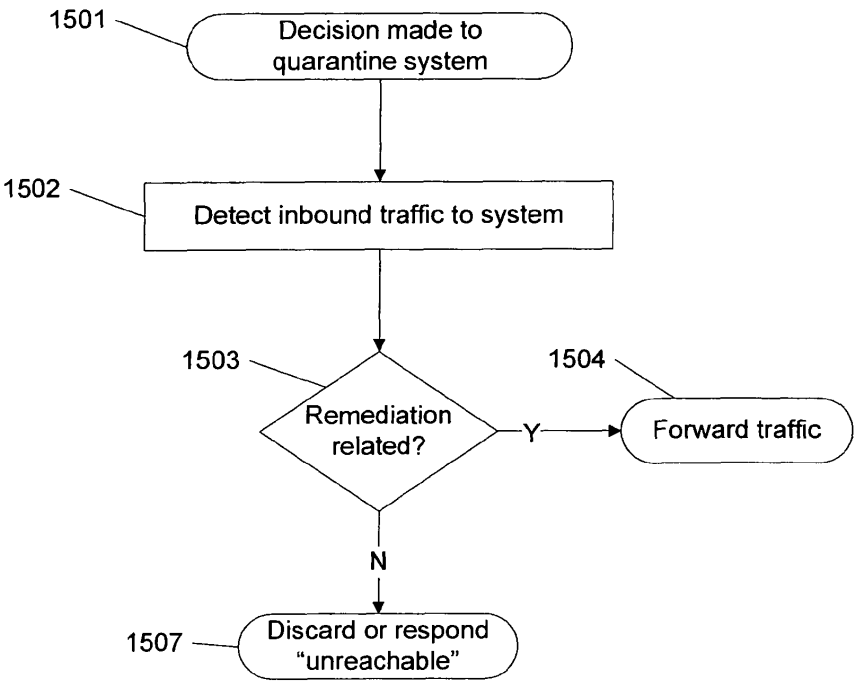

1501
Decision made to quarantine system
1502
Detect inbound traffic to system
1503
Remediation related?
Y
1504
Forward traffic
N
1507
Discard or respond "unreachable"

Figure 15

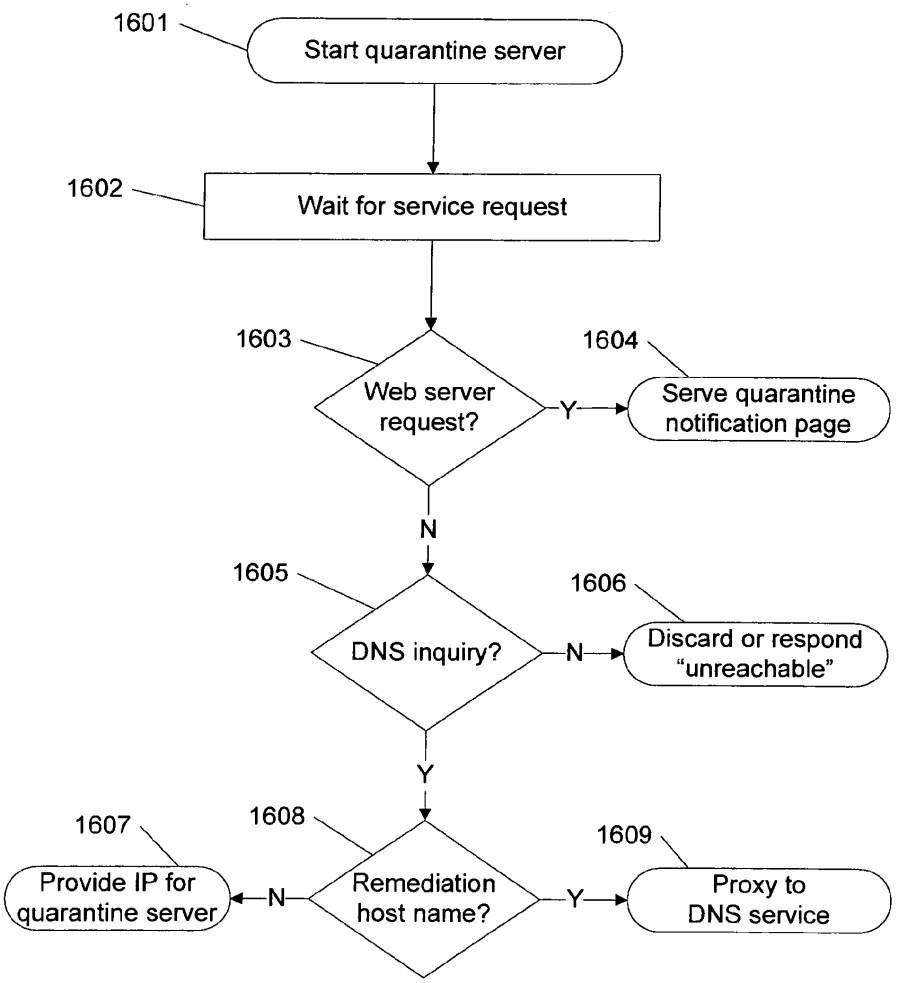

1601
Start quarantine server
1602
Wait for service request
1603
Web server request?
Y
1604
Serve quarantine notification page
N
1605
DNS inquiry?
N
1606
Discard or respond "unreachable"
Y
1607
Provide IP for quarantine server
N
1608
Remediation host name?
Y
1609
Proxy to DNS service

Figure 16

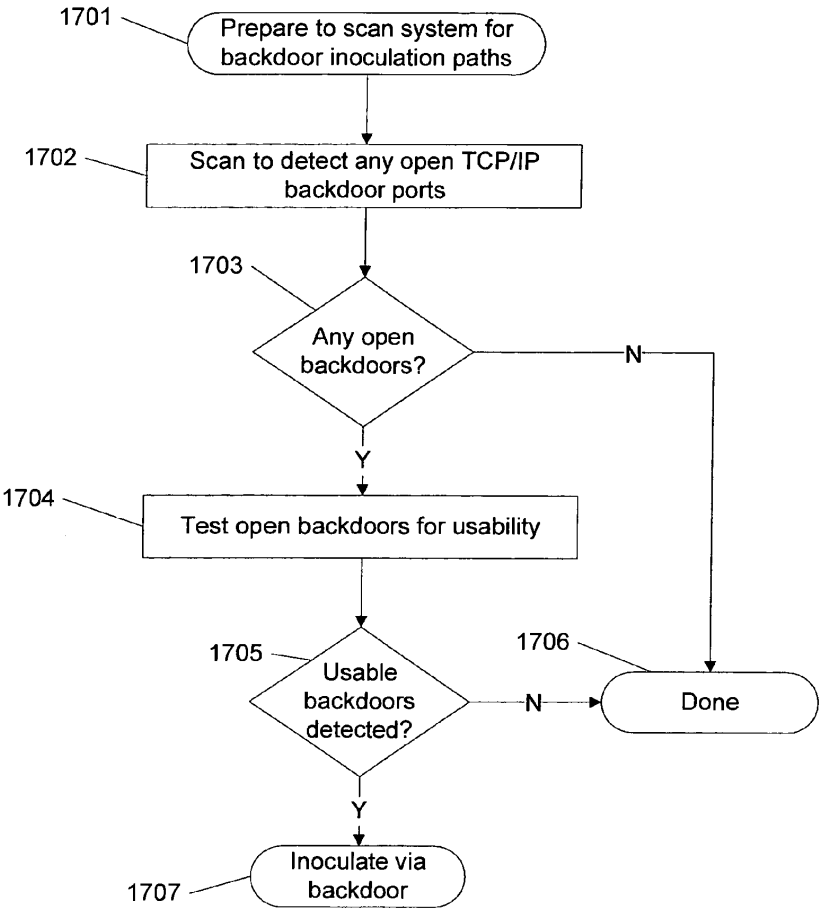

1701
Prepare to scan system for backdoor inoculation paths
1702
Scan to detect any open TCP/IP backdoor ports
1703
Any open backdoors?
N
Y
1704
Test open backdoors for usability
1705
Usable backdoors detected?
N
1706
Done
Y
1707
Inoculate via backdoor

Figure 17

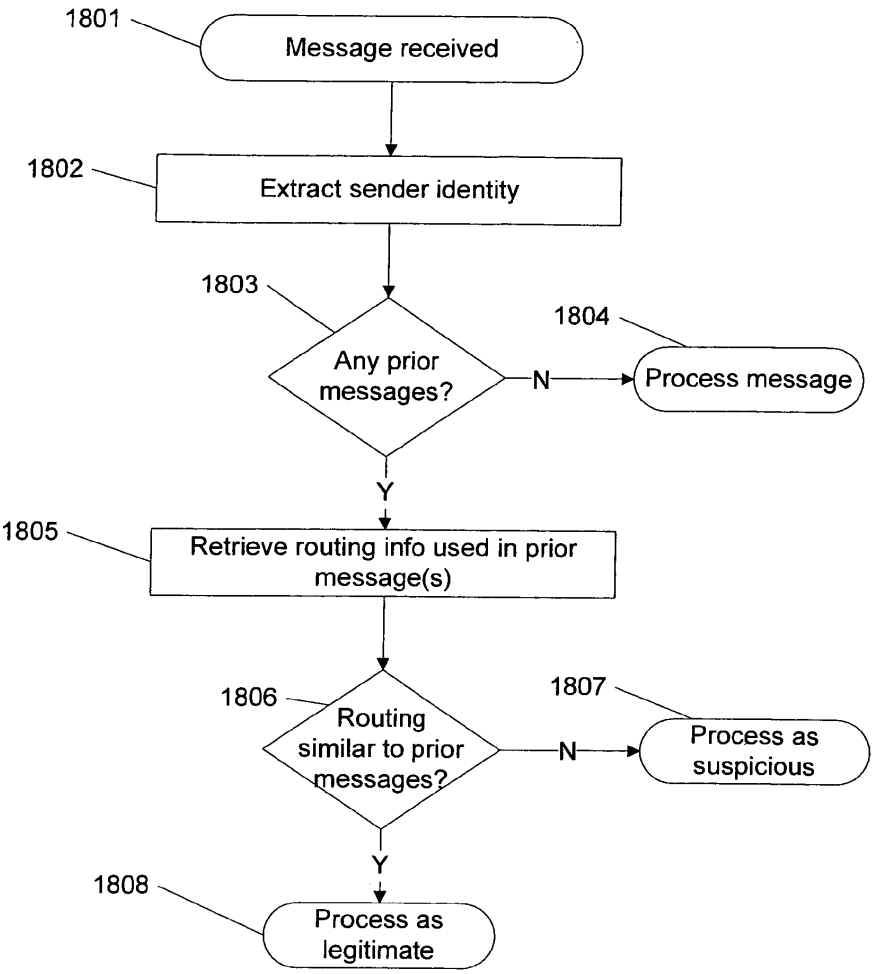

1801
Message received
1802
Extract sender identity
1803
Any prior messages?
N
1804
Process message
Y
1805
Retrieve routing info used in prior message(s)
1806
Routing similar to prior messages?
N
1807
Process as suspicious
Y
1808
Process as legitimate

Figure 18

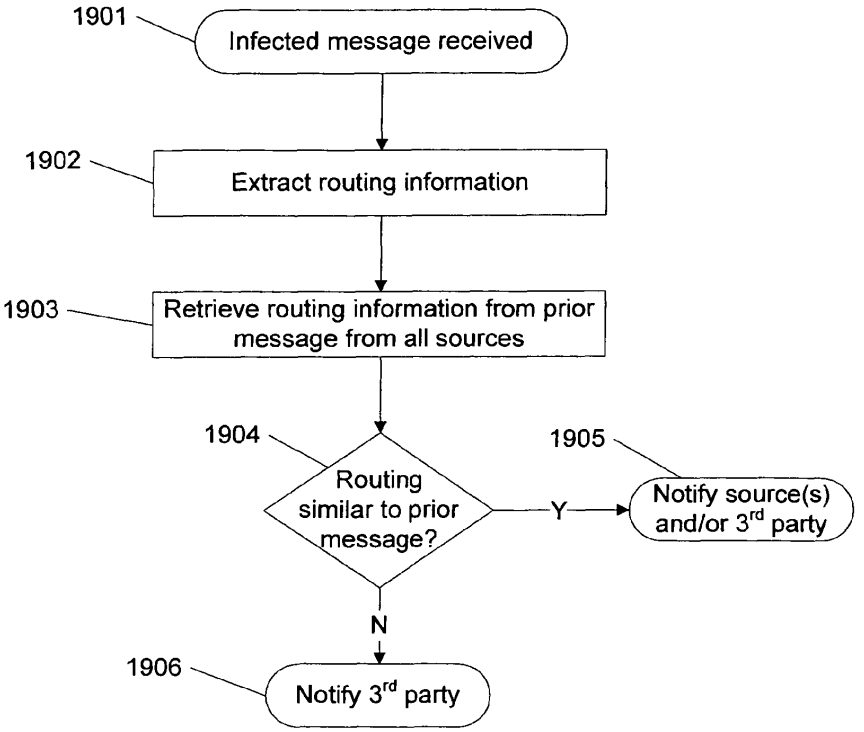

1901
Infected message received
1902
Extract routing information
1903
Retrieve routing information from prior message from all sources
1904
Routing similar to prior message?
Y
1905
Notify source(s) and/or 3rd party
N
1906
Notify 3rd party

Figure 19A

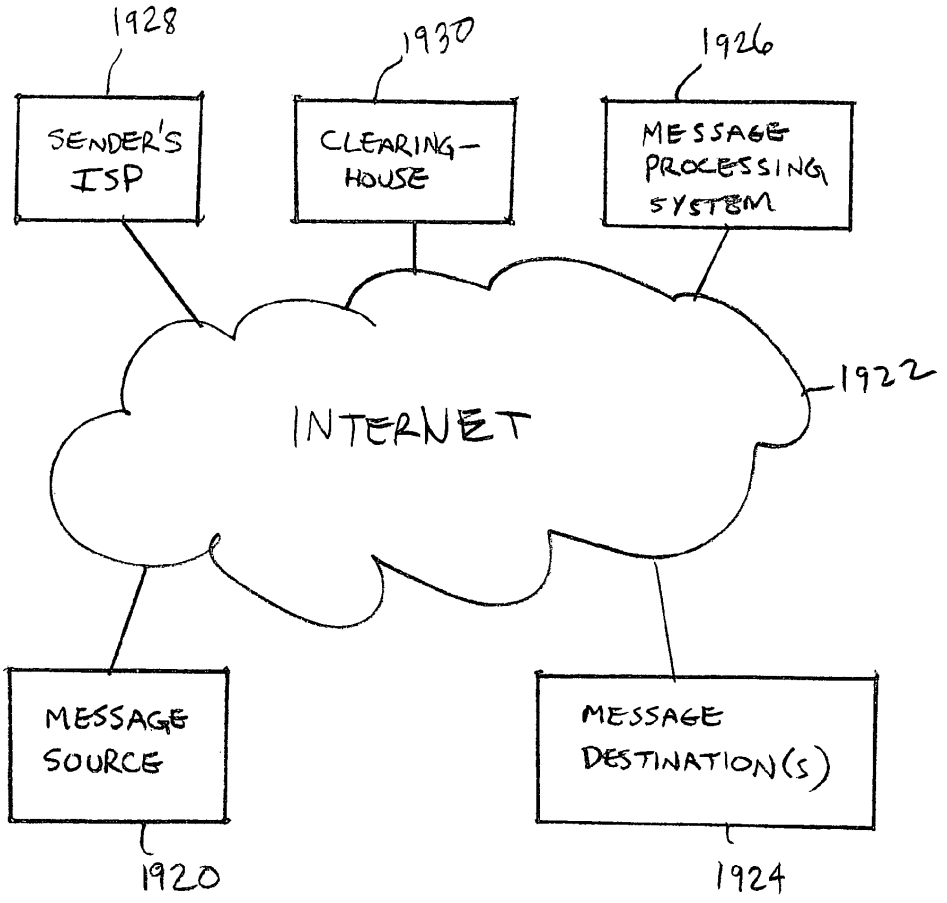

1928
SENDER'S ISP
1930
CLEARING-HOUSE
1926
MESSAGE PROCESSING SYSTEM
1922
INTERNET
MESSAGE SOURCE
1920
MESSAGE DESTINATION(S)
1924

FIG. 19B

CONTAGION ISOLATION AND INOCULATION

CROSS REFERENCE TO OTHER APPLICATIONS

This application claims priority to U.S. Provisional Patent Application No. 60/613,909 entitled CONTAGION ISOLATION AND INOCULATION filed Sep. 27, 2004, which is incorporated herein by reference for all purposes.

BACKGROUND OF THE INVENTION

Laptop and wireless computers and other mobile systems pose a threat to elements comprising and/or connected to a network service provider, enterprise, or other protected networks to which they reconnect after a period of connection to one or more networks and/or systems that are not part of the service provider, enterprise, or other protected network. By roaming to unknown domains, such as the Internet, and/or connecting to such domains through public, wireless, and/or otherwise less secure access nodes, such mobile systems may become infected by computer viruses, worms, backdoors, and/or countless other threats and/or exploits and/or have unauthorized software installed; have software installed on the mobile system by an operator of the protected network for the protection of the mobile system and/or the protected network removed or altered without authorization; and/or have configurations, settings, security data, and/or other data added, removed, and/or changed in unauthorized ways and/or by unauthorized person. Similarly, a stationary system may become infected, e.g., due to receipt and execution of malicious code via a network or other communication and/or a diskette and/or other removable media. Upon connecting to a protected network, a system may infect or otherwise harm resources associated with the protected network before measures can be taken to detect and prevent the spread of such infections or harm. Therefore, there is a need for a reliable way to ensure that a system does not infect or otherwise harm other network resources when connected to a protected network.

BRIEF DESCRIPTION OF THE DRAWINGS

Various embodiments of the invention are disclosed in the following detailed description and the accompanying drawings.

FIG. **1** is a flow diagram of a method for reducing an anti-contagion update interval during periods of suspicious network activity according to some embodiments.

FIG. **2**A is a flow diagram of a method for cooperatively scheduling an anti-contagion update interval according to some embodiments.

FIG. **2**B is a block diagram of a network environment in which anti-contagion updates are distributed according to some embodiments.

FIG. **3**A is a flow diagram of a method for contacting an anti-contagion source and optionally getting an update according to some embodiments.

FIGS. **3**B-**3**D illustrate distributed update distribution as implemented according to some embodiments.

FIG. **4**A is a flow diagram of a method for acquiring an update according to some embodiments.

FIG. **4**B is a block diagram of a client system configured to obtain an update according to some embodiments.

FIG. **4**C is a block diagram of a system used to distribute an update according to some embodiments.

FIG. **5** is a flow diagram of a method for distributing updates according to some embodiments.

FIG. **6** is a flow diagram of a method for establishing a redistribution policy according to some embodiments.

FIG. **7** is a flow diagram of a method for redistributing content such as updates according to some embodiments.

FIG. **8** is a flow diagram of a method for using peers or other indirect data sources to acquire content such as updates according to some embodiments.

FIG. **9** is a flow diagram of a method for redistributing content such as updates according to some embodiments.

FIG. **10**A is a flow diagram of a method for identifying and remedying problems according to some embodiments.

FIG. **10**B is a block diagram illustrating a network environment in which infected hosts and/or networks are quarantined according to some embodiments.

FIG. **11** is a flow diagram of a method for identifying vulnerabilities according to some embodiments.

FIG. **12** is a flow diagram of a method for identifying an infestation according to some embodiments.

FIG. **13** is a flow diagram of a method for monitoring one or more computers for infestation according to some embodiments.

FIG. **14** is a flow diagram of a method for quarantining a computer according to some embodiments.

FIG. **15** is a flow diagram of a method for quarantining a computer according to some embodiments.

FIG. **16** is a flow diagram of a method for a quarantine server to respond to requests according to some embodiments.

FIG. **17** is a flow diagram of a method for scanning backdoors and inoculating according to some embodiments.

FIG. **18** is a flow diagram of a method for categorizing messages according to some embodiments.

FIG. **19**A is a flow diagram of a method for notifying possible sources of infection according to some embodiments.

FIG. **19**B is a block diagram illustrating a network environment in which suspicious messages are identified and/or one or more notifications are sent when a malicious message is detected.

DETAILED DESCRIPTION

The invention can be implemented in numerous ways, including as a process, an apparatus, a system, a composition of matter, a computer readable medium such as a computer readable storage medium or a computer network wherein program instructions are sent over optical or electronic communication links. In this specification, these implementations, or any other form that the invention may take, may be referred to as techniques. A component such as a processor or a memory described as being configured to perform a task includes both a general component that is temporarily configured to perform the task at a given time and a specific component that is manufactured to perform the task. In general, the order of the steps of disclosed processes may be altered within the scope of the invention.

A detailed description of one or more embodiments of the invention is provided below along with accompanying figures that illustrate the principles of the invention. The invention is described in connection with such embodiments, but the invention is not limited to any embodiment. The scope of the invention is limited only by the claims and the invention encompasses numerous alternatives, modifications and equivalents. Numerous specific details are set forth in the following description in order to provide a thorough under-

standing of the invention. These details are provided for the purpose of example and the invention may be practiced according to the claims without some or all of these specific details. For the purpose of clarity, technical material that is known in the technical fields related to the invention has not been described in detail so that the invention is not unnecessarily obscured.

Contagion isolation and inoculation is disclosed. In some embodiments, a request is received from a host, e.g., via a network interface, to connect to a protected network. It is determined whether the host is required to be quarantined. If the host is required to be quarantined, the host is provided only limited access to the protected network. In some embodiments, a quarantined host is permitted to access the protected network only as required to remedy a condition that caused the quarantine to be imposed, such as to download a software patch, update, or definition; install, remove, and/or configure software and/or settings as required by a policy; and/or to have a scan or other diagnostic and/or remedial operation performed. In various embodiments attempts to communicate with hosts not involved in remediation are redirected to a quarantine system, such as a server, that provides information, notices, updates, and/or instructions to the user.

Contagion refers herein to software programs and/or code, data values, configurations, and settings, that expose a computer or other system to security threats, such as unauthorized access and/or use of the system and/or data stored thereon; corruption or loss of data; and/or damage, loss, and/or unauthorized reconfiguration of system software and/or hardware components. Examples of contagion include software programs and/or code designed to perform undesirable and/or unauthorized actions from the perspective of the (authorized) user of a computer, and at the same time perform desired actions from the perspective of the creator or controller of such contagion. Examples of contagion include computer worms, viruses, and Trojan horses. Anti-contagion software refers herein to software that prevents, impedes or remediates contagion, such as Norton Antivirus from Symantec, or VirusScan from McAfee, which identifies and/or removes contagion from a user's computer. Another example of anti-contagion software is software that repairs a security vulnerability through which contagion may spread. Many anti-contagion methods rely on contagion descriptions, such as virus signatures, to identify the contagion, and must be updated regularly to be effective.

FIG. **1** is a flow diagram of a method for reducing an anti-contagion update interval during periods of suspicious network activity according to some embodiments. An update interval refers herein to a period of time between acquisition of updates, such as security updates, virus signature file updates, or virus scanner updates. In this example, network traffic statistics may be monitored (**101**). An example of traffic statistics is latency experienced by email, such as average latency, or median latency. Another example of traffic statistics is the rate of unacceptable attempted TCP/IP traffic, such as attempts to connect to blocked or restricted ports, or insufficiently authenticated (or authorized) connections to ports, or connections that provided invalid or illegal content such as extraordinarily long strings, or specific content detected to be malicious. Recent activity may be evaluated (**102**) to see if it is anomalous. Examples of anomalous activity include median email latency in mail queues beyond some threshold, for example above a 4 hour median queuing delay, and rates of unacceptable attempted TCP/IP traffic beyond some threshold, for example greater than two attempts per minutes over a one hour period. Another example of a threshold for anomalous activity is a comparison with recent historical statistics, such as unacceptable attempted TCP/IP traffic at a rate greater than ten times the expected rate over a 30 minute period. The expected rate may for example be an average over a recent historic period, such as an average rate over the last week of connectivity. If activity is not deemed anomalous (**103**), then in this example, processing continues with no change (**104**) in the next update interval. If the activity is deemed anomalous (**103**), then in this example, the next update interval is reduced, for example reduced to some proportion of the planned update interval, for example reduced to 25% of the previous or default update interval. Another example of reducing the next update interval is to reduce the currently remaining time until an update, for example reducing by a proportion, such as 50%, the remaining time until an update is requested.

FIG. **2**A is a flow diagram of a method for cooperatively scheduling an anti-contagion update interval according to some embodiments. In this example, the planned time for contacting an anti-contagion source may arrive (**201**). An old version identifier may be retrieved (**202**), such as a checksum or hash for a file, or a version number for software stored in a file system or registry, or a version number for an anti-virus signature file stored in a file system or registry. An update may be requested from an anti-contagion source (**203**), for example as is discussed in conjunction with FIG. **3**A, which may include updating or concluding that an update is not needed. A time for next contacting an anti-contagion source is decided upon (**204**). In some embodiments, the scheduled time may be made sooner when the source of the update, such as a vendor of software to which the update pertains, anticipates the impending release of a fully tested update, such as scheduling a next contact time in a few hours, or a few days. If the current computer's version of anti-contagion software contains resolutions to all known problems and otherwise anomalous traffic patterns then, in this example, the update interval may be made longer, such as the full default interval for a computer, for example one week. In some embodiments, the selection of the next update interval may be dependent on the specific product. For example, a "pro version" may have a short default interval, such as one day, and a "home version" may have a relatively long default interval, such as two weeks or one month.

FIG. **2**B is a block diagram of a network environment in which anti-contagion updates are distributed according to some embodiments. A plurality of clients **1** to "n", represented in FIG. **2**B by clients **220**, **222**, and **224**, are connected to the Internet **226**. An update server **228** also is connected to the Internet **226**. In various embodiments, one or more of the clients **220**-**224** is configured to request and obtain updates, if any, from update server **228** via the Internet, e.g., as described above in connection with FIG. **2**A.

FIG. **3**A is a flow diagram of a method for contacting an anti-contagion source and optionally getting an update according to some embodiments. In this example, an old version identifier may be retrieved (**301**), for example as was discussed in **202** of FIG. **2**A. Contact may be made with an anti-contagion source (**302**) via a network such as the Internet. The old version identifier may be provided, for example using a transport protocol such as TCP/IP, and a data encapsulation format such as XML, or using a higher level protocol and format such as SOAP. A decision may be made as to whether an update is needed (**303**). In some embodiments, this update decision may be made by a server, upon review of the old version identifier. In some embodiments, this update decision may be made by the user's computer. In some embodiments, the update decision may be made by incorporating preferences held on a user's computer, such as a pref-

erence to always have the newest updates, or preferences to avoid updates that have been available for less than a threshold length of time (for example, 2 weeks), or preferences that accept an additional update when the source indicates urgency associated with the update. In some embodiments, the update decision may be made based on considerations such as the size of the potential update, for example by avoiding the overhead of small updates, and only performing an update when a threshold amount of change has taken place, or when a threshold length of time has elapsed since an actual update was made. An example of a threshold amount of change taking place is the addition of a prescribed number of viral signatures, such as 2 new signatures. An example of a threshold length of time is 3 weeks. If the update is not needed (**303**) then the process of contacting the source is complete (**304**) in this example. If an update is needed (**303**) then an update is acquired in this example via direct or indirect channels (**305**), for example as further described in conjunction with FIG. **4**A. After the update has been acquired (**305**), the contact may be complete (**306**).

Distributed update distribution is disclosed. In some embodiments, at least some users receive an update indirectly, e.g., from another user that received it previously, either directly from the original source or indirectly from a source that had itself earlier received it, again either directly or indirectly, and so on. FIGS. **3**B-**3**D illustrate distributed update distribution as implemented according to some embodiments. In the example shown in FIGS. **3**B-**3**D, it is assumed for expository purposes that each host that receives the update will serve as a redistribution node, each recipient has the ability to redistribute at the same rate as the original source, and there is continuous demand for the update from all sources. As shown in FIG. **3**B, at a time t**1** a first direct recipient of the update D**1** (**332**) downloads the update from the original source S (**330**). As shown in FIG. **3**C, at a time t**2** a second direct recipient D**2** (**334**) downloads the update from the original source and an indirect recipient R**11** (**336**) downloads the update from the first direct recipient D**1** (**332**), which in this case acts as a redistribution node. As shown in FIG. **3**D, at a time t**3** a third direct recipient D**3** (**338**) downloads the update from original source S (**330**), a second indirect recipient R**12** (**340**) downloads the update from the first direct recipient D**1** (**332**), a third indirect recipient R**21** (**342**) downloads the update from the second direct recipient D**2**, and a fourth indirect recipient R**111** (**344**) downloads the update from first indirect recipient R**11** (**336**). In this example, in the three download cycles shown in FIGS. **3**B-**3**D on its own the original source would only have been able to download the update to the three direct recipients, but by using previous recipients to redistribute, in the example shown in FIGS. **3**B-**3**D a total of seven recipients have been able to obtain the update. In a generalized case, under the assumptions used in this example in "n" download cycles/periods a total of $2n-1$ recipients are able to obtain the update, whereas only "n" recipients, i.e., one per cycle, would otherwise have been able to obtain the update if it were only available directly from the original source. In an even more generalized case, under an assumption that each server may download roughly k copies in a time interval, for example where more than one download is provided by a server at a time, then in "n" download cycles (or periods) a total of roughly $(k+1)n-1$ recipients are able to obtain the update, where as only $k*n$ recipients, i.e., k per cycle, would otherwise have been able to obtain the update if it were only available directly from the original source.

FIG. **4**A is a flow diagram of a method for acquiring an update according to some embodiments. In this example, an update source is contacted (**401**), such as via an Internet connection, for example via TCP/IP. An offer to serve as a redistribution source is made (**402**). In some embodiments, a redistribution source may be contacted by other computers to acquire an update. An example of an offer to be a redistribution server is described in conjunction with FIG. **6**. In various embodiments, **402** is optional, performed only if a criterion or test is satisfied, or omitted. An update or an update identifier is acquired (**403**). An example of an update identifier is a version number for an update. Another example of an update identifier is a checksum for an update. Another example of an update identifier is a set of checksums (such as MD5, SHA-1, or CRC-32) for a corresponding set of byte ranges that form the update. Another example of an update identifier is a combination two or more identifiers, such as the identifiers enumerated above. If the update was completed (**404**) then the user confirms redistribution sourcing status (**408**) in this example. In some embodiments, confirmation includes asserting that the client is prepared to redistribute copies of the received data to one or more recipients. If the update was not completed (**404**), then a list of peer or other sources is acquired (**405**) in this example. In some embodiments, the list is or may be explicit. In some embodiments, the list is or may be implicit, for example represented by an iterator or accessor, which may retrieve elements on an as needed basis. In some embodiments, implicit lists are dynamic, and calculated or created in part on an as needed or as requested basis. In some embodiments, the list of other sources contain addresses that may be contacted to acquire an update, such as a list of IP addresses and ports, as well as optionally some authentication information to be used when contacting a listed source. In some embodiments, a requesting host that agrees to serve as a redistribution source is provided with a list of sources from which the update is available immediately, or at least sooner than it will be made available to hosts that will not redistribute. One or more attempts may be made to acquire the update from a listed source (**406**), for example as exemplified in FIG. **8**. If the update is completed via a listed source (**407**), then redistribution status is confirmed (**408**) in this example. If the update is not complete (**407**) then a reservation is optionally negotiated and/or acquired (**409**) in this example. In some embodiments, a reservation may include a time and/or a specific source or IP address. In some embodiments, a reservation address may be selected from a collection of source addresses, for example in order to distribute the load across more than one server, or to provide a server that is closer to a recipient. An example of selecting a server close to a recipient is to select a server having relatively lower latency during a download, or relatively lower cost, for example by selecting a source within a recipient's ISP complex or within a recipient's intranet or home network. If a reservation time was acquired, then a wait for that reservation time (**410**) is performed in this example. At the reservation time, a further attempt to acquire an update or update identifier from the source (**403**) is made, and the flow proceeds as described earlier. This flow may continue until an acquisition is complete (**408**), as also described earlier. In various embodiments, one or more of confirming redistribution sourcing status (**408**), attempting to obtain the update from a redistribution source (**405-407**), and/or acquiring and/or waiting for a reservation time (**409-410**) is/are optional, conditional, or omitted.

FIG. **4**B is a block diagram of a client system configured to obtain an update according to some embodiments. In some embodiments, the client system shown in FIG. **4**B is used to implement one or more of clients **220-224** of FIG. **2**B. The client system **420** includes a communication interface **422**,

such as a network interface card, configured to send/receive communications via a network, such as the Internet. The client system **420** further includes an update agent and/or browser **424**. In some embodiments, the update agent/browser **424** comprises browser software used to navigate the World Wide Web, e.g., to send/receive data to/from a website from which an update is available for download. In some embodiments, update agent/browser **424** includes an agent associated with anti-contagion software installed on the client system **420**. In various embodiments, the update agent/browser **424** comprises one or more components and is implemented as one or more software programs running on a processor and/or one or more hardware components. A settings data store **426** stores one or more settings and/or other data associated with the software with which an update and/or software to which the update does or may pertain, such as periodicity or next scheduled (e.g., reservation) time to check for updates, a list of update identifiers, locators, and/or redistribution sources from which an update is or may be available, policies for whether and/or under what terms, conditions, circumstances, etc. the client system **420** may be used redistribute an update, a last update time, an old version identifier (see, e.g., **202** of FIG. **2**A), etc.

FIG. **4**C is a block diagram of a system used to distribute an update according to some embodiments. In some embodiments, the system of FIG. **4**C is used to provide an update server, such as update server **228** of FIG. **2**B. In this example, the distribution system **440** includes a communication interface **442** configured to send/receive data via a network, such as the Internet. An update distribution engine **444** is configured to receive and service requests for updates. For updates provided directly to a request source, the update distribution engine **444** obtains the update from an update storage location **446** and provides the update to the request source via the communication interface **442**. In some embodiments, if data included in the request indicates the source of the request is or may be willing to serve as a redistribution source, the update distribution engine **444** hands the request off to a redistribution coordinator **448**, which negotiates with the request source, e.g., based on policies, settings, rules, etc. stored in a policies and lists data store **450**, through communications sent/received via communication interface **442**, to determine if the requesting source will serve as a redistribution source and, if so, to determine the terms and conditions of such service and data, such as an address, locator, or identifier, that can be provided to other requesting systems and/or users to enable them to attempt to obtain the update from the redistribution source. In some embodiments, if update distribution engine **444** determines that it is not able to provide the update to the request source directly (e.g., due to capacity limits or because a policy applicable to the request and/or the request source indicates the request source should be directed to a redistribution source, e.g., because the request source has a less expensive version of the software or is not willing to serve as a redistribution source), the update distribution'engine **444** hands the request off to redistribution coordinator **448**, which provides to the request source an update identifier and a list of one or more redistribution sources from which the update is or may be available to the request source.

FIG. **5** is a flow diagram of a method for distributing updates according to some embodiments. In this example, a source prepares to supply updates to recipients (**501**), for example by starting up a server, for example an HTTP or FTP server, or a multicast server, such as an RTP multicast server. The server waits for an event (**502**), such as a contact on a predefined port, or a notification that the server should start a multicast. If a multicast-time event appears (**503**) then an update is multicast (**504**) in this example. In some embodiments, a multicast time event may be generated internally or externally to the server to start a multicast. In some embodiments, a multicast-time event may be automatically generated at predetermined times, such as hourly, weekly, or daily, or on individually specified dates and times. In some embodiments, a multicast-time event may include a specific identifier for the multicast. If an event of the form of an update request arrives (**503**) then a test for authenticity is optionally performed (**505**) in this example. An example of determining whether a request is authentic is to check a credential. Examples of credentials include a provided shared secret, such as a secret used by all requestors to this server, or a different secret used by each requestor such as a user serial number, or a cryptographically signed authentication, such as credentials or a certificate issued to users that had paid for an update service, for example, by inclusion in a purchased software package. In some embodiments, all requests are assumed to be authentic without the use of any authenticator. If the request was not authentic (**505**) then, in this example, the server continues to wait for an event (**502**). If the request was authentic (**505**) then the server may determine a necessary update (**506**) in this example. In some embodiments, determination of an update may be made based on the requestor's state, which in some embodiments may be indicated in a received request for update. A redistribution contract may be negotiated (**507**). In some embodiments, some requestors may agree to redistribute updates to other requestors. In some embodiments, some requestors may decline to redistribute updates. In some embodiments, a source may decline redistribution proposals from a requestor. An example of a redistribution contract is an agreement not to redistribute. Another example of a redistribution contract is an agreement to redistribute a fixed number of copies, such as five copies. Another example of a redistribution contract is an agreement to redistribute for a fixed period of time, such as one day. Another example of a redistribution contract is an agreement to redistribute within a specified geographic region within a fixed period of time. Another example of a redistribution contract is an agreement to redistribute indefinitely, for example until a replacement update is provided. A redistribution contract may be made with terms that are combinations of the above mentioned exemplary term limitations. An update may be provided directly or indirectly (**508**). A direct update may for example be provided by allowing an FTP download, or by encapsulating the transmission in an HTTP transfer, for example using XML and/or an appropriate coding, such as uuencode. An indirect update may be provided by directing the requestor to an alternate source, such as a redistribution server, or a server at a different address, or a server at a different time, such as a future time, or a combination of two or more such specifications. In some embodiments, a decision to provide direct vs. indirect updates may be based in part on the redistribution contract, for example a requestor that is willing to redistribute may be updated sooner or more directly. An example of a redistribution contract is a promise by a user to provide redistribution in exchange for purchase price concession, such as a reduced initial purchase price, or a quantity discount price. In some embodiments, a decision to provide direct vs. indirect updates may be based in part on previous agreements, such as prepayment for expedited service, such as may be included in a more expensive version of a product that utilizes the updates. The server may wait for an event (**502**). In some embodiments, events may be processed in parallel with previous events, or may be sequentially pro-

cessed, or some combination, for example up to a threshold number of events in parallel, such as a maximum of 32 events in parallel.

FIG. **6** is a flow diagram of a method for establishing a redistribution policy according to some embodiments. In this example, a computer may prepare to offer to be a redistribution source (**601**). In some embodiments, preparations may include retrieving preferences, such as the number of redistributions that may be done, or the duration of time that the computer is willing to act as a redistribution source. In some embodiments, preferences may be defaults, or may be configured by a user and stored persistently, such as in a registry or file system. A contact address may be proposed to the source and a policy may be negotiated (**602**). A contact address may for example be a globally routable IP address and a port, or a local IP address (reachable from a LAN behind a firewall) and port that can be accessed by other computers on the LAN, such as computers that shared a common NAT address in their contact with the original source. In some embodiments, negotiation may include proposing a policy. In some embodiments, negotiations may involve a counter proposal in response to a proposal. An example of a counter proposal criterion is to require a minimum duration for redistribution. If terms were not acceptable (**603**) then, in this example, the computer does nothing (**604**). If the terms were acceptable (**603**) then, in this example, the computer prepares to service redistribution requests (**605**), for example by establishing a server to respond at the agreed contact address, reserving storage to hold the content for redistribution, etc. In some embodiments, redistribution policy may be a broader contractual matter, where an update is provided more quickly or more directly in exchange for a promise to redistribute. In some embodiments, an auditing mechanism may be used to measure compliance and penalize contract breakers, such as failure to accept connections for redistribution, where penalties may for example include refusal to accept future similar contracts. As another example, a lower purchase or subscription price may be offered for operators of a software product, such as antivirus software, who agree to act as redistribution points. In some embodiments such purchase agreements may be configured into the software, for example by establishing agreements to act as a redistribution point for updates as defaults that a user cannot modify.

FIG. **7** is a flow diagram of a method for redistributing content such as updates according to some embodiments. In this example, a redistribution computer is contacted and confirms its sourcing status (**701**), for example by accepting a TCP/IP connection on a predefined port, and processing a request for content. The redistribution computer may optionally contact another computer, for example a more original source, and affirm its redistribution sourcing status (**702**). A more original source may be a source that has provided content in the past to the redistribution computer, directly or indirectly. In some embodiments, contacting a more original source demonstrates that the redistribution computer is performing redistribution. In some embodiments, contacting a more original source allows the redistribution computer to further authenticate the computer that requested the content, for example by getting confirmation from a more original source that a specific computer was authorized to request the content. An authentication to be used to authenticate requests is negotiated (**703**). In some embodiments, authentication includes providing a shared secret to the redistribution computer. In some embodiments, authentication includes presenting an authorization cryptographically signed by a more original source, such as a certificate. In some embodiments, all requests are presumed authentic. If the request is not authentic (**704**), then nothing further is done (**705**) in this example. If the request is authentic (**704**), then the requested update is supplied (**706**) in this example. The updates may be supplied for example via a transport protocol TCP/IP, or a higher level protocol such as FTP, HTTP, or HTTPS, etc.

FIG. **8** is a flow diagram of a method for using peers or other indirect data sources to acquire content such as updates according to some embodiments. In this example, a list of one or more peers or other indirect sources for acquiring updates is obtained (**801**), for example as was discussed in conjunction with **405** of FIG. **4**A. In one example, an indirect source may be a peer, such as another computer that has downloaded the content. In another example, an indirect source may be a computer configured to redistribute content without other use of the content, such as an ISP server established to redistribute content to subscribers. In another example, an indirect source may be a server created or authorized by the original source to perform additional distributions. In some embodiments, an obtained list of indirect sources may be implicit, for example by referring to a list using an iterator or accessor construct that can itemize entries in the list when requested, for example using a query to a service when an additional item in the list is required, which may for example dynamically complete the list on an as requested basis. A next address and (optional) authenticator in the list of sources is retrieved (**802**). Any order may be used. In some embodiments, an order of processing may correspond to the original list order. In some embodiments, an order of processing may be such that the earlier processed addresses would be expected to have lower latency and/or higher bandwidth, for example by first processing a local address on a LAN. In some embodiments, elements of the list are or may be processed in parallel, and portions (such as byte ranges) of the content may be acquired from different addresses in the list. An address on the list may be contacted to request content (**803**), which may in some embodiments be performed as described in conjunction with FIG. **7**. In one example, a source address may consist of an identifier for a broadcast with optional time(s) for a broadcast, and contact with the source may include receiving the broadcast. In another example, contact with the source may include making a connection such as a TCP/IP connection with the source. If the acquisition is complete (**804**), then the processing is complete (**805**) in this example. If the acquisition was not completed (**804**) then additional sources in the list are considered in this example. If there are more sources in the list (**806**) then, in this example, a next source is retrieved along with an optional authenticator (**802**) and the processing continues. If there are no more sources (**806**) then the attempted acquisition from source(s) is unsuccessful (**807**) in this example.

FIG. **9** is a flow diagram of a method for redistributing content such as updates according to some embodiments. In this example, a computer prepares to supply content (or updates) to a requestor/recipient (**901**), for example by starting up one or more servers. Examples of a server include an HTTP server, an FTP server, and a multicast server. A server waits for an event (**902**), such as a contact on a predefined port, or a notification that the server should be terminated, or a notification that a time for a multicast has arrived. If a service expiration event appears (**904**) then the supply service is terminated (**905**) in this example. A service expiration event may be automatically generated, for example after a predetermined duration of service, or after a predetermined number of copies of an update have been redistributed. If a request event arrives (**904**), then a test for authenticity is performed (**906**) in this example. Authenticity may be established in the request, for example by receiving and validating, or perform-

ing a mutual proof of possession of, a shared secret, such as a secret used by all requestors to this server, or a different secret used by each requestor of this service and held in a list of single-use secrets by this server, or by providing a cryptographically signed authentication. In some embodiments, all requests are considered authentic without the use of any authenticator. If the request was not authentic (**906**) then, in this example, the server continues to wait for an event (**902**) as described earlier. If the request was authentic (**906**) then, in this example, the server transmits a copy of the content to the requestor (**907**), for example by allowing an FTP download, or by encapsulating the transmission in an HTTP transfer, for example using XML and/or an appropriate coding, such as uuencode.

FIG. **10**A is a flow diagram of a method for identifying and remedying problems according to some embodiments. A problem may be identified as some event or status that may cause direct or potential harm. Exemplary problems include vulnerability or infestation. In this example, a problem such as a vulnerability or infestation is detected (**1001**). Infestation herein refers to the current presence and/or execution of contagion. In some embodiments, detection may be made by a router, firewall, or other network component, separately or in concert, using techniques such as those illustrated by FIGS. **11**, **12**, and **13**. In some embodiments, detection may be made by a computing device, for example a computing device operated by an ISP. After a detection of a problem, a computer or network containing the problem is quarantined (**1002**). In some embodiments, quarantining includes preventing unauthorized communications from being sent from the quarantined computer, such as efforts to infect, probe, or harass other computers, or efforts to communicate data without authorization, including for example file contents and keyboard entry logs. In some embodiments, no communications are authorized. In some embodiments, quarantining includes preventing undesired communications from reaching the quarantined computer, such as control messages or infectious messages. Examples of quarantining activities are illustrated in FIGS. **14** and **15**. In this example, the quarantined computer is provided access to remediation services (**1003**). In some embodiments, access to remediation includes or may include allowing connections to servers providing security patches or advisories, for example allowing connections to a vendor's server, or providing virus and worm disinfecting tools, such as focused removal tools, or data updates for previously installed repair tools. In some embodiments, access to remediation includes allowing direct access to vendor sites known to provide remediation assistance, such as to Microsoft's Windows Update service, where security patches may be obtained. In some embodiments, access to vendor sites is restricted by the quarantine to certain classes of traffic, such as HTTP requests, certain port access, such as port **80** or port **443**, certain content requests, such as specific URLs or specific file downloads. In some embodiments, remediation content may be substituted for other requested content, for example as discussed in conjunction with FIG. **14**.

FIG. **10**B is a block diagram illustrating a network environment in which infected hosts and/or networks are quarantined according to some embodiments. A plurality of hosts **1** to "m", represented in FIG. **10**B by hosts **1020**, **1022**, and **1024**, connect via a router **1026** and a gateway **1028** to the Internet **1030**. In various embodiments the router **1026** is omitted or included in the gateway **1028**. In various embodiments, gateway **1028** comprises a gateway, router, firewall, or other device configured to provide and control access between a protected network and the Internet and/or another public or private network. In various embodiments, the process of FIG. **10**A is implemented, in the event of quarantine of one or more of hosts **1020**-**1024**, for example, on the affected (quarantined) host, router **1026**, and/or gateway **1028**. In some embodiments, a quarantined host (or a host associated with a quarantined network or sub-network) is permitted to access a remediation server **1032**, e.g., to download a patch, more current threat definition, etc. In some embodiments, requests to connection to a host other than the remediation server **1032** are redirected to a quarantine server **1034** configured to provide a notice and/or other information and/or instructions to a user of the quarantined host. In various alternative embodiments the quarantine server **1034** is omitted or resides on the router **1026** and/or gateway **1028**.

FIG. **11** is a flow diagram of a method for identifying vulnerabilities according to some embodiments. In this example, preparation is made to scan for one or more vulnerabilities (**1101**), for example by retrieving one or more IP addresses of computers that need to be scanned. For example, an IP address may be provided by a DHCP sever hosted in an ISP network complex, to identify current subscriber IP addresses. As another example, a subnet range may be scanned, for example by taking an address of the form A.B.C.X, and using some or all values of X in the range 0 to 255 to enumerate distinct IP addresses. A scan of one or more IP addresses may optionally be performed to detect an open port (**1102**). An example of scanning an IP address is scanning to detect open TCP/IP ports (**1102**). An open port may be identified by attempting to form a connection, such as a TCP/IP connection to that port. In some embodiments, only open ports on a given computer will provide acknowledgment messages needed to establish a TCP/IP connections, which for example causes a success code to be returned by Unix "connect( )" and a closed port will provide no response to the attempt to connect, which for example would cause an error code to be returned by the Unix function call "connect( )". If no open ports are detected (**1103**), then the scanning is complete (**1106**) in this example. If open ports are detected (**1103**) then those ports are tested for vulnerabilities (**1104**) in this example. In some embodiments, an open port is assumed (**1103**) and a test for vulnerability is performed or attempted (**1104**), for example by communicating a vulnerability test without independently testing to see if one or more requisite ports are open. In some embodiments, a vulnerability test of a port may include sending data that is known to create a discernable response when a vulnerability is present, such as sending a long string known to induce a buffer overflow with a consequent errant response, where the errant response may for example include a characteristic communication, premature termination of the connection, or other erroneous response(s). In some embodiments, a test for a vulnerability is an attempt to exercise a service, such as a sendmail or an RPC service, in search of a misconfiguration of that service that creates a vulnerability. One example of a misconfiguration of a service is provision of any such service. Another example of a misconfigured service is one where a password should be required for security purposes, but no password is required by the misconfiguration. Another example of a misconfiguration is the use of an easily guessable password, such as the default installation password, when resulting access creates a security vulnerability. Another example of a misconfigured service is support for a service provided by an unauthorized agent, such as a worm or virus. If no vulnerabilities are detected (**1105**) then scanning is complete (**1106**) in this example. If vulnerabilities are detected (**1105**) then an associated IP address is flagged as vulnerable (**1107**) in this example. In the case of a subscriber to an ISP, an IP address may be translated to a particular subscriber identity, for

example by checking in DHCP or static IP allocation tables for the subscriber's identity, or for example by extracting a MAC address from traffic associated with the IP address, and such identity may be appropriately flagged or recorded. In some embodiments, the record of the vulnerability may additionally include the port, and the details of the vulnerability that was detected, such as which buffer overrun was vulnerable, or which service was misconfigured to create a vulnerability.

FIG. **12** is a flow diagram of a method for identifying an infestation according to some embodiments. In this example, a device near a computer, such as a firewall on a LAN or an ISP's router, is used in some embodiments to monitor a computer for infestation (**1201**). Network traffic is monitored for anomalous traffic volume, port, or IP destination (**1202**), for example by detecting large volumes of outbound traffic, such as email traffic or instant messaging traffic. Another example of anomalous traffic is a TCP/IP connection to a server that is known to control a collection of infested machines, in a way that suggests the connecting machine is controlled by the server, such as connecting and transmitting an assertion that it is controlled or available to comply. Another example of anomalous traffic is a connection to a computer that is a current target of a denial of service attack, with a pattern that is consistent with the attack, such as a connection to a targeted port, or the use of a falsified return IP address. In some embodiments, a falsified return address may be detected by a router, such as an ISP router, that is aware that the return address is not on the sourcing subnet, or not associated with the MAC address that has supplied the corresponding packets. If an anomaly is detected (**1203**) then an apparent infestation is noted (**1204**) in this example. In some embodiments, a response to infestation may include quarantining of the computer or network containing the computer, for example as discussed in conjunction with FIGS. **14**, **15**, and **16**. In some embodiments, a response to infestation may include an attempt to remediate the problem, for example by inoculation, such as discussed in conjunction with FIG. **17**. If no anomaly is detected (**1203**) then monitoring is complete (**1205**) in this example. In some embodiments, monitoring for anomalous traffic may be continuous. In some embodiments, monitoring for anomalous traffic may be periodic. For example, periodic monitoring may consist of monitoring some number of computers, such as 1 computer, for some period of time, such as 2 minutes, and then not monitoring those computers again for another period of time, such as 1 hour. As another example, periodic monitoring may be randomized, for example by changing the duration of monitoring, pattern of selecting computers to monitor, and/or the unmonitored duration, to be difficult or impossible to predict, for example by being driven by a pseudo random number generator, subject to restrictions of available monitoring computers.

FIG. **13** is a flow diagram of a method for monitoring one or more computers for infestation according to some embodiments. In this example monitoring of a computer for infestation begins (**1301**), for example by retrieving a list of one or more addresses of computers, such as addresses of participating subscribers. In some embodiments, this list may be a list of addresses for some or all subscribers for a given ISP. In some embodiments, the list may be a list of IP addresses issued by a DHCP server, such as on a LAN, or a list of IP addresses observed by a router, such as a router on a LAN, or a complete list of IP addresses in a given subnet. A computer associated with an address identified in a list is queried for a cleanliness assertion (**1302**), for example by contacting a trusted computing base within a computer, and requesting an authenticated infestation scan by trusted software. An example of a trusted computing base within a computer is the Paladium security initiative under development by Microsoft and supported by Intel and American Micro Devices. Another example of a trusted computing base is described in various TCG specifications, such as the TCG Architecture Overview, published by the Trusted Computing Group. Trusted code bases may for example execute antivirus scans of the remainder of the computer, including untrusted portions of the disk and/or operating system. In some embodiments, trusted code bases may digitally sign assertions about the cleanliness (e.g. infestation status) and/or state of their computers. In some embodiments, the query for cleanliness (**1302**) may be responded to by anti-contagion software, such as antivirus software, with assertions about the currency of a scan, such as the last time a scan was performed, or a version associated with a current anti-contagion software or definition file in use, wherein a sufficiently updated software and/or scan may act as a cleanliness assertion. In some embodiments, an operating system may respond with information associated with its patch level, wherein a sufficiently recent patch level may be interpreted as an assertion of cleanliness. If a computer asserts it is clean (**1303**) then monitoring is complete (**1305**) in this example. If a cleanliness assertion is not provided (**1303**) then an infestation or vulnerability is presumed (**1304**) in this example.

FIG. **14** is a flow diagram of a method for quarantining a computer according to some embodiments. In this example, a decision has been made to quarantine a computer (**1401**), such as an ISP deciding to quarantine a subscriber, or a firewall or router for a LAN deciding to quarantine a local computer. In some embodiments, a decision to quarantine may be made based on apparent infestations or apparent vulnerabilities, for example as illustrated in conjunction with FIG. **11**, **12**, or **13**. In some embodiments a decision to quarantine may be based on the accumulation of evidence above a threshold, such as a threshold of one such piece of evidence, or a threshold of two such pieces of evidence. In some embodiments, a decision to quarantine may be made based on configuration rules, for example all new computers must be quarantined until they have contacted their OS vendor and been updated, or by user specification, such as a user requesting a quarantine for a specific computer. In some embodiments, requests for a quarantine may be made on an individual computer via settings, such as control panel in Microsoft Windows, or through a user interface to a firewall or router, for example by specifying an IP address, name, or MAC address to quarantine, or a list of addresses that are not to be quarantined (where other addresses may be quarantined). During a quarantine period outbound traffic is detected (**1402**), for example by a device such as a router, or by a software network stack on the quarantined computer, or by an upstream device, such as a device at an ISP. Outbound traffic is tested in this example to see if it is associated with a remediation request (**1403**), such as contact with an appropriate anti-contagion software distributor, or contact with a security patch update provider, such as Microsoft Windows Update, or contact with a verified remediation site, such as an internal ISP site. If outbound traffic is deemed to be associated with a remediation request (**1403**), then the traffic is routed or forwarded to its destination (**1404**) in this example. If outbound traffic is not associated with a remediation request (**1403**), then, in this example, it is tested to see if it is an approved service request (**1405**), such as a request for a web page. If the request was for an approved service (**1405**) then, in this example, the request is re-routed to a special quarantine server (**1407**), such as an ISP web server that

provides information and links to assist in remediation. Rerouting of request traffic may be accomplished in numerous ways, for example by providing an alternate DNS service that consistently returns a fixed IP address of a quarantine server in response to all DNS requests. Another example of rerouting a web page request is to respond with a redirect message, which would direct a browser on a quarantined computer to contact a quarantine server that provides remediation information. If the request was not for an approved service (**1406**), then the outbound traffic is discarded or responded to with in indication that the desired traffic destination is unreachable (**1406**) in this example.

FIG. **15** is a flow diagram of a method for quarantining a computer according to some embodiments. In this example, a decision has been made to quarantine a computer (**1501**), for example as described in conjunction with **1401** of FIG. **14**. Inbound traffic originally destined for a quarantined computer is detected (**1502**), for example by a network device such as by a router. An example of detecting inbound traffic destined for a quarantined computer is to have a list of quarantined IP addresses, and filter or provide additional processing on traffic destined for an address on the list. Detected inbound traffic is tested in this example to see if it is remediation related (**1503**), such as a response to a remediation request. Direct responses to remediation requests may for example be identified using stateful packet inspection firewall techniques known to those skilled in the art. In various embodiments, at least certain types of traffic from approved sites such as anti-contagion vendors, security update providers, and/or internal ISP sites are deemed remediation related. If traffic is remediation related (**1503**) then it is forwarded to the quarantined computer (**1504**) in this example. If traffic is not remediation related (**1503**) then, in this example, it is discarded or responded to with a message indicating that the destination is unreachable (**1507**).

In some embodiments, a configuration based quarantine during an installation of an operating system, such as Microsoft Windows, may be automatically setup, for example by a default installation setting up restrictions in a firewall installed with the OS or by an installation setting defaults for use when network connectivity is enabled, that establishes a quarantine, for example restricting internet connections, for example allowing internet connections only to a security update site such as Microsoft Windows Update. In some embodiments, settings that may establish a quarantine, such as default settings in an operating system installation, may be altered programmatically to remove the quarantine in response to a remediation site, such as a security update service such as Microsoft Windows Update, asserting that there are no other security updates pending. In some embodiments, configuration settings that may establish a quarantine may be altered programmatically to enable a quarantine in response to the presence of one or more security updates, such as provided by an operating system update service such as Microsoft Windows Update, that are available for installation on a computer.

FIG. **16** is a flow diagram of a method for a quarantine server to respond to requests according to some embodiments. In this example, a quarantine server is started (**1601**), for example by starting a web server, or DNS server, or proxy server, or ftp server, or some combination server. The server waits for a request (**1602**), for example by listening on a port and waiting for a connection. In some embodiments, a device such as a router may forward outbound traffic from a quarantined computer to a quarantine server, for example as illustrated in conjunction with **1407** of FIG. **14**. If a received connection is a web server request (**1603**), such as an HTTP request, then the server responds with a quarantine notification page (**1604**) in this example. An example of a quarantine notification page is a web page that provides notification that the computer is quarantined, and/or provides links to remediation sites appropriate to the quarantine, such as a link to a site that provides anti-contagion software for removing a virus that the quarantined computer is believed to contain. The links on the quarantine notification page may be examples of HTTP addresses that are for use in remediation. Another example of a quarantine notification page is a page that redirects to a quarantine notification page. If the request is not a web server request (**1603**), then it is tested to see if it is a DNS inquiry (**1605**) in this example. A DNS inquiry may for example appear on port **53** using TCP/IP or UDP. If the request is not a DNS inquiry (**1605**) then it may be discarded or responded to as "unreachable" (**1606**) in this example. If the request is a DNS inquiry (**1605**) then, in this example, the inquiry is tested to see if it is a DNS request for a remediation host name (**1608**). A remediation host name may include a host name as appropriate to the reason for a quarantine, such as a host providing anti-contagion software, or a host providing security updates, or a host internal to (or partnered with) an ISP. If the DNS inquiry was for a remediation host name (**1608**) then the request is proxied to an external DNS service provider (**1609**) in this example. A proxy to a DNS service (**1609**) may return a stored or cached response, such as the actual IP address of the host name, or it may contact an external DNS service to relay back the proposed IP address. If the DNS inquiry was not for a remediation host name (**1608**) then an IP address for a quarantine server is provided (**1607**) in this example. In some embodiments, when a redirected IP address is provided, such as that of a quarantine server, the DNS response may include a short time to live, for example **0** seconds. A short time to live ensures that a further DNS query will be sent prior to subsequent communications being sent from the quarantined computer to the host that was the subject of the DNS query, aiding a return to normal routing of traffic once the computer is no longer under quarantine. In some embodiments, the server(s) in this example may continue to wait for a service request (**1602**) in parallel with the processing of the requests as described above, or may wait for a service request again after disposition of the previous request as described above.

FIG. **17** is a flow diagram of a method for scanning backdoors and inoculating according to some embodiments. In this example, preparations are made for scanning a computer for a backdoor inoculation path (**1701**), for example by identifying the computer to be scanned, and enumerating or retrieving a list of one or more backdoors. In one example, an IP address to be scanned may be selected by an ISP from a list of subscribers. In one example, a list of backdoors may include information associated with preloaded software approved by the computer owner for access and inoculation purposes, such as an ISP provided utility that is installed by the subscriber. In some embodiments, an ISP defined backdoor may have an authentication protocol to preclude control by unauthorized parties, such as restrictions on IP address(es) that may contact the backdoor, or shared secrets such as a password required by the sender, or cryptographic authentication such as client authentication on an SSL connection. In some embodiments, a user is requested and/or required to authorize in advance use of an authorized and/or any unauthorized backdoor for purposes of inoculation. As another example, a list of backdoors may include virus or worm installed backdoors, such as those installed by a SubSeven virus or Back Orifice. As another example, backdoors may include exploitable security vulnerabilities, such as a weak-

ness in a web server, or a weakness in an RPC service. A scan may optionally be conducted to detect open TCP/IP backdoor ports (**1702**) similarly to the scan discussed in conjunction with FIG. **11**. If none of the backdoors are open (**1703**), for example there is no response on any of the ports, then the scan is complete (**1706**) in this example. If one or more backdoor ports are open (**1703**) then they are tested for usability (**1704**) in this example. In some embodiments, a backdoor port may be assumed open (**1703**) and may be tested for usability (**1704**) without an independent test to see if the port is open. An example of a usability test is no test at all, with the conclusion that a specific port is usable any time it is open. Another example of a usability test is based on the receipt of confirmation of a functional installation of ISP provided backdoor software, such as a confirmation that may be provided by the installed software. Another example of a usability test involves the use of a less invasive test of vulnerabilities such a buffer overrun exploit that does not have a permanent impact but provides observable validation (such as a dropped connection). If a backdoor is usable (**1705**) then the computer is inoculated via a backdoor (**1707**) in this example. One example of such an inoculation is the remotely controlled download of one or more remedy files into one or more directories, such as an operating system directory, such as the "\windows\system" directory in Microsoft Windows 2000. Another example of an inoculation is the download of an executable file to the computer, followed by the execution of that file. Another example of an inoculation is the download of a data file, such as an anti-contagion signature file, that is interpreted or processed by another executable already on the computer. Another example of an inoculation is the deletion of one or more files from the computer. Another example of an inoculation is the editing of one or more registry entries, or of one or more files, such as deleting, appending or rewriting entries in a registry or file. Another example of inoculation includes a combination of one or more of above techniques, optionally in conjunction with one or more remotely instigated reboot operations. Another example of inoculation includes a combination of one or more of the above techniques, along with an optional dialog box requesting a user's permission to proceed with the inoculation. Another example of inoculation is the use of a dialog box in conjunction with a timer that causes the inoculation to proceed if no user response is given in a prescribed period of time, such as 1 hour.

FIG. **18** is a flow diagram of a method for categorizing messages according to some embodiments. In this example, a message is received (**1801**), such as an email message or instant message. The purported sender's identity is extracted (**1802**), for example by examining the "From:" header line in an email message, or by examining the envelope information conveying an email message, or by examining the sender's ID in an instant message. A check is performed to see if the sender has sent any prior messages (**1803**), for example by scanning an inbox of prior email, or by checking for the sender in a buddy list. If there were no prior messages from the sender (**1803**) then the message is processed (**1804**) in this example. Processing the message may for example include presenting the message to the user, and/or recording routing information for comparison and categorization of future messages. If there were prior messages from the sender (**1803**) then routing information used in prior messages is retrieved (**1805**) in this example. Routing information may for example be extracted from the header in email. Routing information may for example be obtained during an instant message session by retrieving the IP address of the party being communicated with, for example as is done when an instant message session coordinates a peer-to-peer activity such as a direct file transfer with a remote party. In some embodiments, routing information may be stored implicitly, such as in headers of a previously received message, or explicitly, for example by extracting routing information and storing it in a database or file system. Routing information for the current message is compared to prior messages (**1806**), for example by checking that the IP address for the party being communicated with is in a similar subnet, or allocated within the same geographical region, for example the same continent, as a previous IP address for the same party. IP address to geographical location may be estimated by various software applications and services, such as services provided by MaxMind and described on http://www.maxmind.com. In some embodiments, routing information for an email message may be deduced by examining header information related to mail transfer agents. In some embodiments, similarity in IP addresses for mail transfer agents may be interpreted as implying similar routing. If routing is similar to previous messages from the same sender (**1806**), then the message is processed as legitimate (**1808**) in this example. If the routing information for a message purporting to be from a sender is not similar to previous messages from the same sender (**1806**) then the message is processed as suspicious (**1807**) in this example. In some embodiments, processing as suspicious may include contributing to a scoring that labels the message as an undesirable message, such as spam, viral, or phishing. In some embodiments, higher probability suspicious messages may be scrutinized further, or may be restricted to some extent in their impact on the computer. An example of restricting impact of a message may include discarding the message. Another example of restricting impact or further scrutiny of a message may include placing a message in a spam inbox. Another example of a restricting impact of a message may include preventing execution of content contained in the message, and/or requiring additional user interactions to bypass the restriction, for example by having a user agree to ignore warnings which describe the suspicious circumstances. In various embodiments, the process of FIG. **18** is implemented on a mail, messaging, or other server associated with the sender and/or the sender's ISP; a mail, messaging, or other server associated with the recipient and/or the recipient's ISP; and/or one or more intermediate nodes.

FIG. **19**A is a flow diagram of a method for notifying possible sources of infection according to some embodiments. In this example, an infected message is received (**1901**). An example of an infected message is a message classified as such by anti-contagion software. Routing information is extracted from the message (**1902**). An example of extracting routing information in an email message is to extract header lines inserted by mail transfer agents that propagated the message. Another example of extracting routing information is to acquire an IP address of a party making contact via instant messaging, for example as is provided by AOL's instant message infrastructure for a peer-to-peer file transfer. Routing information from previously received messages is retrieved (**1903**). An example of retrieving email routing information from prior messages is scanning all received messages, such as messages in an email inbox, and extracting routing information. Another example of retrieving routing information is to query a database or file system where routing information from previous messages has been gathered. Routing information for the infected message is compared with the retrieved routing information (**1904**). If no similarity with previous messages is found (**1904**) then in this example a third party notification is made (**1906**). An example of a third party notification is sending notification to the ISP

that provides network access to the message source. Another example of notifying a third party is notifying a viral clearinghouse of the apparent source of a viral message. Another example of a notifying a third party is notifying a service that determines the credibility of a report, such as a collaborative filter service, that accumulates accusations of viral disseminations. In some embodiments, similarity of routing of an infected message with a previous message may include having a previous message from the IP address of the infected message source. If a similarity is found with previous messages (**1904**) then in this example a notification is sent (**1905**). In some embodiments, notification may be sent (**1905**) to one or more sources of the prior message(s) that best match the infected message, and/or to third parties. As an example, notification may be sent to an email source that had the same originating IP address as the infected message. In some embodiments, a user may be asked to approve notification(s), for example via a pop-up dialog. In some embodiments, notification may be automatically sent without user intervention.

FIG. **19**B is a block diagram illustrating a network environment in which suspicious messages are identified and/or one or more notifications are sent when a malicious message is detected. A message source **1920** sends a message via the Internet **1922** to a message processing system **1926** for delivery via the Internet to one or more message destinations **1924**. In various embodiments, the message processing system **1926** comprises a mail or messaging server associated with the message source **1920**, a mail or messaging server associated with the destination(s) **1924**, and/or an intermediate processing node. In some embodiments, the message source **1920** is connected directly and/or via a local network, and not via the Internet **1922** as shown in FIG. **19**B, to message processing system **1926**. In some embodiments, message processing system **1926** is configured to identify suspicious messages using current and historic routing information, e.g., as described above in connection with FIG. **18**. In some embodiments, the process of FIG. **19**A is implemented on message processing system **1926**. For example, in some embodiments if message processing system **1926** determines that a message sent by message source **1920** is infected, message processing system **1926** compares routing information associated with the malicious message with corresponding information for one or more messages received previously from the message source **1920**. If the routing information is similar, in some embodiments the message processing system **1926** notifies the message source **1920**. If the routing information is not similar, in various embodiments the message processing system **1926** notifies the message source **1920**, the sender's ISP **1928**, and/or a clearinghouse or similar applicable service **1930**, as described above.

Although the foregoing embodiments have been described in some detail for purposes of clarity of understanding, the invention is not limited to the details provided. There are many alternative ways of implementing the invention. The disclosed embodiments are illustrative and not restrictive.

What is claimed is:

1. A method for protecting a network, comprising:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition; and

permitting the first host to communicate with the remediation host.

2. A method as recited in claim **1**, wherein detecting an insecure condition further includes at least one of the following: scanning for a vulnerability, scanning for malicious data, determining whether a security software is installed, and detecting anomalous network traffic.

3. A method as recited in claim **1**, wherein detecting an insecure condition includes determining that the first host should be quarantined until an update to an operating system has been installed.

4. A method as recited in claim **1**, wherein detecting an insecure condition includes configuring an operating system to quarantine the first host upon initial startup after installation of the operating system.

5. A method as recited in claim **1**, wherein permitting the first host to communicate with the remediation host includes:

detecting an outbound communication from the first host; and

forwarding the outbound communication if it is addressed to the remediation host.

6. A method as recited in claim **1**, wherein preventing the first host from sending data to the one or more other hosts includes:

detecting an outbound communication from the first host; and

redirecting the outbound communication to a quarantine server if it comprises a request for an approved service and is not addressed to a remediation host.

7. A method as recited in claim **1**, wherein quarantining the first host further includes preventing the first host from receiving via the protected network data not related to remediation of the insecure condition.

8. A method as recited in claim **1**, performed at an Internet service provider.

9. A method as recited in claim **1**, wherein the software component on the first host is an operating system.

10. A method as recited in claim **1**, wherein determining that the response does not include a valid digitally signed attestation of cleanliness includes determining that the software component on the first host is not sufficiently updated.

11. A method as recited in claim **1**, wherein determining that the software component on the first host is not sufficiently updated includes determining that a patch level associated with the software component on the first host is not sufficiently recent.

12. A system for protecting a network, comprising:

a processor configured to:

detect an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantine the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition; and

permit the first host to communicate with the remediation host; and

a memory coupled to the processor and configured to provide instructions to the processor.

13. A system as recited in claim **12**, wherein the processor is configured to detect an insecure condition at least in part by performing one or more of the following: scanning for a vulnerability, scanning for malicious data, determining whether a security software is installed, and detecting anomalous network traffic.

14. A system as recited in claim **12**, wherein the processor is configured to detect an insecure condition at least in part by determining that an initial startup after installation of an operating system is being performed.

15. A system as recited in claim **12**, wherein the processor is configured to quarantine the first host at least in part by preventing the first host from receiving via the protected network data not related to remediation of the insecure condition.

16. A system as recited in claim **12**, wherein the software component on the first host is an operating system.

17. A system as recited in claim **12**, wherein determining that the response does not include a valid digitally signed attestation of cleanliness includes determining that the software component on the first host is not sufficiently updated.

18. A system as recited in claim **17**, wherein determining that the software component on the first host is not sufficiently updated includes determining that a patch level associated with the software component on the first host is not sufficiently recent.

19. A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition; and

permitting the first host to communicate with the remediation host.

* * * * *

EXHIBIT 2

US009516048B1

(12) **United States Patent**
Emigh et al.

(10) **Patent No.: US 9,516,048 B1**
(45) **Date of Patent: Dec. 6, 2016**

(54) **CONTAGION ISOLATION AND INOCULATION VIA QUARANTINE**

(71) Applicants: **Aaron Emigh**, Incline Village, NV (US); **James Roskind**, Redwood City, CA (US)

(72) Inventors: **Aaron Emigh**, Incline Village, NV (US); **James Roskind**, Redwood City, CA (US)

(73) Assignee: **Radix Holdings, LLC**, Incline Village, NV (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **15/206,227**

(22) Filed: **Jul. 9, 2016**

Related U.S. Application Data

(63) Continuation of application No. 11/237,004, filed on Sep. 27, 2005, now Pat. No. 9,398,037.

(60) Provisional application No. 60/613,909, filed on Sep. 27, 2004.

(51) **Int. Cl.**

G06F 21/50	(2013.01)
G06F 21/56	(2013.01)
H04L 12/22	(2006.01)
H04L 12/24	(2006.01)
H04L 12/26	(2006.01)
H04L 29/06	(2006.01)

(52) **U.S. Cl.**
CPC ***H04L 63/1416*** (2013.01); ***G06F 21/56*** (2013.01); ***H04L 41/082*** (2013.01); ***H04L 41/20*** (2013.01); ***H04L 63/102*** (2013.01); ***H04L 63/126*** (2013.01); ***H04L 63/1425*** (2013.01); ***H04L 63/1433*** (2013.01)

(58) **Field of Classification Search**
CPC .. G06F 21/50–21/577; H04L 41/08–41/0896; H04L 41/20; H04L 41/28; H04L 63/08–63/205
USPC 726/22–25, 2–15
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,823,479	B1	11/2004	McElhaney et al.
6,996,845	B1	2/2006	Hurst et al.
7,263,720	B2	8/2007	Silvester
7,287,278	B2	10/2007	Liang
7,424,610	B2	9/2008	Ranganathan
2002/0083328	A1	6/2002	Riordan
2003/0188173	A1	10/2003	Zimmer et al.
2003/0188194	A1	10/2003	Currie et al.
2005/0018618	A1	1/2005	Mualem et al.
2005/0033987	A1	2/2005	Yan et al.
2005/0131997	A1	6/2005	Lewis et al.
2006/0053426	A1	3/2006	Dive-Reclus et al.

OTHER PUBLICATIONS

Jing F. Sims, "Office Action, U.S. Appl. No. 11/237,003," U.S. Patent and Trademark Office, Jun. 21, 2012. Relevant passage on pp. 13-15 in the section entitled "Examiner's Statement of Reasons for Allowance."

Mark V. Glover, "Integrating a Trusted Computing Base Extension Server and Secure Session Server into the LINUX Operating System." Sep. 2001, Naval Postgraduate School.

"OLS: Linux and Trusted Computing." Jul. 22, 2005; http://lwn.net/Articles/144681/ (as retrieved on Jul. 15, 2010).

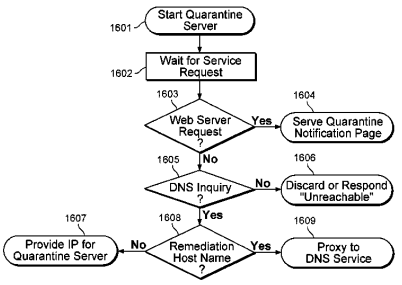
A. Bohra et al, "Remote Repair of Operating System State Using Backdoors." Proceedings of International Conference on Autonomic Computing on May 17-18, 2004, pp. 256-263.

Primary Examiner — Timothy J Weidner

(57) **ABSTRACT**

In some embodiments, techniques for protecting a network may include detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, including contacting a trusted computing base and determining whether a response includes a valid digitally signed attestation of cleanliness (an attestation that the first host is not infested and/or that the presence of a patch or patch level of a software component on the first host); if not, quarantining the first host, including by re-routing a service request sent by the first host by responding with a redirect that causes a browser on the first host to be directed to a quarantine server configured to serve a quarantine notification page that provides remediation information; and permitting the first host to communicate with a remediation host configured to provide data usable to remedy the insecure condition.

20 Claims, 25 Drawing Sheets



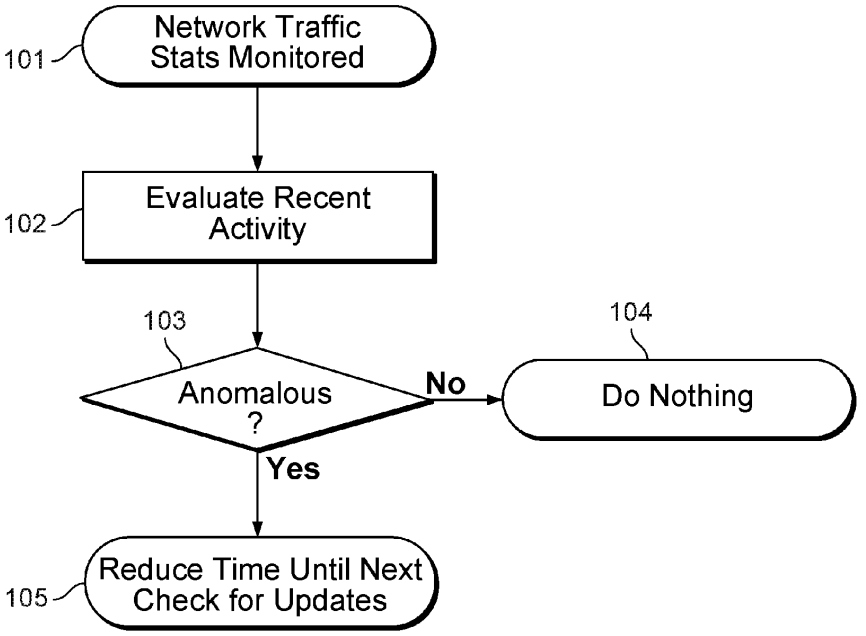

101
Network Traffic Stats Monitored
102
Evaluate Recent Activity
103
Anomalous ?
No
104
Do Nothing
Yes
105
Reduce Time Until Next Check for Updates

FIG. 1

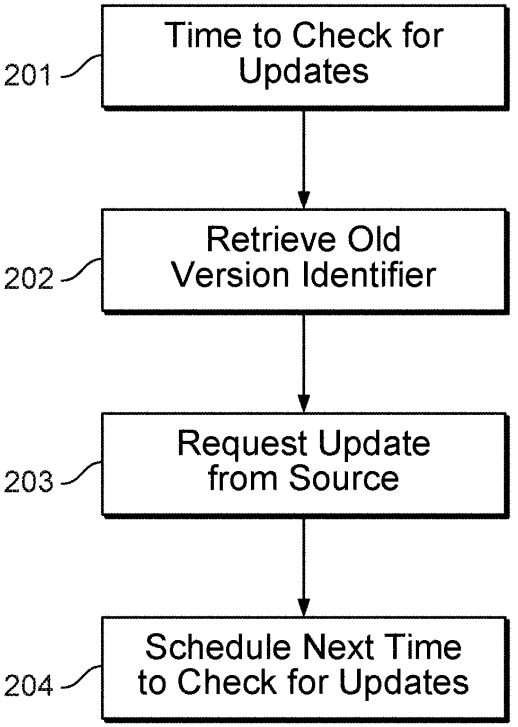

201
Time to Check for Updates
202
Retrieve Old Version Identifier
203
Request Update from Source
204
Schedule Next Time to Check for Updates

FIG. 2A

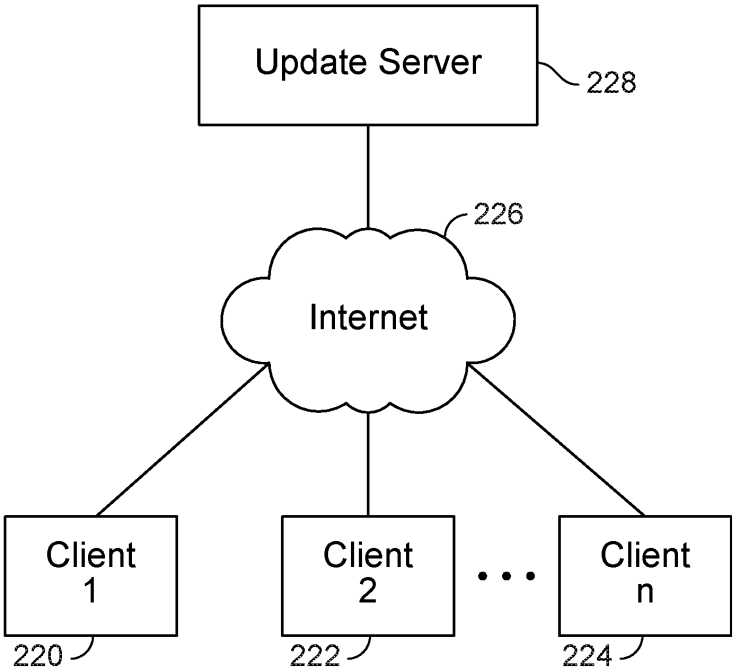

Update Server
228
226
Internet
Client 1
Client 2
Client n
220
222
224

FIG. 2B

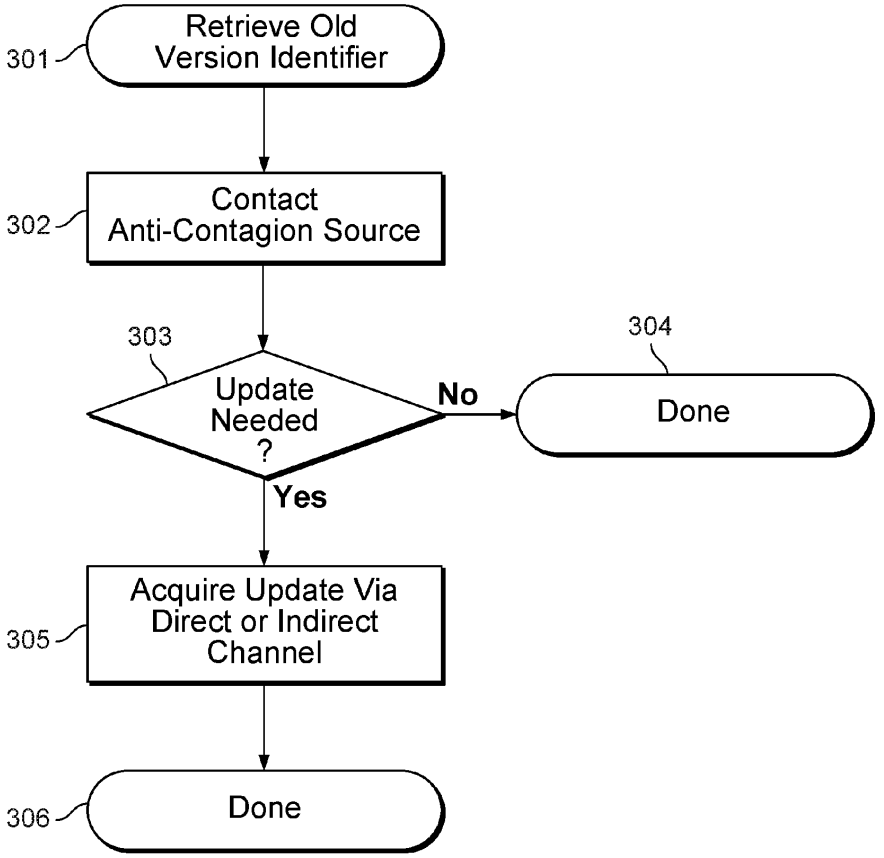

301
Retrieve Old Version Identifier
302
Contact Anti-Contagion Source
303
Update Needed ?
No
304
Done
Yes
305
Acquire Update Via Direct or Indirect Channel
306
Done

FIG. 3A

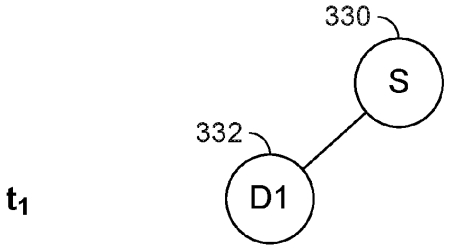


FIG. 3B

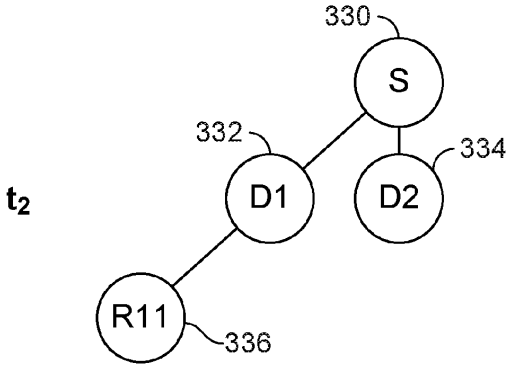


FIG. 3C

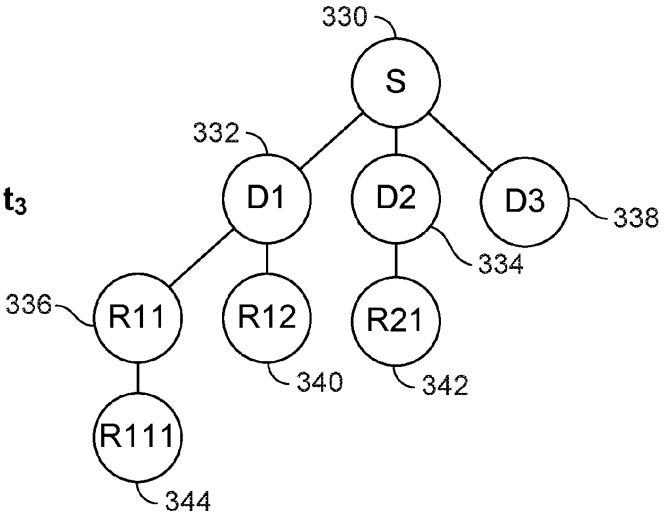


FIG. 3D

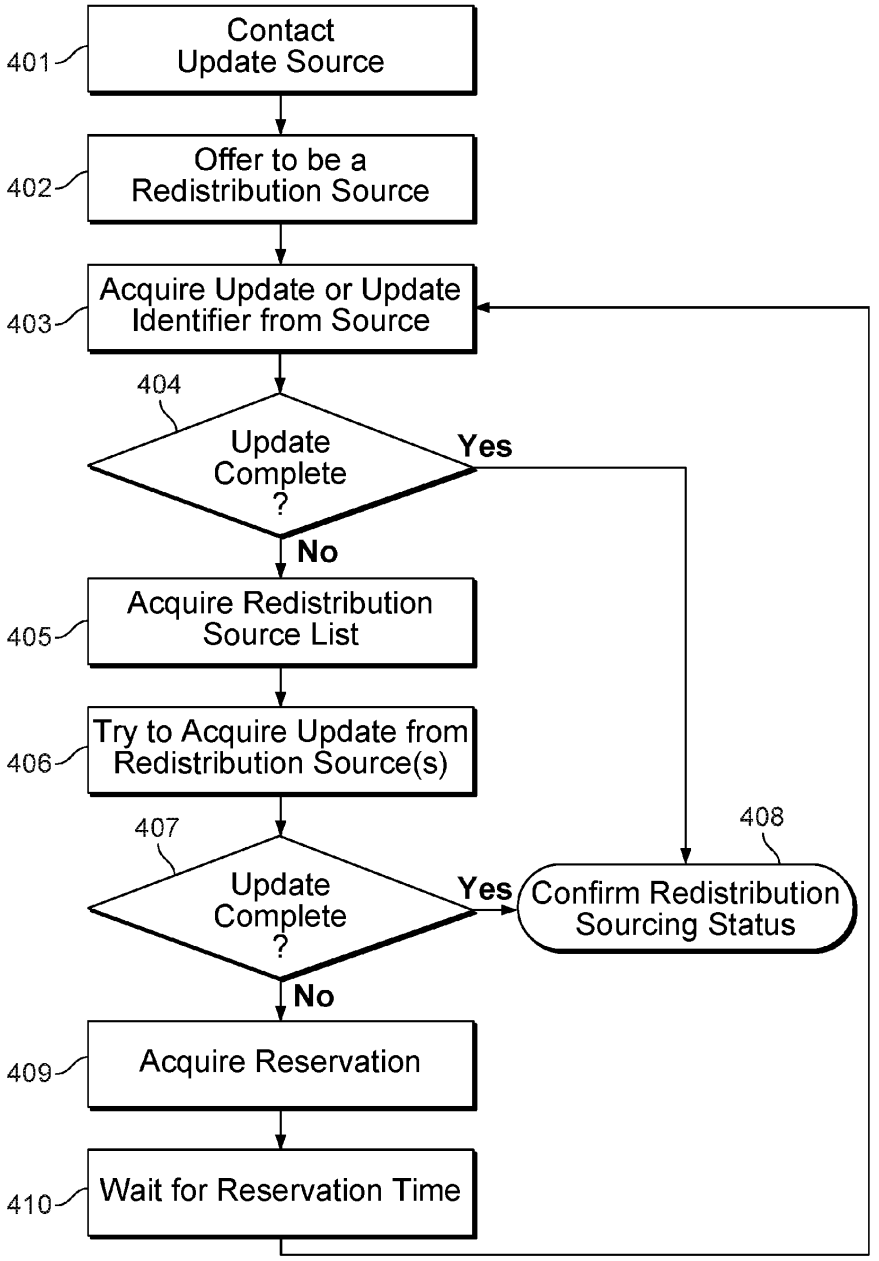

401
Contact Update Source
402
Offer to be a Redistribution Source
403
Acquire Update or Update Identifier from Source
404
Update Complete ?
Yes
No
405
Acquire Redistribution Source List
406
Try to Acquire Update from Redistribution Source(s)
407
Update Complete ?
Yes
No
408
Confirm Redistribution Sourcing Status
409
Acquire Reservation
410
Wait for Reservation Time

FIG. 4A

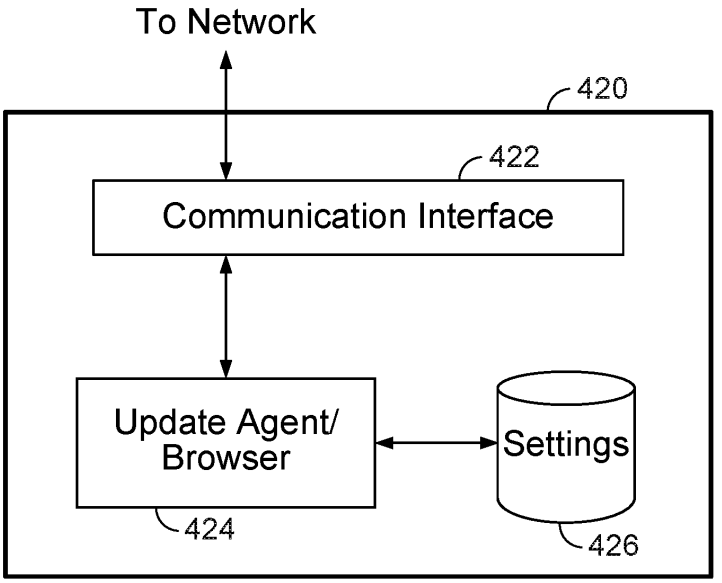

To Network
420
422
Communication Interface
Update Agent/
Browser
Settings
424
426

FIG. 4B

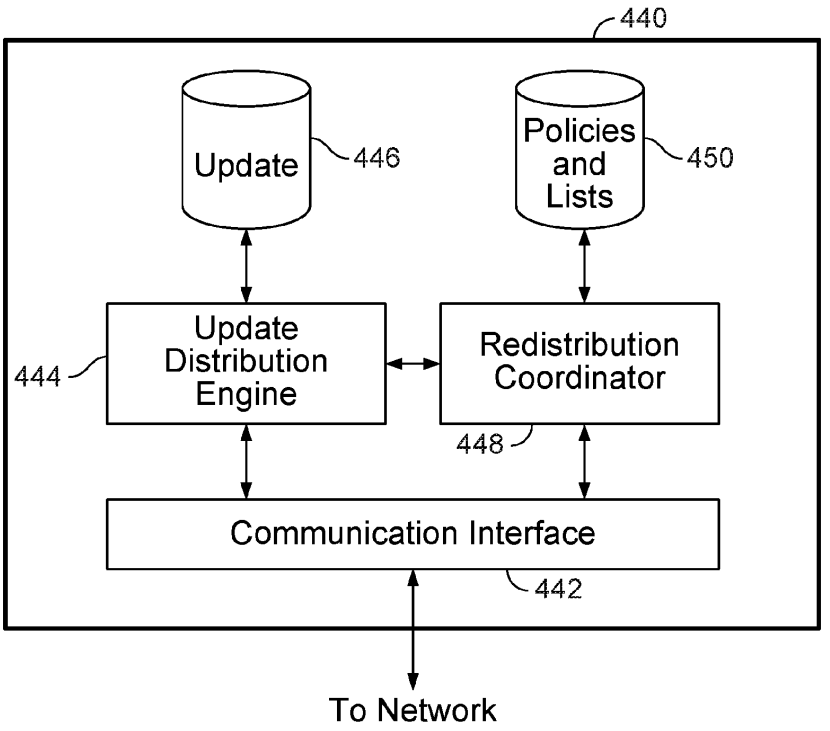

440
Update
446
Policies and Lists
450
Update Distribution Engine
444
Redistribution Coordinator
448
Communication Interface
442
To Network

FIG. 4C

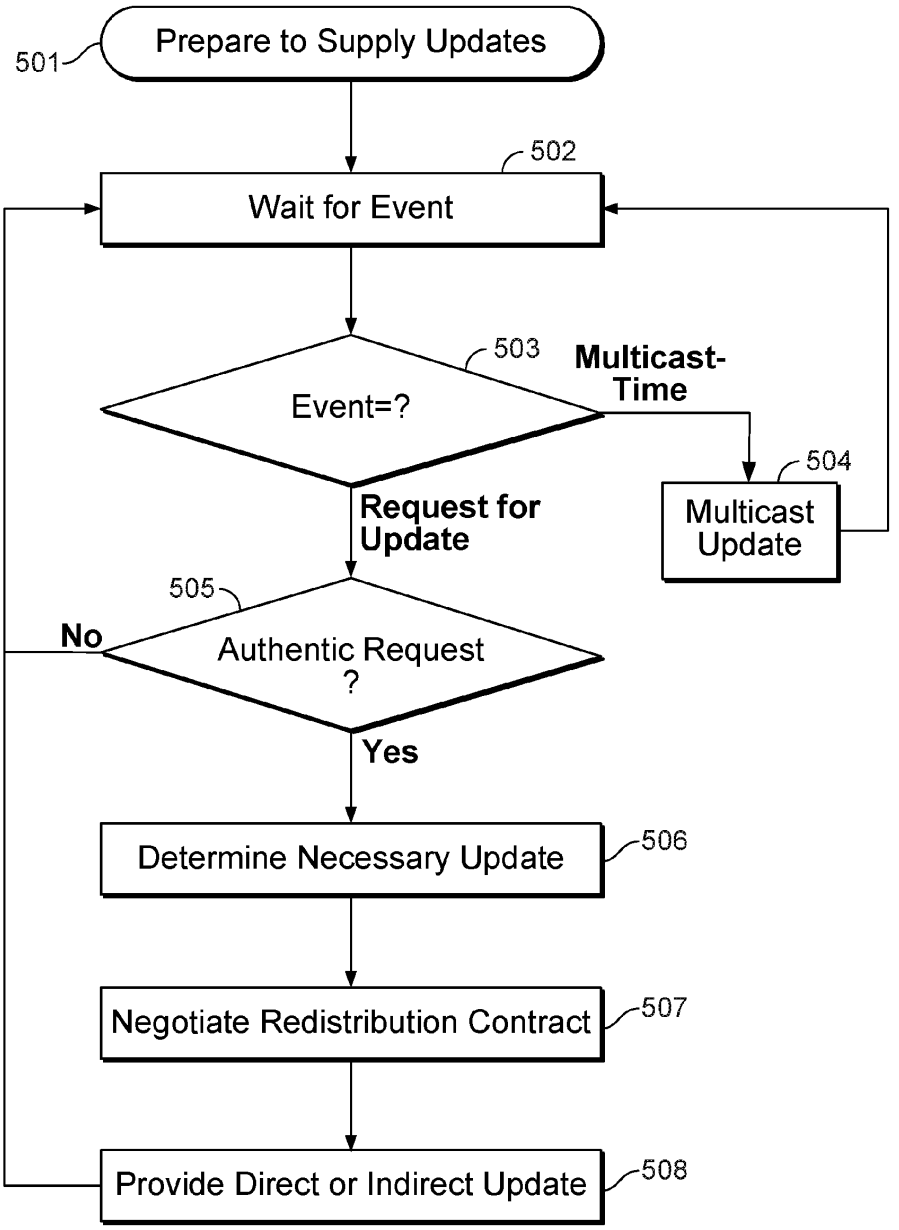

501
Prepare to Supply Updates
502
Wait for Event
503
Event=?
Multicast-Time
504
Multicast Update
Request for Update
505
No
Authentic Request ?
Yes
Determine Necessary Update
506
Negotiate Redistribution Contract
507
Provide Direct or Indirect Update
508

FIG. 5

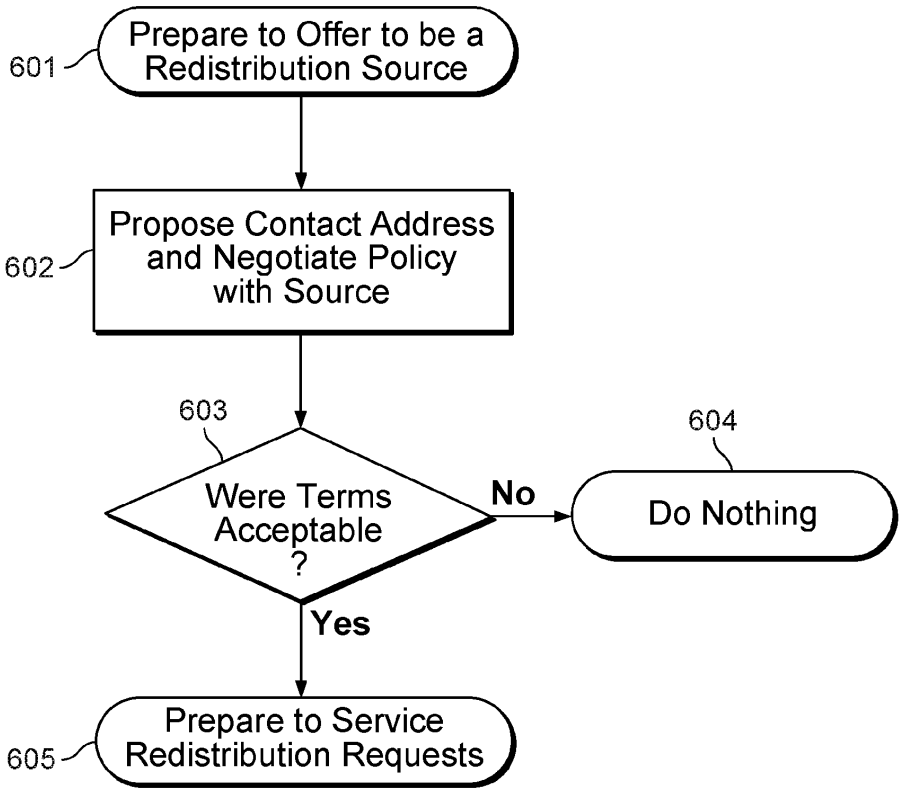

601
Prepare to Offer to be a Redistribution Source
602
Propose Contact Address and Negotiate Policy with Source
603
Were Terms Acceptable ?
No
604
Do Nothing
Yes
605
Prepare to Service Redistribution Requests

FIG. 6

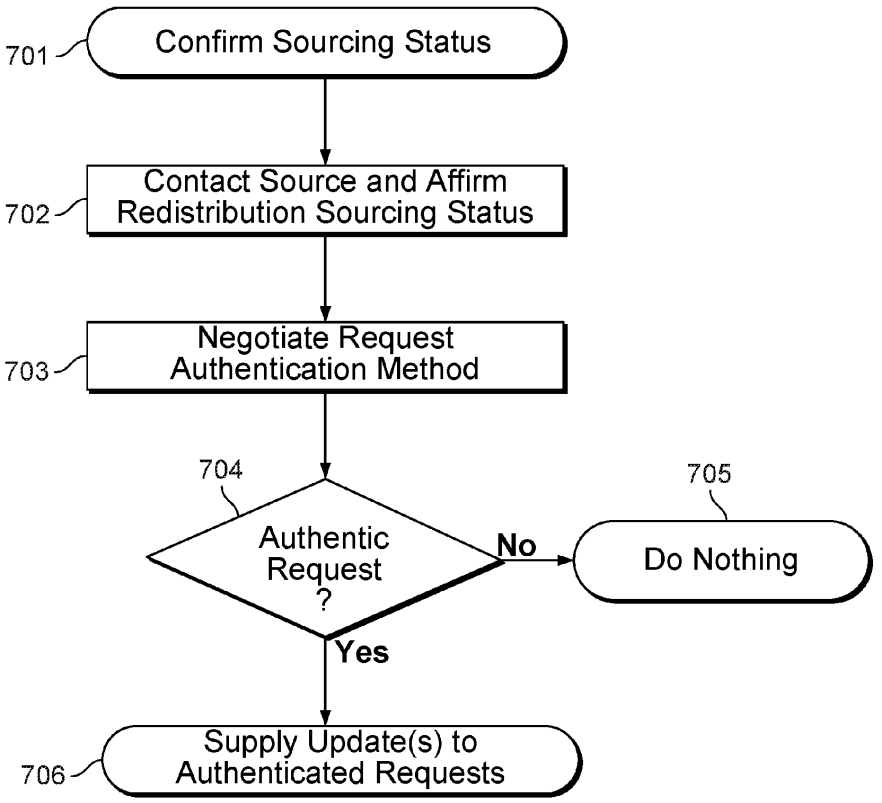

701
Confirm Sourcing Status
702
Contact Source and Affirm Redistribution Sourcing Status
703
Negotiate Request Authentication Method
704
Authentic Request ?
No
705
Do Nothing
Yes
706
Supply Update(s) to Authenticated Requests

FIG. 7

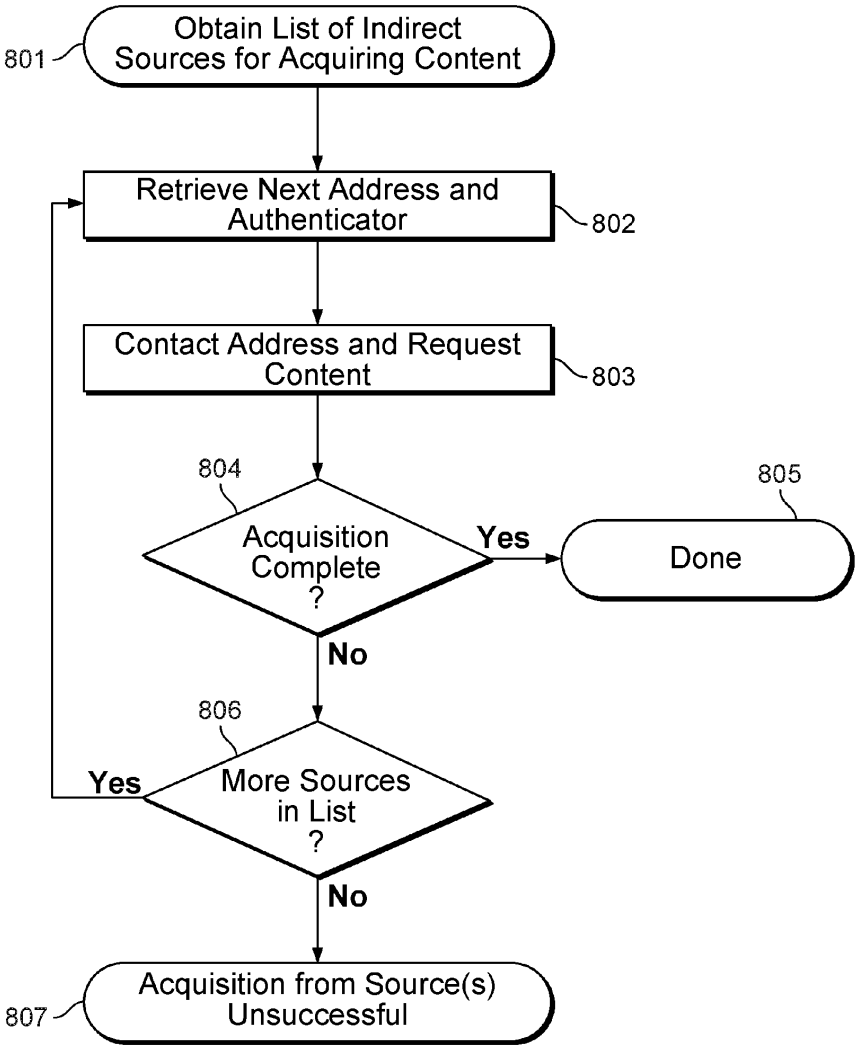

801
Obtain List of Indirect Sources for Acquiring Content
Retrieve Next Address and Authenticator
802
Contact Address and Request Content
803
804
Acquisition Complete ?
Yes
805
Done
No
806
More Sources in List ?
Yes
No
807
Acquisition from Source(s) Unsuccessful

FIG. 8

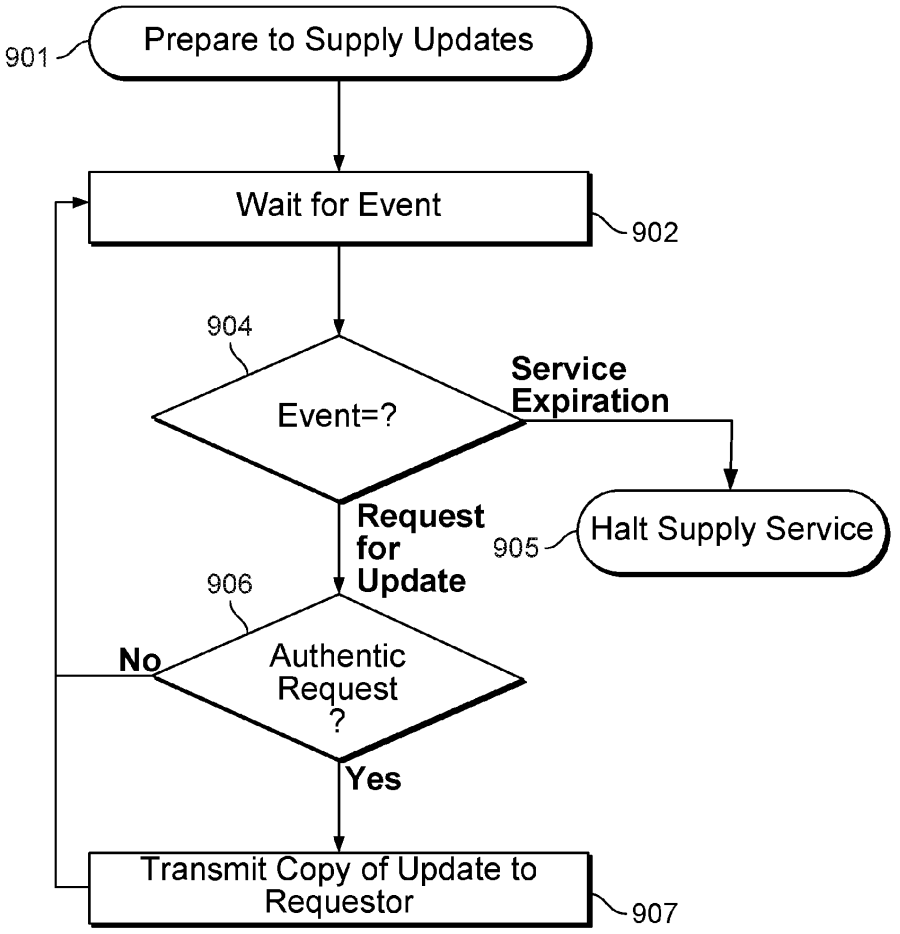

901
Prepare to Supply Updates
Wait for Event
902
904
Event=?
Service Expiration
Halt Supply Service
905
Request for Update
906
Authentic Request ?
No
Yes
Transmit Copy of Update to Requestor
907

FIG. 9

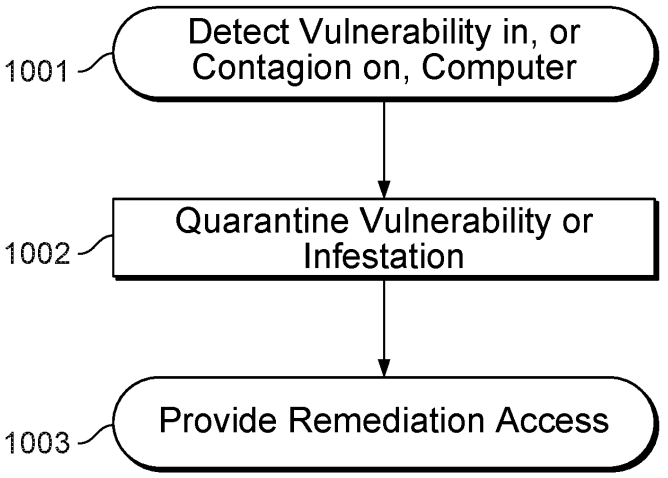

1001
Detect Vulnerability in, or Contagion on, Computer
1002
Quarantine Vulnerability or Infestation
1003
Provide Remediation Access

FIG. 10A

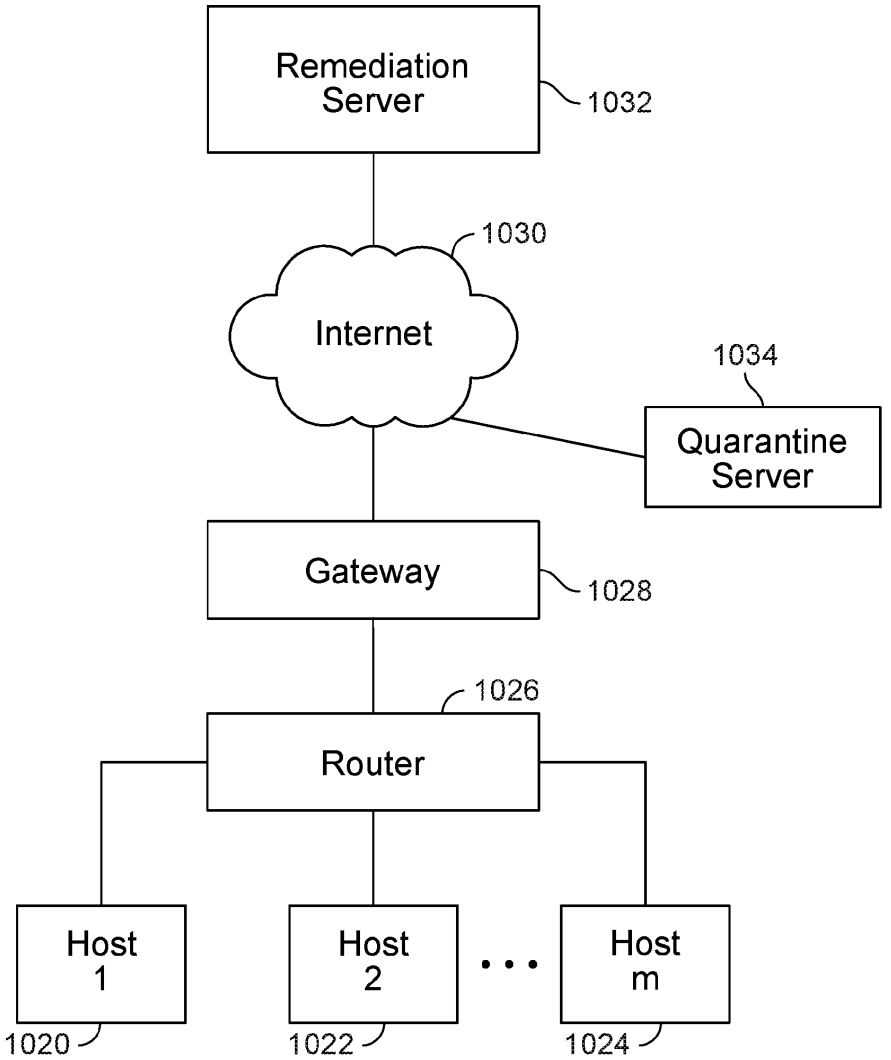

Remediation Server
1032
1030
Internet
1034
Quarantine Server
Gateway
1028
1026
Router
Host 1
Host 2
Host m
1020
1022
1024

FIG. 10B

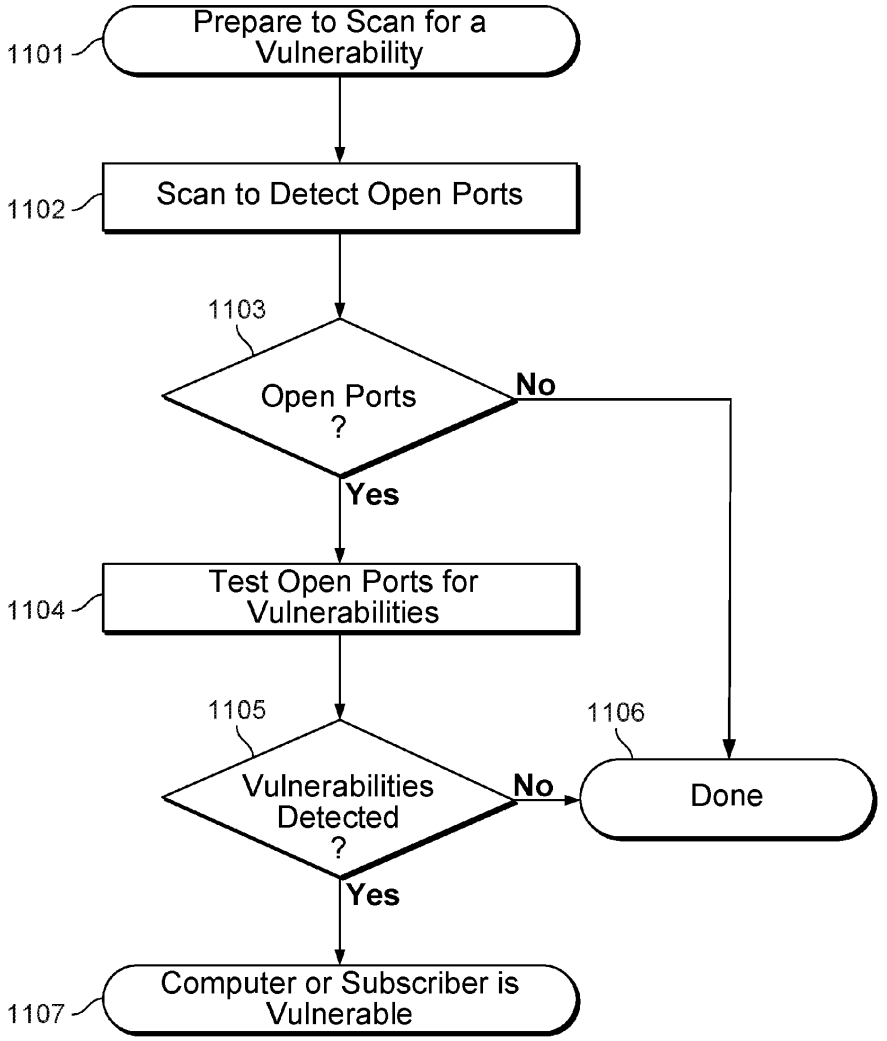

1101
Prepare to Scan for a Vulnerability
1102
Scan to Detect Open Ports
1103
Open Ports ?
No
Yes
1104
Test Open Ports for Vulnerabilities
1105
Vulnerabilities Detected ?
No
1106
Done
Yes
1107
Computer or Subscriber is Vulnerable

FIG. 11

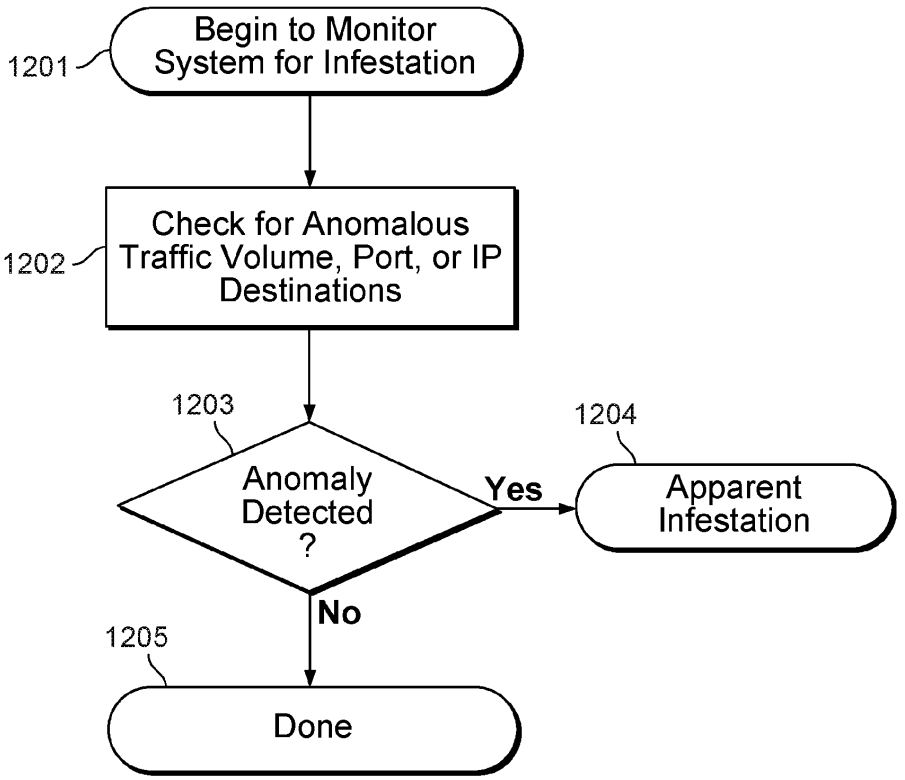

1201
Begin to Monitor System for Infestation
1202
Check for Anomalous Traffic Volume, Port, or IP Destinations
1203
Anomaly Detected ?
Yes
1204
Apparent Infestation
No
1205
Done

FIG. 12

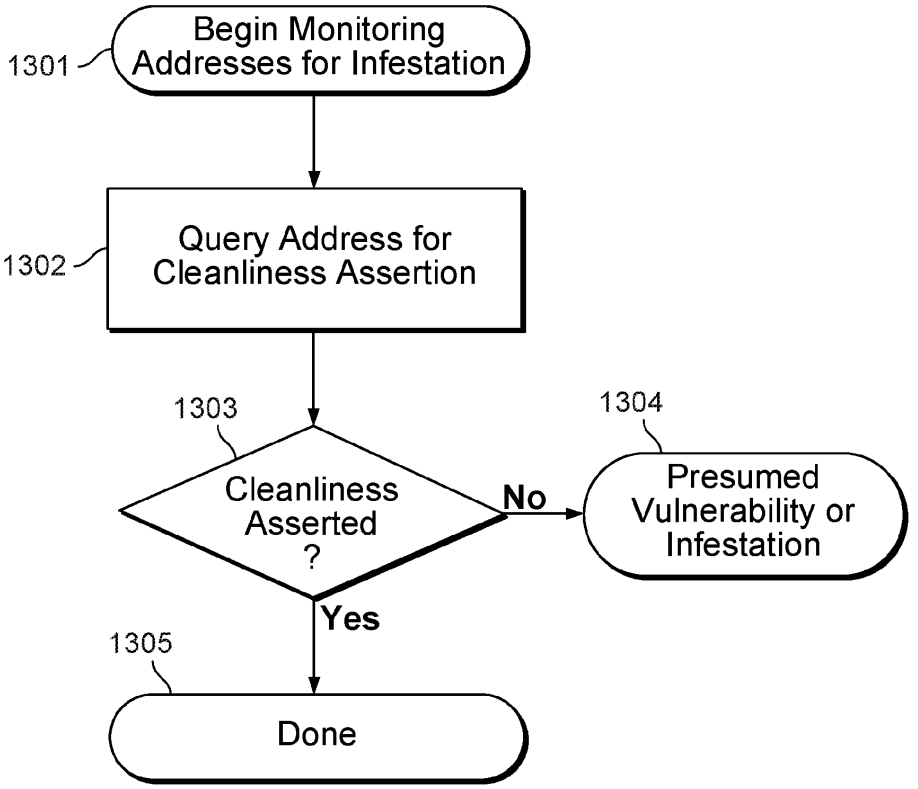

1301
Begin Monitoring Addresses for Infestation
1302
Query Address for Cleanliness Assertion
1303
Cleanliness Asserted ?
No
1304
Presumed Vulnerability or Infestation
Yes
1305
Done

FIG. 13

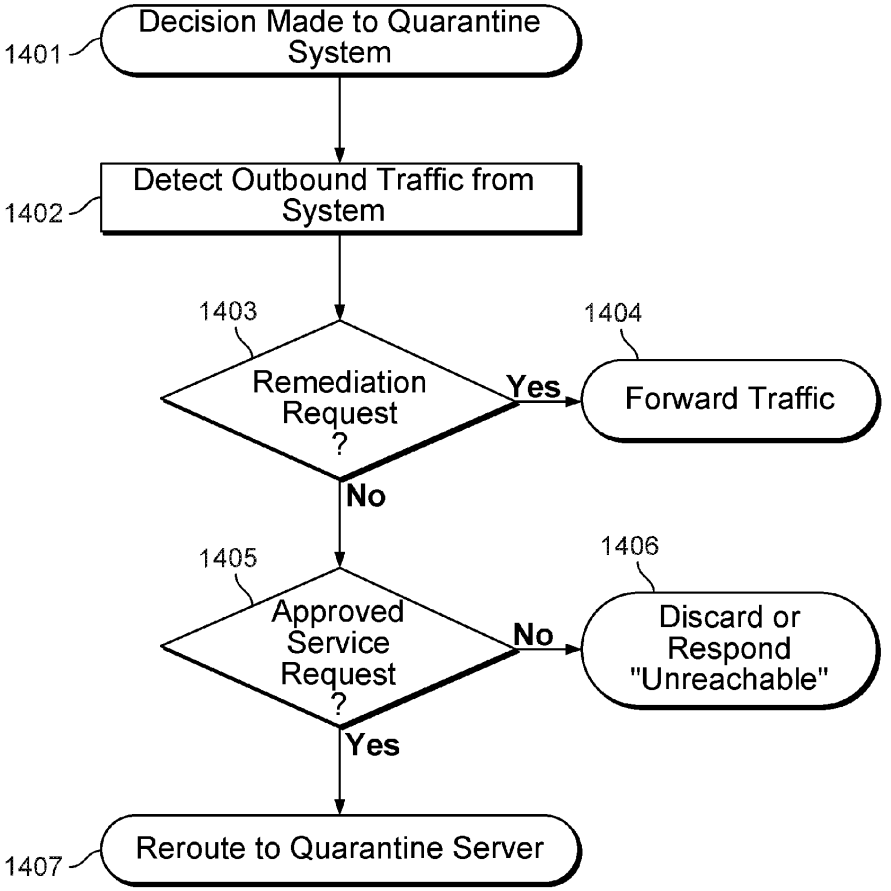

1401
Decision Made to Quarantine System
1402
Detect Outbound Traffic from System
1403
Remediation Request ?
Yes
1404
Forward Traffic
No
1405
Approved Service Request ?
No
1406
Discard or Respond "Unreachable"
Yes
1407
Reroute to Quarantine Server

FIG. 14

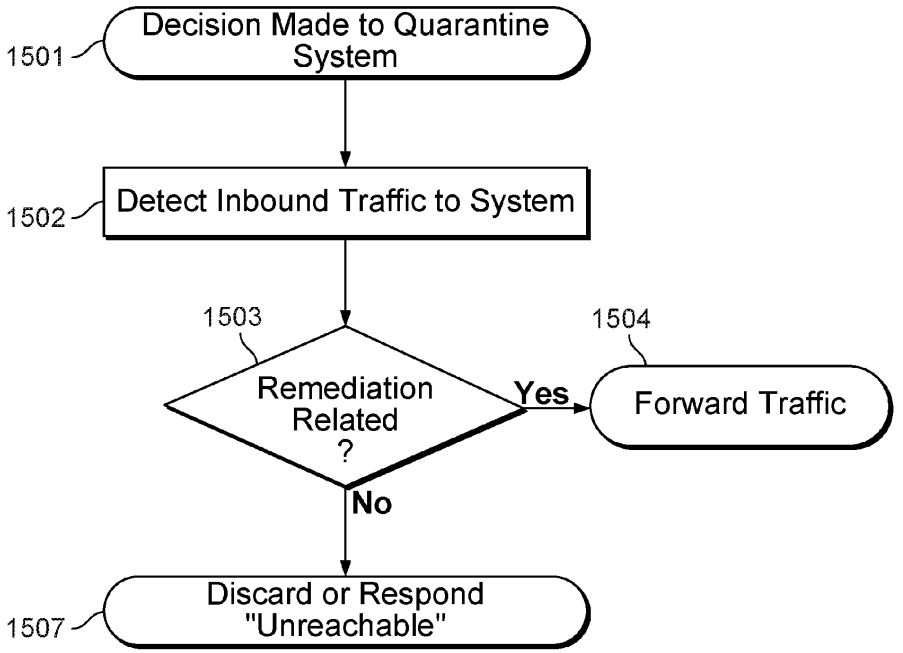

1501
Decision Made to Quarantine System
1502
Detect Inbound Traffic to System
1503
Remediation Related ?
Yes
1504
Forward Traffic
No
1507
Discard or Respond "Unreachable"

FIG. 15

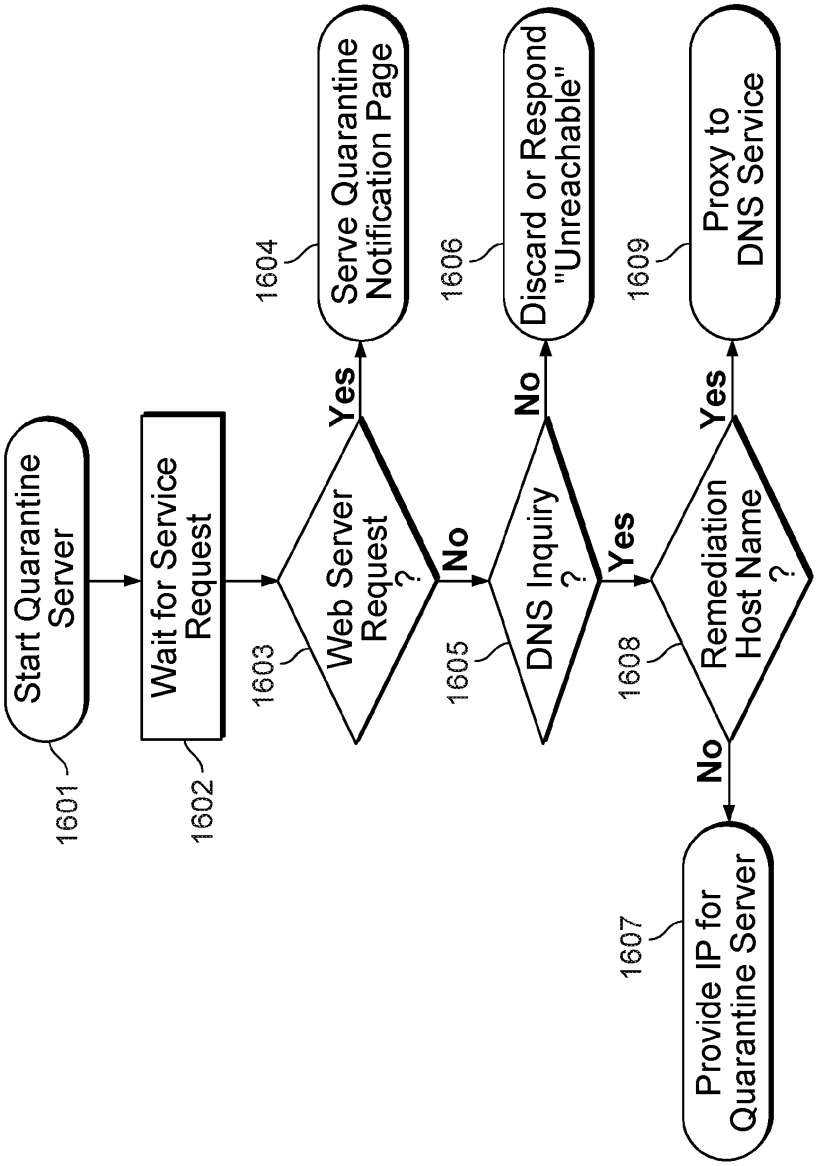

Start Quarantine Server
1601
Wait for Service Request
1602
Web Server Request ?
1603
Yes
Serve Quarantine Notification Page
1604
No
DNS Inquiry ?
1605
No
Discard or Respond "Unreachable"
1606
Yes
Remediation Host Name ?
1608
No
Provide IP for Quarantine Server
1607
Yes
Proxy to DNS Service
1609

FIG. 16

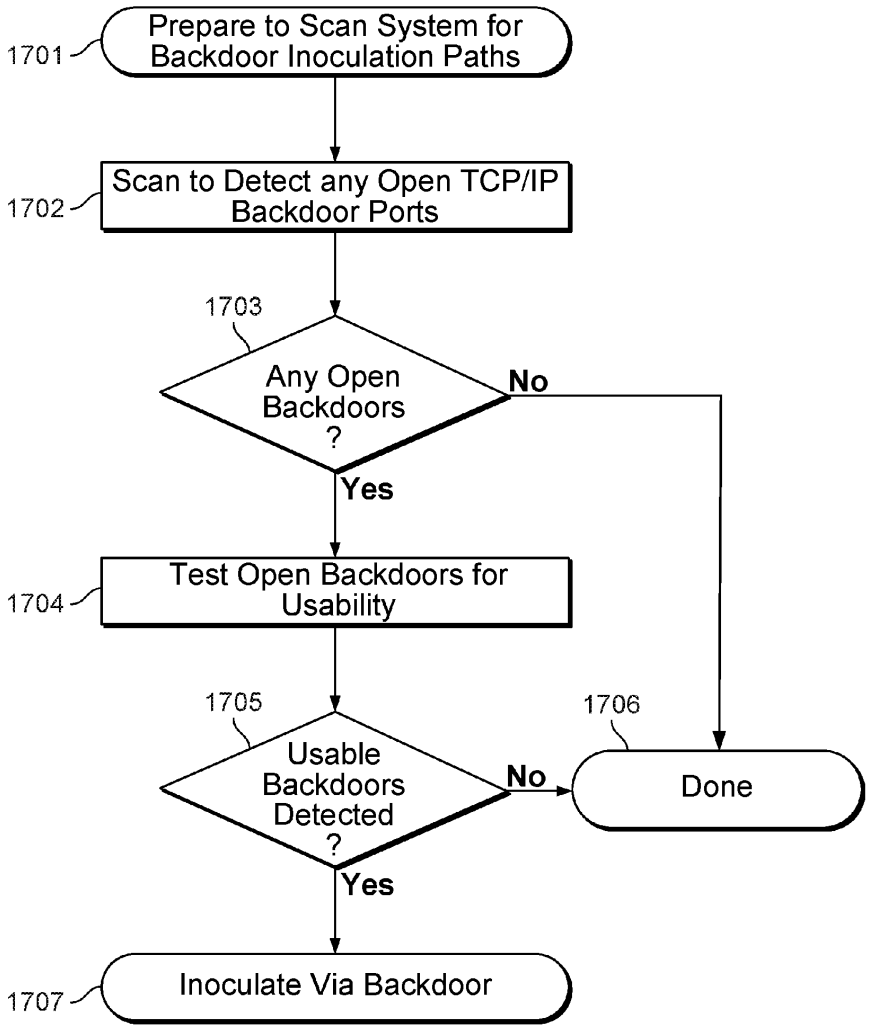

1701
Prepare to Scan System for Backdoor Inoculation Paths
1702
Scan to Detect any Open TCP/IP Backdoor Ports
1703
Any Open Backdoors ?
No
Yes
1704
Test Open Backdoors for Usability
1705
Usable Backdoors Detected ?
No
1706
Done
Yes
1707
Inoculate Via Backdoor

FIG. 17

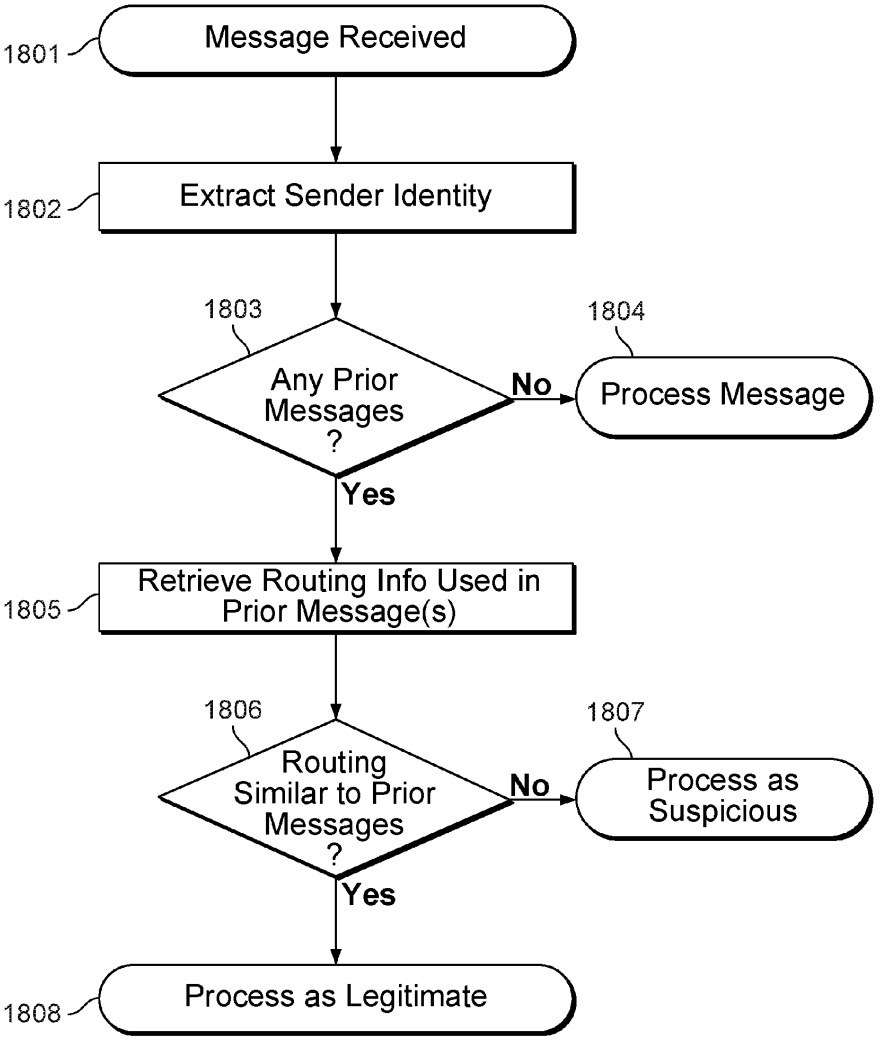

1801
Message Received
1802
Extract Sender Identity
1803
Any Prior Messages ?
No
1804
Process Message
Yes
1805
Retrieve Routing Info Used in Prior Message(s)
1806
Routing Similar to Prior Messages ?
No
1807
Process as Suspicious
Yes
1808
Process as Legitimate

FIG. 18

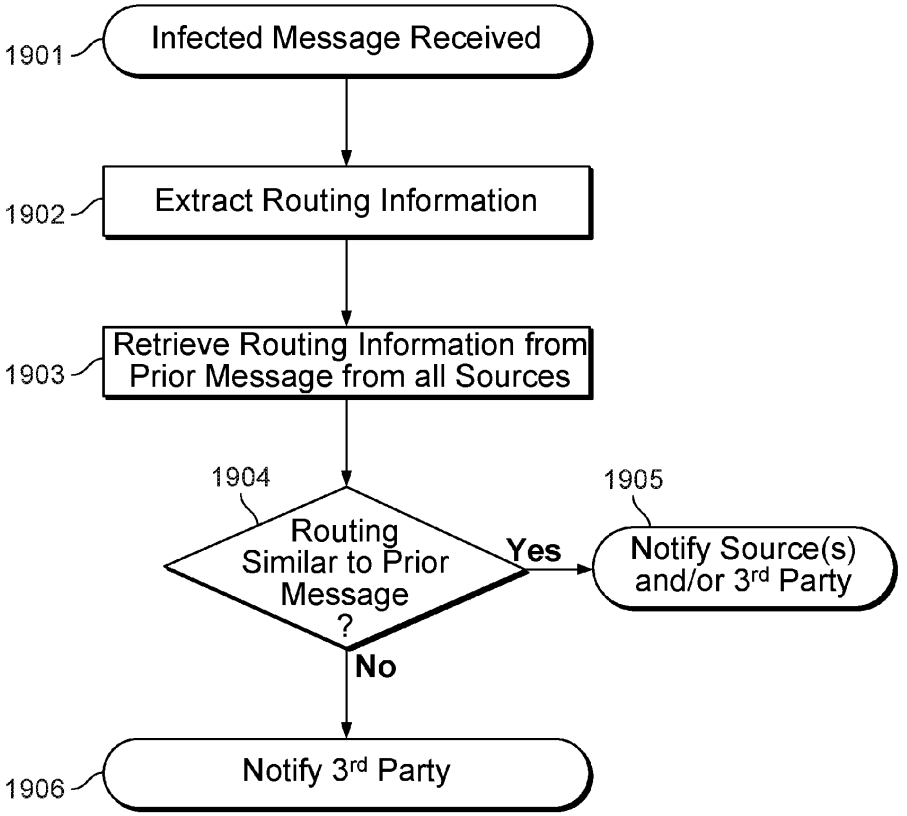

1901
Infected Message Received
1902
Extract Routing Information
1903
Retrieve Routing Information from Prior Message from all Sources
1904
Routing Similar to Prior Message ?
Yes
No
1905
Notify Source(s) and/or 3rd Party
1906
Notify 3rd Party

FIG. 19A

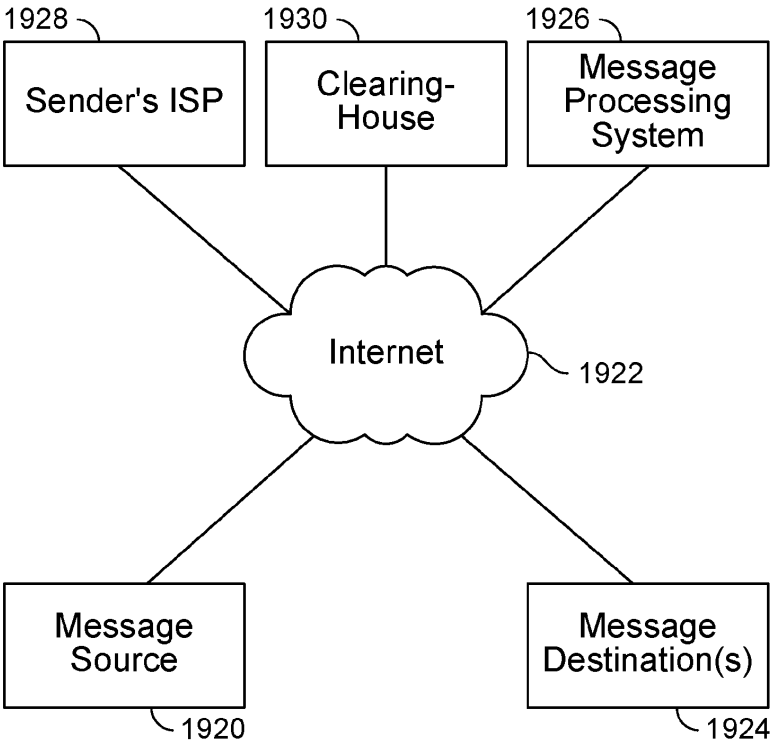

1928
Sender's ISP
1930
Clearing-House
1926
Message Processing System
Internet
1922
Message Source
1920
Message Destination(s)
1924

FIG. 19B

CONTAGION ISOLATION AND INOCULATION VIA QUARANTINE

CROSS REFERENCE TO OTHER APPLICATIONS

This application is a continuation of co-pending U.S. patent application Ser. No. 11/237,004, entitled Detecting and Processing Suspicious Network Communications, filed Sep. 27, 2005, which is incorporated herein by reference for all purposes, which claims priority to U.S. Provisional Patent Application No. 60/613,909 entitled CONTAGION ISOLATION AND INOCULATION filed Sep. 27, 2004, which is incorporated herein by reference for all purposes.

This application is related to U.S. patent application Ser. No. 11/237,003, entitled Contagion Isolation and Inoculation, filed Sep. 27, 2005, now U.S. Pat. No. 8,234,705.

BACKGROUND OF THE INVENTION

Unwanted and/or malicious network communications, such as spam, phishing, worm propagation, etc., hamper productivity and the use and enjoyment of computer and network resources by end users, burden affected networks with unauthorized and/or undesired traffic, and expose recipients to the risk of theft, fraud, etc. Current technologies for dealing with such unauthorized and/or unwanted communications focus on filtering messages at a point in the communication path close to the recipient system, such as anti-spam and/or other security software installed on the destination host and/or a local mail (or other messaging) server to which communications to the destination host are directed for delivery to the destination host, e.g., by detecting messages sent by blacklisted senders and/or messages containing header or other content that indicates the message is suspicious. While this approach can be an effective way to protect an individual host or a group of destination hosts served by a local server, it does not address the network congestion problem and leaves vulnerable hosts not protected by destination (or destination mail or messaging server) based filtering software. In addition, in many cases an attacker causes each of one or more compromised hosts to send multiple instances of a malicious communication, often simultaneously or nearly so, magnifying the effect on the network. Therefore, there is a need for an effective way to intercept and take corrective action with respect to unauthorized, unwanted, and/or otherwise malicious electronic mail and/or other network communications that better protects the network and provides protection to destination hosts that are not protected by destination or destination mail or messaging server based filtering software.

BRIEF DESCRIPTION OF THE DRAWINGS

Various embodiments of the invention are disclosed in the following detailed description and the accompanying drawings.

FIG. **1** is a flow diagram of a method for reducing an anti-contagion update interval during periods of suspicious network activity according to some embodiments.

FIG. **2**A is a flow diagram of a method for cooperatively scheduling an anti-contagion update interval according to some embodiments.

FIG. **2**B is a block diagram of a network environment in which anti-contagion updates are distributed according to some embodiments.

FIG. **3**A is a flow diagram of a method for contacting an anti-contagion source and optionally getting an update according to some embodiments.

FIGS. **3**B-**3**D illustrate distributed update distribution as implemented according to some embodiments.

FIG. **4**A is a flow diagram of a method for acquiring an update according to some embodiments.

FIG. **4**B is a block diagram of a client system configured to obtain an update according to some embodiments.

FIG. **4**C is a block diagram of a system used to distribute an update according to some embodiments.

FIG. **5** is a flow diagram of a method for distributing updates according to some embodiments.

FIG. **6** is a flow diagram of a method for establishing a redistribution policy according to some embodiments.

FIG. **7** is a flow diagram of a method for redistributing content such as updates according to some embodiments.

FIG. **8** is a flow diagram of a method for using peers or other indirect data sources to acquire content such as updates according to some embodiments.

FIG. **9** is a flow diagram of a method for redistributing content such as updates according to some embodiments.

FIG. **10**A is a flow diagram of a method for identifying and remedying problems according to some embodiments.

FIG. **10**B is a block diagram illustrating a network environment in which infected hosts and/or networks are quarantined according to some embodiments.

FIG. **11** is a flow diagram of a method for identifying vulnerabilities according to some embodiments.

FIG. **12** is a flow diagram of a method for identifying an infestation according to some embodiments.

FIG. **13** is a flow diagram of a method for monitoring one or more computers for infestation according to some embodiments.

FIG. **14** is a flow diagram of a method for quarantining a computer according to some embodiments.

FIG. **15** is a flow diagram of a method for quarantining a computer according to some embodiments.

FIG. **16** is a flow diagram of a method for a quarantine server to respond to requests according to some embodiments.

FIG. **17** is a flow diagram of a method for scanning backdoors and inoculating according to some embodiments.

FIG. **18** is a flow diagram of a method for categorizing messages according to some embodiments.

FIG. **19**A is a flow diagram of a method for notifying possible sources of infection according to some embodiments.

FIG. **19**B is a block diagram illustrating a network environment in which suspicious messages are identified and/or one or more notifications are sent when a malicious message is detected.

DETAILED DESCRIPTION

The invention can be implemented in numerous ways, including as a process, an apparatus, a system, a composition of matter, a computer readable medium such as a computer readable storage medium or a computer network wherein program instructions are sent over optical or electronic communication links. In this specification, these implementations, or any other form that the invention may take, may be referred to as techniques. A component such as a processor or a memory described as being configured to perform a task includes both a general component that is temporarily configured to perform the task at a given time and a specific component that is manufactured to perform

the task. In general, the order of the steps of disclosed processes may be altered within the scope of the invention.

A detailed description of one or more embodiments of the invention is provided below along with accompanying figures that illustrate the principles of the invention. The invention is described in connection with such embodiments, but the invention is not limited to any embodiment. The scope of the invention is limited only by the claims and the invention encompasses numerous alternatives, modifications and equivalents. Numerous specific details are set forth in the following description in order to provide a thorough understanding of the invention. These details are provided for the purpose of example and the invention may be practiced according to the claims without some or all of these specific details. For the purpose of clarity, technical material that is known in the technical fields related to the invention has not been described in detail so that the invention is not unnecessarily obscured.

Analyzing routing information to identify and intercept unauthorized, unwanted, and/or otherwise malicious communications is disclosed. In some embodiments, routing information associated with a message currently being processed is checked against corresponding information associated with a message from the same source that was processed previously. In some embodiments, a message processing system that handled and/or otherwise has access to routing information for both messages performs the check, such as a network messaging and/or access service provider with which the sender is associated, performs the check. If the routing information for the current message does not match the corresponding information observed in the previous message, responsive action is taken.

Contagion refers herein to software programs and/or code, data values, configurations, and settings, that expose a computer or other system to security threats, such as unauthorized access and/or use of the system and/or data stored thereon; corruption or loss of data; and/or damage, loss, and/or unauthorized reconfiguration of system software and/or hardware components. Examples of contagion include software programs and/or code designed to perform undesirable and/or unauthorized actions from the perspective of the (authorized) user of a computer, and at the same time perform desired actions from the perspective of the creator or controller of such contagion. Examples of contagion include computer worms, viruses, and Trojan horses. Anti-contagion software refers herein to software that prevents, impedes or remediates contagion, such as Norton Antivirus from Symantec, or VirusScan from McAfee, which identifies and/or removes contagion from a user's computer. Another example of anti-contagion software is software that repairs a security vulnerability through which contagion may spread. Many anti-contagion methods rely on contagion descriptions, such as virus signatures, to identify the contagion, and must be updated regularly to be effective.

FIG. **1** is a flow diagram of a method for reducing an anti-contagion update interval during periods of suspicious network activity according to some embodiments. An update interval refers herein to a period of time between acquisition of updates, such as security updates, virus signature file updates, or virus scanner updates. In this example, network traffic statistics may be monitored (**101**). An example of traffic statistics is latency experienced by email, such as average latency, or median latency. Another example of traffic statistics is the rate of unacceptable attempted TCP/IP traffic, such as attempts to connect to blocked or restricted ports, or insufficiently authenticated (or authorized) connections to ports, or connections that provided invalid or illegal content such as extraordinarily long strings, or specific content detected to be malicious. Recent activity may be evaluated (**102**) to see if it is anomalous. Examples of anomalous activity include median email latency in mail queues beyond some threshold, for example above a 4 hour median queuing delay, and rates of unacceptable attempted TCP/IP traffic beyond some threshold, for example greater than two attempts per minutes over a one hour period. Another example of a threshold for anomalous activity is a comparison with recent historical statistics, such as unacceptable attempted TCP/IP traffic at a rate greater than ten times the expected rate over a 30 minute period. The expected rate may for example be an average over a recent historic period, such as an average rate over the last week of connectivity. If activity is not deemed anomalous (**103**), then in this example, processing continues with no change (**104**) in the next update interval. If the activity is deemed anomalous (**103**), then in this example, the next update interval is reduced, for example reduced to some proportion of the planned update interval, for example reduced to 25% of the previous or default update interval. Another example of reducing the next update interval is to reduce the currently remaining time until an update, for example reducing by a proportion, such as 50%, the remaining time until an update is requested.

FIG. **2**A is a flow diagram of a method for cooperatively scheduling an anti-contagion update interval according to some embodiments. In this example, the planned time for contacting an anti-contagion source may arrive (**201**). An old version identifier may be retrieved (**202**), such as a checksum or hash for a file, or a version number for software stored in a file system or registry, or a version number for an antivirus signature file stored in a file system or registry. An update may be requested from an anti-contagion source (**203**), for example as is discussed in conjunction with FIG. **3**A, which may include updating or concluding that an update is not needed. A time for next contacting an anti-contagion source is decided upon (**204**). In some embodiments, the scheduled time may be made sooner when the source of the update, such as a vendor of software to which the update pertains, anticipates the impending release of a fully tested update, such as scheduling a next contact time in a few hours, or a few days. If the current computer's version of anti-contagion software contains resolutions to all known problems and otherwise anomalous traffic patterns then, in this example, the update interval may be made longer, such as the full default interval for a computer, for example one week. In some embodiments, the selection of the next update interval may be dependent on the specific product. For example, a "pro version" may have a short default interval, such as one day, and a "home version" may have a relatively long default interval, such as two weeks or one month.

FIG. **2**B is a block diagram of a network environment in which anti-contagion updates are distributed according to some embodiments. A plurality of clients 1 to "n", represented in FIG. **2**B by clients **220**, **222**, and **224**, are connected to the Internet **226**. An update server **228** also is connected to the Internet **226**. In various embodiments, one or more of the clients **220**-**224** is configured to request and obtain updates, if any, from update server **228** via the Internet, e.g., as described above in connection with FIG. **2**A.

FIG. **3**A is a flow diagram of a method for contacting an anti-contagion source and optionally getting an update according to some embodiments. In this example, an old version identifier may be retrieved (**301**), for example as was discussed in **202** of FIG. **2**A. Contact may be made with an

anti-contagion source (**302**) via a network such as the Internet. The old version identifier may be provided, for example using a transport protocol such as TCP/IP, and a data encapsulation format such as XML, or using a higher level protocol and format such as SOAP. A decision may be made as to whether an update is needed (**303**). In some embodiments, this update decision may be made by a server, upon review of the old version identifier. In some embodiments, this update decision may be made by the user's computer. In some embodiments, the update decision may be made by incorporating preferences held on a user's computer, such as a preference to always have the newest updates, or preferences to avoid updates that have been available for less than a threshold length of time (for example, 2 weeks), or preferences that accept an additional update when the source indicates urgency associated with the update. In some embodiments, the update decision may be made based on considerations such as the size of the potential update, for example by avoiding the overhead of small updates, and only performing an update when a threshold amount of change has taken place, or when a threshold length of time has elapsed since an actual update was made. An example of a threshold amount of change taking place is the addition of a prescribed number of viral signatures, such as 2 new signatures. An example of a threshold length of time is 3 weeks. If the update is not needed (**303**) then the process of contacting the source is complete (**304**) in this example. If an update is needed (**303**) then an update is acquired in this example via direct or indirect channels (**305**), for example as further described in conjunction with FIG. **4**A. After the update has been acquired (**305**), the contact may be complete (**306**).

Distributed update distribution is disclosed. In some embodiments, at least some users receive an update indirectly, e.g., from another user that received it previously, either directly from the original source or indirectly from a source that had itself earlier received it, again either directly or indirectly, and so on. FIGS. **3**B-**3**D illustrate distributed update distribution as implemented according to some embodiments. In the example shown in FIGS. **3**B-**3**D, it is assumed for expository purposes that each host that receives the update will serve as a redistribution node, each recipient has the ability to redistribute at the same rate as the original source, and there is continuous demand for the update from all sources. As shown in FIG. **3**B, at a time t**1** a first direct recipient of the update D**1** (**332**) downloads the update from the original source S**1** (**330**). As shown in FIG. **3**C, at a time t**2** a second direct recipient D**2** (**334**) downloads the update from the original source and an indirect recipient R**11** (**336**) downloads the update from the first direct recipient D**1** (**332**), which in this case acts as a redistribution node. As shown in FIG. **3**D, at a time t**3** a third direct recipient D**3** (**338**) downloads the update from original source S (**330**), a second indirect recipient R**12** (**340**) downloads the update from the first direct recipient D**1** (**332**), a third indirect recipient R**21** (**342**) downloads the update from the second direct recipient D**2**, and a fourth indirect recipient R**111** (**344**) downloads the update from first indirect recipient R**11** (**336**). In this example, in the three download cycles shown in FIGS. **3**B-**3**D on its own the original source would only have been able to download the update to the three direct recipients, but by using previous recipients to redistribute, in the example shown in FIGS. **3**B-**3**D a total of seven recipients have been able to obtain the update. In a generalized case, under the assumptions used in this example in "n" download cycles/periods a total of 2^n-1 recipients are able to obtain the update, whereas only "n" recipients, i.e., one per cycle, would otherwise have been able to obtain the update if it were only available directly from the original source. In an even more generalized case, under an assumption that each server may download roughly k copies in a time interval, for example where more than one download is provided by a server at a time, then in "n" download cycles (or periods) a total of roughly $(k+1)^n-1$ recipients are able to obtain the update, where as only k*n recipients, i.e., k per cycle, would otherwise have been able to obtain the update if it were only available directly from the original source.

FIG. **4**A is a flow diagram of a method for acquiring an update according to some embodiments. In this example, an update source is contacted (**401**), such as via an Internet connection, for example via TCP/IP. An offer to serve as a redistribution source is made (**402**). In some embodiments, a redistribution source may be contacted by other computers to acquire an update. An example of an offer to be a redistribution server is described in conjunction with FIG. **6**. In various embodiments, **402** is optional, performed only if a criterion or test is satisfied, or omitted. An update or an update identifier is acquired (**403**). An example of an update identifier is a version number for an update. Another example of an update identifier is a checksum for an update. Another example of an update identifier is a set of checksums (such as MD5, SHA-1, or CRC-32) for a corresponding set of byte ranges that form the update. Another example of an update identifier is a combination two or more identifiers, such as the identifiers enumerated above. If the update was completed (**404**) then the user confirms redistribution sourcing status (**408**) in this example. In some embodiments, confirmation includes asserting that the client is prepared to redistribute copies of the received data to one or more recipients. If the update was not completed (**404**), then a list of peer or other sources is acquired (**405**) in this example. In some embodiments, the list is or may be explicit. In some embodiments, the list is or may be implicit, for example represented by an iterator or accessor, which may retrieve elements on an as needed basis. In some embodiments, implicit lists are dynamic, and calculated or created in part on an as needed or as requested basis. In some embodiments, the list of other sources contain addresses that may be contacted to acquire an update, such as a list of IP addresses and ports, as well as optionally some authentication information to be used when contacting a listed source. In some embodiments, a requesting host that agrees to serve as a redistribution source is provided with a list of sources from which the update is available immediately, or at least sooner than it will be made available to hosts that will not redistribute. One or more attempts may be made to acquire the update from a listed source (**406**), for example as exemplified in FIG. **8**. If the update is completed via a listed source (**407**), then redistribution status is confirmed (**408**) in this example. If the update is not complete (**407**) then a reservation is optionally negotiated and/or acquired (**409**) in this example. In some embodiments, a reservation may include a time and/or a specific source or IP address. In some embodiments, a reservation address may be selected from a collection of source addresses, for example in order to distribute the load across more than one server, or to provide a server that is closer to a recipient. An example of selecting a server close to a recipient is to select a server having relatively lower latency during a download, or relatively lower cost, for example by selecting a source within a recipient's ISP complex or within a recipient's intranet or home network. If a reservation time was acquired, then a wait for that reservation time (**410**) is performed in this example. At the reservation time, a further attempt to acquire

an update or update identifier from the source (**403**) is made, and the flow proceeds as described earlier. This flow may continue until an acquisition is complete (**408**), as also described earlier. In various embodiments, one or more of confirming redistribution sourcing status (**408**), attempting to obtain the update from a redistribution source (**405-407**), and/or acquiring and/or waiting for a reservation time (**409-410**) is/are optional, conditional, or omitted.

FIG. **4**B is a block diagram of a client system configured to obtain an update according to some embodiments. In some embodiments, the client system shown in FIG. **4**B is used to implement one or more of clients **220-224** of FIG. **2**B. The client system **420** includes a communication interface **422**, such as a network interface card, configured to send/receive communications via a network, such as the Internet. The client system **420** further includes an update agent and/or browser **424**. In some embodiments, the update agent/browser **424** comprises browser software used to navigate the World Wide Web, e.g., to send/receive data to/from a website from which an update is available for download. In some embodiments, update agent/browser **424** includes an agent associated with anti-contagion software installed on the client system **420**. In various embodiments, the update agent/browser **424** comprises one or more components and is implemented as one or more software programs running on a processor and/or one or more hardware components. A settings data store **426** stores one or more settings and/or other data associated with the software with which an update and/or software to which the update does or may pertain, such as periodicity or next scheduled (e.g., reservation) time to check for updates, a list of update identifiers, locators, and/or redistribution sources from which an update is or may be available, policies for whether and/or under what terms, conditions, circumstances, etc. the client system **420** may be used redistribute an update, a last update time, an old version identifier (see, e.g., **202** of FIG. **2**A), etc.

FIG. **4**C is a block diagram of a system used to distribute an update according to some embodiments. In some embodiments, the system of FIG. **4**C is used to provide an update server, such as update server **228** of FIG. **2**B. In this example, the distribution system **440** includes a communication interface **442** configured to send/receive data via a network, such as the Internet. An update distribution engine **444** is configured to receive and service requests for updates. For updates provided directly to a request source, the update distribution engine **444** obtains the update from an update storage location **446** and provides the update to the request source via the communication interface **442**. In some embodiments, if data included in the request indicates the source of the request is or may be willing to serve as a redistribution source, the update distribution engine **444** hands the request off to a redistribution coordinator **448**, which negotiates with the request source, e.g., based on policies, settings, rules, etc. stored in a policies and lists data store **450**, through communications sent/received via communication interface **442**, to determine if the requesting source will serve as a redistribution source and, if so, to determine the terms and conditions of such service and data, such as an address, locator, or identifier, that can be provided to other requesting systems and/or users to enable them to attempt to obtain the update from the redistribution source. In some embodiments, if update distribution engine **444** determines that it is not able to provide the update to the request source directly (e.g., due to capacity limits or because a policy applicable to the request and/or the request source indicates the request source should be directed to a redistribution source, e.g., because the request source has a less expensive version of the software or is not willing to serve as a redistribution source), the update distribution engine **444** hands the request off to redistribution coordinator **448**, which provides to the request source an update identifier and a list of one or more redistribution sources from which the update is or may be available to the request source.

FIG. **5** is a flow diagram of a method for distributing updates according to some embodiments. In this example, a source prepares to supply updates to recipients (**501**), for example by starting up a server, for example an HTTP or FTP server, or a multicast server, such as an RTP multicast server. The server waits for an event (**502**), such as a contact on a predefined port, or a notification that the server should start a multicast. If a multicast-time event appears (**503**) then an update is multicast (**504**) in this example. In some embodiments, a multicast time event may be generated internally or externally to the server to start a multicast. In some embodiments, a multicast-time event may be automatically generated at predetermined times, such as hourly, weekly, or daily, or on individually specified dates and times. In some embodiments, a multicast-time event may include a specific identifier for the multicast. If an event of the form of an update request arrives (**503**) then a test for authenticity is optionally performed (**505**) in this example. An example of determining whether a request is authentic is to check a credential. Examples of credentials include a provided shared secret, such as a secret used by all requestors to this server, or a different secret used by each requestor such as a user serial number, or a cryptographically signed authentication, such as credentials or a certificate issued to users that had paid for an update service, for example, by inclusion in a purchased software package. In some embodiments, all requests are assumed to be authentic without the use of any authenticator. If the request was not authentic (**505**) then, in this example, the server continues to wait for an event (**502**). If the request was authentic (**505**) then the server may determine a necessary update (**506**) in this example. In some embodiments, determination of an update may be made based on the requestor's state, which in some embodiments may be indicated in a received request for update. A redistribution contract may be negotiated (**507**). In some embodiments, some requestors may agree to redistribute updates to other requestors. In some embodiments, some requestors may decline to redistribute updates. In some embodiments, a source may decline redistribution proposals from a requestor. An example of a redistribution contract is an agreement not to redistribute. Another example of a redistribution contract is an agreement to redistribute a fixed number of copies, such as five copies. Another example of a redistribution contract is an agreement to redistribute for a fixed period of time, such as one day. Another example of a redistribution contract is an agreement to redistribute within a specified geographic region within a fixed period of time. Another example of a redistribution contract is an agreement to redistribute indefinitely, for example until a replacement update is provided. A redistribution contract may be made with terms that are combinations of the above mentioned exemplary term limitations. An update may be provided directly or indirectly (**508**). A direct update may for example be provided by allowing an FTP download, or by encapsulating the transmission in an HTTP transfer, for example using XML and/or an appropriate coding, such as uuencode. An indirect update may be provided by directing the requestor to an alternate source, such as a redistribution server, or a server at a different address, or a server at a

different time, such as a future time, or a combination of two or more such specifications. In some embodiments, a decision to provide direct vs. indirect updates may be based in part on the redistribution contract, for example a requestor that is willing to redistribute may be updated sooner or more directly. An example of a redistribution contract is a promise by a user to provide redistribution in exchange for purchase price concession, such as a reduced initial purchase price, or a quantity discount price. In some embodiments, a decision to provide direct vs. indirect updates may be based in part on previous agreements, such as prepayment for expedited service, such as may be included in a more expensive version of a product that utilizes the updates. The server may wait for an event (**502**). In some embodiments, events may be processed in parallel with previous events, or may be sequentially processed, or some combination, for example up to a threshold number of events in parallel, such as a maximum of 32 events in parallel.

FIG. **6** is a flow diagram of a method for establishing a redistribution policy according to some embodiments. In this example, a computer may prepare to offer to be a redistribution source (**601**). In some embodiments, preparations may include retrieving preferences, such as the number of redistributions that may be done, or the duration of time that the computer is willing to act as a redistribution source. In some embodiments, preferences may be defaults, or may be configured by a user and stored persistently, such as in a registry or file system. A contact address may be proposed to the source and a policy may be negotiated (**602**). A contact address may for example be a globally routable IP address and a port, or a local IP address (reachable from a LAN behind a firewall) and port that can be accessed by other computers on the LAN, such as computers that shared a common NAT address in their contact with the original source. In some embodiments, negotiation may include proposing a policy. In some embodiments, negotiations may involve a counter proposal in response to a proposal. An example of a counter proposal criterion is to require a minimum duration for redistribution. If terms were not acceptable (**603**) then, in this example, the computer does nothing (**604**). If the terms were acceptable (**603**) then, in this example, the computer prepares to service redistribution requests (**605**), for example by establishing a server to respond at the agreed contact address, reserving storage to hold the content for redistribution, etc. In some embodiments, redistribution policy may be a broader contractual matter, where an update is provided more quickly or more directly in exchange for a promise to redistribute. In some embodiments, an auditing mechanism may be used to measure compliance and penalize contract breakers, such as failure to accept connections for redistribution, where penalties may for example include refusal to accept future similar contracts. As another example, a lower purchase or subscription price may be offered for operators of a software product, such as antivirus software, who agree to act as redistribution points. In some embodiments such purchase agreements may be configured into the software, for example by establishing agreements to act as a redistribution point for updates as defaults that a user cannot modify.

FIG. **7** is a flow diagram of a method for redistributing content such as updates according to some embodiments. In this example, a redistribution computer is contacted and confirms its sourcing status (**701**), for example by accepting a TCP/IP connection on a predefined port, and processing a request for content. The redistribution computer may optionally contact another computer, for example a more original source, and affirm its redistribution sourcing status (**702**). A more original source may be a source that has provided content in the past to the redistribution computer, directly or indirectly. In some embodiments, contacting a more original source demonstrates that the redistribution computer is performing redistribution. In some embodiments, contacting a more original source allows the redistribution computer to further authenticate the computer that requested the content, for example by getting confirmation from a more original source that a specific computer was authorized to request the content. An authentication to be used to authenticate requests is negotiated (**703**). In some embodiments, authentication includes providing a shared secret to the redistribution computer. In some embodiments, authentication includes presenting an authorization cryptographically signed by a more original source, such as a certificate. In some embodiments, all requests are presumed authentic. If the request is not authentic (**704**), then nothing further is done (**705**) in this example. If the request is authentic (**704**), then the requested update is supplied (**706**) in this example. The updates may be supplied for example via a transport protocol TCP/IP, or a higher level protocol such as FTP, HTTP, or HTTPS, etc.

FIG. **8** is a flow diagram of a method for using peers or other indirect data sources to acquire content such as updates according to some embodiments. In this example, a list of one or more peers or other indirect sources for acquiring updates is obtained (**801**), for example as was discussed in conjunction with **405** of FIG. **4**A. In one example, an indirect source may be a peer, such as another computer that has downloaded the content. In another example, an indirect source may be a computer configured to redistribute content without other use of the content, such as an ISP server established to redistribute content to subscribers. In another example, an indirect source may be a server created or authorized by the original source to perform additional distributions. In some embodiments, an obtained list of indirect sources may be implicit, for example by referring to a list using an iterator or accessor construct that can itemize entries in the list when requested, for example using a query to a service when an additional item in the list is required, which may for example dynamically complete the list on an as requested basis. A next address and (optional) authenticator in the list of sources is retrieved (**802**). Any order may be used. In some embodiments, an order of processing may correspond to the original list order. In some embodiments, an order of processing may be such that the earlier processed addresses would be expected to have lower latency and/or higher bandwidth, for example by first processing a local address on a LAN. In some embodiments, elements of the list are or may be processed in parallel, and portions (such as byte ranges) of the content may be acquired from different addresses in the list. An address on the list may be contacted to request content (**803**), which may in some embodiments be performed as described in conjunction with FIG. **7**. In one example, a source address may consist of an identifier for a broadcast with optional time(s) for a broadcast, and contact with the source may include receiving the broadcast. In another example, contact with the source may include making a connection such as a TCP/IP connection with the source. If the acquisition is complete (**804**), then the processing is complete (**805**) in this example. If the acquisition was not completed (**804**) then additional sources in the list are considered in this example. If there are more sources in the list (**806**) then, in this example, a next source is retrieved along with an optional authenticator (**802**) and the process-

ing continues. If there are no more sources (**806**) then the attempted acquisition from source(s) is unsuccessful (**807**) in this example.

FIG. **9** is a flow diagram of a method for redistributing content such as updates according to some embodiments. In this example, a computer prepares to supply content (or updates) to a requestor/recipient (**901**), for example by starting up one or more servers. Examples of a server include an HTTP server, an FTP server, and a multicast server. A server waits for an event (**902**), such as a contact on a predefined port, or a notification that the server should be terminated, or a notification that a time for a multicast has arrived. If a service expiration event appears (**904**) then the supply service is terminated (**905**) in this example. A service expiration event may be automatically generated, for example after a predetermined duration of service, or after a predetermined number of copies of an update have been redistributed. If a request event arrives (**904**), then a test for authenticity is performed (**906**) in this example. Authenticity may be established in the request, for example by receiving and validating, or performing a mutual proof of possession of, a shared secret, such as a secret used by all requestors to this server, or a different secret used by each requestor of this service and held in a list of single-use secrets by this server, or by providing a cryptographically signed authentication. In some embodiments, all requests are considered authentic without the use of any authenticator. If the request was not authentic (**906**) then, in this example, the server continues to wait for an event (**902**) as described earlier. If the request was authentic (**906**) then, in this example, the server transmits a copy of the content to the requestor (**907**), for example by allowing an FTP download, or by encapsulating the transmission in an HTTP transfer, for example using XML and/or an appropriate coding, such as uuencode.

FIG. **10**A is a flow diagram of a method for identifying and remedying problems according to some embodiments. A problem may be identified as some event or status that may cause direct or potential harm. Exemplary problems include vulnerability or infestation. In this example, a problem such as a vulnerability or infestation is detected (**1001**). Infestation herein refers to the current presence and/or execution of contagion. In some embodiments, detection may be made by a router, firewall, or other network component, separately or in concert, using techniques such as those illustrated by FIGS. **11**, **12**, and **13**. In some embodiments, detection may be made by a computing device, for example a computing device operated by an ISP. After a detection of a problem, a computer or network containing the problem is quarantined (**1002**). In some embodiments, quarantining includes preventing unauthorized communications from being sent from the quarantined computer, such as efforts to infect, probe, or harass other computers, or efforts to communicate data without authorization, including for example file contents and keyboard entry logs. In some embodiments, no communications are authorized. In some embodiments, quarantining includes preventing undesired communications from reaching the quarantined computer, such as control messages or infectious messages. Examples of quarantining activities are illustrated in FIGS. **14** and **15**. In this example, the quarantined computer is provided access to remediation services (**1003**). In some embodiments, access to remediation includes or may include allowing connections to servers providing security patches or advisories, for example allowing connections to a vendor's server, or providing virus and worm disinfecting tools, such as focused removal tools, or data updates for previously installed repair tools. In some embodiments, access to remediation includes allowing direct access to vendor sites known to provide remediation assistance, such as to Microsoft's Windows Update service, where security patches may be obtained. In some embodiments, access to vendor sites is restricted by the quarantine to certain classes of traffic, such as HTTP requests, certain port access, such as port **80** or port **443**, certain content requests, such as specific URLs or specific file downloads. In some embodiments, remediation content may be substituted for other requested content, for example as discussed in conjunction with FIG. **14**.

FIG. **10**B is a block diagram illustrating a network environment in which infected hosts and/or networks are quarantined according to some embodiments. A plurality of hosts 1 to "m", represented in FIG. **10**B by hosts **1020**, **1022**, and **1024**, connect via a router **1026** and a gateway **1028** to the Internet **1030**. In various embodiments the router **1026** is omitted or included in the gateway **1028**. In various embodiments, gateway **1028** comprises a gateway, router, firewall, or other device configured to provide and control access between a protected network and the Internet and/or another public or private network. In various embodiments, the process of FIG. **10**A is implemented, in the event of quarantine of one or more of hosts **1020-1024**, for example, on the affected (quarantined) host, router **1026**, and/or gateway **1028**. In some embodiments, a quarantined host (or a host associated with a quarantined network or sub-network) is permitted to access a remediation server **1032**, e.g., to download a patch, more current threat definition, etc. In some embodiments, requests to connection to a host other than the remediation server **1032** are redirected to a quarantine server **1034** configured to provide a notice and/or other information and/or instructions to a user of the quarantined host. In various alternative embodiments the quarantine server **1034** is omitted or resides on the router **1026** and/or gateway **1028**.

FIG. **11** is a flow diagram of a method for identifying vulnerabilities according to some embodiments. In this example, preparation is made to scan for one or more vulnerabilities (**1101**), for example by retrieving one or more IP addresses of computers that need to be scanned. For example, an IP address may be provided by a DHCP sever hosted in an ISP network complex, to identify current subscriber IP addresses. As another example, a subnet range may be scanned, for example by taking an address of the form A.B.C.X, and using some or all values of X in the range 0 to 255 to enumerate distinct IP addresses. A scan of one or more IP addresses may optionally be performed to detect an open port (**1102**). An example of scanning an IP address is scanning to detect open TCP/IP ports (**1102**). An open port may be identified by attempting to form a connection, such as a TCP/IP connection to that port. In some embodiments, only open ports on a given computer will provide acknowledgment messages needed to establish a TCP/IP connections, which for example causes a success code to be returned by Unix "connect( )" and a closed port will provide no response to the attempt to connect, which for example would cause an error code to be returned by the Unix function call "connect( )". If no open ports are detected (**1103**), then the scanning is complete (**1106**) in this example. If open ports are detected (**1103**) then those ports are tested for vulnerabilities (**1104**) in this example. In some embodiments, an open port is assumed (**1103**) and a test for vulnerability is performed or attempted (**1104**), for example by communicating a vulnerability test without independently testing to see if one or more requisite ports are open. In some embodiments, a vulnerability test of a port may include sending data that is known to create a discernable

response when a vulnerability is present, such as sending a long string known to induce a buffer overflow with a consequent errant response, where the errant response may for example include a characteristic communication, premature termination of the connection, or other erroneous response(s). In some embodiments, a test for a vulnerability is an attempt to exercise a service, such as a sendmail or an RPC service, in search of a misconfiguration of that service that creates a vulnerability. One example of a misconfiguration of a service is provision of any such service. Another example of a misconfigured service is one where a password should be required for security purposes, but no password is required by the misconfiguration. Another example of a misconfiguration is the use of an easily guessable password, such as the default installation password, when resulting access creates a security vulnerability. Another example of a misconfigured service is support for a service provided by an unauthorized agent, such as a worm or virus. If no vulnerabilities are detected (**1105**) then scanning is complete (**1106**) in this example. If vulnerabilities are detected (**1105**) then an associated IP address is flagged as vulnerable (**1107**) in this example. In the case of a subscriber to an ISP, an IP address may be translated to a particular subscriber identity, for example by checking in DHCP or static IP allocation tables for the subscriber's identity, or for example by extracting a MAC address from traffic associated with the IP address, and such identity may be appropriately flagged or recorded. In some embodiments, the record of the vulnerability may additionally include the port, and the details of the vulnerability that was detected, such as which buffer overrun was vulnerable, or which service was misconfigured to create a vulnerability.

FIG. **12** is a flow diagram of a method for identifying an infestation according to some embodiments. In this example, a device near a computer, such as a firewall on a LAN or an ISP's router, is used in some embodiments to monitor a computer for infestation (**1201**). Network traffic is monitored for anomalous traffic volume, port, or IP destination (**1202**), for example by detecting large volumes of outbound traffic, such as email traffic or instant messaging traffic. Another example of anomalous traffic is a TCP/IP connection to a server that is known to control a collection of infested machines, in a way that suggests the connecting machine is controlled by the server, such as connecting and transmitting an assertion that it is controlled or available to comply. Another example of anomalous traffic is a connection to a computer that is a current target of a denial of service attack, with a pattern that is consistent with the attack, such as a connection to a targeted port, or the use of a falsified return IP address. In some embodiments, a falsified return address may be detected by a router, such as an ISP router, that is aware that the return address is not on the sourcing subnet, or not associated with the MAC address that has supplied the corresponding packets. If an anomaly is detected (**1203**) then an apparent infestation is noted (**1204**) in this example. In some embodiments, a response to infestation may include quarantining of the computer or network containing the computer, for example as discussed in conjunction with FIGS. **14**, **15**, and **16**. In some embodiments, a response to infestation may include an attempt to remediate the problem, for example by inoculation, such as discussed in conjunction with FIG. **17**. If no anomaly is detected (**1203**) then monitoring is complete (**1205**) in this example. In some embodiments, monitoring for anomalous traffic may be continuous. In some embodiments, monitoring for anomalous traffic may be periodic. For example, periodic monitoring may consist of monitoring some number of computers, such as 1 computer, for some period of time, such as 2 minutes, and then not monitoring those computers again for another period of time, such as 1 hour. As another example, periodic monitoring may be randomized, for example by changing the duration of monitoring, pattern of selecting computers to monitor, and/or the unmonitored duration, to be difficult or impossible to predict, for example by being driven by a pseudo random number generator, subject to restrictions of available monitoring computers.

FIG. **13** is a flow diagram of a method for monitoring one or more computers for infestation according to some embodiments. In this example monitoring of a computer for infestation begins (**1301**), for example by retrieving a list of one or more addresses of computers, such as addresses of participating subscribers. In some embodiments, this list may be a list of addresses for some or all subscribers for a given ISP. In some embodiments, the list may be a list of IP addresses issued by a DHCP server, such as on a LAN, or a list of IP addresses observed by a router, such as a router on a LAN, or a complete list of IP addresses in a given subnet. A computer associated with an address identified in a list is queried for a cleanliness assertion (**1302**), for example by contacting a trusted computing base within a computer, and requesting an authenticated infestation scan by trusted software. An example of a trusted computing base within a computer is the Paladium security initiative under development by Microsoft and supported by Intel and American Micro Devices. Another example of a trusted computing base is described in various TCG specifications, such as the TCG Architecture Overview, published by the Trusted Computing Group. Trusted code bases may for example execute antivirus scans of the remainder of the computer, including untrusted portions of the disk and/or operating system. In some embodiments, trusted code bases may digitally sign assertions about the cleanliness (e.g. infestation status) and/or state of their computers. In some embodiments, the query for cleanliness (**1302**) may be responded to by anti-contagion software, such as antivirus software, with assertions about the currency of a scan, such as the last time a scan was performed, or a version associated with a current anti-contagion software or definition file in use, wherein a sufficiently updated software and/or scan may act as a cleanliness assertion. In some embodiments, an operating system may respond with information associated with its patch level, wherein a sufficiently recent patch level may be interpreted as an assertion of cleanliness. If a computer asserts it is clean (**1303**) then monitoring is complete (**1305**) in this example. If a cleanliness assertion is not provided (**1303**) then an infestation or vulnerability is presumed (**1304**) in this example.

FIG. **14** is a flow diagram of a method for quarantining a computer according to some embodiments. In this example, a decision has been made to quarantine a computer (**1401**), such as an ISP deciding to quarantine a subscriber, or a firewall or router for a LAN deciding to quarantine a local computer. In some embodiments, a decision to quarantine may be made based on apparent infestations or apparent vulnerabilities, for example as illustrated in conjunction with FIG. **11**, **12**, or **13**. In some embodiments a decision to quarantine may be based on the accumulation of evidence above a threshold, such as a threshold of one such piece of evidence, or a threshold of two such pieces of evidence. In some embodiments, a decision to quarantine may be made based on configuration rules, for example all new computers must be quarantined until they have contacted their OS vendor and been updated, or by user specification, such as a

user requesting a quarantine for a specific computer. In some embodiments, requests for a quarantine may be made on an individual computer via settings, such as control panel in Microsoft Windows, or through a user interface to a firewall or router, for example by specifying an IP address, name, or MAC address to quarantine, or a list of addresses that are not to be quarantined (where other addresses may be quarantined). During a quarantine period outbound traffic is detected (**1402**), for example by a device such as a router, or by a software network stack on the quarantined computer, or by an upstream device, such as a device at an ISP. Outbound traffic is tested in this example to see if it is associated with a remediation request (**1403**), such as contact with an appropriate anti-contagion software distributor, or contact with a security patch update provider, such as Microsoft Windows Update, or contact with a verified remediation site, such as an internal ISP site. If outbound traffic is deemed to be associated with a remediation request (**1403**), then the traffic is routed or forwarded to its destination (**1404**) in this example. If outbound traffic is not associated with a remediation request (**1403**), then, in this example, it is tested to see if it is an approved service request (**1405**), such as a request for a web page. If the request was for an approved service (**1405**) then, in this example, the request is re-routed to a special quarantine server (**1407**), such as an ISP web server that provides information and links to assist in remediation. Re-routing of request traffic may be accomplished in numerous ways, for example by providing an alternate DNS service that consistently returns a fixed IP address of a quarantine server in response to all DNS requests. Another example of re-routing a web page request is to respond with a redirect message, which would direct a browser on a quarantined computer to contact a quarantine server that provides remediation information. If the request was not for an approved service (**1406**), then the outbound traffic is discarded or responded to with in indication that the desired traffic destination is unreachable (**1406**) in this example.

FIG. **15** is a flow diagram of a method for quarantining a computer according to some embodiments. In this example, a decision has been made to quarantine a computer (**1501**), for example as described in conjunction with **1401** of FIG. **14**. Inbound traffic originally destined for a quarantined computer is detected (**1502**), for example by a network device such as by a router. An example of detecting inbound traffic destined for a quarantined computer is to have a list of quarantined IP addresses, and filter or provide additional processing on traffic destined for an address on the list. Detected inbound traffic is tested in this example to see if it is remediation related (**1503**), such as a response to a remediation request. Direct responses to remediation requests may for example be identified using stateful packet inspection firewall techniques known to those skilled in the art. In various embodiments, at least certain types of traffic from approved sites such as anti-contagion vendors, security update providers, and/or internal ISP sites are deemed remediation related. If traffic is remediation related (**1503**) then it is forwarded to the quarantined computer (**1504**) in this example. If traffic is not remediation related (**1503**) then, in this example, it is discarded or responded to with a message indicating that the destination is unreachable (**1507**).

In some embodiments, a configuration based quarantine during an installation of an operating system, such as Microsoft Windows, may be automatically setup, for example by a default installation setting up restrictions in a firewall installed with the OS or by an installation setting defaults for use when network connectivity is enabled, that establishes a quarantine, for example restricting internet connections, for example allowing internet connections only to a security update site such as Microsoft Windows Update. In some embodiments, settings that may establish a quarantine, such as default settings in an operating system installation, may be altered programmatically to remove the quarantine in response to a remediation site, such as a security update service such as Microsoft Windows Update, asserting that there are no other security updates pending. In some embodiments, configuration settings that may establish a quarantine may be altered programmatically to enable a quarantine in response to the presence of one or more security updates, such as provided by an operating system update service such as Microsoft Windows Update, that are available for installation on a computer.

FIG. **16** is a flow diagram of a method for a quarantine server to respond to requests according to some embodiments. In this example, a quarantine server is started (**1601**), for example by starting a web server, or DNS server, or proxy server, or ftp server, or some combination server. The server waits for a request (**1602**), for example by listening on a port and waiting for a connection. In some embodiments, a device such as a router may forward outbound traffic from a quarantined computer to a quarantine server, for example as illustrated in conjunction with **1407** of FIG. **14**. If a received connection is a web server request (**1603**), such as an HTTP request, then the server responds with a quarantine notification page (**1604**) in this example. An example of a quarantine notification page is a web page that provides notification that the computer is quarantined, and/or provides links to remediation sites appropriate to the quarantine, such as a link to a site that provides anti-contagion software for removing a virus that the quarantined computer is believed to contain. The links on the quarantine notification page may be examples of HTTP addresses that are for use in remediation. Another example of a quarantine notification page is a page that redirects to a quarantine notification page. If the request is not a web server request (**1603**), then it is tested to see if it is a DNS inquiry (**1605**) in this example. A DNS inquiry may for example appear on port **53** using TCP/IP or UDP. If the request is not a DNS inquiry (**1605**) then it may be discarded or responded to as "unreachable" (**1606**) in this example. If the request is a DNS inquiry (**1605**) then, in this example, the inquiry is tested to see if it is a DNS request for a remediation host name (**1608**). A remediation host name may include a host name as appropriate to the reason for a quarantine, such as a host providing anti-contagion software, or a host providing security updates, or a host internal to (or partnered with) an ISP. If the DNS inquiry was for a remediation host name (**1608**) then the request is proxied to an external DNS service provider (**1609**) in this example. A proxy to a DNS service (**1609**) may return a stored or cached response, such as the actual IP address of the host name, or it may contact an external DNS service to relay back the proposed IP address. If the DNS inquiry was not for a remediation host name (**1608**) then an IP address for a quarantine server is provided (**1607**) in this example. In some embodiments, when a redirected IP address is provided, such as that of a quarantine server, the DNS response may include a short time to live, for example 0 seconds. A short time to live ensures that a further DNS query will be sent prior to subsequent communications being sent from the quarantined computer to the host that was the subject of the DNS query, aiding a return to normal routing of traffic once the computer is no longer under quarantine. In some embodiments, the server(s) in this example may con-

tinue to wait for a service request (**1602**) in parallel with the processing of the requests as described above, or may wait for a service request again after disposition of the previous request as described above.

FIG. **17** is a flow diagram of a method for scanning backdoors and inoculating according to some embodiments. In this example, preparations are made for scanning a computer for a backdoor inoculation path (**1701**), for example by identifying the computer to be scanned, and enumerating or retrieving a list of one or more backdoors. In one example, an IP address to be scanned may be selected by an ISP from a list of subscribers. In one example, a list of backdoors may include information associated with preloaded software approved by the computer owner for access and inoculation purposes, such as an ISP provided utility that is installed by the subscriber. In some embodiments, an ISP defined backdoor may have an authentication protocol to preclude control by unauthorized parties, such as restrictions on IP address(es) that may contact the backdoor, or shared secrets such as a password required by the sender, or cryptographic authentication such as client authentication on an SSL connection. In some embodiments, a user is requested and/or required to authorize in advance use of an authorized and/or any unauthorized backdoor for purposes of inoculation. As another example, a list of backdoors may include virus or worm installed backdoors, such as those installed by a SubSeven virus or Back Orifice. As another example, backdoors may include exploitable security vulnerabilities, such as a weakness in a web server, or a weakness in an RPC service. A scan may optionally be conducted to detect open TCP/IP backdoor ports (**1702**) similarly to the scan discussed in conjunction with FIG. **11**. If none of the backdoors are open (**1703**), for example there is no response on any of the ports, then the scan is complete (**1706**) in this example. If one or more backdoor ports are open (**1703**) then they are tested for usability (**1704**) in this example. In some embodiments, a backdoor port may be assumed open (**1703**) and may be tested for usability (**1704**) without an independent test to see if the port is open. An example of a usability test is no test at all, with the conclusion that a specific port is usable any time it is open. Another example of a usability test is based on the receipt of confirmation of a functional installation of ISP provided backdoor software, such as a confirmation that may be provided by the installed software. Another example of a usability test involves the use of a less invasive test of vulnerabilities such a buffer overrun exploit that does not have a permanent impact but provides observable validation (such as a dropped connection). If a backdoor is usable (**1705**) then the computer is inoculated via a backdoor (**1707**) in this example. One example of such an inoculation is the remotely controlled download of one or more remedy files into one or more directories, such as an operating system directory, such as the "\windows\system" directory in Microsoft Windows 2000. Another example of an inoculation is the download of an executable file to the computer, followed by the execution of that file. Another example of an inoculation is the download of a data file, such as an anti-contagion signature file, that is interpreted or processed by another executable already on the computer. Another example of an inoculation is the deletion of one or more files from the computer. Another example of an inoculation is the editing of one or more registry entries, or of one or more files, such as deleting, appending or rewriting entries in a registry or file. Another example of inoculation includes a combination of one or more of above techniques, optionally in conjunction with one or more remotely instigated reboot operations. Another example of inoculation includes a combination of one or more of the above techniques, along with an optional dialog box requesting a user's permission to proceed with the inoculation. Another example of inoculation is the use of a dialog box in conjunction with a timer that causes the inoculation to proceed if no user response is given in a prescribed period of time, such as 1 hour.

FIG. **18** is a flow diagram of a method for categorizing messages according to some embodiments. In this example, a message is received (**1801**), such as an email message or instant message. The purported sender's identity is extracted (**1802**), for example by examining the "From:" header line in an email message, or by examining the envelope information conveying an email message, or by examining the sender's ID in an instant message. A check is performed to see if the sender has sent any prior messages (**1803**), for example by scanning an inbox of prior email, or by checking for the sender in a buddy list. If there were no prior messages from the sender (**1803**) then the message is processed (**1804**) in this example. Processing the message may for example include presenting the message to the user, and/or recording routing information for comparison and categorization of future messages. If there were prior messages from the sender (**1803**) then routing information used in prior messages is retrieved (**1805**) in this example. Routing information may for example be extracted from the header in email. Routing information may for example be obtained during an instant message session by retrieving the IP address of the party being communicated with, for example as is done when an instant message session coordinates a peer-to-peer activity such as a direct file transfer with a remote party. In some embodiments, routing information may be stored implicitly, such as in headers of a previously received message, or explicitly, for example by extracting routing information and storing it in a database or file system. Routing information for the current message is compared to prior messages (**1806**), for example by checking that the IP address for the party being communicated with is in a similar subnet, or allocated within the same geographical region, for example the same continent, as a previous IP address for the same party. IP address to geographical location may be estimated by various software applications and services, such as services provided by MaxMind and described on http://www.maxmind.com. In some embodiments, routing information for an email message may be deduced by examining header information related to mail transfer agents. In some embodiments, similarity in IP addresses for mail transfer agents may be interpreted as implying similar routing. If routing is similar to previous messages from the same sender (**1806**), then the message is processed as legitimate (**1808**) in this example. If the routing information for a message purporting to be from a sender is not similar to previous messages from the same sender (**1806**) then the message is processed as suspicious (**1807**) in this example. In some embodiments, processing as suspicious may include contributing to a scoring that labels the message as an undesirable message, such as spam, viral, or phishing. In some embodiments, higher probability suspicious messages may be scrutinized further, or may be restricted to some extent in their impact on the computer. An example of restricting impact of a message may include discarding the message. Another example of restricting impact or further scrutiny of a message may include placing a message in a spam inbox. Another example of a restricting impact of a message may include preventing execution of content contained in the message, and/or requiring additional user interactions to bypass the restriction, for example by having

a user agree to ignore warnings which describe the suspicious circumstances. In various embodiments, the process of FIG. **18** is implemented on a mail, messaging, or other server associated with the sender and/or the sender's ISP; a mail, messaging, or other server associated with the recipient and/or the recipient's ISP; and/or one or more intermediate nodes.

FIG. **19**A is a flow diagram of a method for notifying possible sources of infection according to some embodiments. In this example, an infected message is received (**1901**). An example of an infected message is a message classified as such by anti-contagion software. Routing information is extracted from the message (**1902**). An example of extracting routing information in an email message is to extract header lines inserted by mail transfer agents that propagated the message. Another example of extracting routing information is to acquire an IP address of a party making contact via instant messaging, for example as is provided by AOL's instant message infrastructure for a peer-to-peer file transfer. Routing information from previously received messages is retrieved (**1903**). An example of retrieving email routing information from prior messages is scanning all received messages, such as messages in an email inbox, and extracting routing information. Another example of retrieving routing information is to query a database or file system where routing information from previous messages has been gathered. Routing information for the infected message is compared with the retrieved routing information (**1904**). If no similarity with previous messages is found (**1904**) then in this example a third party notification is made (**1906**). An example of a third party notification is sending notification to the ISP that provides network access to the message source. Another example of notifying a third party is notifying a viral clearinghouse of the apparent source of a viral message. Another example of a notifying a third party is notifying a service that determines the credibility of a report, such as a collaborative filter service, that accumulates accusations of viral disseminations. In some embodiments, similarity of routing of an infected message with a previous message may include having a previous message from the IP address of the infected message source. If a similarity is found with previous messages (**1904**) then in this example a notification is sent (**1905**). In some embodiments, notification may be sent (**1905**) to one or more sources of the prior message(s) that best match the infected message, and/or to third parties. As an example, notification may be sent to an email source that had the same originating IP address as the infected message. In some embodiments, a user may be asked to approve notification(s), for example via a pop-up dialog. In some embodiments, notification may be automatically sent without user intervention.

FIG. **19**B is a block diagram illustrating a network environment in which suspicious messages are identified and/or one or more notifications are sent when a malicious message is detected. A message source **1920** sends a message via the Internet **1922** to a message processing system **1926** for delivery via the Internet to one or more message destinations **1924**. In various embodiments, the message processing system **1926** comprises a mail or messaging server associated with the message source **1920**, a mail or messaging server associated with the destination(s) **1924**, and/or an intermediate processing node. In some embodiments, the message source **1920** is connected directly and/or via a local network, and not via the Internet **1922** as shown in FIG. **19**B, to message processing system **1926**. In some embodiments, message processing system **1926** is configured to identify suspicious messages using current and historic routing information, e.g., as described above in connection with FIG. **18**. In some embodiments, the process of FIG. **19**A is implemented on message processing system **1926**. For example, in some embodiments if message processing system **1926** determines that a message sent by message source **1920** is infected, message processing system **1926** compares routing information associated with the malicious message with corresponding information for one or more messages received previously from the message source **1920**. If the routing information is similar, in some embodiments the message processing system **1926** notifies the message source **1920**. If the routing information is not similar, in various embodiments the message processing system **1926** notifies the message source **1920**, the sender's ISP **1928**, and/or a clearinghouse or similar applicable service **1930**, as described above.

Although the foregoing embodiments have been described in some detail for purposes of clarity of understanding, the invention is not limited to the details provided. There are many alternative ways of implementing the invention. The disclosed embodiments are illustrative and not restrictive.

What is claimed is:

1. A method, comprising:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one attestation selected from the group consisting of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, determining whether the service request sent by the first host is associated with a remediation request, and when it is determined that the service request sent by the first host is not associated with a remediation request, serving a quarantine notification page that provides remediation information to the first host if the service request sent by the first host comprises a web server request, wherein serving the quarantine notification page to the first host includes re-routing by responding to the service request sent by the first host with a redirect that causes a browser on the first host to be directed to a quarantine server configured to serve the quarantine notification page; and

permitting the first host to communicate with a remediation host configured to provide data usable to remedy the insecure condition.

2. A method as recited in claim **1**, wherein detecting the insecure condition further includes at least one of the group consisting of scanning for a vulnerability, scanning for

malicious data, determining whether a security software is installed, and detecting anomalous network traffic.

3. A method as recited in claim **1**, wherein detecting the insecure condition includes determining that the first host should be quarantined until an update to an operating system has been installed.

4. A method as recited in claim **1**, wherein permitting the first host to communicate with the remediation host includes:

detecting an outbound communication from the first host; and

forwarding the outbound communication if it is addressed to the remediation host.

5. A method as recited in claim **1**, wherein quarantining the first host further includes preventing the first host from receiving via the protected network data not related to remediation of the insecure condition.

6. A method as recited in claim **1**, performed at an Internet service provider.

7. A method as recited in claim **1**, wherein the software component on the first host is an operating system.

8. A method as recited in claim **1**, wherein determining that the response does not include a valid digitally signed attestation of cleanliness includes determining that the software component on the first host is not sufficiently updated.

9. A method as recited in claim **8**, wherein determining that the software component on the first host is not sufficiently updated includes determining that a patch level associated with the software component on the first host is not sufficiently recent.

10. A system, comprising:

a processor configured to:

detect an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one attestation selected from the group consisting of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantine the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, determining whether the service request sent by the first host is associated with a remediation request, and when it is determined that the service request sent by the first host is not associated with a remediation request, serving a quarantine notification page that provides remediation information to the first host if the service request sent by the first host comprises a web server request, wherein serving the quarantine notification page to the first host includes re-routing by responding to the service request sent by the first host with a redirect that causes a browser on the first host to be directed to a quarantine server configured to serve the quarantine notification page; and

permit the first host to communicate with a remediation host configured to provide data usable to remedy the insecure condition; and

a memory coupled to the processor and configured to provide instructions to the processor.

11. A system as recited in claim **10**, wherein the processor is configured to detect an insecure condition at least in part by performing one or more of the following: scanning for a vulnerability, scanning for malicious data, determining whether a security software is installed, and detecting anomalous network traffic.

12. A system as recited in claim **10**, wherein the processor is configured to detect an insecure condition at least in part by determining that an initial startup after installation of an operating system is being performed.

13. A system as recited in claim **10**, wherein the processor is configured to quarantine the first host at least in part by preventing the first host from receiving via the protected network data not related to remediation of the insecure condition.

14. A system as recited in claim **10**, wherein the software component on the first host is an operating system.

15. A system as recited in claim **10**, wherein determining that the response does not include a valid digitally signed attestation of cleanliness includes determining that the software component on the first host is not sufficiently updated.

16. A system as recited in claim **15**, wherein determining that the software component on the first host is not sufficiently updated includes determining that a patch level associated with the software component on the first host is not sufficiently recent.

17. A computer program product, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one attestation selected from the group consisting of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, determining whether the service request sent by the first host is associated with a remediation request, and when it is determined that the service request sent by the first host is not associated with a remediation request, serving a quarantine notification page that provides remediation information to the first host if the service request sent by the first host comprises a web

server request, wherein serving the quarantine notification page to the first host includes re-routing by responding to the service request sent by the first host with a redirect that causes a browser on the first host to be directed to a quarantine server configured to serve the quarantine notification page; and

permitting the first host to communicate with a remediation host configured to provide data usable to remedy the insecure condition.

18. A computer program product as recited in claim **17**, wherein the software component on the first host is an operating system.

19. A computer program product as recited in claim **17**, wherein determining that the response does not include a valid digitally signed attestation of cleanliness includes determining that the software component on the first host is not sufficiently updated.

20. A computer program product as recited in claim **19**, wherein determining that the software component on the first host is not sufficiently updated includes determining that a patch level associated with the software component on the first host is not sufficiently recent.

* * * * *

EXHIBIT 3

Claim Chart

Contagion isolation and inoculation

US8234705B1

vs

Amazon

US8234705B1 vs Amazon AWS Verified Access

Amazon	
Mapped Product	Chart has been made with respect to Amazon AWS Verified Access
Defendant	Amazon

Patent -US8234705B1

US8234705B1 Contagion isolation and inoculation

K.Mizra Patent - US8234705B1

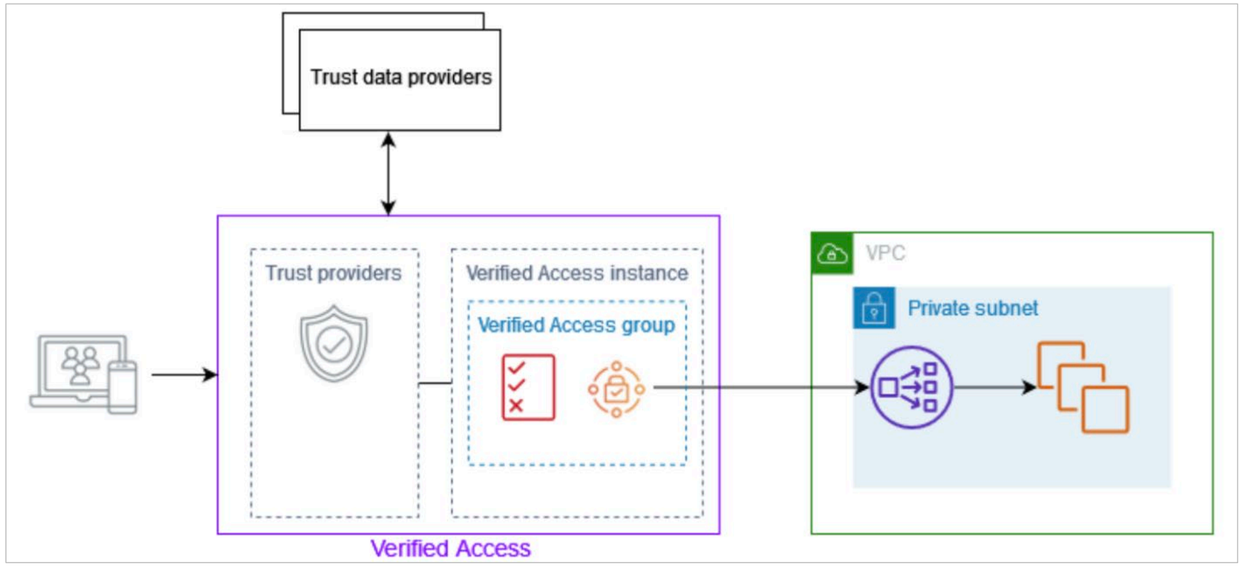
US8234705B1

Contagion isolation and inoculation

Priority Date –September 27, 2004

	• US case filed in Texas Eastern District Court
2005-09-27	• Priority to US11/237,003
2005-09-27	• Application filed by Radix Holdings LLC
2012-07-31	• Publication of US8234705B1
2012-07-31	• Application granted
Status	• Active
2029-11-25	• Adjusted expiration

Show all events



Source: https://aws.amazon.com/blogs/security/how-to-use-aws-verified-access-logs-to-write-and-troubleshoot-access-policies

US8234705B1 vs Amazon AWS Verified Access

Mapped claim(s)	Independent Claim 19

Patent - US8234705B1

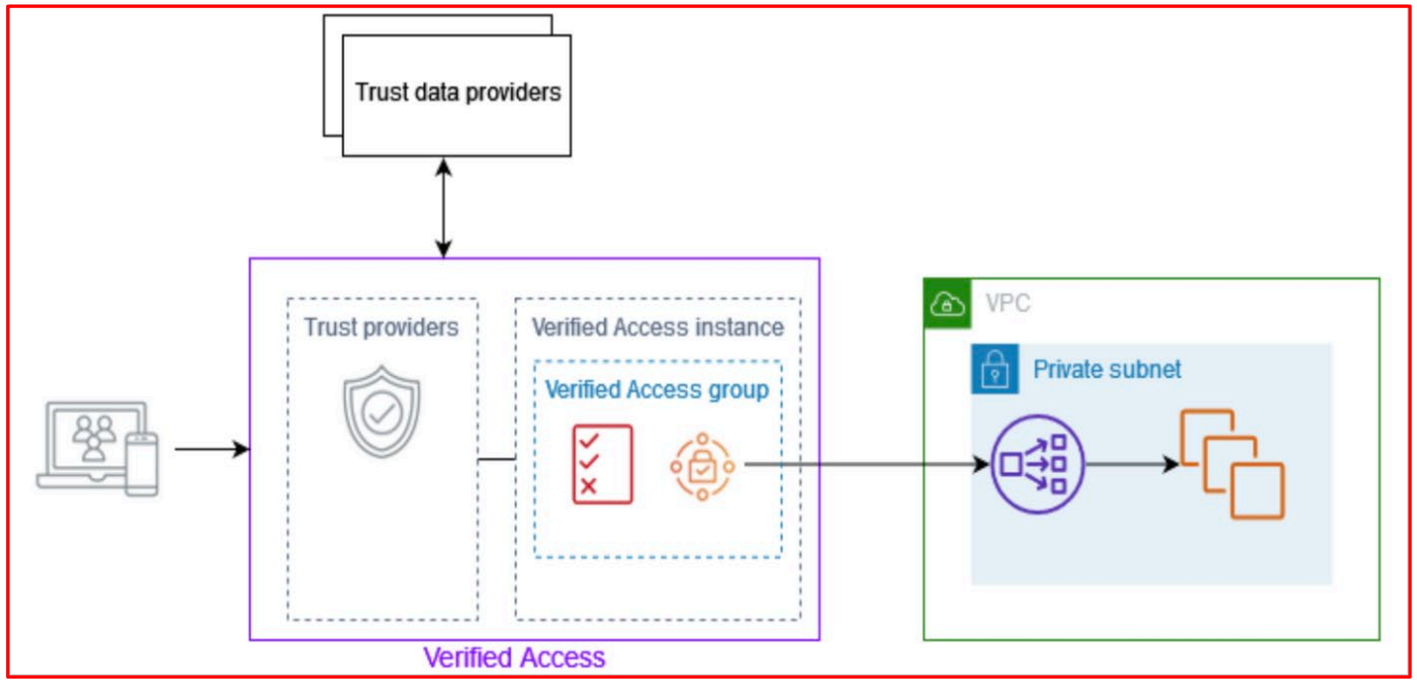
US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

...

Amazon AWS Verified Access



Source: https://aws.amazon.com/blogs/security/how-to-use-aws-verified-access-logs-to-write-and-troubleshoot-access-policies

About AVA

AWS Verified Access (AVA) is a cloud-based service that enables secure access to applications without requiring the use of a virtual private network (VPN). It evaluates each application request and gives users access to each application only when they meet the specified security requirements.

Source: https://aws.amazon.com/blogs/publicsector/aws-verified-access-in-a-tic-3-0-architecture/

Patent - US8234705B1

US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

...

Amazon AWS Verified Access

Pricing table for HTTP(S) applications

There are two pricing dimensions that determine your Verified Access bill for HTTP(S) applications 1) hourly charges for associated applications, and 2) per-GB charges for data processed.

Application hours

You are charged per hour for each application associated with an active HTTP(S) Verified Access endpoint. Each partial application hour (app-hour) consumed is billed as a full hour. For example, if you associate 10 applications with 10 Verified Access endpoints, each for an hour, you will be charged 10 app-hours.

GB of data processed

You are also charged per GB for data processed by Verified Access. When users request access or connect to your applications, Verified Access processes the data flowing between users and the application. For example, if all your applications using Verified Access require 10 GB of total data processing, you will be charged for that 10 GB of data processed.

Tiered pricing for app hours	**Price**
1 to 148,800 app hours (covers up to 200 apps in a month)	$0.27/hr
148,800+ app hours	$0.20/hr
Note: 200 apps associated for 31 days equals 148,800 app hours	
Per GB of data processed	$0.02

Example 2:

You created a Verified Access in US East (Ohio) and associated 300 applications. These 300 applications were associated for a month with 31 days. This will result in 223,200 app hours. Also, each application required 1 GB for data processing. The total charge will be $55,062.00, and the below table provides the details.

First 148,800 app hours	148,800 hours * $0.27	$40,176.00
Next 74,400 (223,200-148,800) app hours	74,400 hours * $0.20	$14,880.00
Data processing	1 GB * 300 applications * $0.02	$6.00
Total charge		**$55,062.00**

Source: https://aws.amazon.com/verified-access/pricing/

Patent - US8234705B1

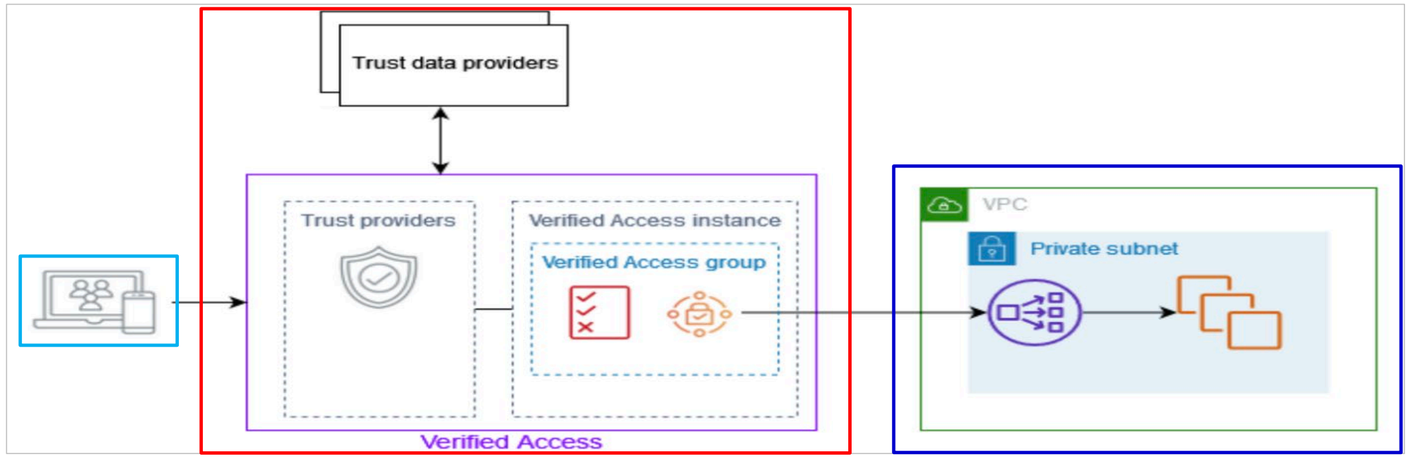
US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

...

Amazon AWS Verified Access



Source: https://aws.amazon.com/blogs/security/how-to-use-aws-verified-access-logs-to-write-and-troubleshoot-access-policies

Why AWS Verified Access?

AWS Verified Access provides secure access to corporate applications and resources without a VPN. It enhances your security posture by allowing you to define fine-grained access policies based on a user's identity and device security state, and enforcing policies on every access request. It simplifies security operations by allowing administrators to create, group, and manage access policies for applications and resources with similar security requirements from a single interface. Verified Access logs each access attempt, so you can efficiently respond to security and connectivity incidents.

Improve your security posture by defining granular access policies based upon user's identity and device security status for your corporate applications including web-based applications accessed using browsers and infrastructure resources such as databases and EC2 instances.

Application owners can apply zero trust access controls to their corporate applications by constantly verifying each access request against granular, conditional access policies. This ensures that access is granted per application only when specific security requirements like user identity and device security posture are met and maintained.

Get comprehensive logging and visibility into access attempts to quickly identify and resolve security and connectivity incidents.

Source: https://aws.amazon.com/verified-access/

Patent - US8234705B1

US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

...

Amazon AWS Verified Access

When a user tries to access an application, Verified Access gets their data from the trust provider and evaluates it against the policies that you set for the application. Verified Access grants access to the requested application only if the user meets your specified security requirements. All application requests are denied by default, until a policy is defined.

In addition, Verified Access logs every access attempt, to help you respond quickly to security incidents and audit requests.

The following diagram provides a high-level overview of Verified Access. Users send requests to access an application. Verified Access evaluates the request against the access policy for the group and any application-specific endpoint policies. If access is allowed, the request is sent to the application through the endpoint.

Source: https://docs.aws.amazon.com/verified-access/latest/ug/how-it-works.html

A trust provider is a service that sends information about users and devices to AWS Verified Access. This information is called trust context. It can include attributes based on user identity, such as an email address or membership in the "sales" organization, or device information such as installed security patches or anti-virus software version.

Verified Access supports the following categories of trust providers:

- **User identity** – An identity provider (IdP) service that stores and manages digital identities for users.
- **Device management** – A device management system for devices such as laptops, tablets, and smartphones.

Source: https://docs.aws.amazon.com/verified-access/latest/ug/trust-providers.html

Patent - US8234705B1

US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

...

Amazon AWS Verified Access

If you plan to incorporate device trust context into your access policies, then you will need either the AWS Verified Access browser extension, or another partner's browser extension. Verified Access currently supports Google Chrome and Mozilla Firefox browsers.

We currently support three device trust providers: Jamf (which supports macOS devices), CrowdStrike (which supports Windows 11 and Windows 10 devices), and JumpCloud (which supports both Windows and MacOS).

- If you're using **Jamf** trust data in your policies, your users must download and install the AWS Verified Access browser extension from the Chrome web store or Firefox Add-on site on their devices.
- If you are using **CrowdStrike** trust data in your policies, first your users need to install the AWS Verified Access Native Messaging Host (direct download link). This component is required to get the trust data from the CrowdStrike agent running on users' devices. Then, after installing this component, users must install the AWS Verified Access browser extension from the Chrome web store or Firefox Add-on site on their devices.
- If you're using **JumpCloud**, your users must have the JumpCloud browser extension from the Chrome web store or Firefox Add-on site installed on their devices.

Source: https://docs.aws.amazon.com/verified-access/latest/ug/trust-data-third-party-trust.html

First, AWS Verified Access evaluates each application request in real time using Zero Trust principles and multiple security factors, such as identity and device posture. Agencies can manage identity through AWS IAM Identity Center (successor to AWS Single Sign-On) or any OpenID Connect (OIDC) compatible solution. AWS Verified Access can also leverage device-level data with browser plugins from supported third-party trust providers, such as JAMF for macOS devices and Crowdstrike for Windows devices. Plugins retrieve client device information and share it with AWS Verified Access through a browser extension.

Second, AWS Verified Access lets administrators define fine-grained access policies through Cedar, a language for defining permissions as policies, which describe who should have access to what.

Source: https://docs.aws.amazon.com/verified-access/latest/ug/trust-data-third-party-trust.html

Patent - US8234705B1

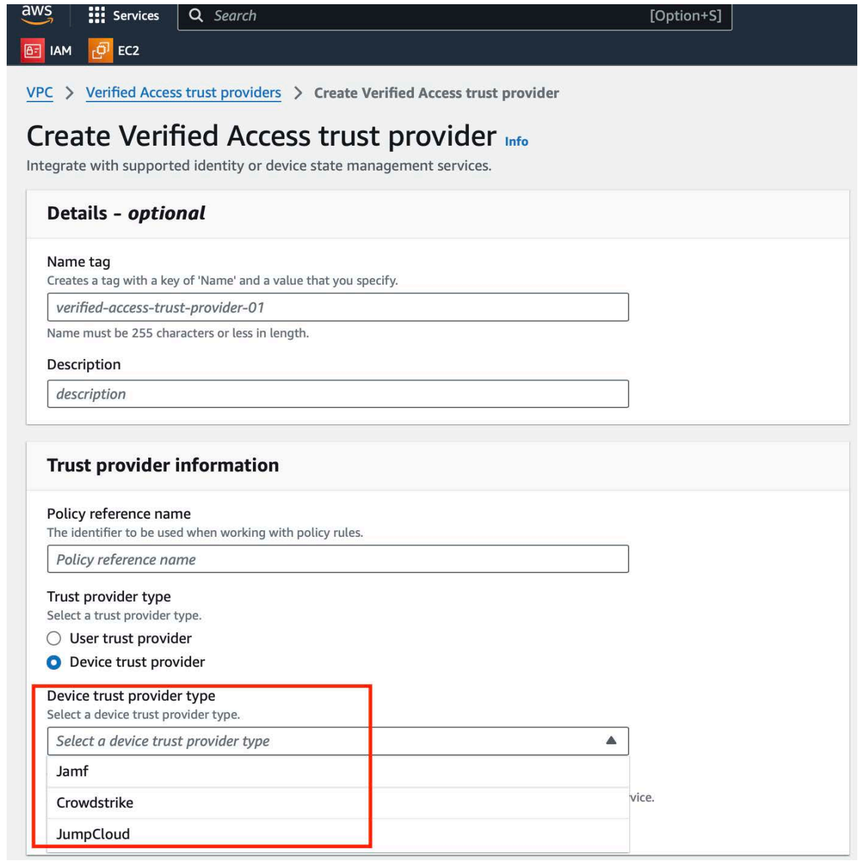
US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

...

Amazon AWS Verified Access



Source: AWS Verified Access service

Patent - US8234705B1

US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

...

Amazon AWS Verified Access

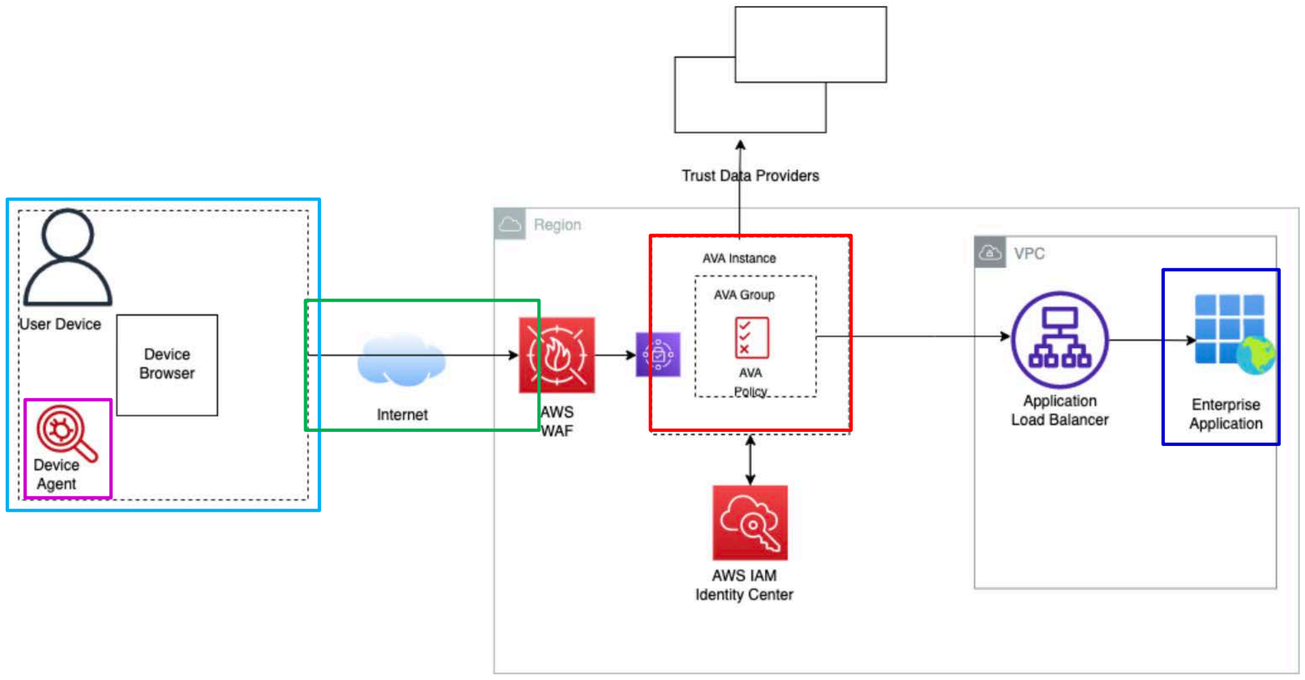


Figure 2. An architecture model in which agency employees can access AWS services directly over the internet. AWS WAF provides protection against common web exploits and AWS Verified Access evaluates every request and provides access to the application.

Source: https://aws.amazon.com/blogs/publicsector/aws-verified-access-sd-wan-streamline-secure-remote-application-access-federal-employees/

Patent - US8234705B1

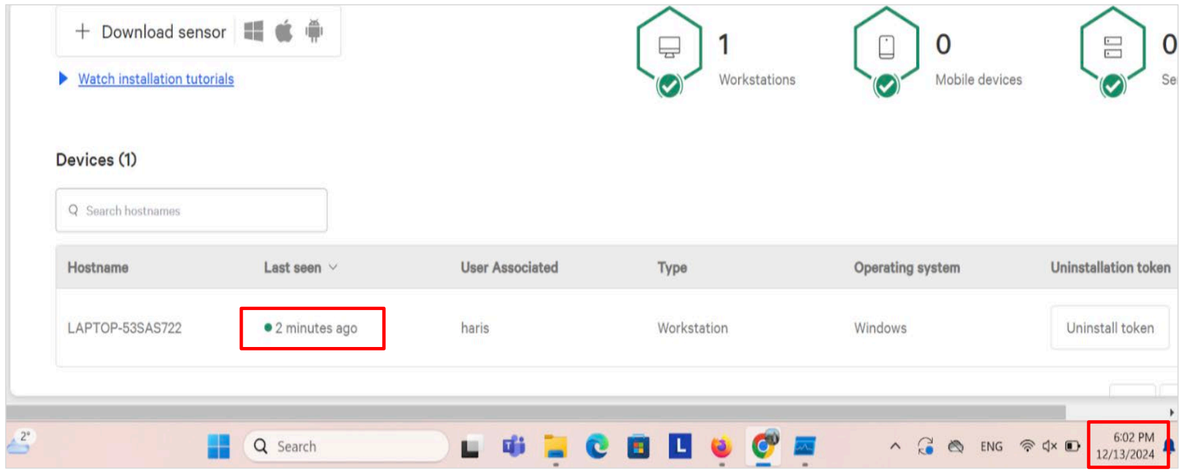
US8234705B1 (Claim 19)

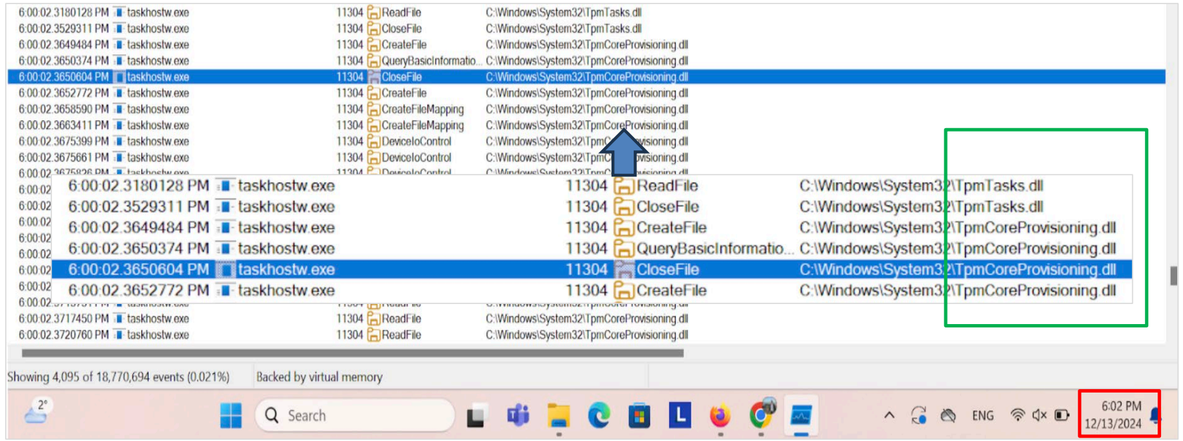
A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on **a first host that has connected or is attempting to connect to a protected network**, wherein detecting the insecure condition includes **contacting a trusted computing base associated with a trusted platform module within the first host**, **receiving a response**, **and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested**, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

...

Amazon AWS Verified Access (CrowdStrike)





Patent - US8234705B1

US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on **a first host that has connected or is attempting to connect to a protected network**, wherein detecting the insecure condition includes **contacting a trusted computing base associated with a trusted platform module within the first host**, **receiving a response**, **and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested**, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

...

Amazon AWS Verified Access (CrowdStrike)

Event Properties

Event | Process | Stack

Image

Host Process for Windows Tasks

Microsoft Corporation

Name: taskhostw.exe

Version: 10.0.22621.4599 (WinBuild.160101.0800)

Path:

C:\Windows\system32\taskhostw.exe

Command Line:

taskhostw.exe SBServicing

PID:	11304	Architecture:	64-bit
Parent PID:	2280	Virtualized:	False
Session ID:	0	Integrity:	Verbindliche Beschriftung\
User:	NT-AUTORITÄT\SYSTEM		
Auth ID:	00000000:000003e7		
Started:	12/13/2024 6:00:02 PM	Ended:	12/13/2024 6:00:02 PM

Modules:

Module	Address	Size	Path	Company	Version	Timestamp
umppc18721.dll	0x24807190000	0x14000	C:\Windows\System32\umppc18721.dll	CrowdStrike, Inc.	7.17.18721.0	9/13/2024 10:10:...
umppc18721.dll	0x248071d0000	0x14000	C:\Windows\System32\umppc18721.dll	CrowdStrike, Inc.	7.17.18721.0	9/13/2024 10:10:...
CsXumd64_1872...	0x24807220000	0x12000	C:\Windows\System32\CsXumd64_18...	CrowdStrike, Inc.	7.17.18721.0	9/13/2024 10:11:...

Patent - US8234705B1

US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on **a first host that has connected or is attempting to connect to a protected network**, wherein detecting the insecure condition includes **contacting a trusted computing base associated with a trusted platform module within the first host**, **receiving a response**, **and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested**, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

...

Amazon AWS Verified Access (JumpCloud)

AWS and JumpCloud have partnered to offer a VPN alternative for securely accessing company applications over the web using JumpCloud Go™, JumpCloud SSO (SAML or OIDC) and AWS Verified Access (AVA). With this collaboration, JumpCloud can authenticate and authorize devices and identities in one console and provide access control for AWS Verified Access for managed devices. See our blog for more information about our partnership.

Source: https://docs.aws.amazon.com/verified-access/latest/ug/trust-providers.html

JumpCloud Go supports the following device types that meet these hardware requirements:

- macOS devices with a Secure Enclave.
- Windows devices with a Trusted Platform Module (TPM) 2.0.
- Linux devices (GNOME-based distros) with a Trusted Platform Module (TPM) 2.0.

The JumpCloud agent has to be installed and running on macOS, Windows, and Linux devices. See Install the JumpCloud Agent.

Source: https://jumpcloud.com/support/get-started-jumpcloud-go

JumpCloud Go anchors its trust to both the device and the user. During the enrollment process the user authenticates to JumpCloud's OpenID Connect service and is issued an ID Token for the JumpCloud Go application. A new DUK (Device User Key) is created from the Secure Enclave (on Apple devices) or the Trusted Platform Module (TPM) (on Windows systems). This is added to a JWT (JSON Web Token) that is then signed by the JumpCloud Agent certificate unique to the device and used by the JumpCloud Agent to authenticate all traffic back to JumpCloud.

Zero trust emphasizes that only trusted devices access resources. JumpCloud Go anchors trust to both the device and the device identity within Apple's Secure Enclave or the Trusted Platform Module (TPM). Using device-specific cryptographic methods instead of passwords boosts security and forms the basis for Zero Trust.

Source: https://jumpcloud.com/wp-content/uploads/2023/11/202310-WP-JumpCloudGo.pdf

Patent - US8234705B1

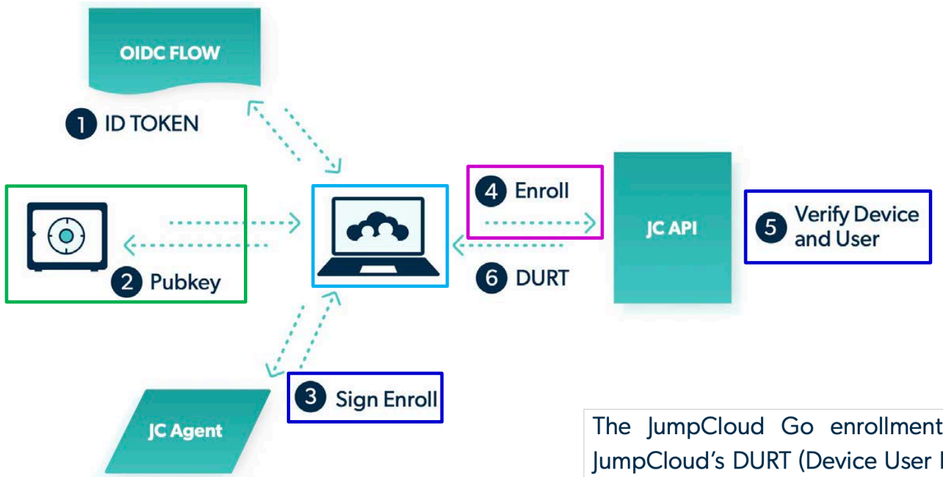
US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes **contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested**, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

...

Amazon AWS Verified Access (JumpCloud)



After registration is complete, a new DURT (Device User Refresh Token) is generated by the DURT API and registered to the unique user-device combination. The DURT is sent back down to the client where it is treated in a similar fashion to an OAuth Access token. The DURT is kept in the local user's keychain along with a reference to the DUK stored in the hardware-backed encryption service on the device.

The JumpCloud Go enrollment request is validated by JumpCloud's DURT (Device User Refresh Token) API through the following process:

1. The JWT's signature is validated by first ensuring the certificate in the x5c claim in the JWT header is a valid JumpCloud Agent certificate in good standing.
2. The JWT signature is checked to ensure it was signed by the private key of the same agent certificate.
3. The JWT is inspected to ensure it hasn't expired and is otherwise valid.
4. The embedded ID Token is validated back to the JumpCloud OpenID Connect service via its embedded signature using the OIDC JWKS.

Source: https://jumpcloud.com/wp-content/uploads/2023/11/202310-WP-JumpCloudGo.pdf

Patent - US8234705B1

US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes **contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested**, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

...

Amazon AWS Verified Access

The following use cases provide examples for scenarios in which the TPM provides security to End Users and Platform Administrators. They also establish the assumptions and drive recommendations for the TPM state before and during all stages of provisioning. This document groups the use cases into classes of use cases. This convention helps the reader more easily identify which recommendations accommodate each class of use cases.

The TPM can provide a standardized, unspoofable, hardware-based identity – in contrast with IP and MAC addresses that are relatively trivial to spoof. Enterprises find this property useful if they need to identify Platforms on their network for many reasons, including, but not limited to, the following:

Compliance Reporting - For Enterprises to reliably match compliance, health, and posture reports with Platforms, they require a durable unique identifier for each Platform. Such an identifier allows Network Administrators to locate, quarantine, or remediate Platforms that have fallen out of compliance with network policy.

Source: https://trustedcomputinggroup.org/wp-content/uploads/TCG-TPM-v2.0-Provisioning-Guidance-Published-v1r1.pdf

Patent - US8234705B1

US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes **contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;**

...

Amazon AWS Verified Access

2.1 TCG Attestation

Within the TCG context, Attestation is a process for determining the identity of a device and the software running on the device. Attestation is broken into two phases, shown in Figure 1:

- During system startup, measurements (i.e., hashes computed as fingerprints of files) are "extended", or stored, in the TPM, along with entries added to an informational log. The measurement process generally follows the Chain of Trust model used in Measured Boot, where each stage of the system measures the next one before launching it.
- Once the device is running and has operational network connectivity, a separate, trusted server (called a Verifier in this document) can interrogate the network device to retrieve the logs and a copy of the digests collected by hashing each software object, signed by a key known only to the TPM.

The result is that the Verifier can verify the device's identity by checking the certificate corresponding to the TPM's attestation key, and can validate the software that was launched by comparing digests in the log with known-good values, and verifying their correctness by comparing with the signed digests from the TPM.

It should be noted that attestation and identity are inextricably linked; signed evidence that a particular version of software was loaded is of little value without cryptographic proof of the identity of the device producing the evidence.[13]

Source: https://trustedcomputinggroup.org/wp-content/uploads/TCG-NetEq-Attestation-Workflow-Outline_v1r9b_pubrev.pdf

5.7 Attestation of Integrity for Network Devices ("Health Check")

Attestation is a mechanism by which *measurements* stored in a TPM can be retrieved by a remote attestation server to determine the configuration of the equipment at boot time as well as at runtime. In the case of Network Equipment, remote attestation might be invoked by a management station (called the Challenger or Verifier), causing a networking device to report its boot (and runtime) configuration. Alternatively, two peer devices might do Mutual Attestation to prove to each other that their configurations are acceptable.

Attestation of both boot time and run time configuration can include a wide range of objects that might have a bearing on the security posture of a device.

Boot time objects typically include:

- The initial firmware loaded (such as UEFI code)
- Options of the initial firmware
- Boot loader
- OS kernel

Source: https://trustedcomputinggroup.org/wp-content/uploads/TCG_Guidance_for_Securing_NetEq_1_0r29.pdf

Patent - US8234705B1

US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

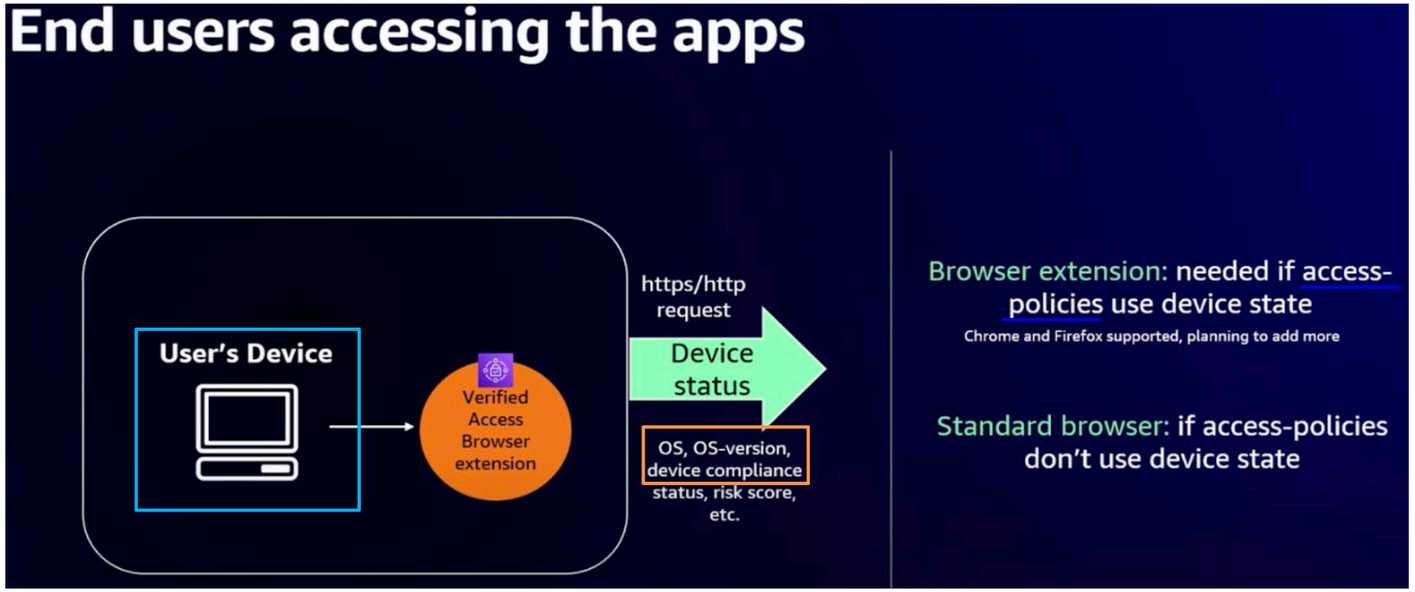
detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, **and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that** the trusted computing base **has ascertained that the first host is not infested**, **and an attestation that** the trusted computing base **has ascertained the presence of a patch or a patch level associated with a software component on the first host**;

...

Amazon AWS Verified Access

A trust provider is a service that sends information about users and devices to AWS Verified Access. This information is called trust context. It can include attributes based on user identity, such as an email address or membership in the "sales" organization, or device information such as installed security patches or anti-virus software version.

Source: https://docs.aws.amazon.com/verified-access/latest/ug/trust-providers.html



Source: https://www.youtube.com/watch?v=GaDqIZ5bQBc at 18:03

Patent - US8234705B1

US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, **and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that** the trusted computing base **has ascertained that the first host is not infested, and an attestation that** the trusted computing base **has ascertained the presence of a patch or a patch level associated with a software component on the first host**;

...

Amazon AWS Verified Access (CrowdStrike)

CrowdStrike

CrowdStrike is a third-party trust provider. When a policy is evaluated, if you define CrowdStrike as a trust provider, Verified Access includes the trust data in the Cedar context under the key you specify as "Policy Reference Name" on the trust provider configuration. You can write a policy that evaluates against the trust data if you choose. The following JSON schema shows which data is included in the evaluation.

```
{
  "title": "CrowdStrike device data specification",
  "type": "object",
  "properties": {
    "assessment": {
      "type": "object",
      "description": "Data about CrowdStrike's assessment of the device",
      "properties": {
        "overall": {
          "type": "integer",
          "description": "A single metric, between 1-100, that accounts as a weighted averc
        },
        "os": {
          "type": "integer",
          "description": "A single metric, between 1-100, that accounts for the OS-specific
        },
        "sensor_config": {
          "type": "integer",
          "description": "A single metric, between 1-100, that accounts for the different s
        },
        "version": {
          "type": "string",
          "description": "The version of the scoring algorithm being used"
        }
      }
    },
    },
    "platform": {
      "type": "string",
      "enum": ["Windows 10", "Windows 11", "macOS"],
      "description": "Operating system of the endpoint"
```

Source: https://docs.aws.amazon.com/verified-access/latest/ug/trust-data-third-party-trust.html

Patent - US8234705B1

US8234705B1 (Claim 19)

A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, **and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that** the trusted computing base **has ascertained that the first host is not infested**, **and an attestation that** the trusted computing base **has ascertained the presence of a patch or a patch level associated with a software component on the first host**;

...

Amazon AWS Verified Access (JumpCloud)

JumpCloud's automated patch management helps you monitor which version and release your Windows, macOS, and Linux devices are currently using, and remotely manage update availability and the installation of those updates.

An OS patch management policy can control when devices can see available OS upgrades and minor updates and how long an update can be deferred. Don't apply multiple patch policies to a device.

These patch policies are available for Mac, Windows, and Linux devices:

OS	Policy Name	Description
macOS	Automatic MacOS Updates	Lets you manage and enforce when major OS upgrades and minor update patches are available and applied to your devices. You can also configure how much flexibility your users have in delaying or installing the update. See Create a MacOS Patch Policy.
Windows	Configure Advanced Windows Updates	Allows you to control when and how Windows updates and upgrades are downloaded and delivered. See Create a Windows Patch Policy.
Linux	Configure Ubuntu Updates	Enables you to defer the availability of OS release upgrades (for example, upgrading from 18.04 to 20.04), as well as restrict updates to only LTS releases. It provides functionality equivalent to the Software and Updates desktop program on the device, plus the additional ability to delay OS release upgrades. See Create a Linux Patch Policy.

Source: https://jumpcloud.com/support/choose-an-os-patch-management-policy

AWS and JumpCloud have partnered to offer a VPN alternative for securely accessing company applications over the web using JumpCloud Go™, JumpCloud SSO (SAML or OIDC) and AWS Verified Access (AVA). With this collaboration, JumpCloud can authenticate and authorize devices and identities in one console and provide access control for AWS Verified Access for managed devices. See our blog for more information about our partnership.

Source: https://jumpcloud.com/support/choose-an-os-patch-management-policy

Patent - US8234705B1

US8234705B1 (Claim 19)

...

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition; and

permitting the first host to communicate with the remediation host.

Amazon AWS Verified Access

The following is an example of a policy that evaluates against the trust data provided by CrowdStrike.

```
permit(principal, action, resource) when {
   context.crowdstrike.assessment.overall > 50
};
```

The following is an example of a policy that evaluates against the trust context provided by JumpCloud.

```
permit(principal, action, resource) when {
   context.jumpcloud.org_id = 'Unique_orgnaization_identifier'
};
```

```
"title": "JumpCloud device data specification",
"type": "object",
"properties": {
  "device": {
    "type": "object",
    "description": "Properties of the device",
    "properties": {
      "is_managed": {
        "type": "boolean",
        "description": "Boolean to indicate if the device is under management"
```

Source: https://docs.aws.amazon.com/verified-access/latest/ug/trust-data-third-party-trust.html

Patent - US8234705B1

US8234705B1 (Claim 19)

...

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition; and

permitting the first host to communicate with the remediation host.

Amazon AWS Verified Access

When a user tries to access an application, Verified Access gets their data from the trust provider and evaluates it against the policies that you set for the application. Verified Access grants access to the requested application only if the user meets your specified security requirements. All application requests are denied by default, until a policy is defined.

In addition, Verified Access logs every access attempt, to help you respond quickly to security incidents and audit requests.

Verified Access instances – An instance evaluates application requests and grants access only when your security requirements are met.

Access policies – A set of user-defined rules that determine whether to allow or deny access to an application. You can specify a combination of factors, including user identity and device security state. You create a group access policy for each Verified Access group, which is inherited by all endpoints in the group. You can optionally create application-specific policies and attach them to specific endpoints.

Trust data – The security-related data for users or devices that your trust provider sends to Verified Access. Also referred to as *user claims* or *trust context*. For example, the email address of a user or the operating system version of a device. Verified Access evaluates this data against your access policies when it receives each request to access an application.

Source: https://docs.aws.amazon.com/verified-access/latest/ug/how-it-works.html

Use a web browser with the Verified Access extension to test access to the application. Before signing in, you should be redirected to the Okta sign-in page. After successful authentication, if the device does not have a ZTA score above the threshold in the group policy, then the user receives a 403 Unauthorized error.

Source: https://aws.amazon.com/blogs/networking-and-content-delivery/integrating-aws-verified-access-with-device-trust-providers/

Patent - US8234705B1

US8234705B1 (Claim 19)

...

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network **includes receiving a service request sent by the first host**, **serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query**, **providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition**; and

permitting the first host to communicate with the remediation host.

Amazon AWS Verified Access

Use a web browser with the Verified Access extension to test access to the application. Before signing in, you should be redirected to the Okta sign-in page. After successful authentication, if the device does not have a ZTA score above the threshold in the group policy, then the user receives a 403 Unauthorized error.

Source: https://aws.amazon.com/blogs/networking-and-content-delivery/integrating-aws-verified-access-with-device-trust-providers/

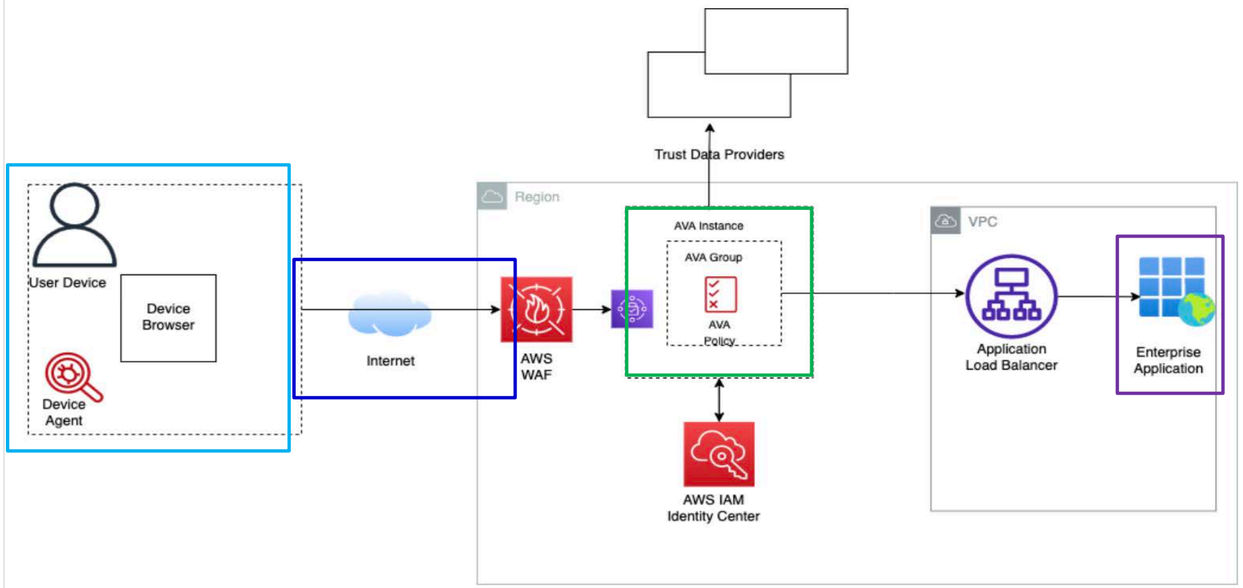


Figure 2. An architecture model in which agency employees can access AWS services directly over the internet. AWS WAF provides protection against common web exploits and AWS Verified Access evaluates every request and provides access to the application.

Source: https://aws.amazon.com/blogs/publicsector/aws-verified-access-sd-wan-streamline-secure-remote-application-access-federal-employees/

Patent - US8234705B1

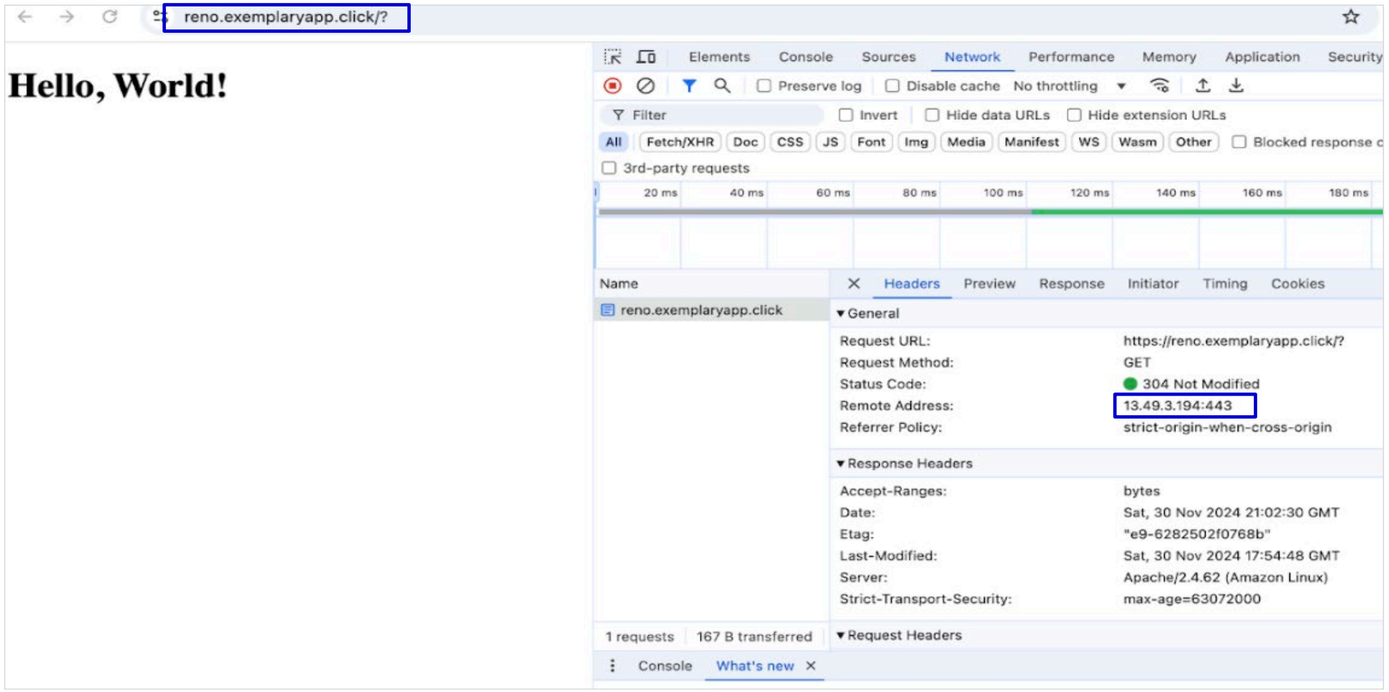
US8234705B1 (Claim 19)

...

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network **includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition**; and

permitting the first host to communicate with the remediation host.

Amazon AWS Verified Access



Source: AWS Verified Access service

Patent - US8234705B1

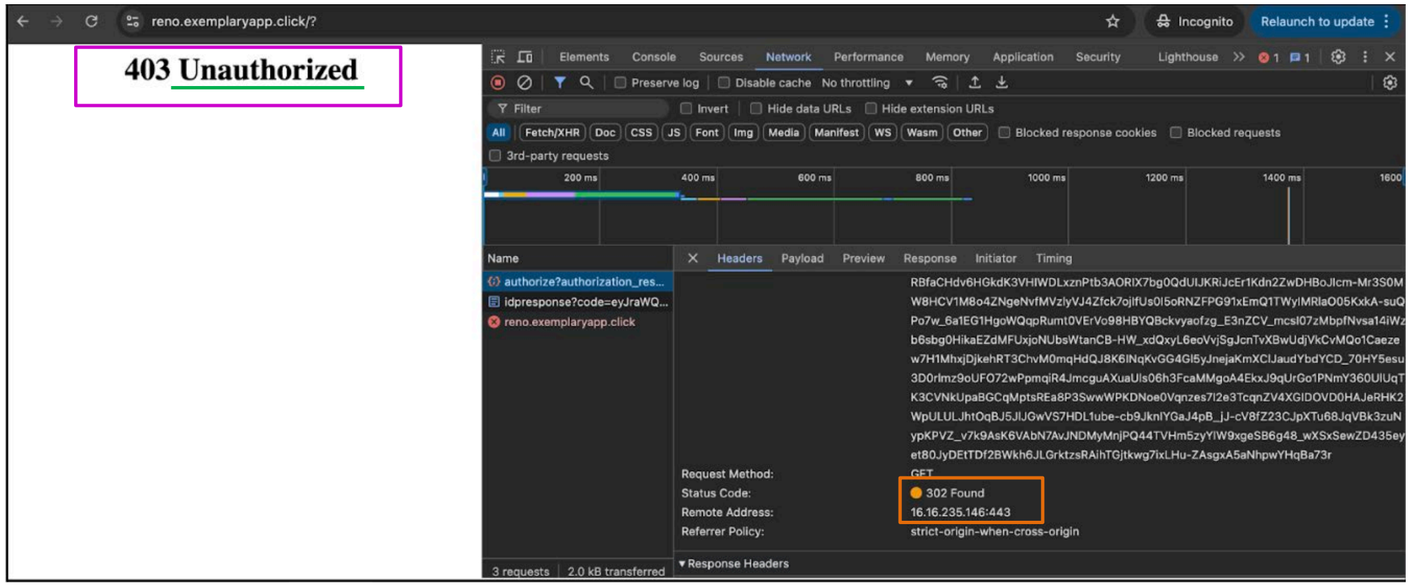
US8234705B1 (Claim 19)

...

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network **includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition**; and

permitting the first host to communicate with the remediation host.

Amazon AWS Verified Access



Source: AWS Verified Access service

Patent - US8234705B1

US8234705B1 (Claim 19)

...

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network **includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition**; and

permitting the first host to communicate with the remediation host.

Amazon AWS Verified Access

Source: https://qiita.com/hayao_k/items/5f3b8a88a0ea75828f95

Patent - US8234705B1

US8234705B1 (Claim 19)

...

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition; and

permitting the first host to communicate with the remediation host.

Amazon AWS Verified Access

A trust provider is a service that sends information about users and devices to AWS Verified Access. This information is called trust context. It can include attributes based on user identity, such as an email address or membership in the "sales" organization, or device information such as installed security patches or anti-virus software version.

Source: https://docs.aws.amazon.com/verified-access/latest/ug/trust-providers.html

Improved security posture – A traditional security model evaluates access once and grants the user access to all applications. Verified Access evaluates each application access request in real time. This makes it difficult for bad actors to move from one application to another.

Source: https://docs.aws.amazon.com/verified-access/latest/ug/what-is-verified-access.html

Application owners can apply zero trust access controls to their corporate applications by constantly verifying each access request against granular, conditional access policies. This ensures that access is granted per application only when specific security requirements like user identity and device security posture are met and maintained.

Source: https://aws.amazon.com/verified-access/