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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 December 2012 (27.12.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/177597 A1

(51) International Patent Classification:
G06F 9/445 (2006.01)

(21) International Application Number:
PCT/US2012/043084

(22) International Filing Date:
19 June 2012 (19.06.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/500,800 24 June 2011 (24.06.2011) US

(71) Applicant *(for all designated States except US)*: **SIEMENS AKTIENGESELLSCHAFT** [DE/DE]; Wittelsbacherplatz 2, 80333 Munich (DE).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **AL FARUQUE, Mohammad Abdullah** [DE/US]; 1608 Fox Run Drive, Plainsboro, New Jersey 08536 (US). **DALLORO, Livio** [IT/US]; 141 Fairway Drive, Princeton, New Jersey 08540 (US). **LUDWIG, Hartmut** [US/US]; 9 Robert Drive, West Windsor, New Jersey 08550 (US). **CLAUS, Joerg** [DE/US]; 3 Major Lane, Plainsboro, New Jersey 08536 (US). **FRÖHLICH, Robert** [DE/DE]; Untere Klosterstr. 34, 76684 Östringen (DE). **BUTUN, Ismail** [TR/US]; 22 S. Linden Lane, Plainsboro, New Jersey 08536 (US).

(74) Agents: **CONOVER, Michele, L.** et al.; Siemens Corporation- Intellectual Property Dept., 170 Wood Avenue South, Iselin, New Jersey 08830 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

(54) Title: NETWORKING ELEMENTS AS A PATCH DISTRIBUTION PLATFORM FOR DISTRIBUTED AUTOMATION AND CONTROL DOMAINS

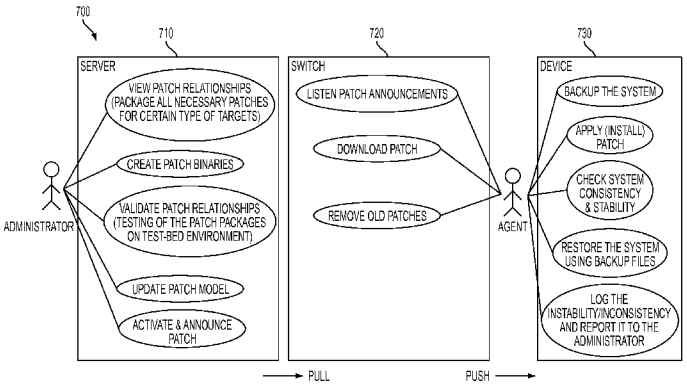

FIG. 7

(57) Abstract: A patch management system and method increases performance, scalability, resiliency and safety by deploying the major part of the patch distribution using the existing computational power of the networking elements, such as switches or routers at a plant location. The patch management function in an automation network is decentralized and shifted downstream, closer to the automation devices. Patches are distributed in a peer-to-peer topology or a server-router-device topology.

WO 2012/177597 A1

NETWORKING ELEMENTS AS A PATCH DISTRIBUTION PLATFORM FOR DISTRIBUTED AUTOMATION AND CONTROL DOMAINS

Cross-Reference to Related Applications

[1] This application claims the benefit of United States Provisional Patent Application Ser. No. 61/500,800, filed June 24, 2011, the contents of which are hereby incorporated by reference herein.

Field of the Invention

[2] This invention relates generally to industrial and other automation systems, and more particularly to methods, systems and computer readable media for distributing software patches to automation devices and associated network elements.

Background of the Invention

[3] A software patch is a piece of software code designed to fix problems with, or update, a computer program or its supporting data. Patches are typically used to add new functionality or usability features to software currently installed in a device, and to increase the performance and stability of the overall system. Patches may also be used to fix security vulnerabilities and other problems with installed software. Patches are used in any system in which software is used, including industrial control systems, building automation systems, factory automation systems and other automation systems.

[4] Patch management is a system management tool for overseeing the installation of patches in a system that includes multiple devices running multiple programs. In general, patch management involves the following: system and

vulnerability assessment, patch aggregation, patch testing, and distribution and installation (deployment) of multiple patches to an administered computing/control system. Patch management may further include maintaining current knowledge of the available patches, deciding which patches are appropriate for each individual connected system, ensuring that patches are installed properly, post-installation testing, and documenting all the associated procedures as required by the specific configurations.

[5] Because software vulnerabilities are dramatically increasing, patch management is becoming more important. In the last decade, the number of vulnerabilities affecting end-users increased exponentially, with vulnerabilities doubling every year over the amount of the previous year. For example, there were around 760 vulnerabilities per computing device in the year 2010. Most commonly, patch management involves the following types of patches: operating system patches, application patches and configuration update patches. Patches may be distributed in three ways: (1) patch as a source code of a program requiring re-compilation; (2) patch to the compiled binary code; and (3) patch as a complete file(s) replacement.

[6] In the past, security was provided in the distributed automation and control domain by obscuring and segregating the plant network from the external world. Due to the increasing trend of connecting the control system to the external world through public networks, the typical control system is becoming vulnerable to the same risks as a typical IT network. As a recent example, Stuxnet, discovered in June 2010, was an application worm that exploited several layers of vulnerabilities of a SCADA system and Programmable Logic Controllers (PLC) at the uranium enrichment facility at Natanz, Iran. The outcome of that cyber-attack decreased a uranium centrifuge operational

capacity by 30%. Because control systems directly monitor and control physical processes, the result of a cyber-attack may be extreme in terms of financial cost and even loss of human life. To protect against security vulnerabilities and/or bugs, a strong, efficient, resilient, secure, scalable, and safe patch management scheme is therefore needed.

Summary of the Invention

[7] The present invention addresses the needs described above by providing a method for distributing software patches in an automation network. The automation network includes an intermediate network element, first and second downstream network elements downstream of the intermediate network element, and a first automation device downstream of the first downstream network element.

[8] A determination is made that software patch is available for firmware installed on the first automation device, and that the software patch is not stored on the first downstream network element. Based on information stored at the intermediate network element, it is further determined that the software patch is stored on the second downstream network element.

[9] The software patch is copied from the second downstream network element to the first downstream network element, and is transmitted from the first downstream network element to the first automation device for installation.

[10] The automation network may additionally include a second automation device downstream of the first downstream element. In that case, it is determined that a second software patch is available for second firmware installed on the second

automation device, and that the second software patch is not stored on either the first or the second downstream network elements. The second software patch is then requested and received from from a patch management server, and is stored on the first downstream network element. The second software patch is transmitted from the first downstream network element to the second automation device for installation.

[11] In another aspect of the invention, a downstream network element includes a non-transitory computer-usable medium having computer readable instructions stored thereon for execution by a processor to perform methods for distributing software patches as described above.

Brief Description of the Drawings

[12] FIG. 1 is a schematic diagram of a presently used patch distribution mechanism.

[13] FIG. 2 is a screen shot of a firmware update interface of a presently used patch distribution mechanism.

[14] FIG. 3 is a schematic diagram of a patch distribution system in accordance with one embodiment of the invention.

[15] FIG. 4 is a life cycle time sequence of a computer software patch, showing a reduction in patch distribution time in accordance with one embodiment of the invention.

[16] FIG. 5 is a schematic diagram of a patch distribution system illustrating a server-router-device model in accordance with one embodiment of the invention.

[17] FIG. 6 is an activity diagram of a patch distribution system in accordance with one embodiment of the invention.

[18] FIG. 7 is an information flow diagram of a patch distribution system illustrating a server-router-device model in accordance with one embodiment of the invention.

[19] FIG. 8 is a schematic diagram of a patch distribution system illustrating a peer-to-peer model in accordance with one embodiment of the invention.

[20] FIG. 9 is a detailed sequence diagram of a patch distribution system in accordance with the invention.

[21] FIG. 10 is a schematic diagram showing a computer system for instantiating a system in accordance with one embodiment of the invention.

Description of the Invention

[22] The present disclosure describes a patch management system and method in which the functionality of existing network elements is exploited to achieve higher performance in terms of parallel patching of multiple devices, scalability, resiliency in terms of natural redundancy of patch files at the device locations and increased safety due to patch maintenance close to the end automation devices. In the following disclosure, the terms “switch” and “router” are used as examples of network elements. In general, a “network element “ is any component having a processor, a memory and a network interface, that is programmed to perform one or more network operations.

[23] Two different patch distribution architectures are discussed. One architecture is a switch/server construct, in which the server, such as a SINEMA Server

manufactured by Siemens, is a dedicated patch management server. The other architecture is a peer-to-peer solution based on network element configurations.

[24] Network elements (routers and/or switches), such as the Siemens SCALANCE X, may be used in the presently disclosed system as the main patch distribution platform. The complete patch distribution function is thus distributed throughout the already-deployed network elements. The miniaturization of transistors has facilitated higher computational power and storage capability within networking elements. It is therefore possible to perform additional computation on top of the existing routing functionalities within a router. The advantages of porting part of the patch distribution functionality to the network elements level are higher performance, higher scalability, increased resiliency, and greater safety.

[25] The presently described patch distribution scheme is an agent-based scheme where network elements such as switches are used as the intermediate drivers for the patch distribution to the automation devices. To support different types of scenarios, two different types of patch distribution architectures are proposed herein: a) a server-switch-device patch distribution architecture; and b) a highly distributed peer-to-peer (end-device to end-device) patch distribution architecture, in which there is no server involvement. Several issues are also discussed below, including rollover file transfer protocol, scheduling constraints related to proposed patch distribution, memory issues for the networking elements, and patch distribution functionalities.

[26] Current state-of-the-art control systems in a physical plant are patched in two different ways based on the connectivity of the plant to an external wide area network such as the Internet. Where there is no mechanism to download or manage new

patches from a vendor, direct human interaction is required to physically transport the patches to every plant. Typically, plant engineers/commissioning engineers/maintenance personnel traveled from one plant to another plant to patch the industrial control devices. The process is inefficient, requiring a significant amount of time which is costly in terms of maintenance. More importantly, that type of patching is problematic in terms of timeliness, considering the speed in which vulnerabilities are exploited and exploitation vectors are propagated in today's highly connected IT world.

[27] An example of a current-technology patch distribution system, in which the control system 100 of the physical plant is connected to the Internet 105, is illustrated in FIG. 1. The driver of the patch distribution system is a server 110 that is administering all the connected automation devices within the local network, such as PLCs, industrial PCs, Human Machine Interfaces (HMIs) and Motion Controllers (MCs). The server 110 may, for example, be a Siemens SINEMA Server, or Siemens STEP 7 engineering system connected to a plant network such as a one using PROFIBUS communication protocol that may also initiate the patch updating. When the server 110 receives a notification through the Internet 105 from a vendor regarding the availability of new patches, it downloads the patches from the vendor Website to the server-specific database at the server. After downloading the patches from the patch source, the assigned server 110 sorts and selects the patches, and distributes them based on system configuration to each of the connected devices. The patches are installed at each of the automation devices 120 sequentially from the patch management server. Although the plant network is connected to cyberspace, it may not be possible to patch all the connected devices online because many legacy devices do not support online firmware updates.

[28] In the following, an example of current technology is discussed in which an engineering system updates connected and unconnected devices from a Siemens STEP 7 engineering system. Previously, firmware updates had been performed exclusively offline, which required additional steps to configure the device to use default parameters. A newer version of STEP 7, version V5.5, allows for some new modules to be updated online and for the remainder of the legacy modules to be updated offline using a memory card, because those devices do not support online update. The two methods to update the devices from STEP 7 V5.5 are elaborated below.

[29] First discussed is the updating, through a wide area network such as the Internet, of a target module / CPU that is installed in rack 0 of an automation device. The screenshot 200 shown in FIG. 2 illustrates a dialog box presented by STEP 7 V5.5 for selecting the firmware file to be uploaded to a displayed target module. The properties of the selected target modules that STEP 7 can identify online are displayed in a target module box 210. Those properties may include the order number, current firmware version, module name, node address and slot. Several special cases apply in using the target module box, as follows:

[30] If an operator selects the DP slaves or IO devices then the option is presented of running firmware updates for any module. In that case, the "Change Slot" button (not shown) is displayed, allowing the operator to select the slot of the module containing firmware that must be updated. The target module box then displays the data of the module in the selected slot. The usual procedure is then followed to update the firmware.

[31] In some cases, modules support a firmware update in redundant mode (e.g. IM 153-2BA00-0XB0). In those cases, both target modules are shown as a right target module and a left target module in the target module box 210. That function is only possible with an active backplane bus in the H system. The firmware update is then automatically transferred to the redundant module.

[32] Another special case for using the target module box is firmware update for H CPUs (only in HW configuration). If two CPUs (for example, CPU 417-4H) are in redundant mode, the data listed above is displayed in the target module box 210 for both CPUs (CPU in rack...). After the "Run" button is clicked, a "Firmware update wizard" supports the user during the update on both H-CPUs. The update is “bumpless”; in other words, the execution of the program continues.

[33] A firmware file selection box 220 is provided for entering the path to the required firmware file. Alternatively, STEP 7 can enter the path automatically via the "Find" button. Up-to-date firmware can be obtained on the Internet at a vendor Web site. Once the path has been defined, the firmware version as well as a table showing the modules suitable for update with this firmware and the firmware version of these modules is shown in a lower portion of the firmware file selection box 220.

[34] A checkbox 230 is provided to activate the firmware after download. An operator enables this check box to reset the module automatically after the file is loaded. The module will now operate with the new firmware. If the check box 230 is not enabled, the 'old' firmware will remain active until the module is reset (e.g. after POWER OFF - POWER ON). The new firmware will not be active unless the module has been reset.

[35] If the corresponding CPU is in RUN mode, activating the firmware may lead to an access error or station failure and can set the CPU to STOP. After one has enabled the check box 230, the station performs a warm restart. When operating with F I/O, for example, that has the effect that the modules are passivity and need to be deactivated subsequently. During that time the safety program cannot be executed.

[36] If SFC99 or SFC109 (with F or H-CPUs) is enabled, the CPU must be changed to STOP with the mode selector (SFC109) or the operator will need to enter the password (SFC99). With some CPUs, the "Activate firmware after download" check box 230 is grayed out and cannot be selected. In that case, the operator must restart the CPU manually.

[37] A second STEP 7 V5.5 updating method will now be discussed, in which a new firmware version or new operating system version is transferred to a module or a CPU off-line by means of a memory card. The update requires the following two steps: (1) creating an "update memory card" by transferring the update files to a memory card with the programming device or PC with an external PROM burning device (or “prommer”), and (2) using the "update memory card," updating the operating system on the CPU. The process requires a memory card with sufficient storage capacity, and a programming device or PC setup to program memory cards.

[38] The transfer process of the updated files to a memory card includes creating a new directory, and transferring the desired update file to that directory and unzipping it there. The directory will then contain the UPD file.

[39] An S7 memory card is inserted into the programming device or prommer. The contents of the memory card is deleted (menu command: File > S7 Memory Card >

Delete in the SIMATIC Manager). The PLC is then selected using the Update Operating System menu command in the SIMATIC Manager.

[40] In the dialog box that is displayed, the directory with the UPD files is selected. The correct UPD file is then double-clicked. That action starts the programming process. When the process is ended, the message "The firmware update for the module was successfully transferred to the S7 memory card" is displayed.

[41] The operating system of the automation device is then updated using the programmed memory card, as follows. The power supply unit for the CPU is switched off. The prepared memory card with the update is then inserted into the CPU, and the power for the CPU is switched back on. The operating system is then transferred from the S7 memory card to the internal FLASH EPROM. During this time, all LEDs on the CPU are lit.

[42] After about two minutes, the update is finished. To indicate that the update is finished, the STOP LED on the CPU flashes slowly, indicating a system request for memory reset. The power is switched off at the power supply unit, and, where appropriate, the S7 memory card that is intended for the operation is inserted. The power is then switched back on, and the CPU executes an automatic memory reset. After that, the CPU is ready for operation.

[43] Although the above-described existing patch distribution system is functional, it suffers from several problems due primarily to the paradigm shift towards more distributed monitoring and control functionalities. One such problem is a scalability issue. As more and more devices are integrated within a single automation and control network, it is difficult to monitor the complete patch distribution

functionalities from the patch management servers. The integration of more servers to the network is not an economic solution.

[44] Another problem with the above-described current model of patch distribution is that it has a single point of failure. As most of the time only a single server is responsible for distributing the related patches to individual devices, it may be a single point of failure in critical situations. In such scenarios, the complete system may remain exposed to security vulnerabilities for an increased amount of time. The above-described current model may also have safety issues. Typically, in an automation and control domain, it is preferred to keep the automation device access as close as possible to the device. In a model where a single server or only a few servers are responsible for patch distribution, the system may suffer from additional safety issues including malicious access to the automation devices during patching.

[45] Advantages of the Presently Described System

[46] The presently described patch distribution model overcomes those problems, providing increased performance and scalability, by porting part of the patch distribution functionality to the network element level. An illustration of a router-based patch distribution system 300 according to the present disclosure is depicted in FIG. 3. Patch management servers (not shown) download from the Internet 310 patches that are required by devices in the system. Those patches are pushed to network elements such as the switches 315, 330 for storage and installation on automation devices such as device 320. Patch storage is typically at a network element along a path to the automation device, and as close as possible to the device.

[47] The following sequence illustrates the operation of the system shown in FIG. 3. An instance of an S7-300 device 325 that is already known to the network is attached to the system by connecting to the switch 315. That switch 315 does not have the patches for that particular device. The switch 315 therefore sends a request to the top-level switch 350 for available patches for the S7-300 device. The top-level switch 350 forwards the request for available patches to a switch 330 that already has the patches stored. The required patches are then forwarded from the switch 330 to the switch 315 to be installed on the newly connected device 325.

[48] The presently described patch management system has several advantages over the state of the art. For example, the presently described system has improved performance. Typically, in a server-based solution, patches are uploaded to each of the end devices in a sequential manner. That process consumes a significant amount of time. In a state-of-the-art patch distribution system such as that shown in FIG. 1, if there are 50 devices connected to the network and a particular patch must be installed on all the devices, then the patch must be copied from the server sequentially 50 times. Typically, the state-of-the-art patch distribution scheme performs the following steps from the user point of view: an operator initiates uploading the patches to the first device. When the first patch uploading is finished, the operator starts uploading patches for the next device. Those steps are repeated sequentially for each of the devices. Therefore, there may be a patch distribution latency of 50X for a scheme. The scheme also requires direct human interaction for patch distribution.

[49] In the presently described scheme, the patches are distributed over the network in a proactive and automatic way. The patches are stored at downstream

network elements that are either located in the automation network close to end network elements, or are themselves end network elements. "End network elements," as used herein, are network elements that are connected in an upstream direction to intermediate and higher level network elements, but have no additional network elements downstream. Automation devices are connected downstream of end network elements. "Downstream network elements," as that term is used herein, are network elements connected downstream of intermediate network elements, but there may be additional network elements downstream of the downstream network elements.

[50] If, in the presently described scheme, all the automation devices are connected to three end network elements, then there is a patch distribution latency of 3X. The scheme is furthermore automatic and requires minimal or no human interaction. The latency estimate assumes that the networking devices are capable of updating the firmware in a parallel fashion, using existing broadcasting or multicasting capabilities of the network elements. In general, the performance improvement depends on the topology of the plant network. If the parallel downloading capability of the network elements is not used, the performance gain may be roughly 17X for the given example. In the following discussion, it is assumed that the network elements are capable of parallel downloading of patches. It may therefore be claimed that the described patch distribution scheme increases the overall performance of the patch management significantly.

[51] The described patch distribution scheme also yields higher scalability and easier integration. The integration of new devices to the system will not create additional bottlenecks in the network. In the current patch distribution system described above, when a newly added device must be updated, the patches must be downloaded from the

server. If the patch is not available at the server, it must be fetched and downloaded from a vendor server. In the presently described patch distribution scheme, when a known device is added, it can be patched from already available patches in other network elements, as described above with reference to FIG. 3. In a large network, the probability that the required patch files may be found in a lower spatial distance is higher in the current-technology approach.

[52] The presently described approach offers increased resiliency. In the proposed scheme, patches may be stored only at the end network elements of the network, such as switches or routers. Patches required to update the firmware of the switch itself will be stored at the switch level. Only pointers to the original patches need be stored at intermediate and high-level network elements, although the patches themselves may be stored there. Further, the network elements function as a caching medium for both the version information of the available patches from the patch management server side and current firmware versions at the device side, as discussed below with reference to FIG. 9. The distributed storage of the patches creates a natural redundancy that increases the overall resiliency of the control system. If the patch management server is down at a particular time and a device must be patched immediately, the proposed network-element-based patch distribution scheme will provide the required solution, as patches are locally stored close to the end devices.

[53] Safety is increased by the presently described patch management system. It is considered a safe and secure maintenance and management practice in the automation and control domain to keep the system management procedures as close as possible to the end device. In the presently described patch management scheme, the

downloaded patches from the vendor servers are kept very close to the automation devices.

[54] Structural and Functional Operation of the Described System

[55] A typical patch management life cycle 400 is shown in Figure 4. After a patch is announced from the vendor, it is the responsibility of the patch distribution system of the control network to distribute the patches to the corresponding end-devices. As discussed earlier, in the state-of-the-art server-based model, it requires significant amount of time, due to the nature of sequential distribution, to distribute the patches among the end-devices. A goal of the presently described scheme is to reduce that distribution time (indicated by ellipse 410 of FIG. 4) by exploiting the computation and storage capability of the intermediate networking elements.

[56] Patch management architectures may be classified into two different classes based on the existence of the software agents running on the patch management/distribution platform to aid the functionalities: (1) an agent-based patch management architecture, and (2) an agent-less patch management architecture. There are several advantages and disadvantages for both the agent-based and the agent-less patch management architectures.

[57] In an agent-based architecture, there is a central server that can serve patch files, and an agent that is installed on the end devices to perform local tasks. Agent-based patching can use either a push (interrupt) type or a pull (polling) type of patching model. Patches can be pushed as soon as they are available for deployment. Alternatively, agents can check in to pull patches at regular intervals, or during a device-side event such

as a re-booting or re-joining a cyber-network. Agent-based architectures can scan a large network very quickly and typically minimize the use of network bandwidth while scanning devices and vendor servers. The disadvantages of an agent-based scheme include the requirement that software agents be installed, running and managing on all the participating devices. If an agent is not running due to failure or misconfiguration, the device may not be patched. Agents furthermore must have administrator or root privileges. That creates the possibility of remote attacks against agents that give attackers administrator privileges.

[58] On the other hand, agent-less patch management architectures require no additional software to be installed on the device/client. Generally standard remote administration tools are used for that purpose. Agent-less architectures, however, are limited to pushing patches to the end-device, using an "interrupt" approach wherein patches are pushed when they arrive. Without limitation of the disclosure, an agent-based patch management architecture is used in the exemplary system described herein.

[59] In the present disclosure, two different networking element based patch distribution architectures are proposed: (1) a server-switch-device architecture, and (2) a peer-to-peer architecture. An example of the proposed server-switch-device architecture, which uses networking elements as the patch distribution platform, is shown in FIG. 5. A server 510 manages the initial part of the patch management and downloading functionality from the vendor server (not shown). In one example, a SINEMA server from Siemens may be used, but the scheme is orthogonal to any server. The server 510 announces a patch whenever it is downloadable. An agent program is executed on a network element such as one of the switches 520, 530. The agent pulls the available

patches and sorts them according to the end-device needs. Finally, agent pushes the patches to the automation devices 540, 550 and causes them to install the patches whenever they are ready.

[60] At least one of the network elements 520, 530 keeps a cached copy of the patch. Other network elements upstream in the network architecture may maintain pointers to the cached copies. The network element closest to an automation device may manage the updating of the device firmware using the patch.

[61] A detailed example activity diagram 600 of the present patch distribution scheme is presented in FIG. 6. The patch distribution server initially views patch relationships at the system level (610) and determines whether a patch is required (612). If no patch is required, the flow returns to step 610. If a patch is required, then patch binaries are created (614) and patch relationships are validated at the system level (616). If the patch cannot be validated (618), then flow returns to step 610. If the patch is valid, then the server updates the patch model (620) and activates and announces the patch (622).

[62] A network element, such as a switch or a router, listens to the patch announcements (640) to determine whether a new patch has arrived (642). If so, then patch relationships are viewed at the automation device level (644), and determination is made whether the patch has been requested (646). If so, then the patch is downloaded (648) and the patch relationships are validated at the automation device level (650). If the patch is valid (652) then it is transmitted to the automation device.

[63] The automation device backs up its system (680) and installs the patch (682). The system is then checked for stability and consistency (684). If the system is

stable (686), then the device is restarted (692), if required. Device restart may be triggered by the network element. If the system is not stable, then the system is restored using the backup files (688), and the instability is logged and reported to an administrator (690) before the device is restarted (692).

[64] A diagram of the server-switch-device patch distribution architecture 700 is presented in FIG. 7. In the architecture, the end network element or switch 720 "pulls" patches and patch information from the server 710, and "pushes" patches to the automation device 730.

[65] The patch distribution scheme of the present disclosure may also utilize a peer-to-peer patch distribution architecture 800, shown in FIG. 8. As with the server-switch-device patch distribution architecture, in a peer-to-peer patch distribution architecture, patch distribution is deployed to the network elements 810. In peer-to-peer architecture, however, network elements are connected peer to peer, and there is no dedicated server. As in the other model, an agent program is executed at the network elements 810. The network elements share the downloaded patches. A mechanism similar to the push-pull mechanism of the previous model is also applicable to the peer-to-peer model. A major difference in the peer-to-peer model is that patches are pulled from vendor servers through the Internet and/or peers 820 rather than through a dedicated server.

[66] A detailed sequence diagram 900 for the presently disclosed patch management system is shown in FIG. 9. Major sequences performed by the system are shown in separate sections 920-950 of the sequence diagram 900.

[67] Current version extraction 920 gathers the current version information of firmware installed in the automation devices. The sequence is initiated by the patch management server 910, which requests information about the operating system, applications and configuration versions of the devices. That request is relayed by a downstream network element or switch 914 to the automation device 916. The device 916 replies by providing information about the firmware installed. The downstream switch 914 and/or the intermediate switch 912 store or cache the information related to the installed version, and forward it to the patch management server 910. Some devices do not provide information about the installed firmware version. In those cases, the end network elements gather that information from the patch management server, or extract the information from a log file of the previous version installed on the device 916.

[68] The "available patch info" sequence 925 provides the up-to-date available patch version to downstream network elements close to the automation device 916. Available patch information is provided by the server 910 to an intermediate switch 912 as a catalog of available relevant patches. The intermediate switch 912 then provides a refined subset of the available patches to the switch 914 close to the device 916. Intermediate switches in the network topology forward to the downstream switch 914 only that information related to available patches for devices connected to the particular downstream switch.

[69] The "patch uploading" sequence 930 is initiated by a downstream switch 914 close to the device 916, which requests a download of the required patches based on the information on available required patches. An intermediate switch 912 forwards the request to the patch management server 910. The patch management server 910

downloads the patches from a vendor server, validates the downloaded patches, and forwards them through the intermediate switch 912 to the switch 914. Only those patches for firmware in devices close to a particular switch are forwarded to that switch. The downstream switch stores the patch and confirms successful downloading of the patch. No patch file is stored at the intermediate switches. Instead, the intermediate switches keep pointers to the locations of the patch files downstream in the topology.

[70] Patch updating 935 to the devices 916 is initiated by a downstream switch 914, which transmits the patch to the device 916. The device then provides the switch 914 with feedback regarding successful transfer of the patch to the device. The device 916 then transfers to the switch 914 image files for those files that will be updated with the patch. If the device updates are not successful, then the device 916 uses the image files stored on the downstream switch 914 to roll back to the previous state. If the update is successful, then the switch 914 is notified, and a re-start of the device 916 from the switch 914 is performed, if necessary.

[71] As to a rollback 940 in case of failure of the device update, the switch 914 uploads the backup image files to the device 916, which restores its firmware using the image files. The device then confirms the successful rollback.

[72] In the sequence "updating firmware of the switches" 945, new patches for switches are loaded from the server 910 to the switches 912, 914, which update their firmware and confirm success with the server 910. The server retains version information about firmware installed on the servers during current feature extraction 920. The "updating firmware" sequence may additionally require propagating topology specific information.

[73] The system additionally handles the attachment 950 of a new device 916 in the automation network at a point close to a downstream switch 914. In that case, the new device requests a new patch from the switch 914. If the switch does not have the required patch, it forwards the request upstream in the topology. If the patch is not present in the network, then the patch management server downloads the patch from a vendor server and transmits the patch through the network to the switch 914 in a manner similar to that described above with reference to patch uploading 930. It is possible, however, that the patch is stored at another downstream switch in the network. If that is so, then a pointer to that patch will be stored on one of the intermediate switches or on the patch management server. The pointer is used to locate the patch at the other server, and the patch is then transmitted from other downstream switch to the downstream switch 914.

[74] In the following, several additional issues related to the described agent-based patch distribution scheme on top of existing network elements are discussed. Memory constraint in a network element is a potential bottleneck for the proposed patch distribution scheme. Due to the higher transistor integration technology, however, it is possible today to integrate higher memory capacity on a router at a very low cost. If on-router memory is limited, external flash memory may be used.

[75] The proposed patch distribution scheme reduces the cost to the company in terms of automatic patching, because no physical transportation of new patches from one plant to another plant is required. For automatic patch distribution, heterogeneous file transfer protocols are necessary. One example of such a protocol is Trivial File Transfer Protocol (TFTP).

[76] If a new patch is not successfully installed at the automation device, the old state of the firmware must be resumed automatically through the use of rollover protocols. To drive the patch distribution, those downstream network elements driving the patch updates at the end of the network topology may require permission to re-start the device, at least during the rollover time. A different memory location to install the updated firmware, such as an external flash memory or internal storage of the device, may also be used to facilitate the rollover functionalities. Moreover, a new file transfer interface such as $I_{patchme}$ may be created at the device side. The details of file transfer protocols are beyond the scope of this discussion.

[77] Typically, patches are stored at network elements close to the devices to patch the connected devices. Intermediate network devices may hold only pointers to the existence of the available patches downstream in the topology. In practice, network elements may also need to be patched. To solve that particular problem, there are two options: (1) storing the patch files in the network elements that must be patched; and (2) driving the patches from the next-highest level network elements.

[78] Both options have some merits and some disadvantages. The first option, in which the patch files for network elements are stored in the network elements themselves, is actually a peer-to-peer patch distribution architecture. For the peer-to-peer patch distribution architecture, there is no server involvement; therefore, keeping the patch files at each network element will provide higher performance in terms of locality. In the present example, the patch files required to update the software/firmware of a network element are kept within the element itself. As to the server-switch-device patch distribution architecture, it must be assumed that the network element will be a hotspot in

terms communication, patch storage, and distribution; therefore, due to safety and performance reasons it may be advantageous to keep the required patch files for the network elements within themselves.

[79] System

[80] The elements of the methodology described above may be implemented in a computer system comprising a single unit or a plurality of units linked by a wired, optical or wireless network. An exemplary automation network system 1000 is shown in FIG. 10.

[81] A patch distribution server 1010, an intermediate network element 1020, an end network element 1030 and an automation device 1040 are linked in a topology wherein the automation device 1040 is in a downstream location and the intermediate network element 1020 is in an upstream location. The patch distribution server 1010 is shown upstream of the intermediate network element1020, but may be placed elsewhere in the network. Actual implementations of the system 100 typically include multiple automation devices 1040 connected to a single end network element 1030, multiple end network elements 1030 connected to a single intermediate network element 1020, etc.

[82] The patch distribution sever 1010 may be a commercial automation server, mainframe computer, a desktop or laptop computer or any other device capable of processing data. The intermediate network element 1020 and the end network element 1030 may be routers, switches or any other network elements capable of communicating, processing and storing data. The automation device 1040 may be a programmable logic controller, an industrial PC, a CNC controller, a building automation controller, or any

building, process or machine controller that requires firmware updates. The patch distribution server 1010 receives data from any number of data sources that may be connected, including a wide area data network (WAN) 1050 such as the Internet.

[83] Each of the elements 1010, 1020, 1030, 1040 includes a central processing unit (CPU) 1012 and a memory 1014. One or more of the elements may be connected to an input and/or output device such as a personal computer 1048 capable of transmitting and receiving information to and from the element. The input may be a mouse, network interface, touch screen, etc., and the output may be a liquid crystal display (LCD), cathode ray tube (CRT) display, printer, etc. Alternatively, commands containing input/output data may be passed via the automation network 1000 and the network 1050. The elements 1010, 1020, 1030, 1040 can be configured to operate and display information by using, e.g., the input and output devices to execute certain tasks.

[84] The CPUs 1012, when configured using software according to the present disclosure, include modules that are configured for performing one or more methods for patch distribution as discussed herein.

[85] The memory 1014 may include a random access memory (RAM) and a read-only memory (ROM). The memory may also include removable media such as a disk drive, tape drive, memory card, etc., or a combination thereof. The RAM functions as a data memory that stores data used during execution of programs in the CPU 1012; the RAM is also used as a work area. The ROM functions as a program memory for storing a program executed in the CPU 1012. The program may reside on the ROM or on any other tangible or non-volatile computer-usable medium as computer readable instructions stored thereon for execution by the CPU or another processor to perform the

methods of the invention. The ROM may also contain data for use by the program or other programs.

[86] The above-described method may be implemented by program modules that are executed by a computer, as described above. Generally, program modules include routines, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types. The term "program" as used herein may connote a single program module or multiple program modules acting in concert. The disclosure may be implemented on a variety of types of computers, including programmable logic controllers, personal computers (PCs), hand-held devices, multi-processor systems, microprocessor-based programmable consumer electronics, network PCs, mini-computers, mainframe computers and the like. The disclosure may also be employed in distributed computing environments, where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, modules may be located in both local and remote memory storage devices.

[87] An exemplary processing module for implementing the methodology above may be hardwired or stored in a separate memory that is read into a main memory of a processor or a plurality of processors from a computer readable medium such as a ROM or other type of hard magnetic drive, optical storage, tape or flash memory. In the case of a program stored in a memory media, execution of sequences of instructions in the module causes the processor to perform the process steps described herein. The embodiments of the present disclosure are not limited to any specific combination of

hardware and software and the computer program code required to implement the foregoing can be developed by a person of ordinary skill in the art.

[88] The term "computer-readable medium" as employed herein refers to any tangible machine-encoded medium that provides or participates in providing instructions to one or more processors. For example, a computer-readable medium may be one or more optical or magnetic memory disks, flash drives and cards, a read-only memory or a random access memory such as a DRAM, which typically constitutes the main memory. Such media excludes propagated signals, which are not tangible. Cached information is considered to be stored on a computer-readable medium. Common expedients of computer-readable media are well-known in the art and need not be described in detail here.

[89] The foregoing detailed description is to be understood as being in every respect illustrative and exemplary, but not restrictive, and the scope of the disclosure herein is not to be determined from the description, but rather from the claims as interpreted according to the full breadth permitted by the patent laws. It is to be understood that various modifications will be implemented by those skilled in the art, without departing from the scope and spirit of the disclosure.

Claims

What is claimed is:

1. A method for distributing software patches in an automation network, the automation network including an intermediate network element, first and second downstream network elements downstream of the intermediate network element, and a first automation device downstream of the first downstream network element, the method comprising:

determining, by a processor of the first downstream network element, that a software patch is available for firmware installed on the first automation device;

determining, by the processor of the first downstream network element, that the software patch is not stored on the first downstream network element;

determining, by the processor of the first downstream network element, based on information stored at the intermediate network element, that the software patch is stored on the second downstream network element;

copying the software patch from the second downstream network element to the first downstream network element; and

transmitting the software patch from the first downstream network element to the first automation device for installation.

2. A method as in claim 1, wherein determining that a software patch is available for the firmware installed on the first automation device further comprises:

at the first downstream network element, receiving from the first automation device a message including version information of the firmware installed on the first automation device;

at the first downstream network element, receiving a message from a patch management server including available patch information; and

comparing the version information of the firmware with the available patch information.

3. A method as in claim 2, wherein the message from the first automation device comprises a notification that the first automation device is a newly-installed device.

4. A method as in claim 1, wherein the automation network further includes a second automation device downstream of the first downstream element, the method further comprising:

determining, by the first downstream network element, that a second software patch is available for second firmware installed on the second automation device;

determining, by the first downstream network element, that the second software patch is not stored on either the first or the second downstream network elements;

requesting, by the first downstream network element, the second software patch from a patch management server;

receiving the second software patch from the patch management server;

storing the second software patch on the first downstream network element; and

transmitting the second software patch from the first downstream network element to the second automation device for installation.

5. A method as in claim 1, wherein the automation network further includes a third downstream network element downstream of the intermediate network element, the method further comprising:

at the first downstream network element, receiving a request from the third downstream network element to transmit a copy of the software patch to the third downstream network element; and

transmitting the copy of the software patch from the first downstream network element to the third downstream network element.

6. A method as in claim 1, further comprising:

at the first downstream network element, receiving from the first automation device a backup image copy of the firmware installed on the first automation device;

after transmitting the software patch from the first downstream network element to the first automation device for installation, receiving from the first automation device a notification of unsuccessful installation;

transmitting the backup image copy of the firmware to the first automation device for rollback; and

receiving from the first automation device a notification of successful rollback.

7. A method as in claim 1, wherein the information stored at the intermediate network element is a pointer to the software patch stored on the second downstream network element.

8. A method as in claim 1, further comprising:

receiving, by the first downstream network element from the intermediate network element, a software patch for firmware installed on the first downstream network element;

installing the software patch on the first downstream network element; and

transmitting to the intermediate network element a notification of a successful installation of the software patch.

9. A method as in claim 1, wherein copying the software patch from the second downstream network element to the first downstream network element comprises receiving a copy of the software patch from the intermediate network element, the software patch not being stored by the intermediate network element.

10. A method as in claim 1, further comprising:

validating, by the first downstream network element, the software patch for installation in the first automation device.

11. A downstream network element in an automation network, the downstream network element and a second downstream network element being downstream of an intermediate network element and further being upstream of a first automation device in the automation network, the downstream network element

comprising a processor and a non-transitory computer-usable medium having computer readable instructions stored thereon for execution by the processor to perform a method of distributing software patches in the automation network, the method comprising:

determining that a software patch is available for firmware installed on the first automation device;

determining that the software patch is not stored on the downstream network element;

determining, based on information stored at the intermediate network element, that the software patch is stored on the second downstream network element;

copying the software patch from the second downstream network element to the downstream network element; and

transmitting the software patch from the downstream network element to the first automation device for installation.

12. A downstream network element as in claim 11, wherein determining that a software patch is available for the firmware installed on the first automation device further comprises:

receiving from the first automation device a message including version information of the firmware installed on the first automation device;

receiving a message from a patch management server including available patch information; and

comparing the version information of the firmware with the available patch information.

13. A downstream network element as in claim 12, wherein the message from the first automation device comprises a notification that the first automation device is a newly-installed device.

14. A downstream network element as in claim 11, wherein the downstream network element is upstream of a second automation device in the automation network, the method further comprising:

determining that a second software patch is available for second firmware installed on the second automation device;

determining that the second software patch is not stored on either the first or the second downstream network elements;

requesting the second software patch from a patch management server;

receiving the second software patch from the patch management server;

storing the second software patch on the downstream network element; and

transmitting the second software patch from the downstream network element to the second automation device for installation.

15. A downstream network element as in claim 11, wherein the automation network further includes a third downstream network element downstream of the intermediate network element, the method further comprising:

receiving a request from the third downstream network element to transmit a copy of the software patch to the third downstream network element; and

transmitting the copy of the software patch to the third downstream network element.

16. A downstream network element as in claim 11, wherein the method further comprises:

receiving from the first automation device a backup image copy of the firmware installed on the first automation device;

after transmitting the software patch to the first automation device for installation, receiving from the first automation device a notification of unsuccessful installation;

transmitting the backup image copy of the firmware to the first automation device for rollback; and

receiving from the first automation device a notification of successful rollback.

17. A downstream network element as in claim 11, wherein the information stored at the intermediate network element is a pointer to the software patch stored on the second downstream network element.

18. A downstream network element as in claim 11, wherein the method further comprises:

receiving, from the intermediate network element, a software patch for firmware installed on the downstream network element;

installing the software patch on the downstream network element; and

transmitting to the intermediate network element a notification of a successful installation of the software patch.

19. A downstream network element as in claim 11, wherein copying the software patch from the second downstream network element to the downstream network

element comprises receiving a copy of the software patch from the intermediate network element, the software patch not being stored by the intermediate network element.

20. A downstream network element as in claim 11, wherein the method further comprises:

validating the software patch for installation in the first automation device.

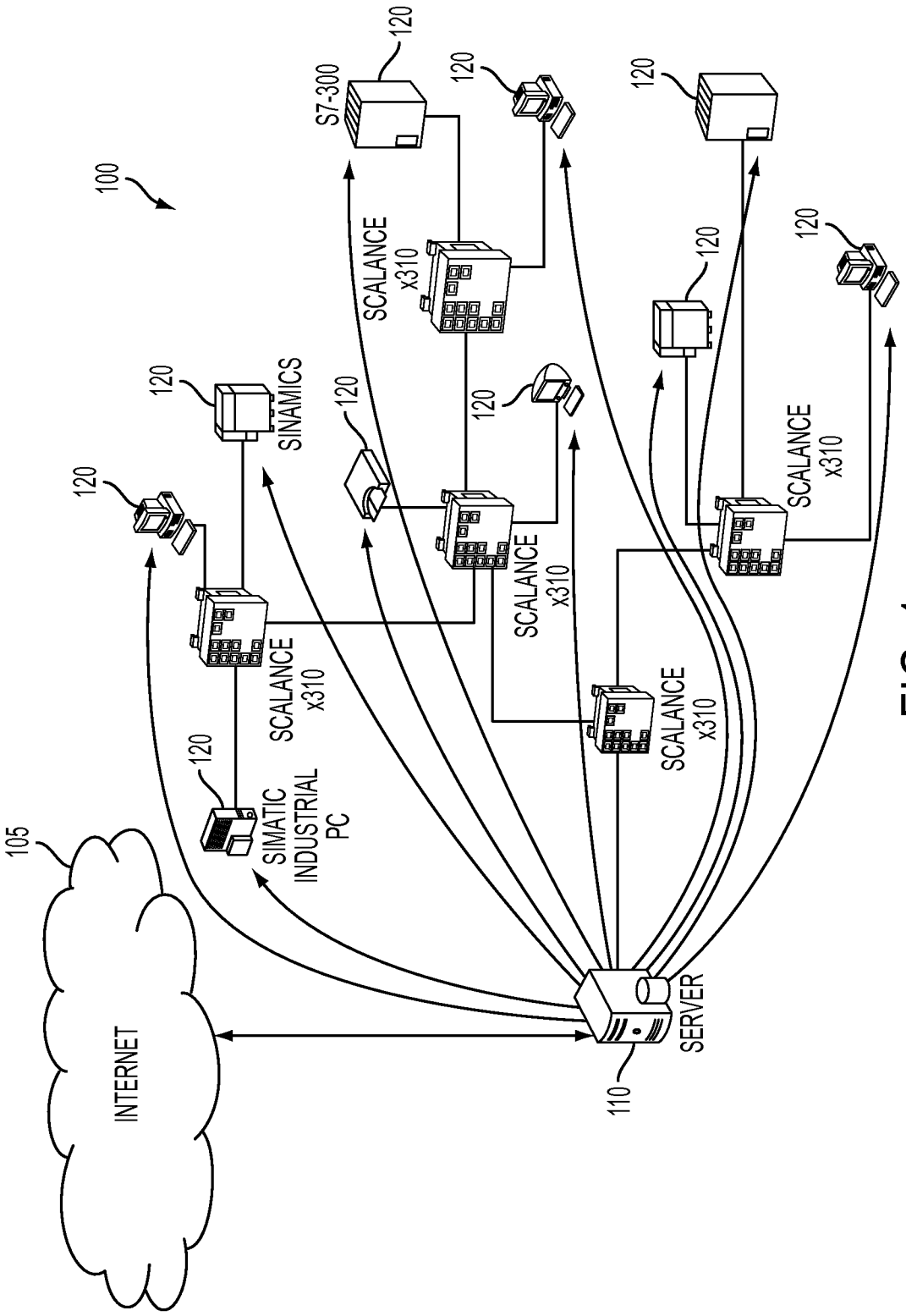

FIG. 1
PRIOR ART

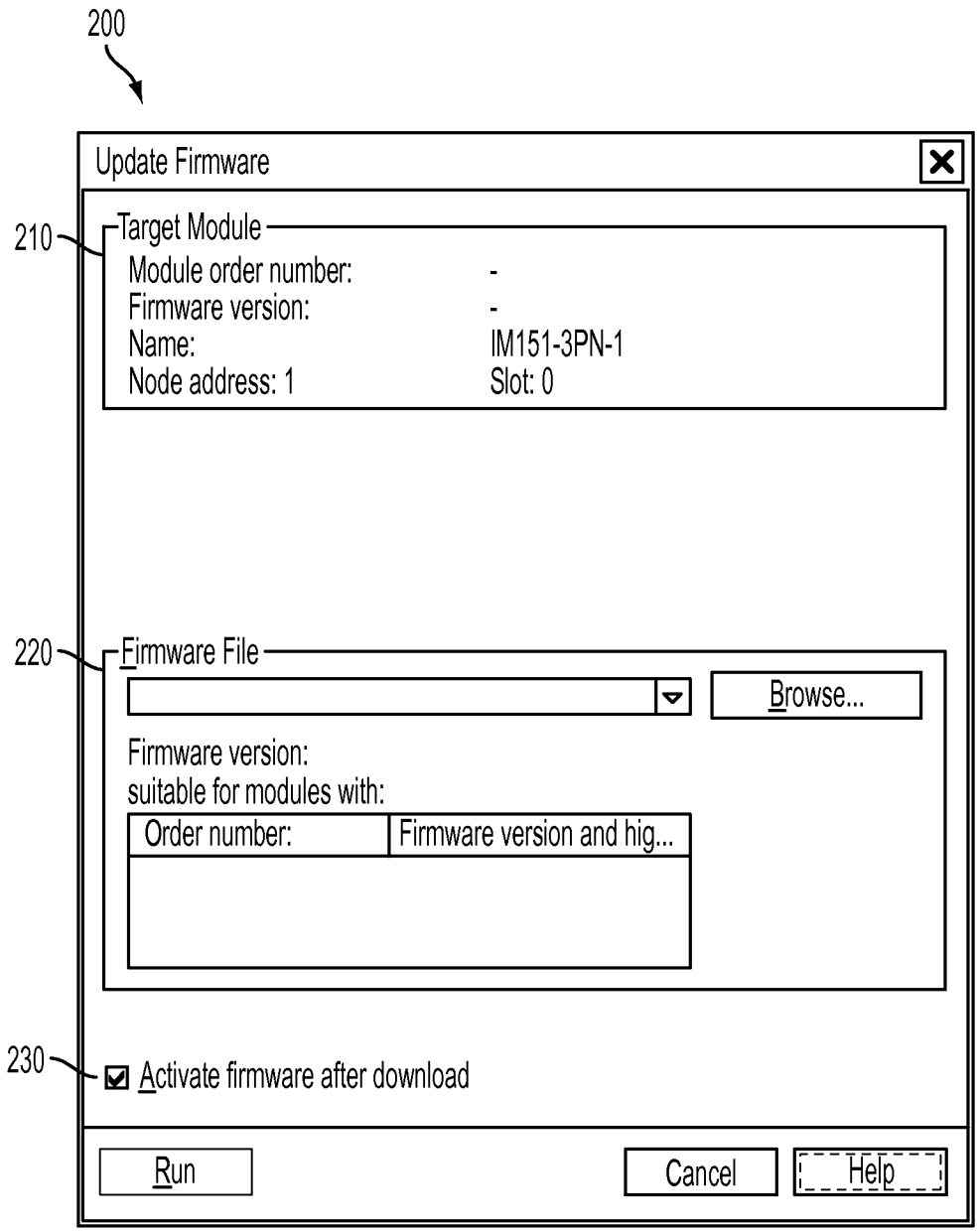

FIG. 2
PRIOR ART

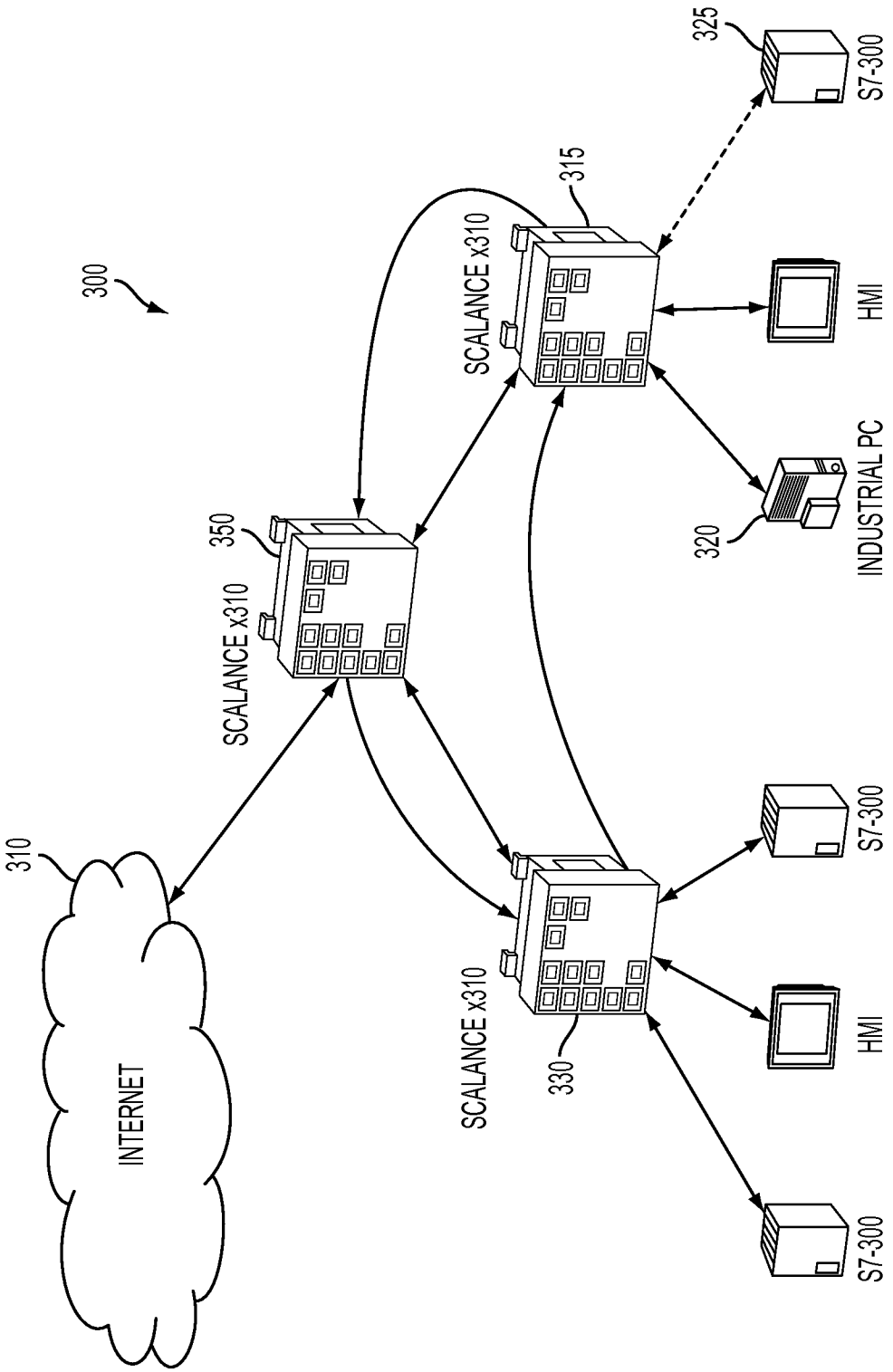
300
310
INTERNET
SCALANCE x310
350
SCALANCE x310
315
SCALANCE x310
330
325
S7-300
HMI
INDUSTRIAL PC
320
S7-300
HMI
S7-300

FIG. 3

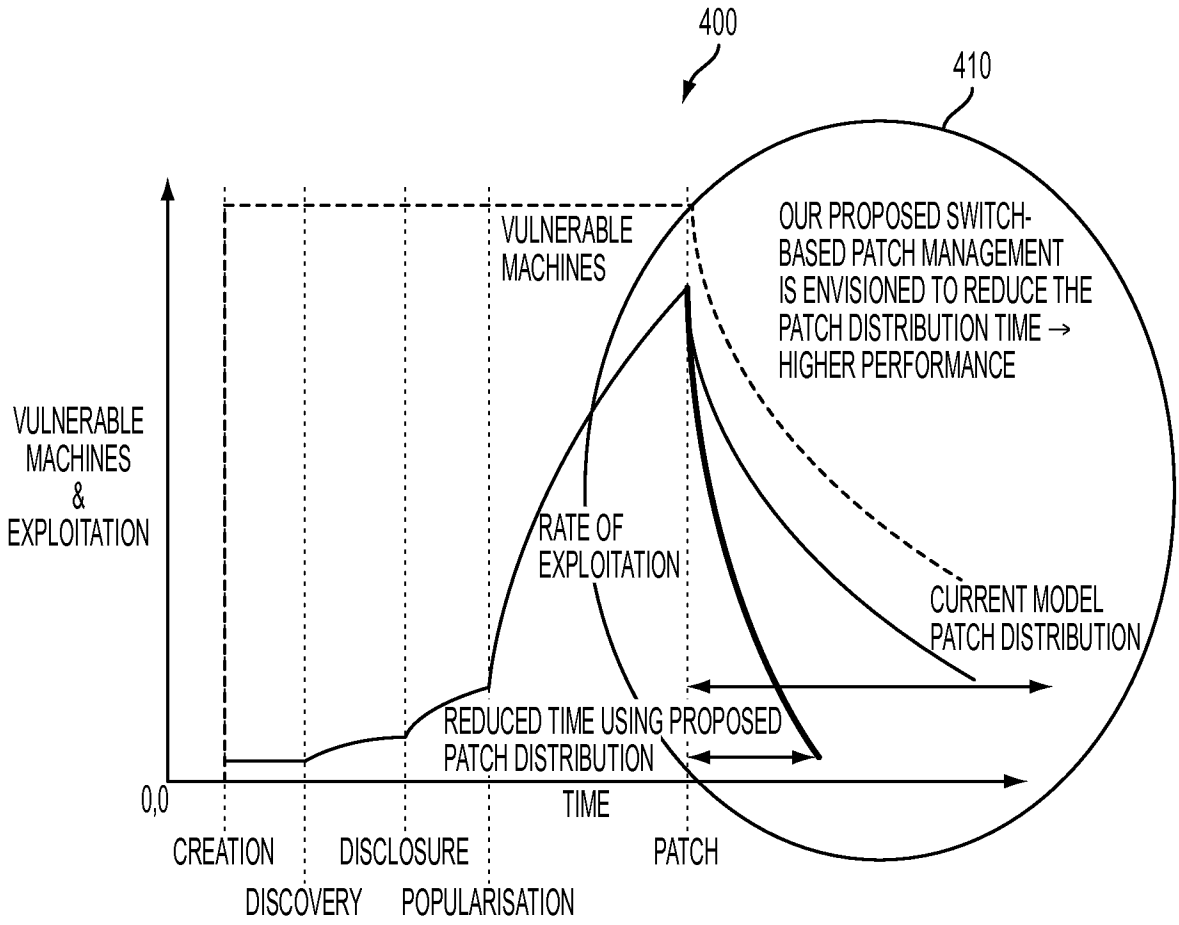
400
410
VULNERABLE MACHINES
OUR PROPOSED SWITCH-BASED PATCH MANAGEMENT IS ENVISIONED TO REDUCE THE PATCH DISTRIBUTION TIME → HIGHER PERFORMANCE
VULNERABLE MACHINES & EXPLOITATION
RATE OF EXPLOITATION
CURRENT MODEL PATCH DISTRIBUTION
REDUCED TIME USING PROPOSED PATCH DISTRIBUTION
0,0
TIME
CREATION
DISCOVERY
DISCLOSURE
POPULARISATION
PATCH

FIG. 4

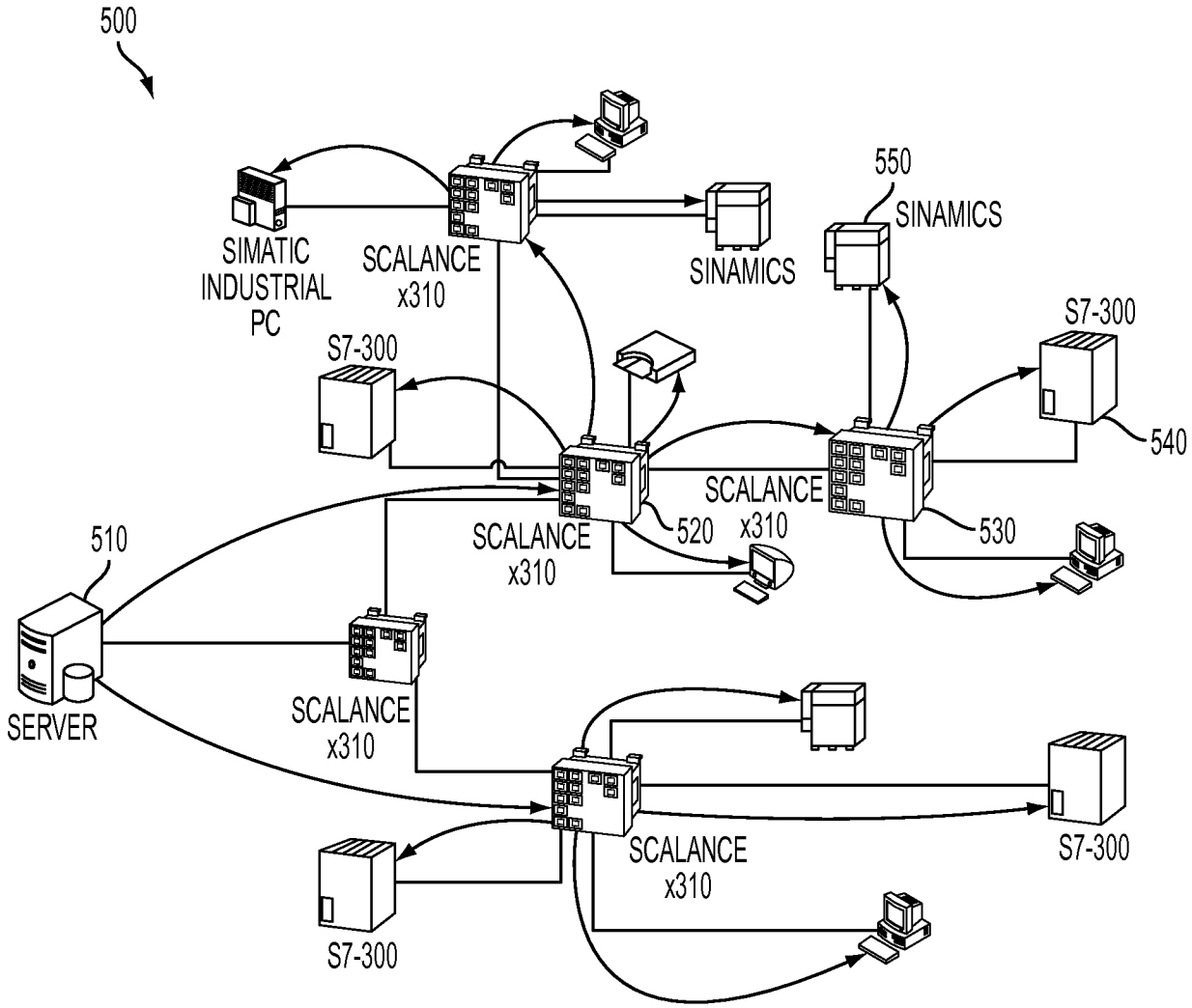
500
SIMATIC INDUSTRIAL PC
SCALANCE x310
SINAMICS
550
SINAMICS
S7-300
S7-300
540
SCALANCE x310
520
SCALANCE x310
530
510
SERVER
SCALANCE x310
SCALANCE x310
S7-300
S7-300

FIG. 5

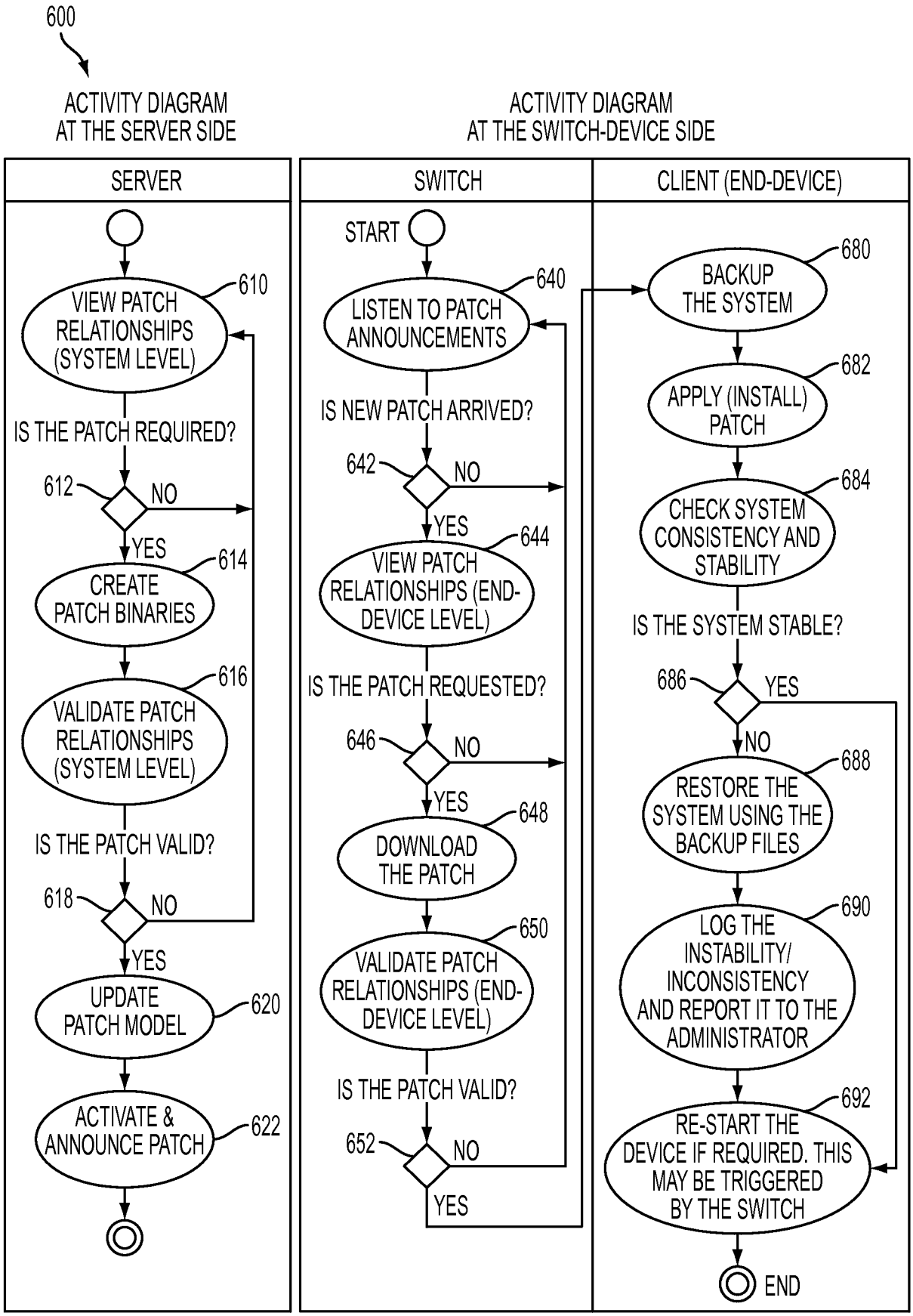
600
ACTIVITY DIAGRAM AT THE SERVER SIDE
ACTIVITY DIAGRAM AT THE SWITCH-DEVICE SIDE
SERVER
SWITCH
CLIENT (END-DEVICE)
START
VIEW PATCH RELATIONSHIPS (SYSTEM LEVEL)
610
IS THE PATCH REQUIRED?
612
NO
YES
CREATE PATCH BINARIES
614
VALIDATE PATCH RELATIONSHIPS (SYSTEM LEVEL)
616
IS THE PATCH VALID?
618
NO
YES
UPDATE PATCH MODEL
620
ACTIVATE & ANNOUNCE PATCH
622
LISTEN TO PATCH ANNOUNCEMENTS
640
IS NEW PATCH ARRIVED?
642
NO
YES
VIEW PATCH RELATIONSHIPS (END-DEVICE LEVEL)
644
IS THE PATCH REQUESTED?
646
NO
YES
DOWNLOAD THE PATCH
648
VALIDATE PATCH RELATIONSHIPS (END-DEVICE LEVEL)
650
IS THE PATCH VALID?
652
NO
YES
BACKUP THE SYSTEM
680
APPLY (INSTALL) PATCH
682
CHECK SYSTEM CONSISTENCY AND STABILITY
684
IS THE SYSTEM STABLE?
686
YES
NO
RESTORE THE SYSTEM USING THE BACKUP FILES
688
LOG THE INSTABILITY/ INCONSISTENCY AND REPORT IT TO THE ADMINISTRATOR
690
RE-START THE DEVICE IF REQUIRED. THIS MAY BE TRIGGERED BY THE SWITCH
692
END

FIG. 6

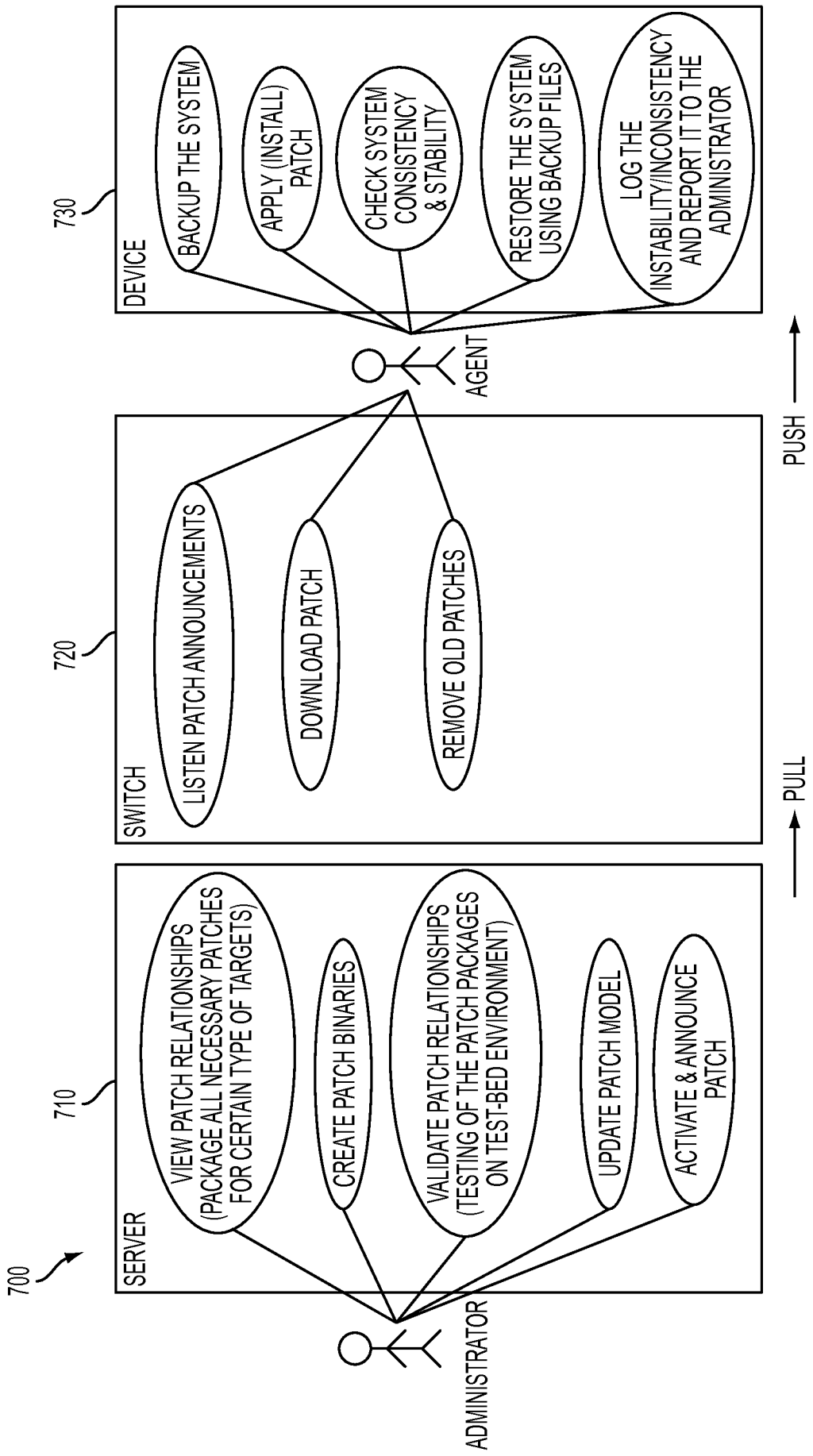
700
SERVER
710
VIEW PATCH RELATIONSHIPS (PACKAGE ALL NECESSARY PATCHES FOR CERTAIN TYPE OF TARGETS)
CREATE PATCH BINARIES
VALIDATE PATCH RELATIONSHIPS (TESTING OF THE PATCH PACKAGES ON TEST-BED ENVIRONMENT)
UPDATE PATCH MODEL
ACTIVATE & ANNOUNCE PATCH
ADMINISTRATOR
PULL
SWITCH
720
LISTEN PATCH ANNOUNCEMENTS
DOWNLOAD PATCH
REMOVE OLD PATCHES
PUSH
DEVICE
730
BACKUP THE SYSTEM
APPLY (INSTALL) PATCH
CHECK SYSTEM CONSISTENCY & STABILITY
RESTORE THE SYSTEM USING BACKUP FILES
LOG THE INSTABILITY/INCONSISTENCY AND REPORT IT TO THE ADMINISTRATOR
AGENT

FIG. 7

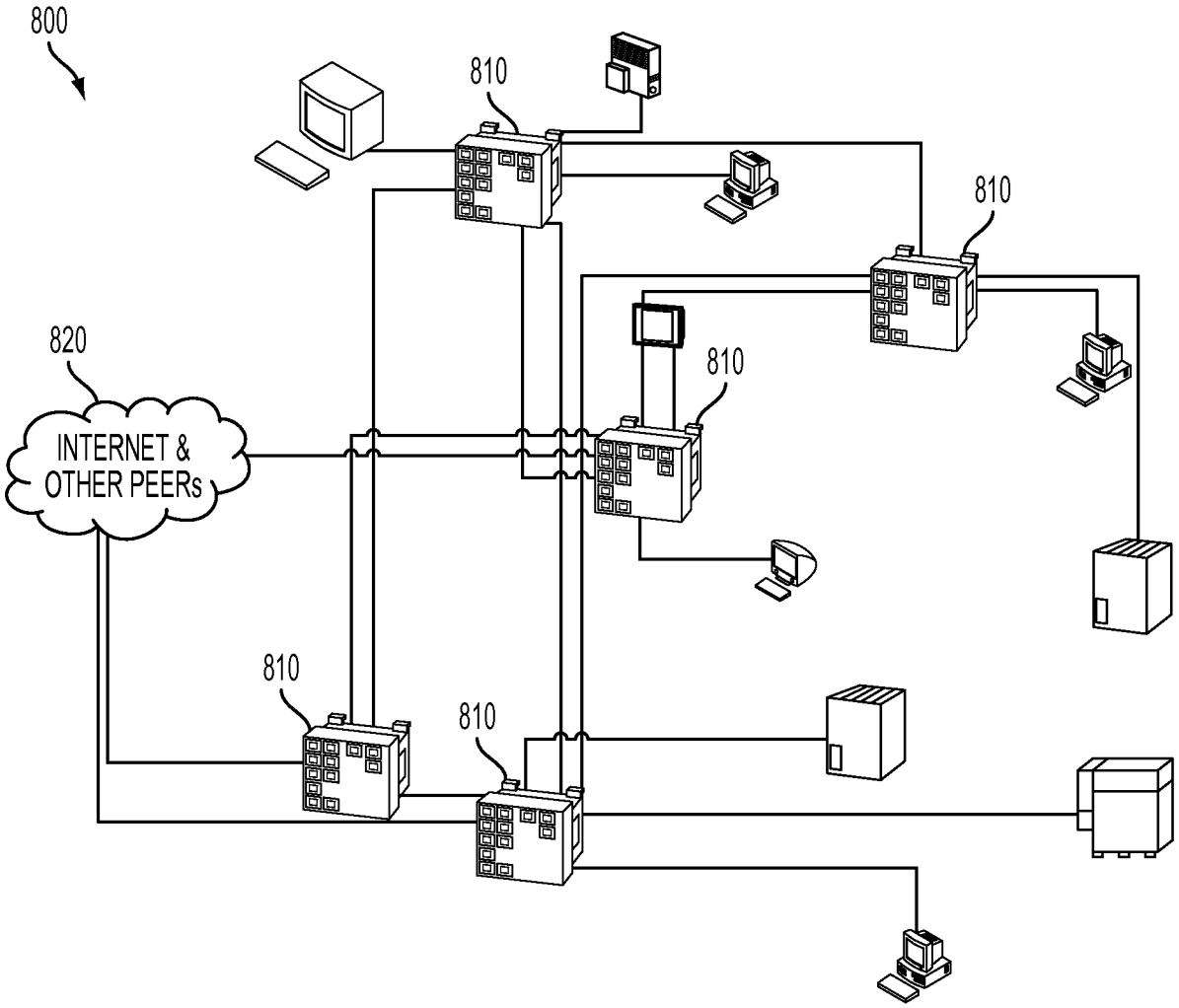

FIG. 8

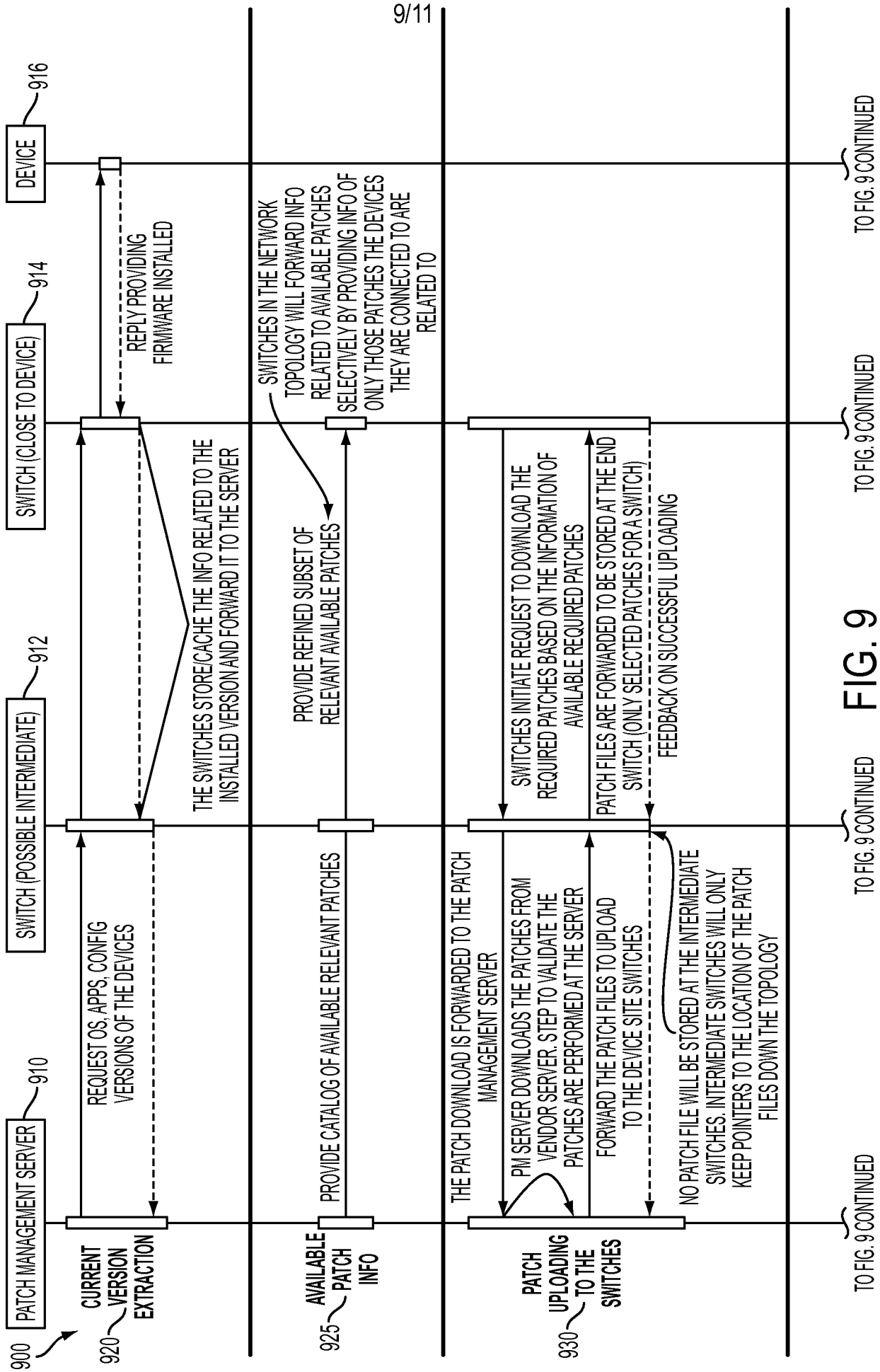
900
PATCH MANAGEMENT SERVER
910
SWITCH (POSSIBLE INTERMEDIATE)
912
SWITCH (CLOSE TO DEVICE)
914
DEVICE
916
CURRENT VERSION EXTRACTION
920
REQUEST OS, APPS, CONFIG VERSIONS OF THE DEVICES
REPLY PROVIDING FIRMWARE INSTALLED
THE SWITCHES STORE/CACHE THE INFO RELATED TO THE INSTALLED VERSION AND FORWARD IT TO THE SERVER
AVAILABLE PATCH INFO
925
PROVIDE CATALOG OF AVAILABLE RELEVANT PATCHES
SWITCHES IN THE NETWORK TOPOLOGY WILL FORWARD INFO RELATED TO AVAILABLE PATCHES SELECTIVELY BY PROVIDING INFO OF ONLY THOSE PATCHES THE DEVICES THEY ARE CONNECTED TO ARE RELATED TO
PROVIDE REFINED SUBSET OF RELEVANT AVAILABLE PATCHES
PATCH UPLOADING TO THE SWITCHES
930
THE PATCH DOWNLOAD IS FORWARDED TO THE PATCH MANAGEMENT SERVER
PM SERVER DOWNLOADS THE PATCHES FROM VENDOR SERVER. STEP TO VALIDATE THE PATCHES ARE PERFORMED AT THE SERVER
FORWARD THE PATCH FILES TO UPLOAD TO THE DEVICE SITE SWITCHES
SWITCHES INITIATE REQUEST TO DOWNLOAD THE REQUIRED PATCHES BASED ON THE INFORMATION OF AVAILABLE REQUIRED PATCHES
PATCH FILES ARE FORWARDED TO BE STORED AT THE END SWITCH (ONLY SELECTED PATCHES FOR A SWITCH)
FEEDBACK ON SUCCESSFUL UPLOADING
NO PATCH FILE WILL BE STORED AT THE INTERMEDIATE SWITCHES. INTERMEDIATE SWITCHES WILL ONLY KEEP POINTERS TO THE LOCATION OF THE PATCH FILES DOWN THE TOPOLOGY
TO FIG. 9 CONTINUED
TO FIG. 9 CONTINUED
TO FIG. 9 CONTINUED
TO FIG. 9 CONTINUED
TO FIG. 9 CONTINUED

FIG. 9

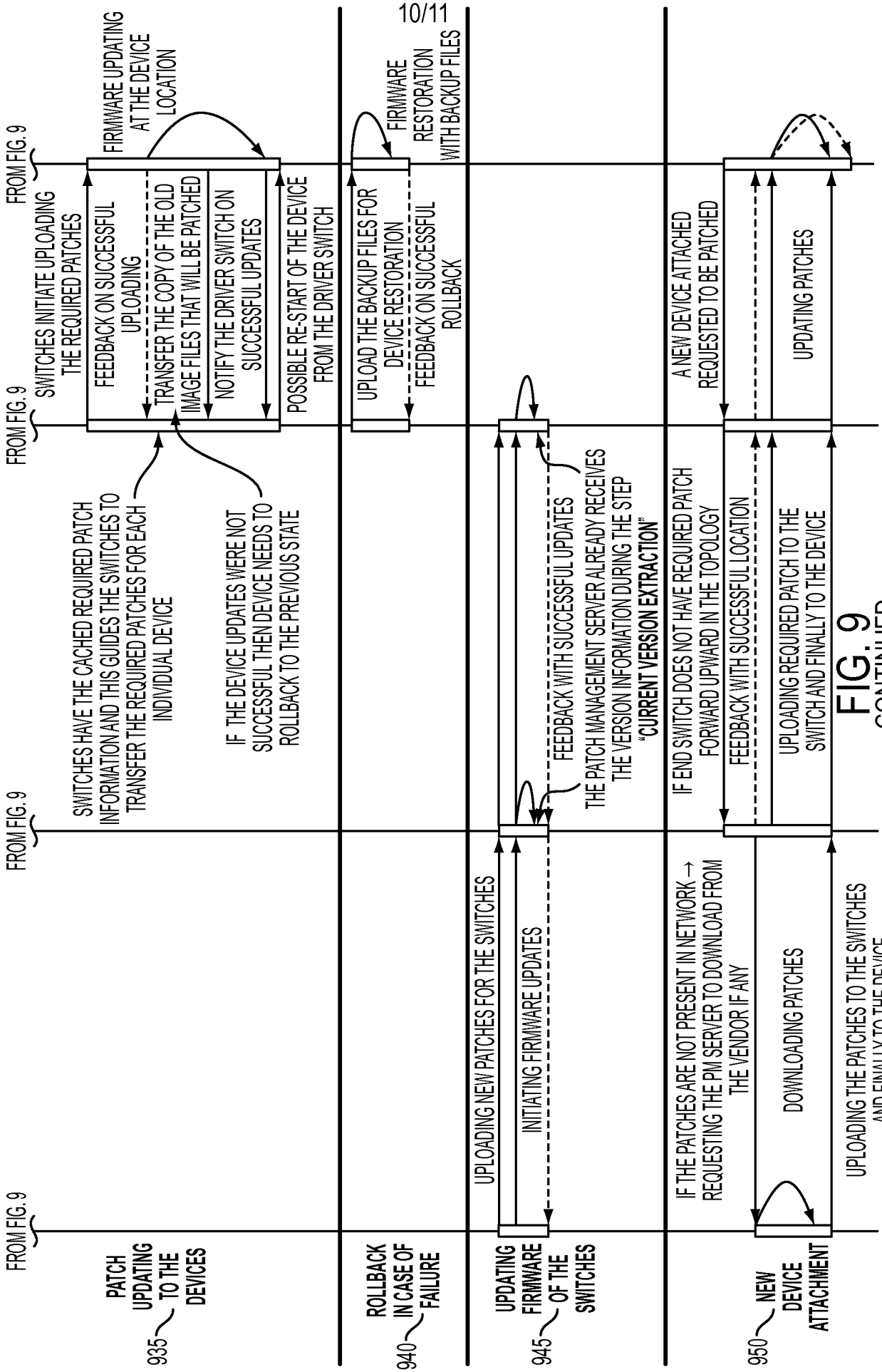

FIG. 9
CONTINUED

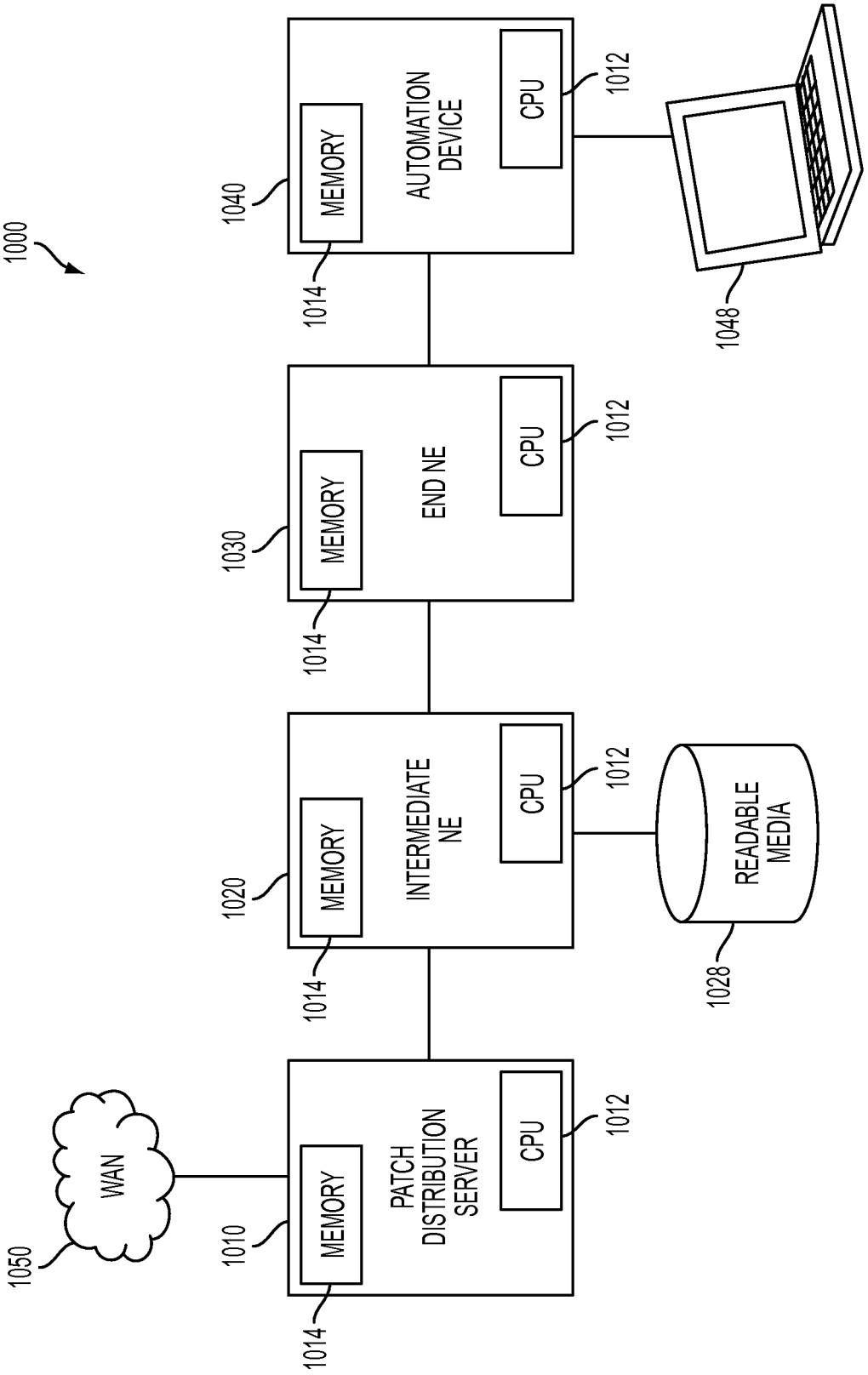
1000
1040
MEMORY
AUTOMATION DEVICE
CPU
1012
1014
1048
1030
MEMORY
END NE
CPU
1012
1014
1020
MEMORY
INTERMEDIATE NE
CPU
1012
READABLE MEDIA
1028
1014
1010
MEMORY
PATCH DISTRIBUTION SERVER
CPU
1012
1050
WAN

FIG. 10

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2012/043084

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/445
ADD.

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2008/028046 A1 (USHIKI TATSUHIKO [JP]) 31 January 2008 (2008-01-31) abstract paragraph [0060] - paragraph [0070]; figure 4 -----	1-20

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
30 August 2012	06/09/2012

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Müller, Tobias

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2012/043084

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2008028046 A1	31-01-2008	JP 2008033445 A	14-02-2008
		US 2008028046 A1	31-01-2008