November 30, 2015

Transparently Sharing Human Interface Devices Between Local And Remote Desktops

Shakeel Butt

Sharjeel Qureshi

TRANSPARENTLY SHARING HUMAN INTERFACE DEVICES (HIDS) BETWEEN LOCAL AND REMOTE DESKTOPS

Use of remote desktop computing is becoming more prevalent. In remote desktop computing, a user of a local machine 100 uses a local interface 105 executed at the local machine 100 to operate a remote desktop app 110 executing on the local machine 100 to control aspects of a remote desktop 200. The user controls the remote desktop 200 as if they were physically present at the remote desktop 200. Thus, the user controls the aspects of the remote desktop 200 virtually because they are only virtually present at the remote desktop 200. The user may have complete control of the remote desktop 200 or control only particular applications, etc. Remote desktop solutions may be retail, open source, or proprietary. The remote desktop 200 can be a virtual computer instance executing on a server, but may be a standalone computing device. The local machine 100 and the remote desktop 200 can be communicatively coupled via any number and/or type(s) of communication paths, devices and/or protocol(s) that can be collectively referred to as a network 300 (or cloud). The example local machine 100 of FIG. 1 may be implemented by the computer device 300 or the mobile computer device 350 of FIG. 3. Different local machines 100 may be implemented by different ones of the devices 300 and 350. The example remote desktop 200 of FIG. 1 may be implemented by the computer device 300 or the mobile computer device 350 of FIG. 3. Different remote desktops 200 may be implemented by different ones of the devices 300 and 350.

Conventional computing devices, as well as remote desktop solutions support human interface devices (HIDs), one of which is designated at reference numeral 115. However, any number and/or type(s) of HIDs may be present, used and/or implemented. Broadly speaking, a HID is any type of electronic or computer device that interacts directly with, and most often takes input from, users and may additionally deliver output to users. Simple HIDs include devices such as keyboards and mice. More complicated HIDs include joysticks, touch tablets, touch screens, trackpads, digitizers, styluses, three-dimensional (3D) mice, etc.

Some remote desktop solutions rely on generic drivers that may not fully support the features of more complicated HIDs. Moreover, the manner in which support for HIDs is implemented often does not provide an intuitive or user friendly experience. For example, when

a user uses a locally connected HID 115 to interact with the remote desktop 200, the HID 115 becomes logically disconnected from any application 120 they had been using on the local machine 100. When the user returns their attention to the local application 120 on the local machine 100, the HID 115 is disconnected from the remote desktop 200 and re-connected to the local machine 100. This disconnecting and connecting (or re-connecting) may decrease user satisfaction because of delays in HID responsiveness and/or the smoothness of control normally provided by HIDs. While some existing solutions allow for a HID to be connected to a remote desktop, or have its data re-directed to a remote desktop, any switch in attention between the local and remote desktop still requires disconnection and connection (or re-connection). To provide a more seamless, cohesive behavior of HIDs across local apps 120 and the remote desktop 200, the local machine 100 implements a device traffic router (DTR) 125 and a focus monitor (FM) 130. In the illustrated example, the DTR 125 is implemented within a kernel 135 of the local machine 100, and the FM 130 is implemented by the remote desktop app 110 within a user space 140.

The FM 130 monitors use of the HID 115 to determine with which application (i.e., the local app 120 or the remote desktop 200) the user is currently focused on, e.g., interacting with. The DTR 125 intercepts or seizes all data going to and/or coming from the HID 115. As shown in FIG. 2, when the FM 130 determines that the user is currently focused on, e.g., interacting with, the remote desktop 200 (block 205), the FM 130 notifies the DTR 125 to redirect the data coming from the HID 115 to the remote desktop 200 via the network 300 (block 210). The DTR 125 also routes data coming from the remote desktop 200 to the HID 115 as appropriate to the type of HID. When the FM 130 determines that the user is currently focused on, e.g., interacting with, the local app 120 (block 215), the FM 130 notifies the DTR 125 to redirect the data coming from the HID 115 to a conventional device driver 145 associated with the HID 115 (block 220). The DTR 125 also routes data coming from the local app 120 to the HID 115 as appropriate to the type of HID.

The device driver 145 is a driver for the HID 115 that is installed at the local machine 100. The device driver 145 enables the local app 120 to process data received from the HID 115 and/or to form data for the HID 115. Similarly, the remote desktop 200 has a device driver 205 for the HID 115 that is installed at the remote desktop 200. The device driver 205 enables the

remote desktop 200 to process data received from the HID 115 and/or to form data for the HID 115. In other words, both the local machine 100 and the remote desktop 200 have a locally installed device drive so that either the local machine 100 or the remote desktop 200 can process signals associated with the HID 115 depending on whether a user is focused on the local app 120 or the remote desktop 200.

In this manner, when the user is interacting with the remote desktop 200 through the remote desktop app 110, they will see and use the HID device 115 as if it was connected to the remote desktop 200, and when the user switches to the local app 120, they will see and use the HID 115 locally. In other words, data coming from the HID 115 is automatically and seamlessly directed between the user space 140 and the remote desktop 200 without requiring disconnection and re-connection of the HID 115. Any method may be used to determine whether the local app 120 or the remote desktop 200 is the object of focus. For example, the position of a mouse pointer on a screen may be tracked, and compared to window location(s).

Many operating systems provide or support device driver frameworks that can be used to implement the DTR 125.

While not shown, the DTR 125 and the FM 130 may be implemented elsewhere. For example, the DTR 125 may be implemented in the user space 140, although seamlessness may be degraded as a change in focus may still result in a disconnection and connection of the HID 115.

While a single remote desktop 200 is shown, the methods described may be used to route HID data between any number and/or combination(s) of remote desktops 200 and local apps 120. Moreover, the DTR 125 could route data for multiple HIDs, or each HID may have a separately implemented DTR. Further, the FM 130 may support multiple HIDs, and/or any number and/or combination(s) of remote desktops 200 and local apps 120.

Any use of descriptive terms such as seamless, smooth, cohesive, etc. merely represent a relative change in system performance that may be perceived by a user.

FIG. 3 shows an example of a generic computer device 300 and a generic mobile computer device 350 that may be used to implement the example local machines 100 and the example remote desktops 200 disclosed herein.. Computing device 300 is intended to represent various forms of digital computers, such as laptops, desktops, tablets, workstations, personal

digital assistants, televisions, servers, blade servers, mainframes, and other appropriate computing devices. Mobile computing device 350 is intended to represent various forms of mobile devices, such as personal digital assistants, cellular telephones, smart phones, and other similar computing devices. The components shown here, their connections and relationships, and their functions, are meant to be exemplary only, and are not meant to limit implementations of the inventions described and/or claimed in this document.

Computing device 300 may be used to implement the example local machines 100 and/or the example remote desktops 200 disclosed herein, and includes a processor 302, memory 304, a storage device 306, a high-speed interface 308 connecting to memory 304 and high-speed expansion ports 310, and a low speed interface 312 connecting to low speed bus 314 and storage device 306. The processor 302 can be a semiconductor-based processor. The memory 304 can be a semiconductor-based memory. Each of the components 302, 304, 306, 308, 310, and 312, are interconnected using various busses, and may be mounted on a common motherboard or in other manners as appropriate. The processor 302 can process instructions for execution within the computing device 300, including instructions stored in the memory 304 or on the storage device 306 to display graphical information for a GUI on an external input/output device, such as display 316 coupled to high speed interface 308. In other implementations, multiple processors and/or multiple buses may be used, as appropriate, along with multiple memories and types of memory. Also, multiple computing devices 300 may be connected, with each device providing portions of the necessary operations (e.g., as a server bank, a group of blade servers, or a multiprocessor system).

The memory 304 stores information within the computing device 300. In one implementation, the memory 304 is a volatile memory unit or units. In another implementation, the memory 304 is a non-volatile memory unit or units. The memory 304 may also be another form of computer-readable medium, such as a magnetic or optical disk.

The storage device 306 is capable of providing mass storage for the computing device 300. In one implementation, the storage device 306 may be or contain a computer-readable medium, such as a floppy disk device, a hard disk device, an optical disk device, or a tape device, a flash memory or other similar solid state memory device, or an array of devices, including devices in a storage area network or other configurations. A computer program product can be

tangibly embodied in an information carrier. The computer program product may also contain instructions that, when executed, perform one or more methods, such as those described above. The information carrier is a computer- or machine-readable medium, such as the memory 304, the storage device 306, or memory on processor 302.

The high speed controller 308 manages bandwidth-intensive operations for the computing device 300, while the low speed controller 312 manages lower bandwidth-intensive operations. Such allocation of functions is exemplary only. In one implementation, the high-speed controller 308 is coupled to memory 304, display 316 (e.g., through a graphics processor or accelerator), and to high-speed expansion ports 310, which may accept various expansion cards (not shown). In the implementation, low-speed controller 312 is coupled to storage device 306 and low-speed expansion port 314. The low-speed expansion port, which may include various communication ports (e.g., USB, Bluetooth, Ethernet, wireless Ethernet) may be coupled to one or more input/output devices, such as a keyboard, a pointing device, a scanner, or a networking device such as a switch or router, e.g., through a network adapter.

The computing device 300 may be implemented in a number of different forms, as shown in the figure. For example, it may be implemented as a standard server 320, or multiple times in a group of such servers. It may also be implemented as part of a rack server system 324. In addition, it may be implemented in a personal computer such as a laptop computer 322. Alternatively, components from computing device 300 may be combined with other components in a mobile device (not shown), such as device 350. Each of such devices may contain one or more of computing device 300, 350, and an entire system may be made up of multiple computing devices 300, 350 communicating with each other.

Computing device 350 may be used to implement the example local machines 100 and/or the example remote desktops 200 disclosed herein, and includes a processor 352, memory 364, an input/output device such as a display 354, a communication interface 366, and a transceiver 368, among other components. The device 350 may also be provided with a storage device, such as a microdrive or other device, to provide additional storage. Each of the components 350, 352, 364, 354, 366, and 368, are interconnected using various buses, and several of the components may be mounted on a common motherboard or in other manners as appropriate.

The processor 352 can execute instructions within the computing device 350, including instructions stored in the memory 364. The processor may be implemented as a chipset of chips that include separate and multiple analog and digital processors. The processor may provide, for example, for coordination of the other components of the device 350, such as control of user interfaces, applications run by device 350, and wireless communication by device 350.

Processor 352 may communicate with a user through control interface 358 and display interface 356 coupled to a display 354. The display 354 may be, for example, a TFT LCD (ThinFilm-Transistor Liquid Crystal Display) or an OLED (Organic Light Emitting Diode) display, or other appropriate display technology. The display interface 356 may comprise appropriate circuitry for driving the display 354 to present graphical and other information to a user. The control interface 358 may receive commands from a user and convert them for submission to the processor 352. In addition, an external interface 362 may be provide in communication with processor 352, so as to enable near area communication of device 350 with other devices. External interface 362 may provide, for example, for wired communication in some implementations, or for wireless communication in other implementations, and multiple interfaces may also be used.

The memory 364 stores information within the computing device 350. The memory 364 can be implemented as one or more of a computer-readable medium or media, a volatile memory unit or units, or a non-volatile memory unit or units. Expansion memory 374 may also be provided and connected to device 350 through expansion interface 372, which may include, for example, a SIMM (Single In Line Memory Module) card interface. Such expansion memory 374 may provide extra storage space for device 350, or may also store applications or other information for device 350. Specifically, expansion memory 374 may include instructions to carry out or supplement the processes described above, and may include secure information also. Thus, for example, expansion memory 374 may be provide as a security module for device 350, and may be programmed with instructions that permit secure use of device 350. In addition, secure applications may be provided via the SIMM cards, along with additional information, such as placing identifying information on the SIMM card in a non-hackable manner.

The memory may include, for example, flash memory and/or NVRAM memory, as discussed below. In one implementation, a computer program product is tangibly embodied in an

information carrier. The computer program product contains instructions that, when executed, perform one or more methods, such as those described above. The information carrier is a computer- or machine-readable medium, such as the memory 364, expansion memory 374, or memory on processor 352, that may be received, for example, over transceiver 368 or external interface 362.

Device 350 may communicate wirelessly through communication interface 366, which may include digital signal processing circuitry where necessary. Communication interface 366 may provide for communications under various modes or protocols, such as GSM voice calls, SMS, EMS, or MMS messaging, CDMA, TDMA, PDC, WCDMA, CDMA2000, or GPRS, among others. Such communication may occur, for example, through radio-frequency transceiver 368. In addition, short-range communication may occur, such as using a Bluetooth, Wi-Fi, or other such transceiver (not shown). In addition, GPS (Global Positioning System) receiver module 370 may provide additional navigation- and location-related wireless data to device 350, which may be used as appropriate by applications running on device 350.

Device 350 may also communicate audibly using audio codec 360, which may receive spoken information from a user and convert it to usable digital information. Audio codec 360 may likewise generate audible sound for a user, such as through a speaker, e.g., in a handset of device 350. Such sound may include sound from voice telephone calls, may include recorded sound (e.g., voice messages, music files, etc.) and may also include sound generated by applications operating on device 350.

The computing device 350 may be implemented in a number of different forms, as shown in the figure. For example, it may be implemented as a cellular telephone 380. It may also be implemented as part of a smart phone 382, personal digital assistant, or other similar mobile device.

Various implementations of the systems and techniques described here can be realized in digital electronic circuitry, integrated circuitry, specially designed ASICs (application specific integrated circuits), computer hardware, firmware, software, and/or combinations thereof. These various implementations can include implementation in one or more computer programs that are executable and/or interpretable on a programmable system including at least one programmable processor, which may be special or general purpose, coupled to receive data and instructions

from, and to transmit data and instructions to, a storage system, at least one input device, and at least one output device.

These computer programs (also known as programs, software, software applications or code) include machine instructions for a programmable processor, and can be implemented in a high-level procedural and/or object-oriented programming language, and/or in assembly/machine language. As used herein, the terms "machine-readable medium" "computer-readable medium" refers to any computer program product, apparatus and/or device (e.g., magnetic discs, optical disks, memory, Programmable Logic Devices (PLDs)) used to provide machine instructions and/or data to a programmable processor, including a machine-readable medium that receives machine instructions as a machine-readable signal. The term "machine-readable signal" refers to any signal used to provide machine instructions and/or data to a programmable processor.

To provide for interaction with a user, the systems and techniques described here can be implemented on a computer having a display device (e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor) for displaying information to the user and a keyboard and a pointing device (e.g., a mouse or a trackball) by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback (e.g., visual feedback, auditory feedback, or tactile feedback); and input from the user can be received in any form, including acoustic, speech, or tactile input.

The systems and techniques described here can be implemented in a computing system that includes a back end component (e.g., as a data server), or that includes a middleware component (e.g., an application server), or that includes a front end component (e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an implementation of the systems and techniques described here), or any combination of such back end, middleware, or front end components. The components of the system can be interconnected by any form or medium of digital data communication (e.g., a communication network). Examples of communication networks include a local area network ("LAN"), a wide area network ("WAN"), and the Internet.

The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The

relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

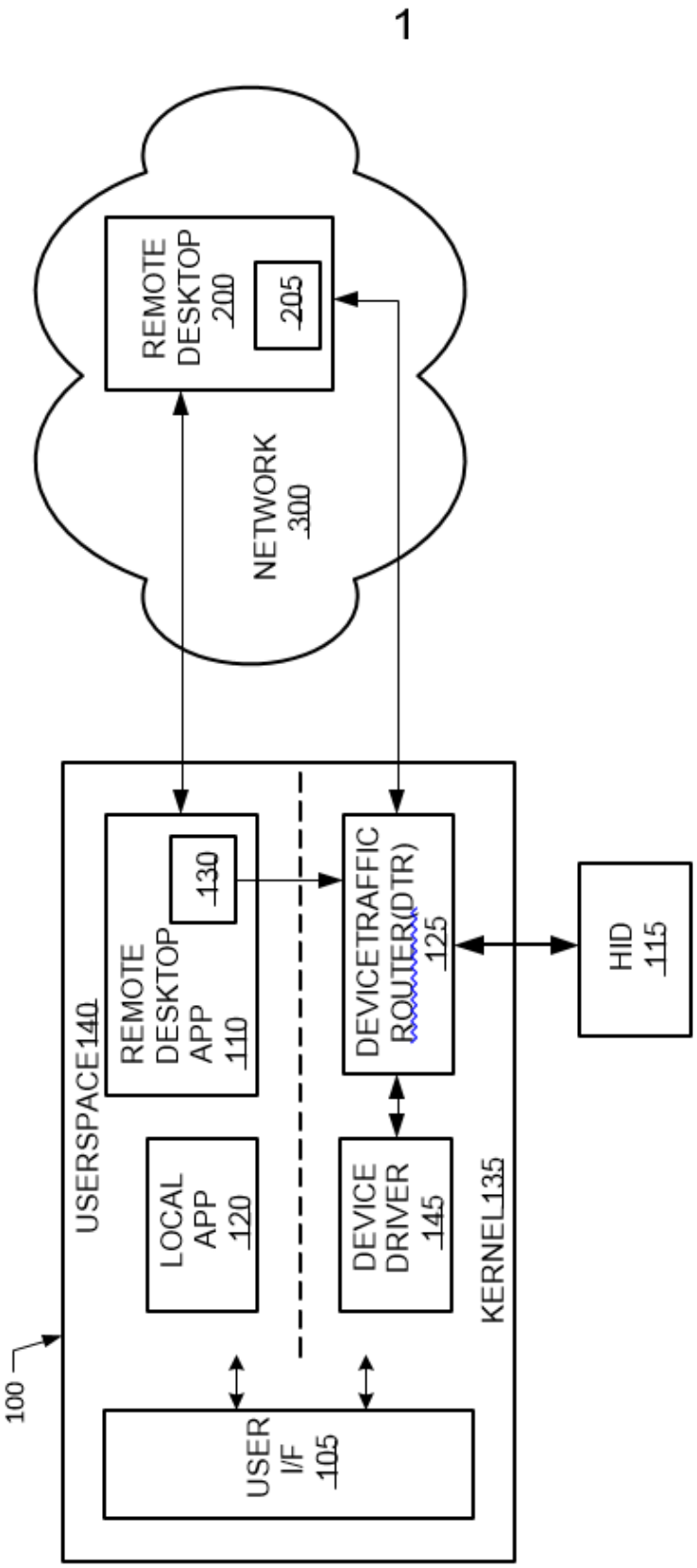
1
REMOTE DESKTOP 200
205
NETWORK 300
USERSPACE 140
REMOTE DESKTOP APP 110
130
LOCAL APP 120
DEVICE DRIVER 145
DEVICETRAFFIC ROUTER(DTR) 125
HID 115
KERNEL 135
USER I/F 105
100

FIG.1

2

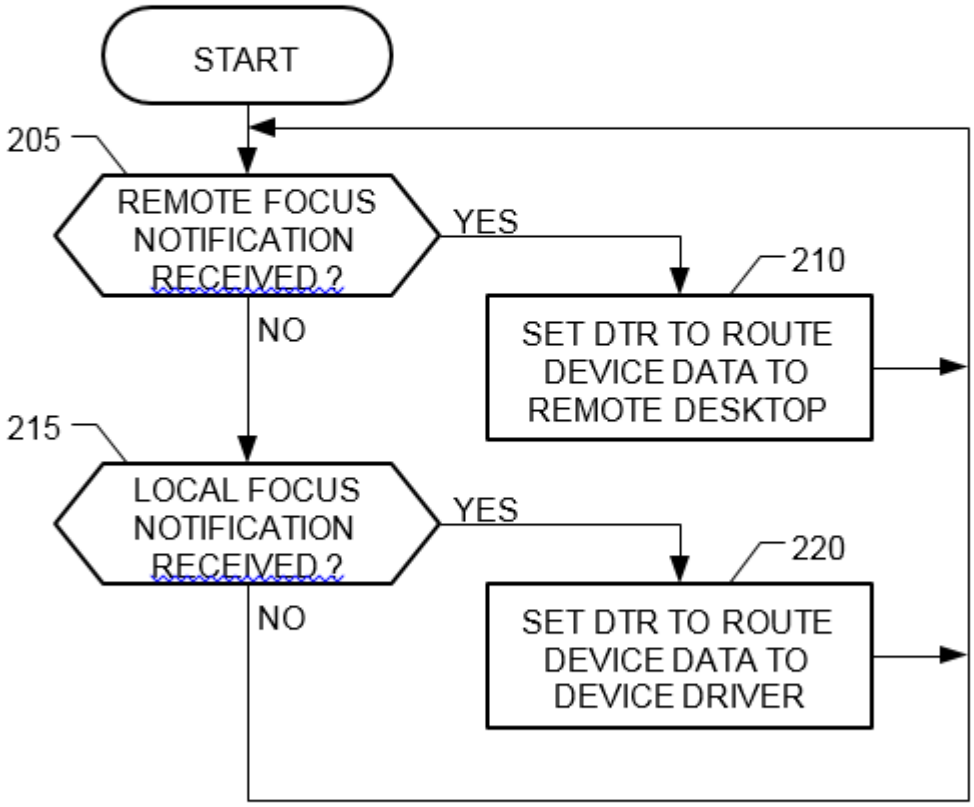
START
205
REMOTE FOCUS NOTIFICATION RECEIVED?
YES
210
SET DTR TO ROUTE DEVICE DATA TO REMOTE DESKTOP
NO
215
LOCAL FOCUS NOTIFICATION RECEIVED?
YES
220
SET DTR TO ROUTE DEVICE DATA TO DEVICE DRIVER
NO

FIG. 2

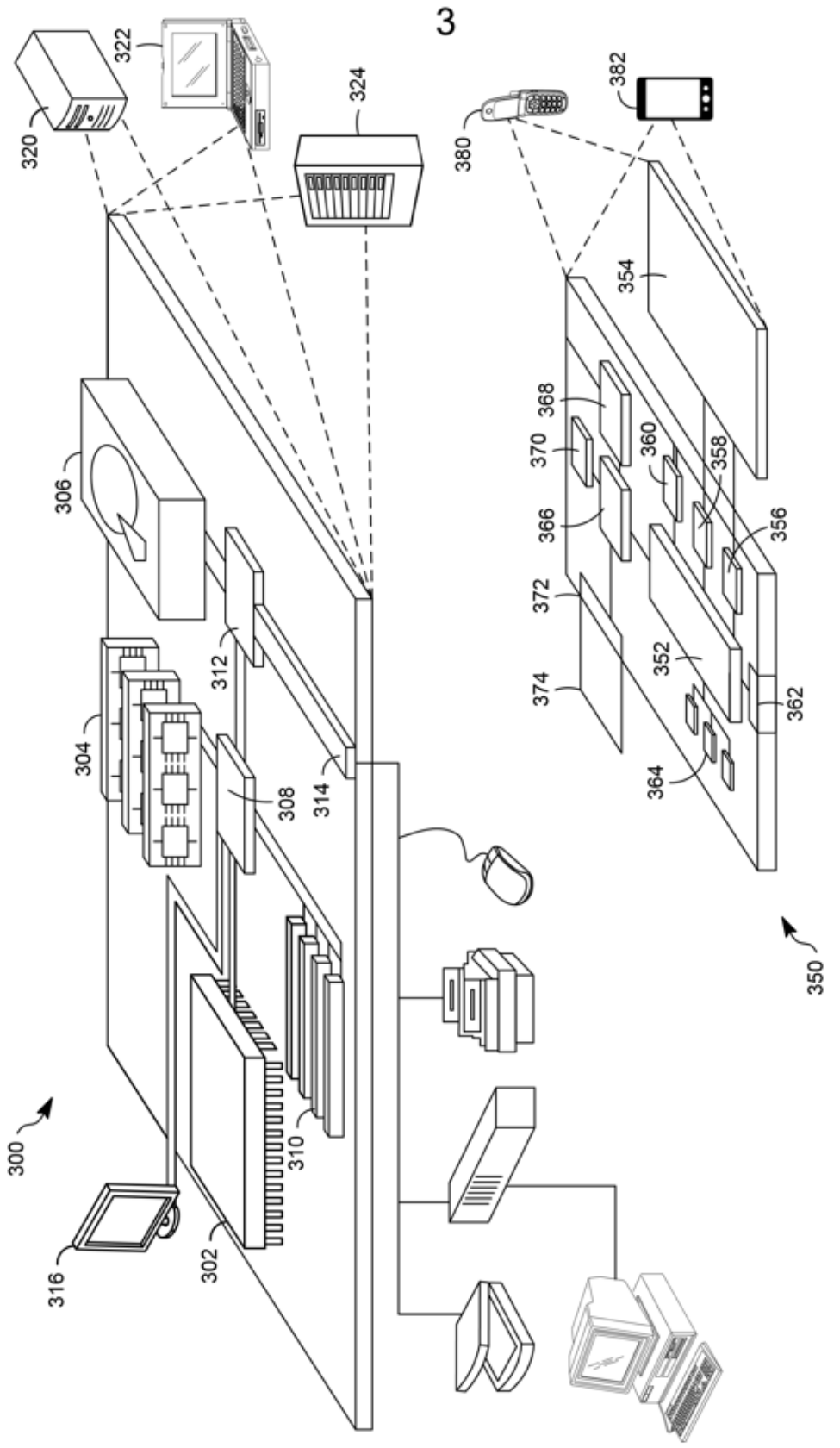
3
320
322
324
380
382
354
368
370
360
358
366
356
372
352
374
362
364
306
312
304
308
314
310
316
302
300
350

FIG. 3