



- (51) **International Patent Classification:**
G06F 21/44 (2013.01) *G06F 11/30* (2006.01)
G06F 9/44 (2006.01)
- (21) **International Application Number:**
PCT/US2016/026003
- (22) **International Filing Date:**
5 April 2016 (05.04.2016)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
14/705,662 6 May 2015 (06.05.2015) US
- (71) **Applicant:** MCAFEE, INC. [US/US]; 2821 Mission College Blvd., Santa Clara, CA 95054 (US).
- (72) **Inventors:** GIRI, Babu, Nath; #57, 4th Main, Domlur 2nd Stage, Bangalore, Karnataka 560 071 (IN). RAMAGOPAL, Prashanth, P.; #414, 1st Floor, 8th E. Main, 1st Block, Hrbr Layout, Kalyan Nagar, Bangalore, Karnataka 560 043 (IN). THOMAS, Vinoo; 7/3 Abu Castle No. 925 & 926, P.H. Road, Pursawalkam, Chennai, Tamil Nadu 600 084 (IN).
- (74) **Agent:** DACOSTA, Octavio, T.; Blank Rome, LLP, 717 Texas Avenue, Suite 1400, Houston, TX 77002 (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, BN, 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, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, 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, ST, 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, KM, ML, MR, NE, SN, TD, TG).

Published:

- *with declaration under Article 17(2)(a); without abstract; title not checked by the International Searching Authority*



ALERTING THE PRESENCE OF BUNDLED SOFTWARE DURING AN INSTALLATION

TECHNICAL FIELD

[0001] Embodiments described herein generally relate to client and server network traffic, and more particularly to systems and methods for alerting a user of the presence of bundled software during installation of a software product.

BACKGROUND ART

[0002] Desktop software or applications may often come with other potentially unwanted programs (PUP's). A PUP is unwanted and/or malicious software, which is distributed by discreetly bundling a requested software product that a user has requested with the unwanted software product that the user may have no knowledge of or interest in. Software agreements that refer to these additional bundled software may often be buried within an end user license agreement (EULA) that is obscured from a user or be hidden under advanced selection buttons as options in a graphical user interface (GUI) instance displayed during an installation process. A user who does not pay close attention to a EULA text or a selection button that indicates the presence of a PUP may unwittingly or without the user's knowledge install the PUP on the user's computer system. These PUPs can be malware such as, for example, adware or spyware that cause undesirable system behavior or alter the security state and privacy posture of the system on which they are installed. Further, spyware may also eavesdrop on a user's activities by collecting personal information like Internet surfing habits, user logins, and bank or credit account information, or may interfere with a user's control of the computer by installing additional harmful software or redirecting web browsers to unintended sites. Malware may also change a computer's settings, which can result in slow Internet connection speeds, unauthorized changes in browser settings, changes to software settings, or the like.

[0003] Current solutions to prevent these PUP's place the burden of preventing installation of these bundled PUP components on end users. For example, end users are expected to read a EULA text or watch carefully the GUI options that are provided during an installation in order to determine whether additional software is included and may be installed during the installation process. These users may have to deselect software components associated with PUPs if they were

not requested by the user. However, non-security savvy or novice users usually choose a default installation option as a quicker or “painless” option, which can result in installation of these additional PUPs. Some anti-virus software solutions may warn a user that software is being installed on a system; however, these solutions do not provide a way to discriminate between an authorized installation of an intended application and an unwanted PUP family. Other software solutions detect adware or spyware; however, they only detect such adware or spyware after it is installed on a user’s computer. A way of detecting the presence of a PUP bundled with user-intended software during installation and alerting the user of the PUP would be desirable.

BRIEF DESCRIPTION OF DRAWINGS

[0004] FIG. 1 is a diagram illustrating a system for detecting bundled software products during an installation of a software product according to one embodiment.

[0005] FIGs. 2A–B are examples of a graphical user interface that is created during installation of a software product according to one embodiment.

[0006] FIG. 3 is another example of a user interface that is created during installation of a software product according to one embodiment.

[0007] FIGs. 4A-4B are flowcharts illustrating a technique for detecting bundled software products during installation of a software product according to one embodiment.

[0008] FIG. 5 is a block diagram illustrating a computing device for use with techniques described herein according to one embodiment.

[0009] FIG. 6 is a block diagram illustrating a computing device for use with techniques described herein according to another embodiment.

[0010] FIG. 7 is a block diagram illustrating a network of programmable devices according to one embodiment.

DESCRIPTION OF EMBODIMENTS

[0011] In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the invention. It will be apparent, however, to one skilled in the art that the invention may be practiced without these specific details.

In other instances, structure and devices are shown in block diagram form in order to avoid obscuring the invention. References to numbers without subscripts or suffixes are understood to reference all instance of subscripts and suffixes corresponding to the referenced number. Moreover, the language used in this disclosure has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter, resort to the claims being necessary to determine such inventive subject matter. Reference in the specification to “one embodiment” or to “an embodiment” means that a particular feature, structure, or characteristic described in connection with the embodiments is included in at least one embodiment of the invention, and multiple references to “one embodiment” or “an embodiment” should not be understood as necessarily all referring to the same embodiment.

[0012] As used herein, the term “computer system” can refer to a single computer or a plurality of computers working together to perform the function described as being performed on or by a computer system.

[0013] As used herein, the term “malware” can refer to any software used to disrupt operation of a programmable device, gather sensitive information or gain access to private systems or networks. Malware includes computer viruses (including worms, Trojan horses, etc.), Bots, ransomware, spyware, adware, tracking cookies, scareware and any other type of malicious programs.

[0014] A technique allows a client computing system with a user client to receive a user-intended software product in an installation package from a server. The user-intended software product may be bundled with a potentially unwanted program. The user client can include logic to determine if the user-intended software product includes other bundled software products by reading one or more software identifiers located in graphical user interface instances that are created during installation of the user-intended software product or other non-GUI files that are included in the installation package. In embodiments, graphical user interface instances are read to identify information that are related to software identifiers for the user-intended software product and software identifiers of potentially unwanted programs that are included in the installation package. The software identifiers may be used to determine whether potentially unwanted programs are bundled with the user intended software product. In an embodiment, a signature file may be received from a server and the software identifiers of the user-intended

software product and the potentially unwanted programs can be checked against the signature file during installation of the user-intended software product in order to detect the presence of potentially unwanted programs that may be bundled with user-intended software product. Although software products may include license text and/or instances of graphical user interface that are provided during installation, users may not be able to discriminate that additional bundled software is included, which may be malware, for example, adware or tracking cookies that may potentially harm a user's compute from a mere reading of the text of a license or graphical user interface provided during installation. The technique allows all software installers including novice software installers to determine whether the bundled software products may be malware although bundled as necessary components of the user-intended software.

[0015] Referring to the figures, FIG. 1 illustrates a system **100** for performing a security check of a software product that may be received by a user client **106** in order to identify the software product a user intends to install, to detect the presence of bundled software in the software product, and to alert the user of the bundled software during installation of the software product (hereinafter "user-intended software product") according to one embodiment. System **100** may include a content server **102**, network **104**, user client machine **106** (or user client **106**), third-party server **112**, and reputation server **116**. Content server **102** may be in communication with user client **106** over network **104** and be configured to communicate software applications, documents, or other content from server **102** to user client **106** in response to a request for software or applications made from user client **106** (i.e., the user-intended software products) via network **104**. In one embodiment, a user may install the user-intended software products from a memory device connected to user client **106**. In an embodiment, the software product that is received from server **102** may include additional software components or applications such as, for example, plug-ins, browser extensions, or software that may be transmitted to user client **106** along with the user-intended software products that may be received from server **102** via network **104**. In an embodiment, these additional software components may be discreetly bundled with the user-intended software products that are downloaded to user client **106**. Network **104** is not limited to a network of interconnected computer networks that use an internet protocol (TCP/IP), and can also include other high-speed data networks and/or telecommunications networks that are configured to pass information back and forth to user client **106**. Also, while three servers **102**,

112, and **116** are shown in the example of system **100**, any number of servers are contemplated to be in communication with user client **106**.

[0016] System **100** may also include a third-party server **112** that is in communication with user client **106** via network **104**. Third-party server **112** is associated with a service provider and may be configured to generate a signature file **110** that may be distributed to user client **106** for use in identifying the user-intended software products that a user intends to install and for identifying additional bundled software that may be unwanted and not a necessary part of the installation package for user-intended software products. In an embodiment, third-party server **112** may be configured to gather information about software names and bundling of software including gathering information associated with blacklisted and whitelisted sites or domains in order to build signature file **110**. In one embodiment, third-party server **112** may be configured to determine the presence of additional bundled software on behalf of user client **106** by receiving software identifiers from user client **106** that were extracted during installation of user-intended software products and comparing the software identifiers against a signature file in order to identify the presence of additional software components.

[0017] User client **106** may include a desktop computer, laptop computer, smart phone, tablet device, and any other types of computing devices having a processor and networking capabilities. User client **106** may include a malware software checker **108** and, optionally, signature file **110** that may be received from third-party server **112**. Signature file **110** can include software names and bundled information as well as names of blacklisted and whitelisted software, sites, domains and other keywords. In one embodiment, malware checker **108** may be configured to scan the user-intended software products and flag each indicator of an instance of a software identifier (for example, a software name) during installation of the user-intended software products in order to determine whether additional bundled software products are being installed during installation of the user-intended software products from the software identifiers. Alternatively, or in addition to determining whether additional bundled software products are being installed from the software identifiers, malware checker **108** may utilize signature file **110** during the security check of the user-intended software products that are downloaded from server **102**, for comparison against software identifiers in the user-intended software products that indicates the presence of bundled software. In an embodiment, user client **106** may communicate, with reputation server **116**,

information related to a match so that reputation server **116** can update statistical information on the user-intended software product and its bundled software. In another embodiment, user client **106** may communicate with reputation server **116** information related to software identifiers for which no known signature exists in signature file **110** upon which user client **106** may either terminate installation or allow installation of user-intended software products on user client **106**.

[0018] Malware software checker **108** may be configured with logic to extract information from GUI's launched during installation of user-intended software products or extracted from non-GUIs that may include information that may serve to identify user-intended software products or other bundled software. For example, textual information may be extracted from EULA text in a EULA GUI or from GUI objects such as check-boxes, non-textual information may be extracted from GUIs such as, for example, graphical objects such as icons, and information may be extracted from files that are not displayed on user client **106** such as, for example, textual or non-textual information in file names with executable files (for example, an installation package for user-intended software product can include a bundled software executable file as an ".exe") or textual information from text files that include identifiers of software products that are included in the installation of user-intended software products. Malware software checker **108** may also be configured to compare information in signature file **110** with information extracted from user-intended software products in order to determine the presence of additional bundled software in the user-intended software products that may potentially be installed during installation. In some embodiments, malware software checker **108** may be configured within an existing anti-virus software residing on user client **106** such as, for example, McAfee[®] anti-virus software by McAfee, Inc. or may be a standalone solution for detecting malware in user-intended software products. In one embodiment, with additional reference to FIGs. **2A-2B** and **3**, malware software checker **108** may be configured to identify software name from name **202** in GUI **200** or within EULA text **212** in GUI **210**, respectively, and compare the software name to keywords in signature file **110** in order to determine the presence of bundled software in GUI **210** user-intended software products. In another embodiment, malware software checker **108** (FIG. **1**) may also be configured to read text and keywords in boxes **302**, **308** and check-box **306** in order to determine the presence of additional bundled software in GUI **300**. use software information in signature file **110** and from information

[0019] FIGs. 2A-2B depict examples of GUIs that may be created during installation of a user-intended software product on system 100 (FIG. 1) according to one embodiment. FIG. 2A depicts an example of an instance of an initial GUI 200 that may be launched or created on user client 106 (FIG. 1) during software installation of a user-intended software product according to one embodiment. In an embodiment, the user-intended software products may be installed using a software installer that is configured to perform software installation by creating GUIs, creating directories, registering shared libraries, updating system configuration settings, or the like. As shown, GUI 200 is initially launched or created when installation of the user-intended software product is initiated. GUI 200 may include software name 202 such as, for example, “Application 1”, and GUI buttons 204 and 206 that a user may use to interact with the software during software installation. As shown in FIG. 2B, GUI 210 depicts an instance of a subsequent GUI that may be created during installation of a user-intended software product. GUI 210 may be created after the initial instance of GUI 200. GUI 210 may also include text in addition to GUI buttons during installation of the user-intended software product, for example, Application 1 on user client 106 (FIG. 1). Text can include text 212 for a EULA that is displayed in window 214 when GUI 210 is launched during installation. GUI 210 may also include scroll bar 216 and buttons 218, 220 and 222. Scroll bar 216 may be selectively movable, for example, movable upwards or downwards in FIG. 2B in order to access additional lines of text 212. Also, buttons 218-222 may be respectively selected by a user during the installation process of the user-intended software product in order to move backwards, forwards, or cancel the installation.

[0020] FIG. 3 is another example of an instance of a GUI 300 that is created on user client 106 (FIG. 1) during a software installation process according to one embodiment. As shown in FIG. 3, GUI 300 may include text box 302 that provides instructions during software installation of user-intended software 304, for example, instructions to install “Application 3” using the Application 2 installer. FIG. 3 also depicts a bundled software product that is bundled with Application 3. For example, an option box 306 (in this example, a check-box) in text box 308 indicates that another software application such as, for example, “Application 4” may be bundled with Application 3 within Application 2 and may be installed with Application 3. A user may manually and selectively uncheck or deselect option box 306 in order to prevent Optimizer Pro from being installed with Application 3 on user client 106.

[0021] FIG. 4 is a flowchart illustrating an embodiment of method 400 that may be used for detecting the presence of bundled software during installation of a user-intended software product that is received from a server 102 (FIG. 1). The method 400 allows software users including novice software users to determine whether bundled components are included that may be unwanted although bundled as necessary components of the user-intended software. As method 400 is performed by system 100, FIG. 1 is also referenced in the description of method 400.

[0022] Method 400 begins in step 405 when installation of user-intended software product is initiated. The user-intended product may be received from server 102 in response to a request sent from user client 106 via network 104. In 410, an initial software process is tracked. For example, a user on user client 106 may initiate installation of the user-intended software product which instantiates a software process that is associated with an instance of the installation of the user-intended software product. This initial software process is tracked or monitored during installation of the user-intended software product. In an embodiment, installation of the user-intended software product may instantiate additional software processes that may be tracked during installation. In 415, system 100 determines if a GUI is displayed during software installation. The GUI may be an initial GUI instance, for example, GUI 200, that may be created on user client 106 during installation. If an initial GUI instance is not displayed or created (i.e., step 415 = "N"), step 415 proceeds to step 420 where it may be determined if software information including software identifiers may be captured from non-GUI files that are copied to memory on user client 106 during installation of the user-intended software product. Non-GUIs can include executable files, text files, or the like. If non-GUI's may not be able to be read (i.e., step 420 = "N"), in 430, the initial software process is "escaped" (or terminated) and installation is terminated or may be allowed to continue.

[0023] However, if software information may be captured from non-GUIs (i.e., step 420 = "Y"), in 425, software identifiers that identify the software the user intends to install as well as other bundled software products may be extracted from non-GUI files that are not displayed on user client 106. For example, software identifiers may be extracted from information within executable files (for example, a software installer executable file having an .exe extension) or may be extracted from information within text files (for example, files with a .txt extension). The extracted from non-GUIs are stored in memory. Step 425 proceeds to step 450.

[0024] However, in **415**, if a GUI instance is displayed (i.e., step **415** = “Y”), then, in **435**, the initial GUI instance is scanned/read and textual information within the GUI is identified and stored in memory on user client **106**. For example, the GUI may be scanned to flag each indicator of a software identifier for a name of the software (e.g., software name **202** in FIG. **2A**) of the user-intended software products that are being installed and continued scanning for each indicator of a software identifier for other bundled software that is included in the installation such as, in some non-limiting examples, textual information within a EULA that is displayed in the GUI, information extracted from GUI objects such as check-boxes and non-textual material and graphical objects such as icons. This scanned information is stored in memory.

[0025] In **440** if a subsequent GUI instance is displayed (i.e., step **440** = “Y”), in **445**, information from the subsequent GUI instance may be scanned to identify software names of user-intended software products and other bundled software in the GUI. A subsequent GUI instance (for example, GUI **210** in FIG. **2B** or GUI **300** in FIG. **3**) may be created when a user selects a button to continue installation, for example, by selecting GUI button **204** in GUI **200** (FIG. **2A**). The subsequent GUI instance may be read to identify software identifiers that can be used to determine user-intended software products and other additional bundled software such as, in some non-limiting examples, textual information within a EULA that is displayed in the GUI, information extracted from GUI objects such as check-boxes and non-textual material and graphical objects such as icons. In one embodiment, the GUI instance may be scanned in order to identify software names as well as identifying installation options provided in the GUI, for example, identifying installation options such as option box **306** in order to determine whether the installation options are components to be included in order to install user-intended software product or whether the installation options are options only associated with bundled PUPs. Software names, keywords and installation options in the subsequent GUI instance are stored in memory on user client **106**. However, if a subsequent GUI instance is not displayed or created (i.e., step **440** = “N”), step **440** proceeds to step **450**.

[0026] In **450**, malware software checker **108** uses software identifiers that were flagged during scanning of user-intended software products to determine if bundled software products are included. User client **106** may compare the extracted software information such as, for example, compare the extracted software identifiers with information on software names in order to identify

the user-intended software products and whether additional PUPs or components may be bundled that are unwanted for installation of the user-intended software products.

[0027] In **455**, malware software checker **108** determines whether additional bundled software products are included in the user-intended software products based on the comparison in step **450** (i.e., step **455** = “Y”), in **460**, a visual alert may be displayed on a display of user client **106** that provides a warning to a user and installation of the user-intended software product may be blocked by terminating the initial software process or a user may proceed to install the user-intended software products upon affirmative action (i.e., user affirmatively selects installation) by the user. Method **400** ends in step **430** when analysis of the user-intended software product is completed or installation is completed.

[0028] However, if malware software checker cannot determine that bundled software products are included (i.e., step **455** = “N”), in **465**, extracted information including software identifiers is compared with information in signature file **110** and with blacklisted and whitelisted information in order to identify the user-intended software products as well as other bundled software as part of the installation of the user-intended software products. User client **106** may receive an updated signature file **110** in response to a request that is sent from user client **106** during installation of the user-intended software products or signature file **110** may receive an updated signature file **110** periodically from third-party server **112**.

[0029] In one embodiment, comparison with the signature file **110** is performed at user client **106**, which receives signature file **110** from third-party server **112**. User client **106** may compare the extracted software information with signature file **110** such as, for example, compare the extracted software identifiers with information on software names with their associated bundled components in signature file **110** in order to identify the user-intended software products and whether additional PUPs or components may be bundled that are unwanted for installation of the user-intended software products. In other embodiments, user client **106** may compare the extracted software information with blacklisted software names or blacklisted sites and domains in signature file **110** i for identifying undesirable applications that are known to contain threats or vulnerabilities and to compare the extracted software information with whitelisted software names and whitelisted sites and domains in order to identify known good applications that may be permitted to execute.

[0030] Alternatively, comparison with the signature file **110** may be performed at third-party server **112**, which receives extracted software information from user client **106**. Third-party server **112** may use signature file **110** to compare the extracted software identifiers with signature file **110** in order to identify the user-intended software products and whether additional PUPs or components are bundled that are unwanted for installation of the user-intended software products. In other embodiments, third-party server **112** may compare the extracted software information with blacklisted software and blacklisted sites and domains in signature file **110** in order to identify undesirable applications that are known to contain threats or vulnerabilities and to compare the extracted software information with whitelisted software and whitelisted sites and domains in order to identify known good applications that may be permitted to execute.

[0031] In **470**, if additional bundled software is included in the user-intended software product that are unwanted based on the comparison with signature file **110** in step **465** (i.e., step **470** = “Y”), in **480**, a visual alert may be displayed on a display of user client **106** that provides a warning to a user and installation of the user-intended software product may be blocked by terminating the initial software process. The visual alert may be provided based on analysis performed by user client **106** during installation of the user-intended software products or may be provided based on analysis performed by third-party server **112** that returns the analysis to user client **106**. In an embodiment, a visual alert may be presented via an anti-virus software on user client **106** by notifying the user that the user-intended software product includes bundled PUPs and that installation is being terminated. A user may also be presented with an option to deselect one or more installation options for the bundled PUPs by displaying a GUI instance with installation options that a user can select or deselect. However, a user may choose to continue the initial software process and continue installation of the user-intended software product and the bundled PUPs without deselecting any installation options. Alternately, embodiments may default to not installing the bundled PUPs without affirmative action selecting the PUPs. However, in **470**, if bundled software is not present based on the comparison with signature file **110** in step **465** (i.e., step **470** = “N”), in **475**, the software installation process of user-intended software product is allowed to proceed until completion. Method **400** ends in step **485** when analysis of the user-intended software product is completed or installation is completed.

[0032] Referring now to FIG. 5, a block diagram illustrates a programmable device **500** that may be used within user client **106** in accordance with one embodiment. The programmable device **500** illustrated in FIG. 5 is a multiprocessor programmable device that includes a first processing element **570** and a second processing element **580**. While two processing elements **570** and **580** are shown, an embodiment of programmable device **500** may also include only one such processing element.

[0033] Programmable device **500** is illustrated as a point-to-point interconnect system, in which the first processing element **570** and second processing element **580** are coupled via a point-to-point interconnect **550**. Any or all of the interconnects illustrated in FIG. 5 may be implemented as a multi-drop bus rather than point-to-point interconnects.

[0034] As illustrated in FIG. 5, each of processing elements **570** and **580** may be multicore processors, including first and second processor cores (i.e., processor cores **574a** and **574b** and processor cores **584a** and **584b**). Such cores **574a**, **574b**, **584a**, **584b** may be configured to execute instruction code in a manner similar to that discussed above in connection with FIGs. 1–4. However, other embodiments may use processing elements that are single core processors as desired. In embodiments with multiple processing elements **570**, **580**, each processing element may be implemented with different numbers of cores as desired.

[0035] Each processing element **570**, **580** may include at least one shared cache **546**. The shared cache **546a**, **546b** may store data (e.g., instructions) that are utilized by one or more components of the processing element, such as the cores **574a**, **574b** and **584a**, **584b**, respectively. For example, the shared cache may locally cache data stored in a memory **532**, **534** for faster access by components of the processing elements **570**, **580**. In one or more embodiments, the shared cache **546a**, **546b** may include one or more mid-level caches, such as level 2 (L2), level 3 (L3), level 4 (L4), or other levels of cache, a last level cache (LLC), or combinations thereof.

[0036] While FIG. 5 illustrates a programmable device with two processing elements **570**, **580** for clarity of the drawing, the scope of the present invention is not so limited and any number of processing elements may be present. Alternatively, one or more of processing elements **570**, **580** may be an element other than a processor, such as an graphics processing unit (GPU), a digital signal processing (DSP) unit, a field programmable gate array, or any other programmable

processing element. Processing element **580** may be heterogeneous or asymmetric to processing element **570**. There may be a variety of differences between processing elements **570**, **580** in terms of a spectrum of metrics of merit including architectural, microarchitectural, thermal, power consumption characteristics, and the like. These differences may effectively manifest themselves as asymmetry and heterogeneity amongst processing elements **570**, **580**. In some embodiments, the various processing elements **570**, **580** may reside in the same die package.

[0037] First processing element **570** may further include memory controller logic (MC) **572** and point-to-point (P-P) interconnects **576** and **578**. Similarly, second processing element **580** may include a MC **582** and P-P interconnects **586** and **588**. As illustrated in FIG. 5, MCs **572** and **582** couple processing elements **570**, **580** to respective memories, namely a memory **532** and a memory **534**, which may be portions of main memory locally attached to the respective processors. While MC logic **572** and **582** is illustrated as integrated into processing elements **570**, **580**, in some embodiments the memory controller logic may be discrete logic outside processing elements **570**, **580** rather than integrated therein.

[0038] Processing element **570** and processing element **580** may be coupled to an I/O subsystem **590** via respective P-P interconnects **576** and **586** through links **552** and **554**. As illustrated in FIG. 5, I/O subsystem **590** includes P-P interconnects **594** and **598**. Furthermore, I/O subsystem **590** includes an interface **592** to couple I/O subsystem **590** with a high performance graphics engine **538**. In one embodiment, a bus (not shown) may be used to couple graphics engine **538** to I/O subsystem **590**. Alternately, a point-to-point interconnect **539** may couple these components.

[0039] In turn, I/O subsystem **590** may be coupled to a first link **516** via an interface **596**. In one embodiment, first link **516** may be a Peripheral Component Interconnect (PCI) bus, or a bus such as a PCI Express bus or another I/O interconnect bus, although the scope of the present invention is not so limited.

[0040] As illustrated in FIG. 5, various I/O devices **514**, **524** may be coupled to first link **516**, along with a bridge **518** which may couple first link **516** to a second link **520**. In one embodiment, second link **520** may be a low pin count (LPC) bus. Various devices may be coupled to second link **520** including, for example, a keyboard/mouse **512**, communication device(s) **526** (which may

in turn be in communication with the computer network **503**), and a data storage unit **528** such as a disk drive or other mass storage device which may include code **530**, in one embodiment. The code **530** may include instructions for performing embodiments of one or more of the techniques described above. Further, an audio I/O **524** may be coupled to second bus **520**.

[0041] Note that other embodiments are contemplated. For example, instead of the point-to-point architecture of FIG. 5, a system may implement a multi-drop bus or another such communication topology. Although links **516** and **520** are illustrated as busses in FIG. 4, any desired type of link may be used. Also, the elements of FIG. 5 may alternatively be partitioned using more or fewer integrated chips than illustrated in FIG. 5.

[0042] Referring now to FIG. 6, a block diagram illustrates a programmable device **600** according to another embodiment. Certain aspects of FIG. 5 have been omitted from FIG. 6 in order to avoid obscuring other aspects of FIG. 6.

[0043] FIG. 6 illustrates that processing elements **670**, **680** may include integrated memory and I/O control logic (“CL”) **672** and **682**, respectively. In some embodiments, the **672**, **682** may include memory control logic (MC) such as that described above in connection with FIG. 5. In addition, CL **672**, **682** may also include I/O control logic. FIG. 6 illustrates that not only may the memories **632**, **634** be coupled to the **672**, **682**, but also that I/O devices **644** may be coupled to the control logic **672**, **682**. Legacy I/O devices **615** may be coupled to the I/O subsystem **690** by interface **696**. Each processing element **670**, **680** may include multiple processor cores, illustrated in FIG. 5 as processor cores **674A**, **674B**, **684A**, and **684B**. As illustrated in FIG. 6, I/O subsystem **690** includes P-P interconnects **694** and **698** that connect to P-P interconnects **676** and **686** of the processing elements **670** and **680** with links **652** and **654**. Processing elements **670** and **680** may also be interconnected by link **650** and interconnects **678** and **688**, respectively.

[0044] The programmable devices depicted in FIGs. 5 and 6 are schematic illustrations of embodiments of programmable devices which may be utilized to implement various embodiments discussed herein. Various components of the programmable devices depicted in FIGs. 5 and 6 may be combined in a system-on-a-chip (SoC) architecture.

[0045] Referring now to FIG. 7, an example infrastructure **700** in which the techniques described above may be implemented is illustrated schematically. Infrastructure **700** contains

computer networks **702**. Computer networks **702** may include many different types of computer networks available today, such as the Internet, a corporate network, or a Local Area Network (LAN). Each of these networks can contain wired or wireless programmable devices and operate using any number of network protocols (*e.g.*, TCP/IP). Networks **702** may be connected to gateways and routers (represented by **708**), end user computers **706**, and computer servers **704**. Infrastructure **700** also includes cellular network **703** for use with mobile communication devices. Mobile cellular networks support mobile phones and many other types of mobile devices. Mobile devices in the infrastructure **700** are illustrated as mobile phones **710**, laptops **712**, and tablets **714**. A mobile device such as mobile phone **710** may interact with one or more mobile provider networks as the mobile device moves, typically interacting with a plurality of mobile network towers **720**, **730**, and **740** for connecting to the cellular network **703**. Although referred to as a cellular network in FIG. 7, a mobile device may interact with towers of more than one provider network, as well as with multiple non-cellular devices such as wireless access points and routers **708**. In addition, the mobile devices **710**, **712**, and **714** may interact with non-mobile devices such as computers **704** and **706** for desired services, which may include checking for the presence of bundled software in an application as described above. The functionality of user client **106** may be implemented in any device or combination of devices illustrated in FIG. 7; however, most commonly is implemented in a firewall or intrusion protection system in a gateway or router.

[0046] The following examples pertain to further embodiments.

[0047] Example 1 is a machine readable medium, on which are stored instructions, comprising instructions that when executed by a processor cause a machine to: identify an instance of a software process associated with installation of a software product; scan the software product for identifying first information in the software product; continue scanning the software product for identifying second information in the software product; compare the first information with second information responsive to scanning the software product; and determine whether one or more additional software products are to be installed with the software product responsive to comparing the first information and the second information.

[0048] In Example 2, the subject matter of Example 1 can optionally include, wherein the instructions further comprise instructions that when executed cause the machine to display an alert

on an user interface responsive to determining that the software product includes the one or more additional software products.

[0049] In Example 3, the subject matter of Example 1 to 2 can optionally include, wherein the instructions to display the alert further comprise instructions that when executed cause the machine to terminate the instance of the software process responsive to determining that the software product includes the one or more additional software products.

[0050] In Example 4, the subject matter of Example 1 to 2 can optionally include, wherein the instructions to identify the first information further comprise instructions that when executed cause the machine to read textual information in at least one graphical user interface (GUI) of the software product.

[0051] In Example 5, the subject matter of Example 1 to 2 can optionally include, wherein the instructions to identify the first information further comprise instructions that when executed cause the machine to read textual information from one or more files that are associated with the software product.

[0052] In Example 6, the subject matter of Example 1 to 2 can optionally include, wherein the instructions further comprise instructions that when executed cause the machine to receive a signature file from a server.

[0053] In Example 7, the subject matter of Example 6 can optionally include, wherein the instructions further comprise instructions that when executed cause the machine to compare the first and the second information to the signature file responsive to receiving the signature file from the server.

[0054] In Example 8, the subject matter of Example 1 to 2 can optionally include, wherein the first information comprises a name of the software product.

[0055] In Example 9, the subject matter of Example 1 to 2 can optionally include, wherein the second information comprises a name for the one or more additional software products.

[0056] Example 10 is a computer system for detecting bundled software during installation of a software product, comprising: one or more processors; and a memory coupled to the one or more processors, on which are stored instructions, comprising instructions that when executed cause one

or more of the processors to: identify an instance of a software process associated with installation of a software product; scan the software product for identifying first information in the software product; continue scanning the software product for identifying second information in the software product; compare the first information and the second information with information in a signature file; and determine whether one or more additional software products are to be installed with the software product responsive to comparing the first information and the second information.

[0057] In Example 11, the subject matter of Example 10 can optionally include, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to terminate the instance of the software process responsive to determining that the one or more additional software products are not to be installed with the software product.

[0058] In Example 12, the subject matter of Example 10 to 11 can optionally include, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to display an alert on an user interface responsive to determining that the software product includes the one or more additional software products.

[0059] In Example 13, the subject matter of Example 10 to 11 can optionally include, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to read textual information in at least one graphical user interface (GUI) of the software product.

[0060] In Example 14, the subject matter of Example 10 to 11 can optionally include, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to read textual information from one or more files that are associated with the software product.

[0061] In Example 15, the subject matter of Example 10 to 11 can optionally include, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to receive the signature file from a server.

[0062] In Example 16, the subject matter of Example 10 to 11 can optionally include, wherein the first information comprises a name of the software product.

[0063] In Example 17, the subject matter of Example 10 to 11 can optionally include, wherein the second information comprises a name for each of the one or more additional software products.

[0064] In Example 18, the subject matter of Example 10 to 11 can optionally include, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to transmit the first information to a server.

[0065] In Example 19, the subject matter of Example 18 can optionally include, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to receive information related to determining that the one or more additional software products are not to be installed with the software product responsive to transmitting the first information to the server.

[0066] Example 20 is a method for detecting bundled software during installation of a software product, comprising: receiving first information related to software names and associated bundled components; receiving second information related to at least one of blacklisted and whitelisted information; generating a signature file from at least one of the first and the second information; receiving a request for the signature file from a user client; and transmitting the signature file to the user client responsive to receiving the request for the signature file.

[0067] In Example 21, the subject matter of Example 20 can optionally include, further comprising receiving software information associated with the software product from the user client.

[0068] In Example 22, the subject matter of Example 21 can optionally include, further comprising: comparing the software information with information in the signature file; and determining whether at least one additional software product is to be installed with the software product responsive to comparing the software information and the signature file.

[0069] In Example 23, the subject matter of Example 22 can optionally include, further comprising transmitting instructions for terminating the installation of the software product responsive to determining that the software product includes the one or more additional software products.

[0070] In Example 24, the subject matter of Example 20 to 23 can optionally include, further comprising receiving textual information in at least one graphical user interface (GUI) of the software product from the user client, wherein the textual information comprises a name of the software product.

[0071] In Example 25, the subject matter of Example 20 to 23 can optionally include, further comprising receiving textual information from one or more files that are associated with the software product.

[0072] Example 26 is a computer system for detecting bundled software during installation of a software product, comprising: one or more processors; and a memory coupled to the one or more processors, on which are stored instructions, comprising instructions that when executed cause one or more of the processors to: identify an instance of a software process associated with installation of a software product; scan the software product for identifying first information in the software product; continue scanning the software product for identifying second information in the software product; compare the first information and the second information with information in a signature file; and determine whether one or more additional software products are to be installed with the software product responsive to comparing the first information and the second information.

[0073] In Example 27, the subject matter of Example 26 can optionally include, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to terminate the instance of the software process responsive to determining that the one or more additional software products are not to be installed with the software product.

[0074] In Example 28, the subject matter of Example 26 to 27 can optionally include, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to display an alert on an user interface responsive to determining that the software product includes the one or more additional software products.

[0075] In Example 29, the subject matter of Example 26 to 27 can optionally include, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to read textual information in at least one graphical user interface (GUI) of the software product.

[0076] In Example 30, the subject matter of Example 26 to 27 can optionally include, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to read textual information from one or more files that are associated with the software product.

[0077] In Example 31, the subject matter of Example 26 to 27 can optionally include, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to receive the signature file from a server.

[0078] In Example 32, the subject matter of Example 26 to 27 can optionally include, wherein the first information comprises a name of the software product.

[0079] In Example 33, the subject matter of Example 26 to 27 can optionally include, wherein the second information comprises a name for each of the one or more additional software products.

[0080] In Example 34, the subject matter of Example 26 to 27 can optionally include, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to transmit the first information to a server.

[0081] In Example 35, the subject matter of Example 34 can optionally include, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to receive information related to determining that the one or more additional software products are not to be installed with the software product responsive to transmitting the first information to the server.

[0082] It is to be understood that the above description is intended to be illustrative, and not restrictive. For example, the above-described embodiments may be used in combination with each other. Many other embodiments will be apparent to those of skill in the art upon reviewing the above description. The scope of the invention therefore should be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.

CLAIMS

What is claimed is:

1. A machine readable medium, on which are stored instructions, comprising instructions that when executed by one or more processors cause a machine to:

receive first monitored information from primary sensors over a first channel, wherein the first monitored information includes at least one primary parameter of a control system;

receive second monitored information from secondary sensors over a second channel for the control system, wherein the second channel is independent of the first channel;

determine at least one secondary parameter for the control system responsive to receiving the second monitored information;

determine a discrepancy in the at least one primary parameter; and

provide an alert responsive to a positive determination of a discrepancy in the at least one primary parameter;

wherein the primary sensors and the secondary sensors are associated with the control system.

2. The machine readable medium of claim 1, wherein the instructions to determine the at least one secondary parameter further comprises instructions that when executed cause the machine to perform cognitive analysis on the second monitored information.

3. The machine readable medium of claim 1 to 2, wherein the instructions to determine the discrepancy further comprises instructions that when executed cause the machine to determine a difference between the at least one primary parameter and the at least one secondary parameter.

4. The machine readable medium of claim 1 to 2, wherein the instructions to determine the discrepancy further comprises instructions that when executed cause the machine to compare the at least one secondary parameter with one or more historical or predetermined operational parameters.

5. The machine readable medium of claim 1 to 2, wherein the at least one secondary parameter is a redundant value determined for the at least one primary parameter.
6. The machine readable medium of claim 1 to 2, wherein each of at least one primary parameter and the at least one secondary parameter comprises a vital system health indicator.
7. The machine readable medium of claim 1 to 2, wherein the primary sensors are digital sensors and the secondary sensors are analog sensors.
8. The machine readable medium of claim 1 to 2, wherein the instructions further comprise instructions that when executed cause the machine to shut-down the control system responsive to receiving the notification.
9. A computer system for cognitive protection of a control system, comprising:
 - one or more processors;
 - a memory coupled to the one or more processors, on which are stored instructions, comprising instructions that when executed cause one or more of the processors to:
 - receive first monitored information from primary sensors over a first channel, wherein the first monitored information includes at least one primary parameter of the control system;
 - receive second monitored information from secondary sensors over a second channel for the control system, wherein the second channel is independent of the first channel;
 - determine at least one secondary parameter for the control system responsive to receiving the second monitored information;
 - determine a discrepancy in the at least one primary parameter; and
 - provide an alert when there is a discrepancy in the at least one primary parameter;wherein the primary sensors and the secondary sensors are associated with the control system.

10. The computer system of claim 9, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to perform cognitive analysis on the second monitored information.

11. The computer system of claim 9 to 10, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to determine a difference between the at least one primary parameter and the at least one secondary parameter.

12. The computer system of aim 9 to 10, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to compare the at least one secondary parameter with one or more historical or predetermined operational parameters.

13. The computer system of aim 9 to 10, wherein the at least one secondary parameter is a redundant value determined for the at least one primary parameter.

14. The computer system of claim 9 to 10, wherein each of at least one primary parameter and the at least one secondary parameter comprises a vital system health indicator.

15. The computer system of claim 9 to 10, wherein the primary sensors are digital sensors and the secondary sensors are analog sensors.

16. The computer system of claim 9 to 10, wherein the instructions further comprise instructions that when executed cause the one or more of the processors to shut-down the control system responsive to receiving the notification.

17. A method for cognitive protection of a control system, comprising:
receiving first monitored information from primary sensors over a first channel, wherein the first monitored information includes at least one primary parameter of the control system;
receiving second monitored information from secondary sensors over a second channel for the control system, wherein the second channel is independent of the first channel;
determining at least one secondary parameter for the control system responsive to receiving the second monitored information;
determining a discrepancy in the at least one primary parameter; and

providing an alert when there is a discrepancy in the at least one primary parameter;
wherein the primary sensors and the secondary sensors are associated with the control system.

18. The method of claim 17, further comprising performing cognitive analysis on the second monitored information.

19. The method of claim 17 to 18, further comprising determining a difference between the at least one primary parameter and the at least one secondary parameter.

20. The method of claim 17 to 18, further comprising comparing the at least one secondary parameter with one or more historical or predetermined operational parameters.

21. The method of claim 17 to 18, further comprising determining the at least one secondary parameter as a redundant value for the at least one primary parameter.

22. The method of claim 17 to 18, wherein each of at least one primary parameter and the at least one secondary parameter comprises a vital system health indicator.

23. The method of claim 17 to 18, wherein the primary sensors are digital sensors and the secondary sensors are analog sensors.

24. The method of claim 17 to 18, further comprising shutting-down the control system responsive to providing the alert.

1/9

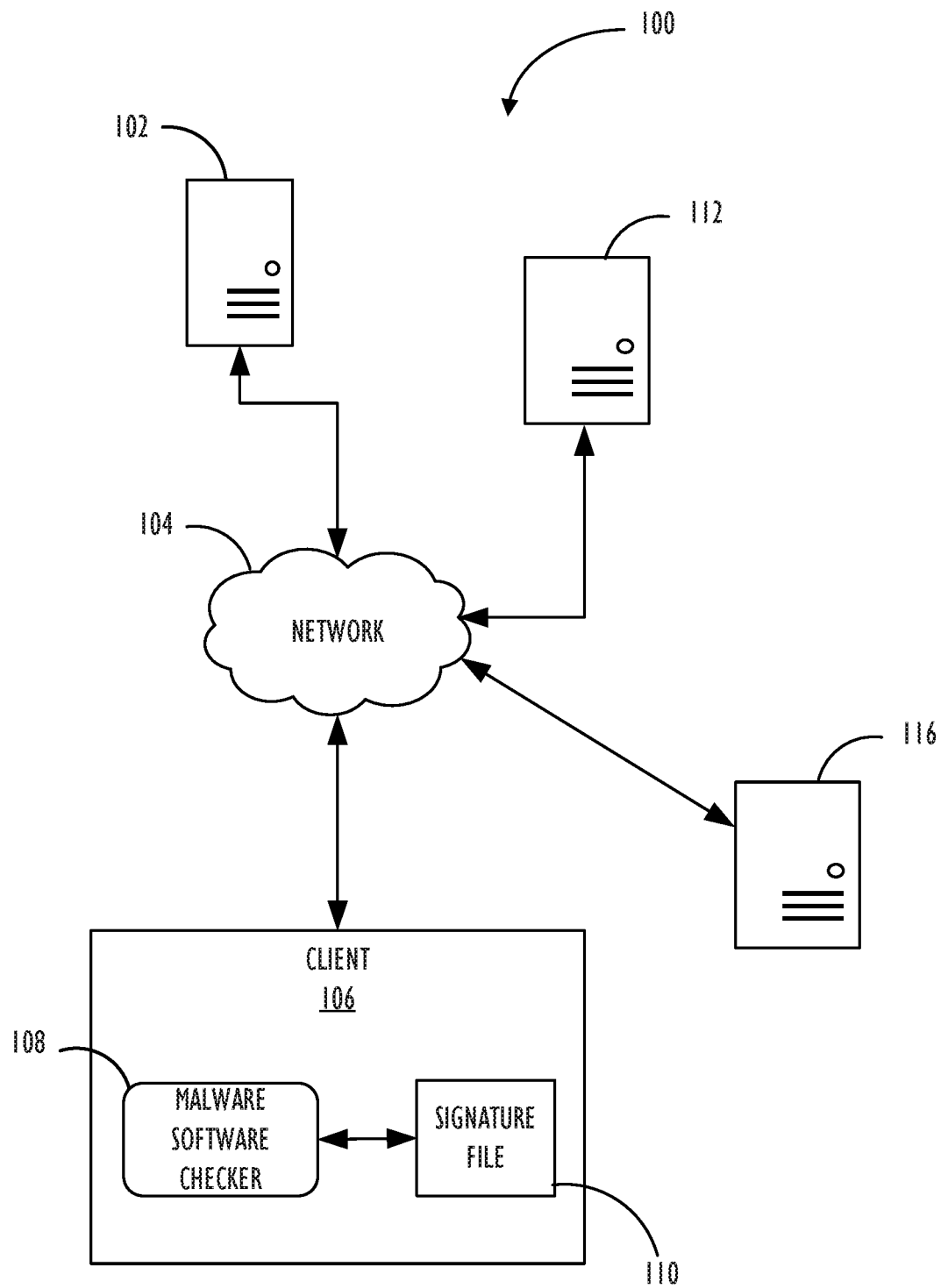


FIG. 1

2/9

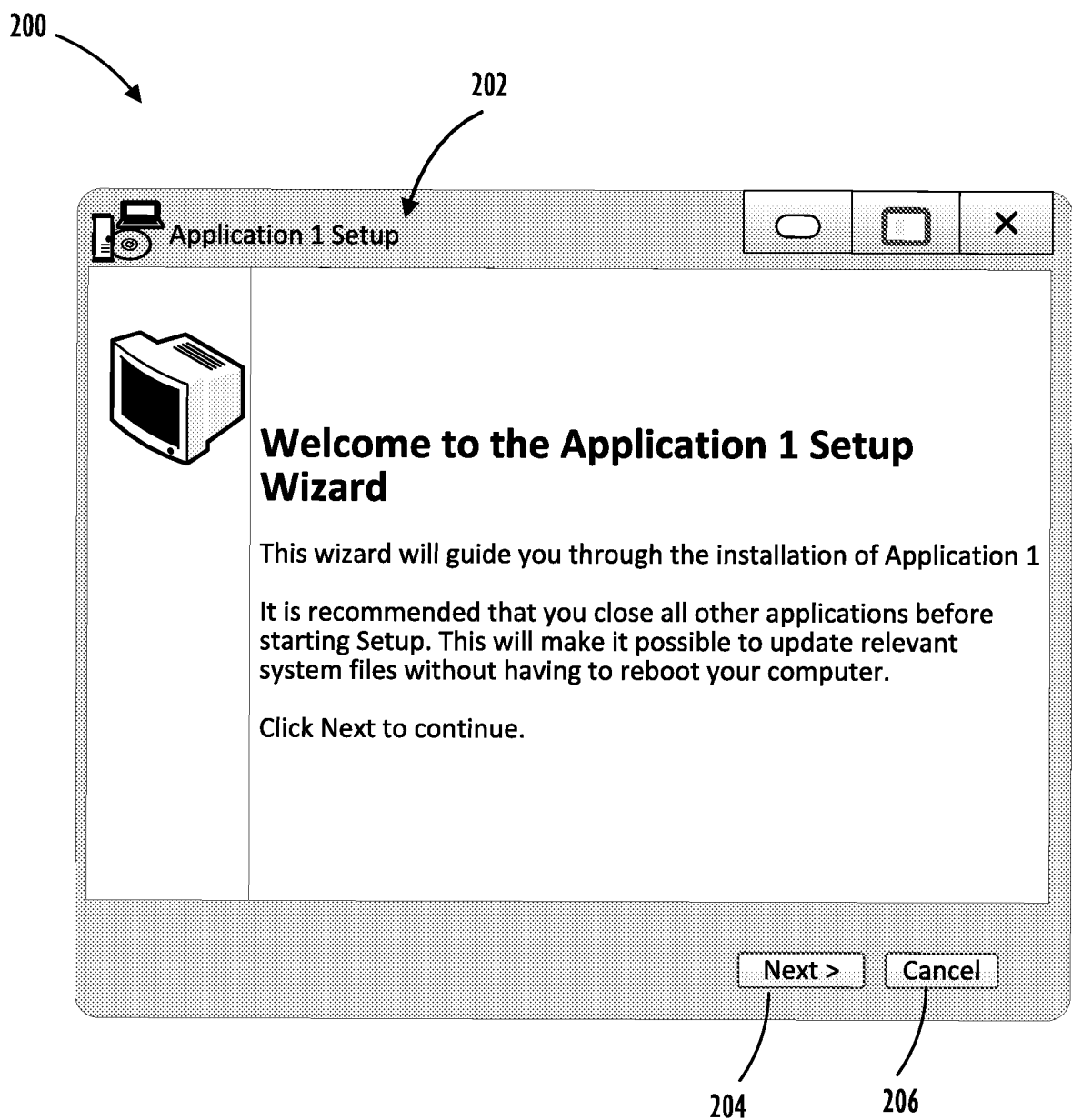


FIG. 2A

3/9

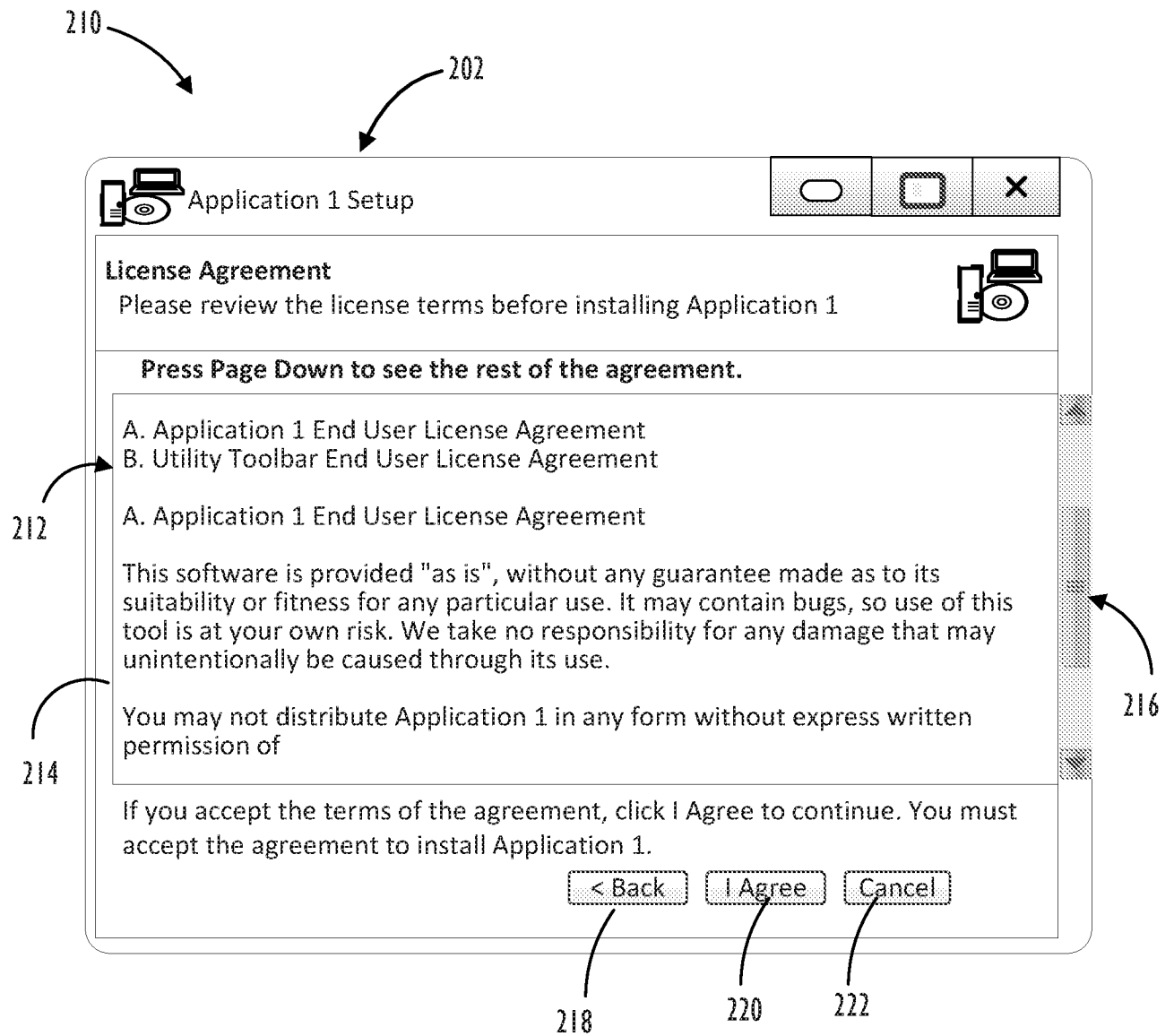


FIG. 2B

4/9

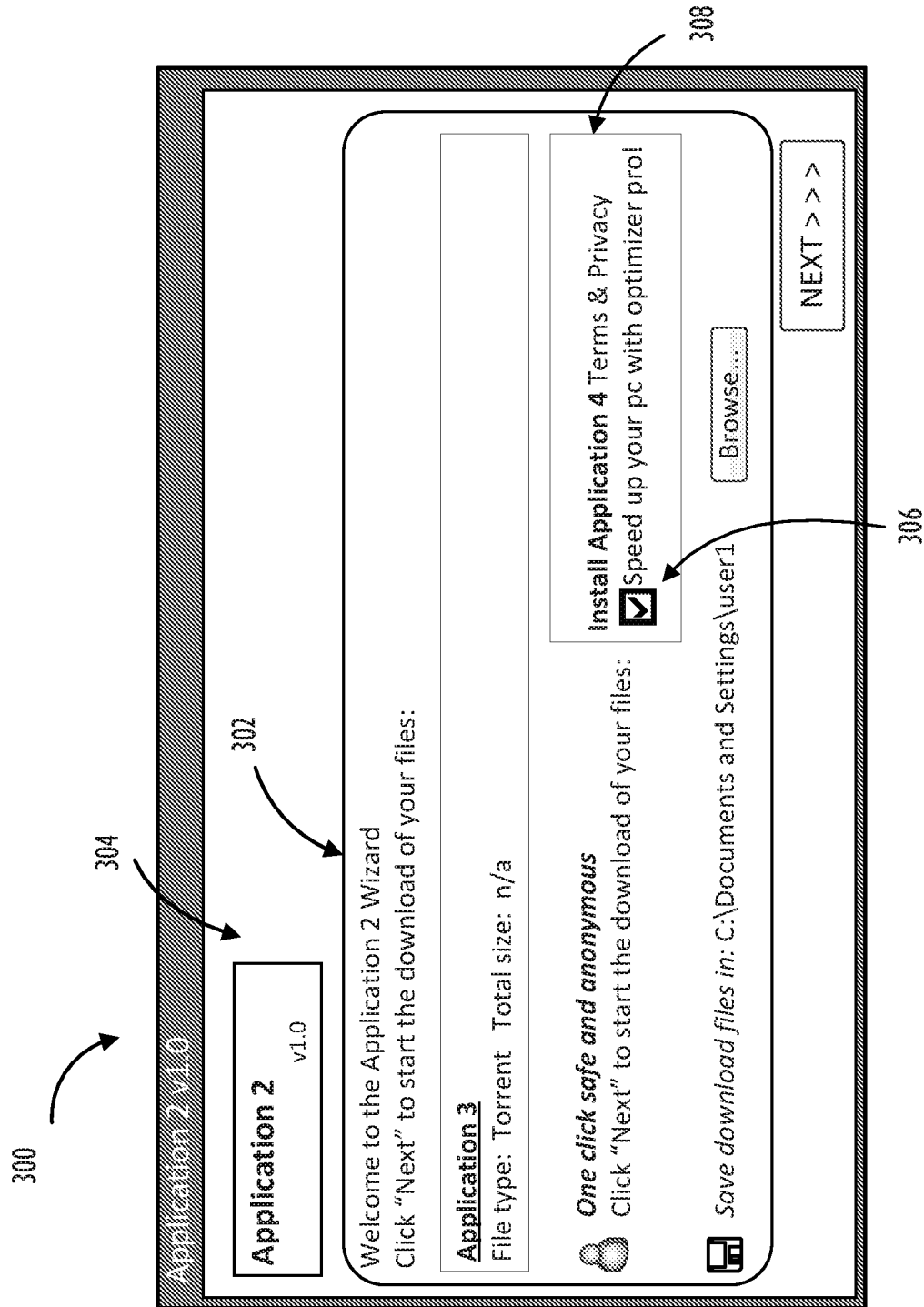


FIG. 3

5/9

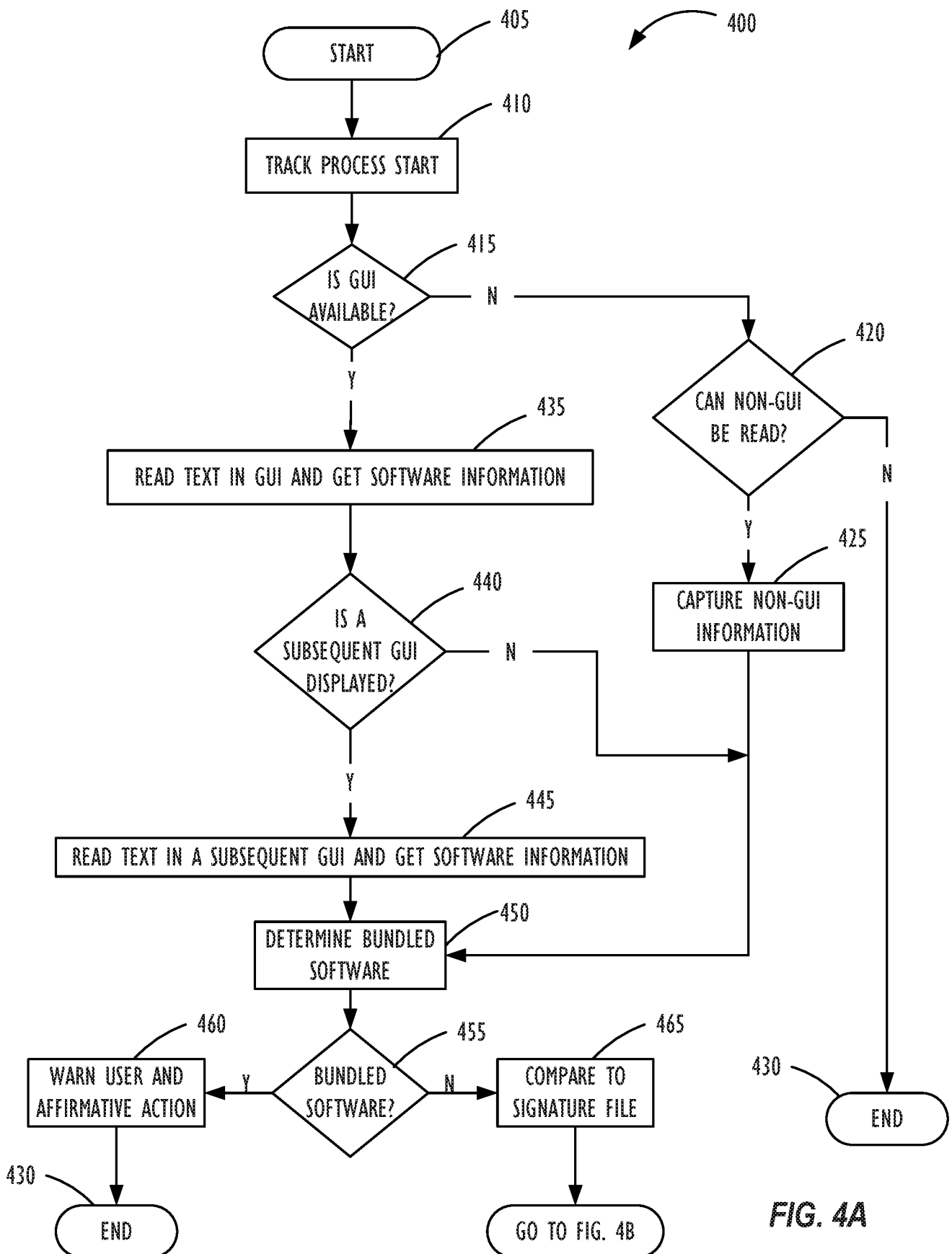


FIG. 4A

6/9

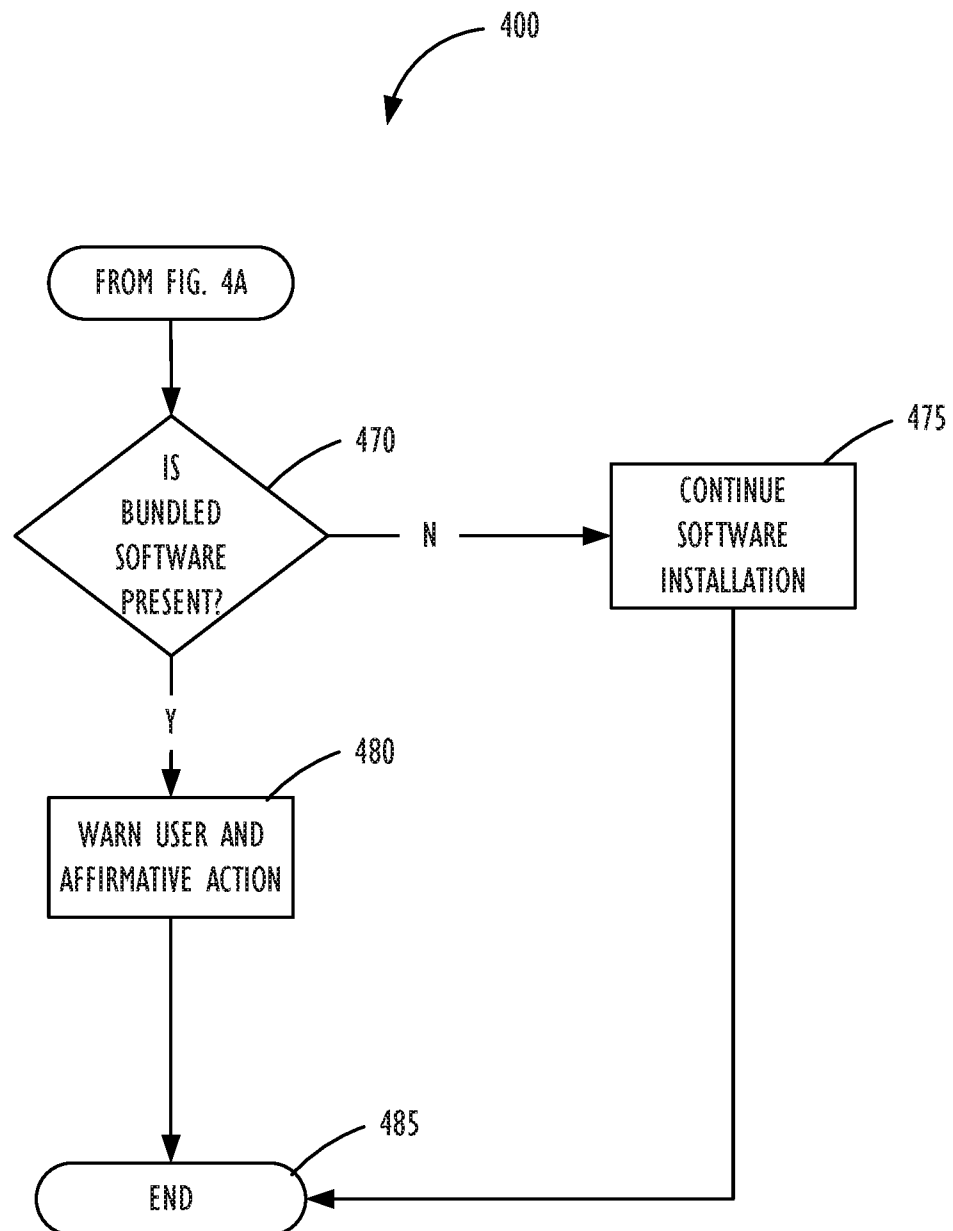


FIG. 4B

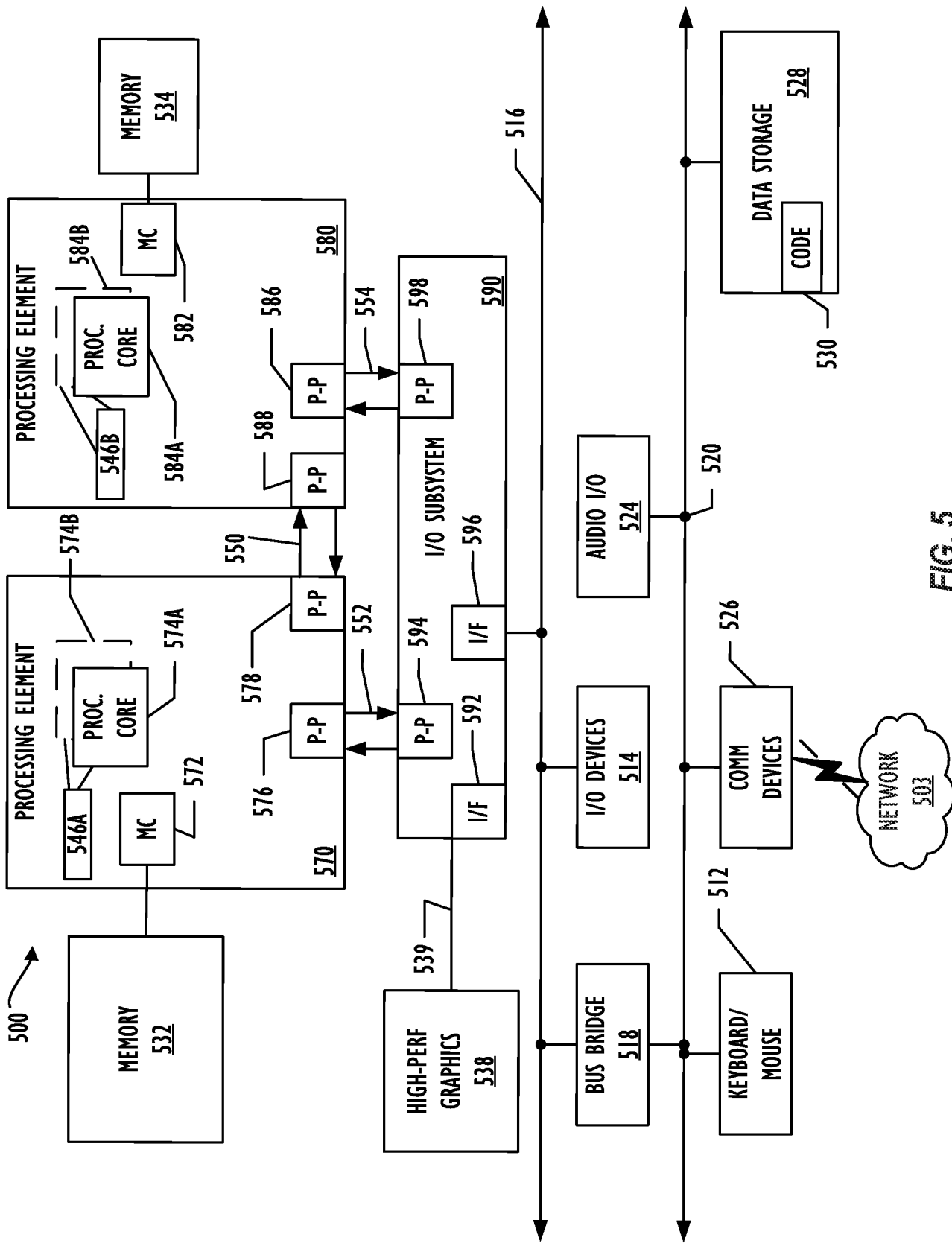


FIG. 5

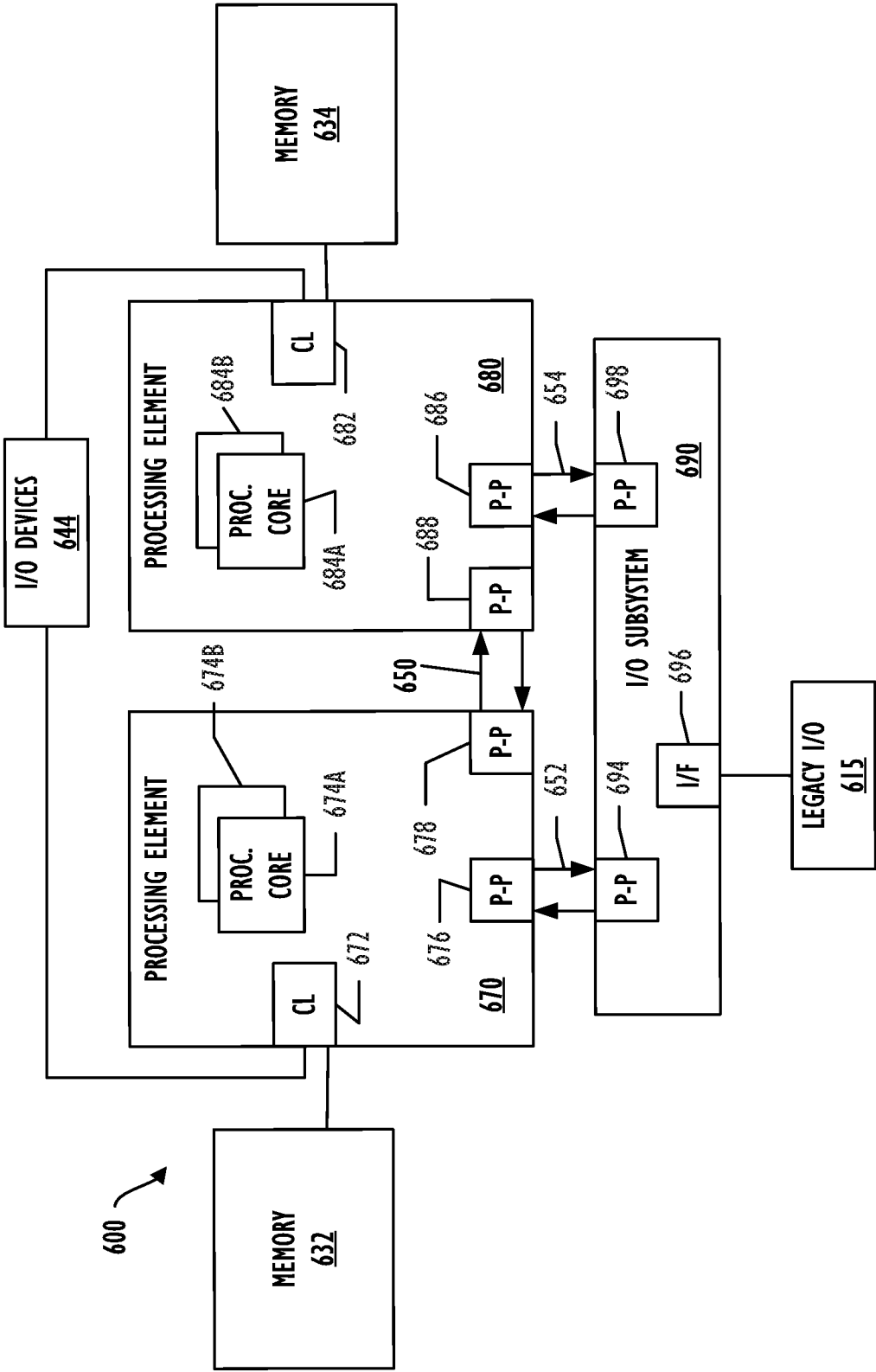


FIG. 6

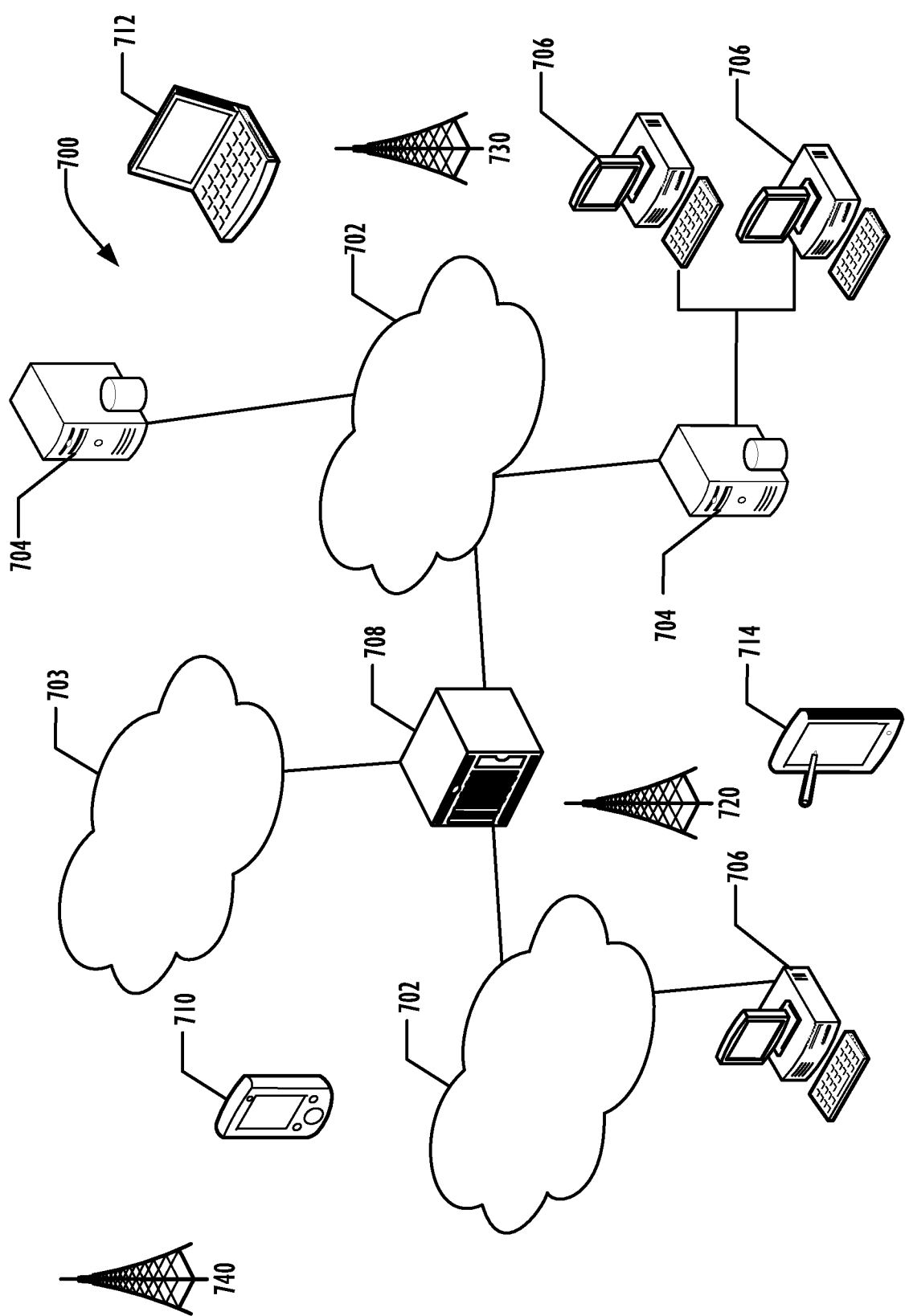


FIG. 7

PATENT COOPERATION TREATY

PCT

DECLARATION OF NON-ESTABLISHMENT OF INTERNATIONAL SEARCH REPORT
(PCT Article 17(2)(a), Rules 13ter.1(c) and (d) and 39)

Applicant's or agent's file reference P80059PCT-382WO	IMPORTANT DECLARATION	Date of mailing (<i>day/month/year</i>) 20 July 2016 (20.07.2016)
International application No. PCT/US2016/026003	International filing date (<i>day/month/year</i>) 05 April 2016 (05.04.2016)	(Earliest) Priority date (<i>day/month/year</i>) 06 May 2015 (06.05.2015)
International Patent Classification (IPC) or both national classification and IPC G06F 21/44(2013.01)i, G06F 9/44(2006.01)i, G06F 11/30(2006.01)i		
Applicant MCAFEE, INC.		

This International Searching Authority hereby declares, according to Article 17(2)(a), that **no international search report will be established** on the international application for the reasons indicated below.

1. ☐ The subject matter of the international application relates to:
 - a. ☐ scientific theories
 - b. ☐ mathematical theories
 - c. ☐ plant varieties
 - d. ☐ animal varieties
 - e. ☐ essentially biological processes for the production of plants and animals, other than microbiological processes and the products of such processes
 - f. ☐ schemes, rules or methods of doing business
 - g. ☐ schemes, rules or methods of performing purely mental acts
 - h. ☐ schemes, rules or methods of playing games
 - i. ☐ methods for treatment of the human body by surgery or therapy
 - j. ☐ methods for treatment of the animal body by surgery or therapy
 - k. ☐ diagnostic methods practised on the human or animal body
 - l. ☐ mere presentation of information
 - m. ☐ computer programs for which this International Searching Authority is not equipped to search prior art
2. ☒ The failure of the following parts of the international application to comply with prescribed requirements prevents a meaningful search from being carried out:

☐ the description
 ☒ the claims
 ☐ the drawings
3. ☐ A meaningful search could not be carried out without the sequence listing; the applicant did not, within the prescribed time limit:

☐ furnish a sequence listing in the form of an Annex C/ST.25 text file, and such listing was not available to the International Searching Authority in a form and manner acceptable to it; or the sequence listing furnished did not comply with the standard provided for in Annex C of the Administrative Instructions.

☐ furnish a sequence listing on paper or in the form of an image file complying with the standard provided for in Annex C of the Administrative Instructions, and such listing was not available to the International Searching Authority in a form and manner acceptable to it; or the sequence listing furnished did not comply with the standard provided for in Annex C of the Administrative Instructions.

☐ pay the required late furnishing fee for the furnishing of a sequence listing in response to an invitation under Rule 13ter.1(a) or (b).
4. Further comments:

Name and mailing address of ISA/KR International Application Division Korean Intellectual Property Office 189 Cheongsu-ro, Seo-gu, Daejeon, 35208, Republic of Korea Facsimile No. +82-42-481-8578	Authorized officer CHIN, Sang Bum Telephone No. +82-42-481-8398
--	---

