US 20240289138A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2024/0289138 A1**

**BOHDAN et al.** (43) **Pub. Date: Aug. 29, 2024**

(54) **SYSTEM AND METHOD FOR MOBILE AND STATIONARY COMPUTING DEVICE INTERWORKING**

(71) Applicant: **SIMPLEWAY TECHNOLOGIES LTD.**, Dublin (IE)

(72) Inventors: **Artem BOHDAN**, Berlin (DE); **Ievgen KRUTOV**, Kryvyi Rih (UA)

(21) Appl. No.: **18/569,933**

(22) PCT Filed: **Jun. 14, 2022**

(86) PCT No.: **PCT/IB2022/055501**

§ 371 (c)(1),
(2) Date: **Dec. 13, 2023**

**Related U.S. Application Data**

(60) Provisional application No. 63/211,169, filed on Jun. 16, 2021.

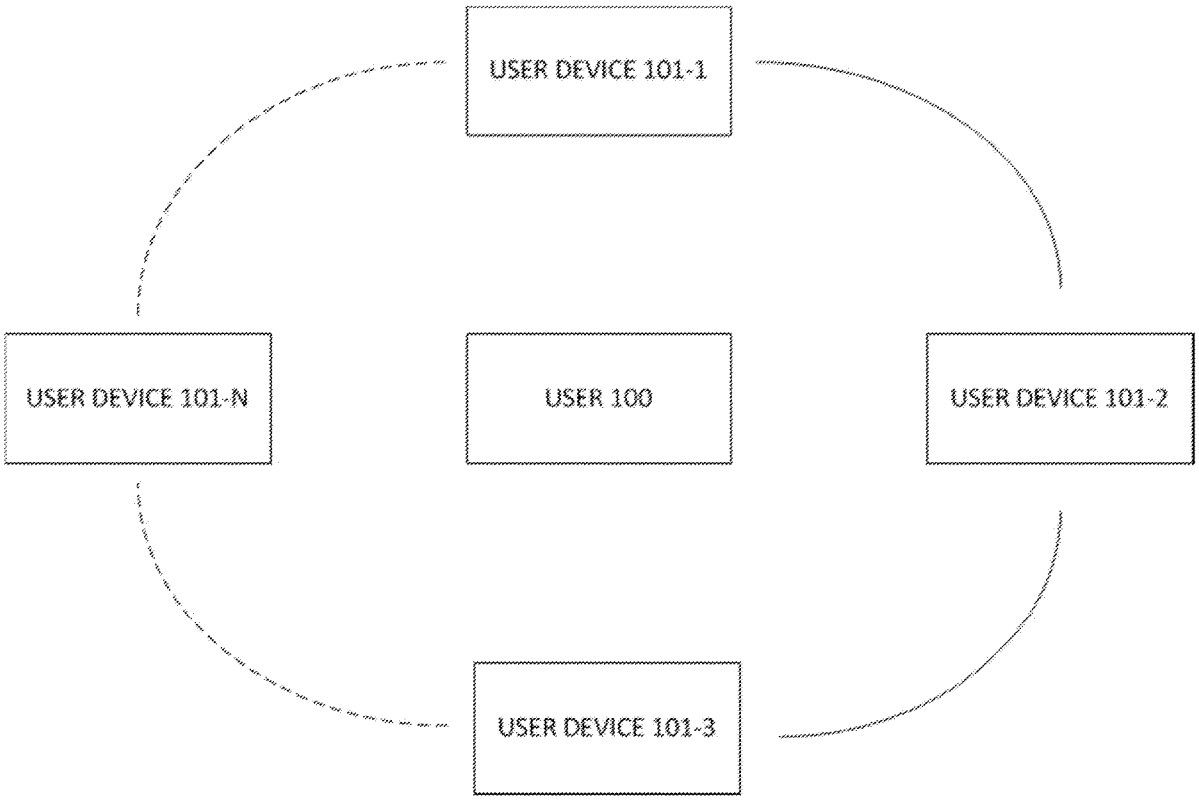
**Publication Classification**

(51) **Int. Cl.**
***G06F 9/4401*** (2006.01)

(52) **U.S. Cl.**
CPC ..... ***G06F 9/441*** (2013.01); ***G06F 9/4418*** (2013.01)

(57) **ABSTRACT**

A system for a mobile computing device and a stationary computing device associated with a user to interwork, wherein: the mobile computing device comprises a device interoperability system having a communications module, wherein a first connection is established between the stationary computing device and the communications module: storage coupled to said communications module, wherein the storage stores an operating system, one or more programs, data associated with the user, further wherein the operating system is booted by the stationary computing device via the first connection, the operating system runs on the stationary computing device, and the operating system uses one or more processing capabilities of the stationary computing device for operation; and one or more processors to support said device interoperability system.



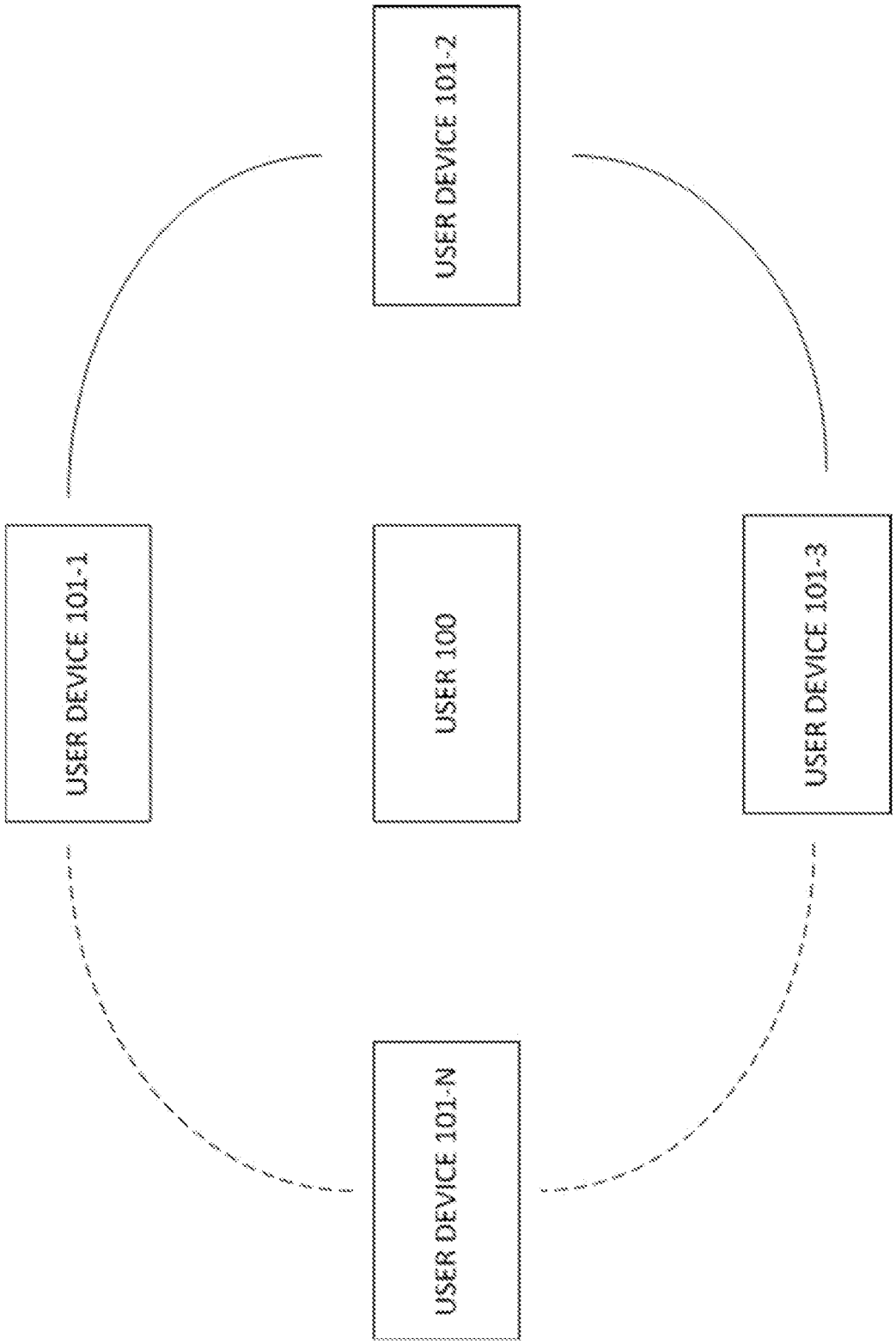

USER DEVICE 101-2
USER DEVICE 101-1
USER 100
USER DEVICE 101-3
USER DEVICE 101-N

FIG. 1

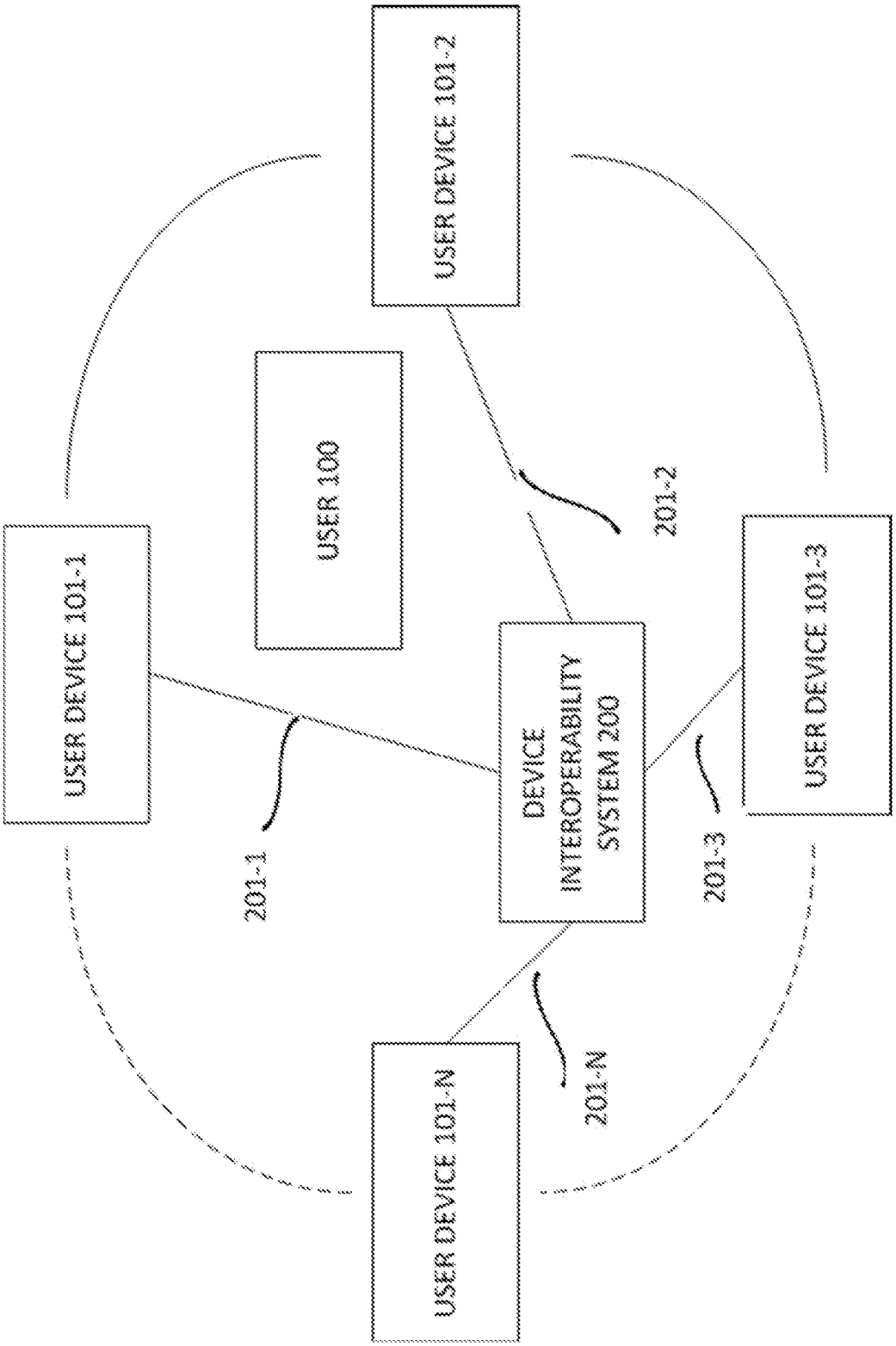

USER DEVICE 101-2
USER 100
USER DEVICE 101-1
DEVICE INTEROPERABILITY SYSTEM 200
USER DEVICE 101-3
USER DEVICE 101-N
201-1
201-2
201-3
201-N

FIG. 2

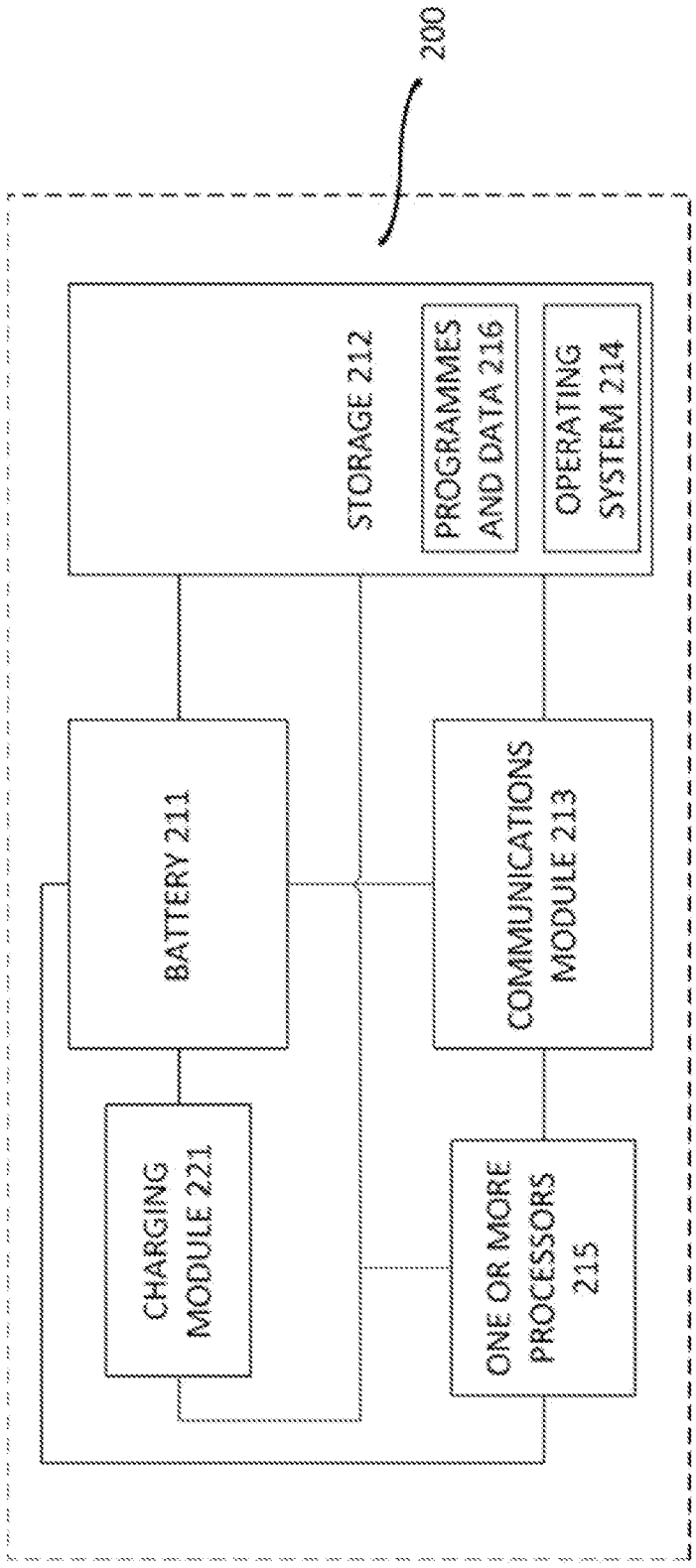

200
STORAGE 212
PROGRAMMES AND DATA 216
OPERATING SYSTEM 214
BATTERY 211
COMMUNICATIONS MODULE 213
CHARGING MODULE 221
ONE OR MORE PROCESSORS 215

FIG. 2B

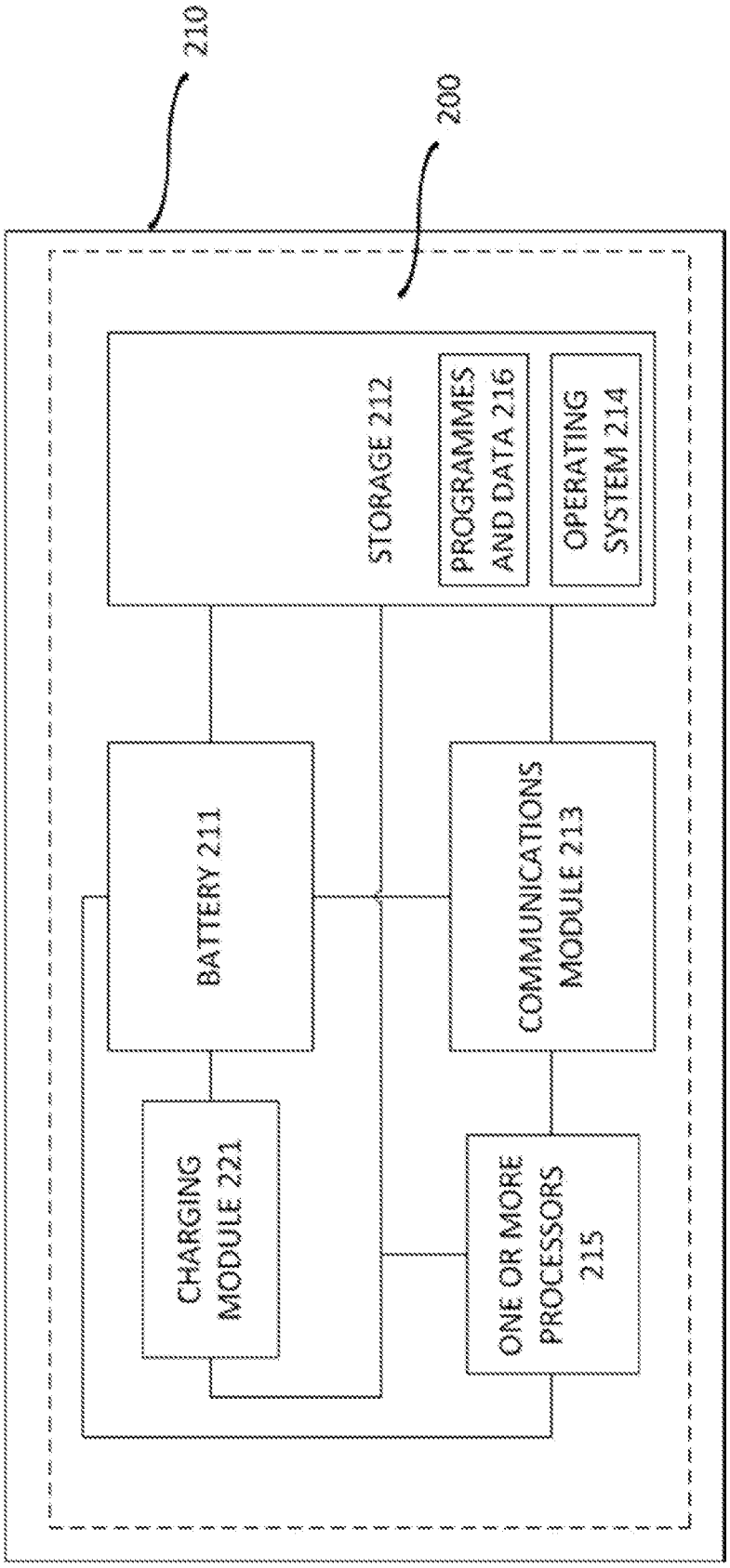

210
200
STORAGE 212
PROGRAMMES AND DATA 216
OPERATING SYSTEM 214
BATTERY 211
COMMUNICATIONS MODULE 213
CHARGING MODULE 221
ONE OR MORE PROCESSORS 215

FIG. 2C

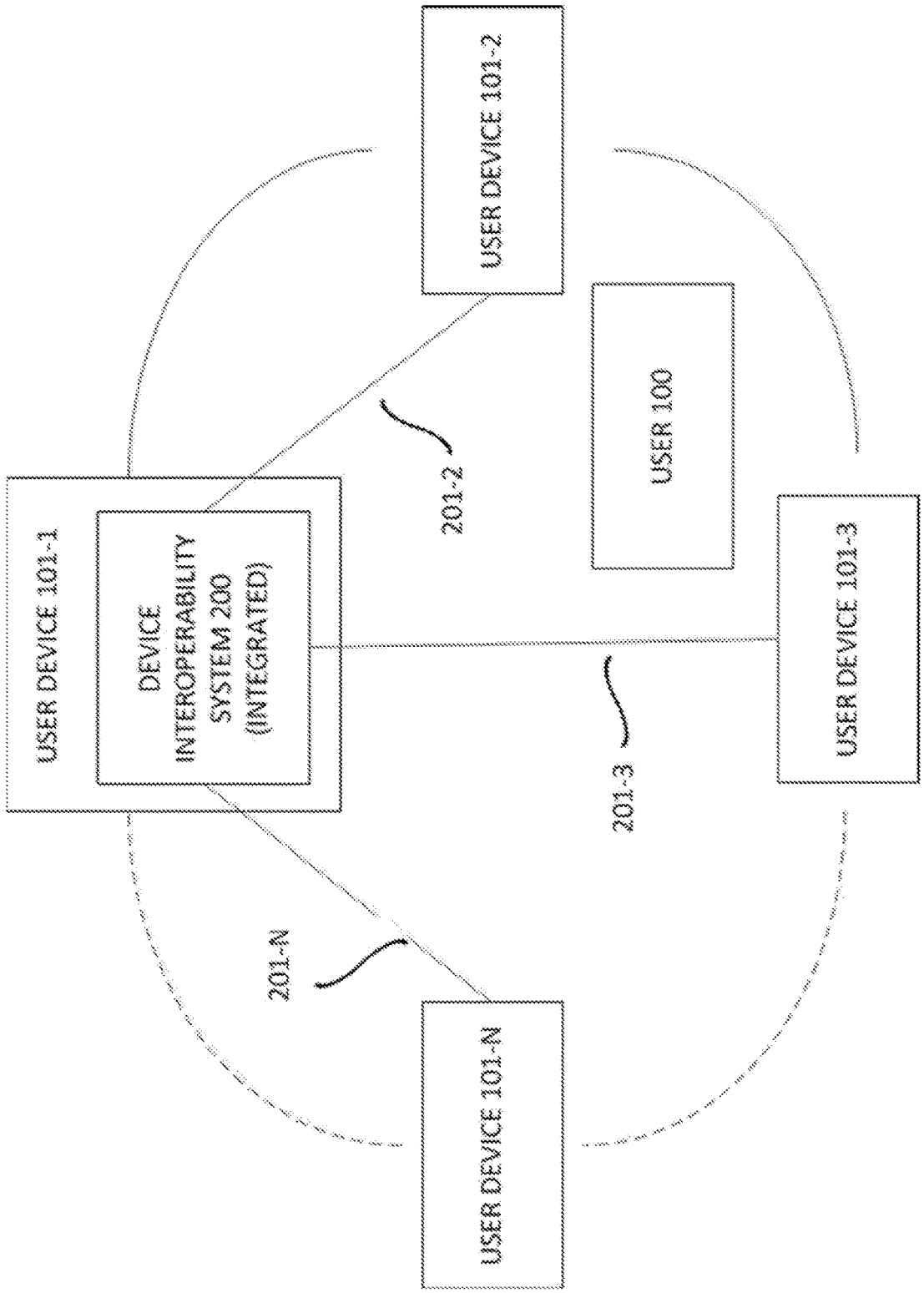

USER DEVICE 101-1
DEVICE INTEROPERABILITY SYSTEM 200 (INTEGRATED)
USER DEVICE 101-2
USER 100
USER DEVICE 101-3
USER DEVICE 101-N
201-2
201-3
201-N

FIG. 2D

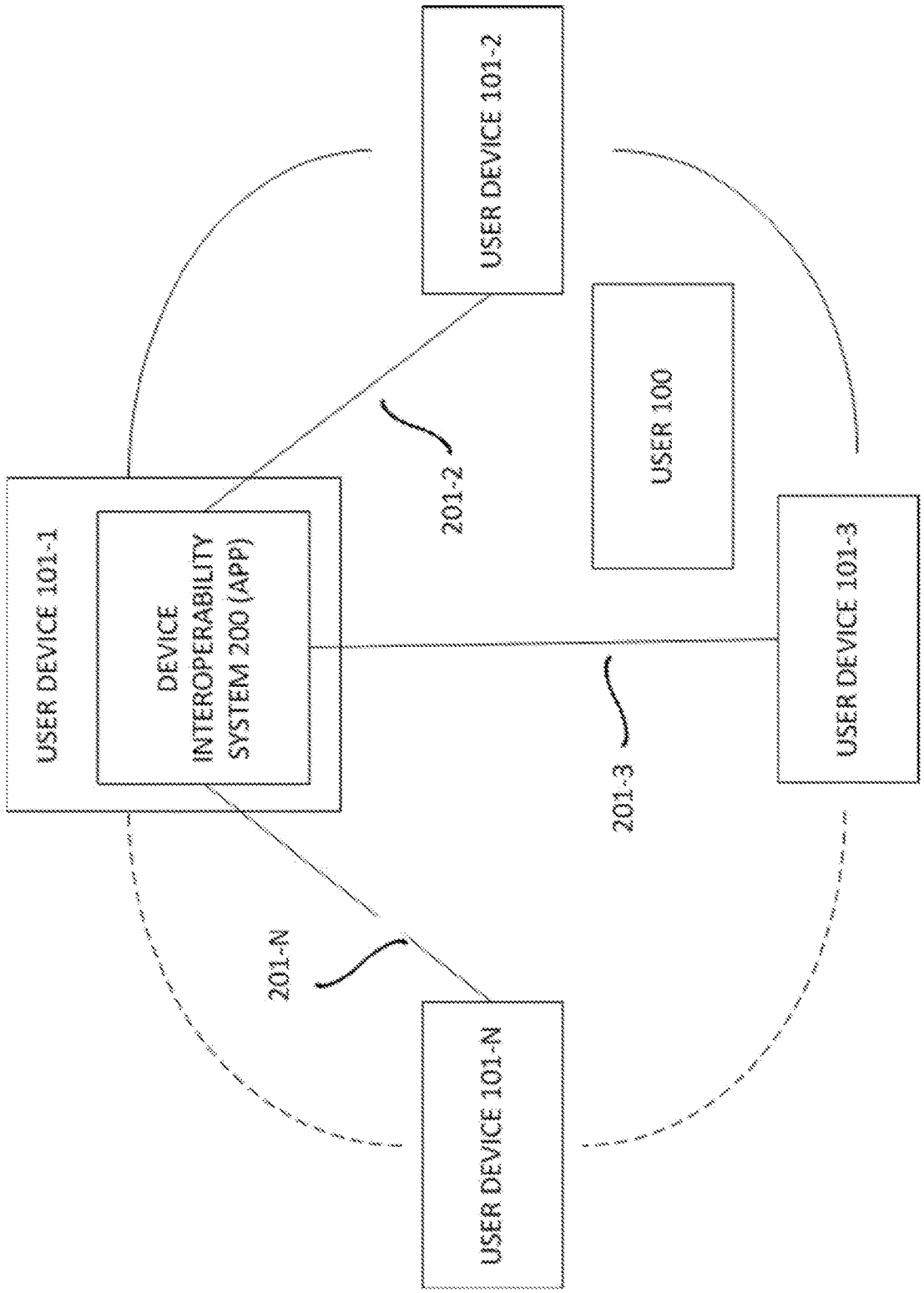

USER DEVICE 101-2
USER 100
USER DEVICE 101-1
DEVICE INTEROPERABILITY SYSTEM 200 (APP)
201-2
USER DEVICE 101-3
201-3
201-N
USER DEVICE 101-N

FIG. 2E

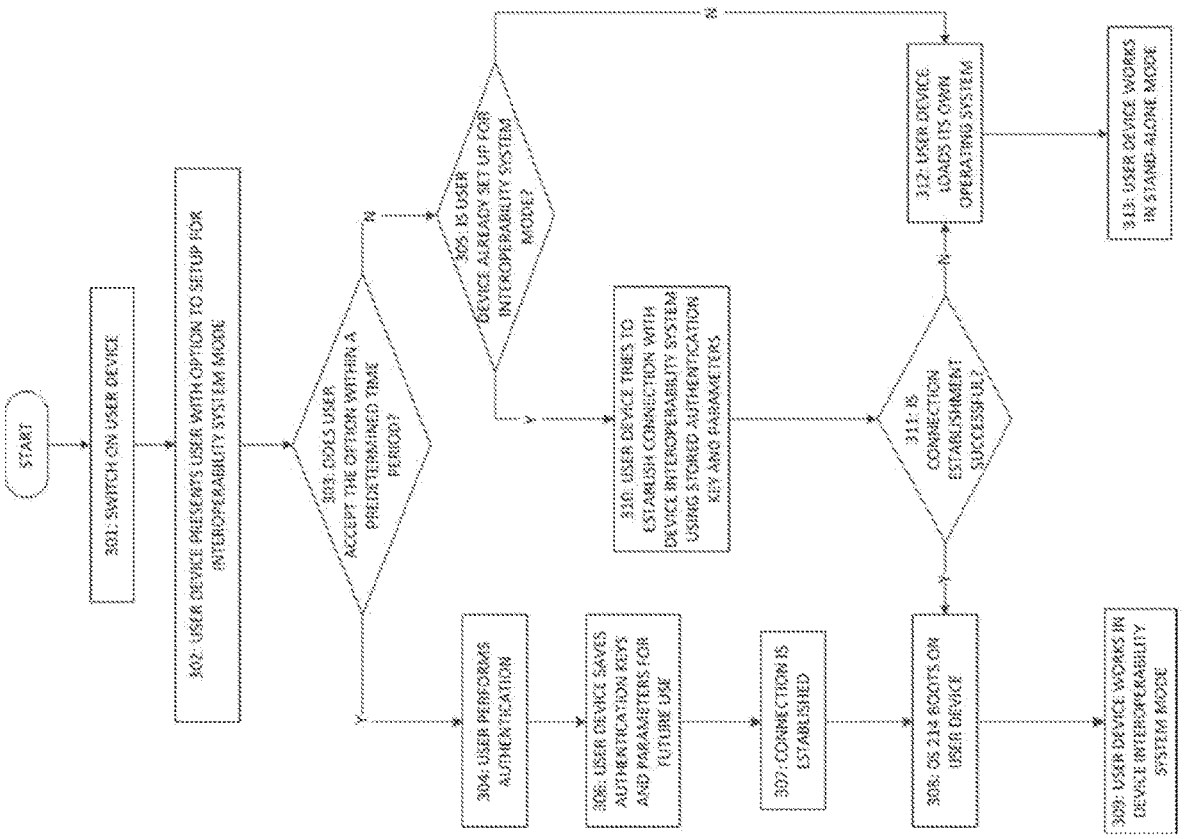

START
301: SWITCH ON USER DEVICE
302: USER DEVICE PRESENTS USER WITH OPTION TO SETUP FOR INTEROPERABILITY SYSTEM MODE
303: DOES USER ACCEPT THE OPTION WITHIN A PREDETERMINED TIME PERIOD?
305: IS USER DEVICE ALREADY SET UP FOR INTEROPERABILITY SYSTEM MODE?
310: USER DEVICE TRIES TO ESTABLISH CONNECTION WITH DEVICE INTEROPERABILITY SYSTEM USING STORED AUTHENTICATION KEY AND PARAMETERS
311: IS CONNECTION ESTABLISHMENT SUCCESSFUL?
312: USER DEVICE LOADS ITS OWN OPERATING SYSTEM
313: USER DEVICE WORKS IN STAND-ALONE MODE
304: USER PERFORMS AUTHENTICATION
306: USER DEVICE SAVES AUTHENTICATION KEYS AND PARAMETERS FOR FUTURE USE
307: CONNECTION IS ESTABLISHED
308: OS 212 BOOTS ON USER DEVICE
309: USER DEVICE WORKS IN INTEROPERABILITY SYSTEM MODE
Y
N

FIG. 3

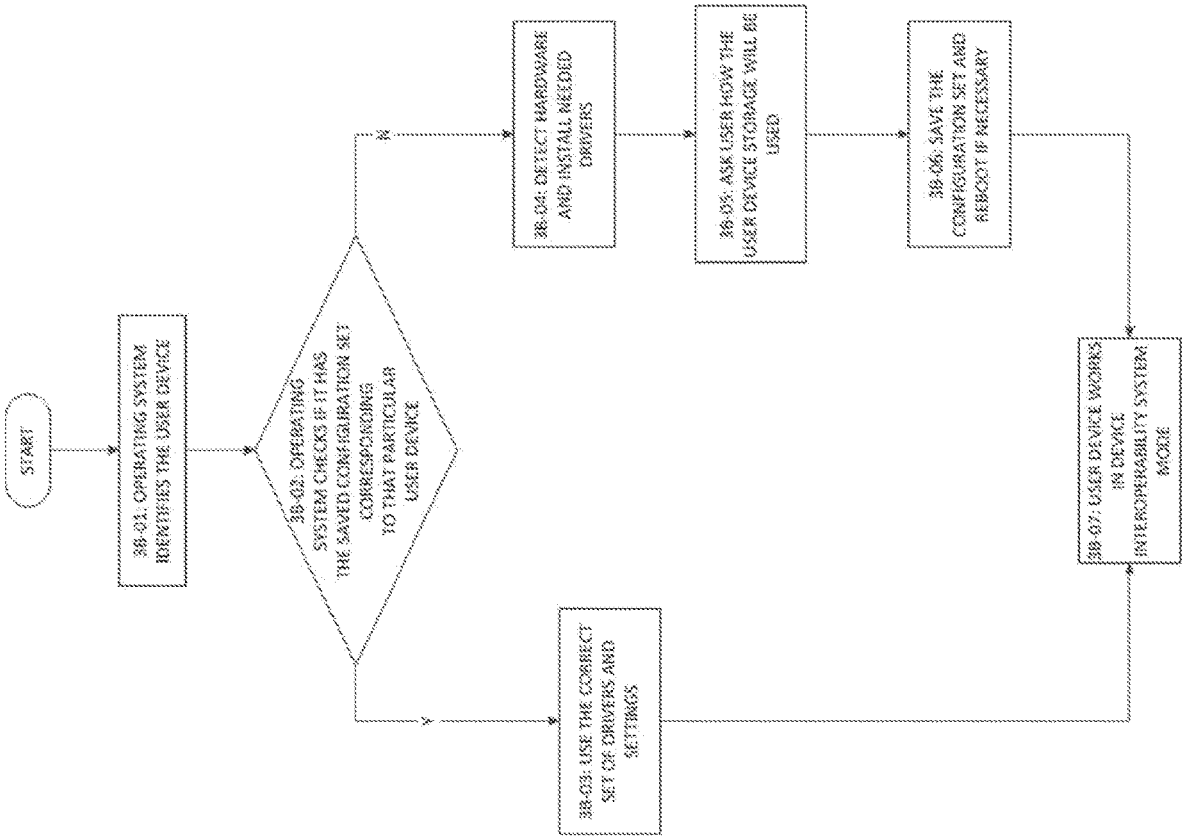

START
3B-01: OPERATING SYSTEM IDENTIFIES THE USER DEVICE
3B-02: OPERATING SYSTEM CHECKS IF IT HAS THE SAVED CONFIGURATION SET CORRESPONDING TO THAT PARTICULAR USER DEVICE
N
3B-04: DETECT HARDWARE AND INSTALL NEEDED DRIVERS
3B-05: ASK USER HOW THE USER DEVICE STORAGE WILL BE USED
3B-06: SAVE THE CONFIGURATION SET AND REBOOT IF NECESSARY
Y
3B-03: USE THE CORRECT SET OF DRIVERS AND SETTINGS
3B-07: USER DEVICE WORKS IN DEVICE INTEROPERABILITY SYSTEM MODE

FIG. 3B

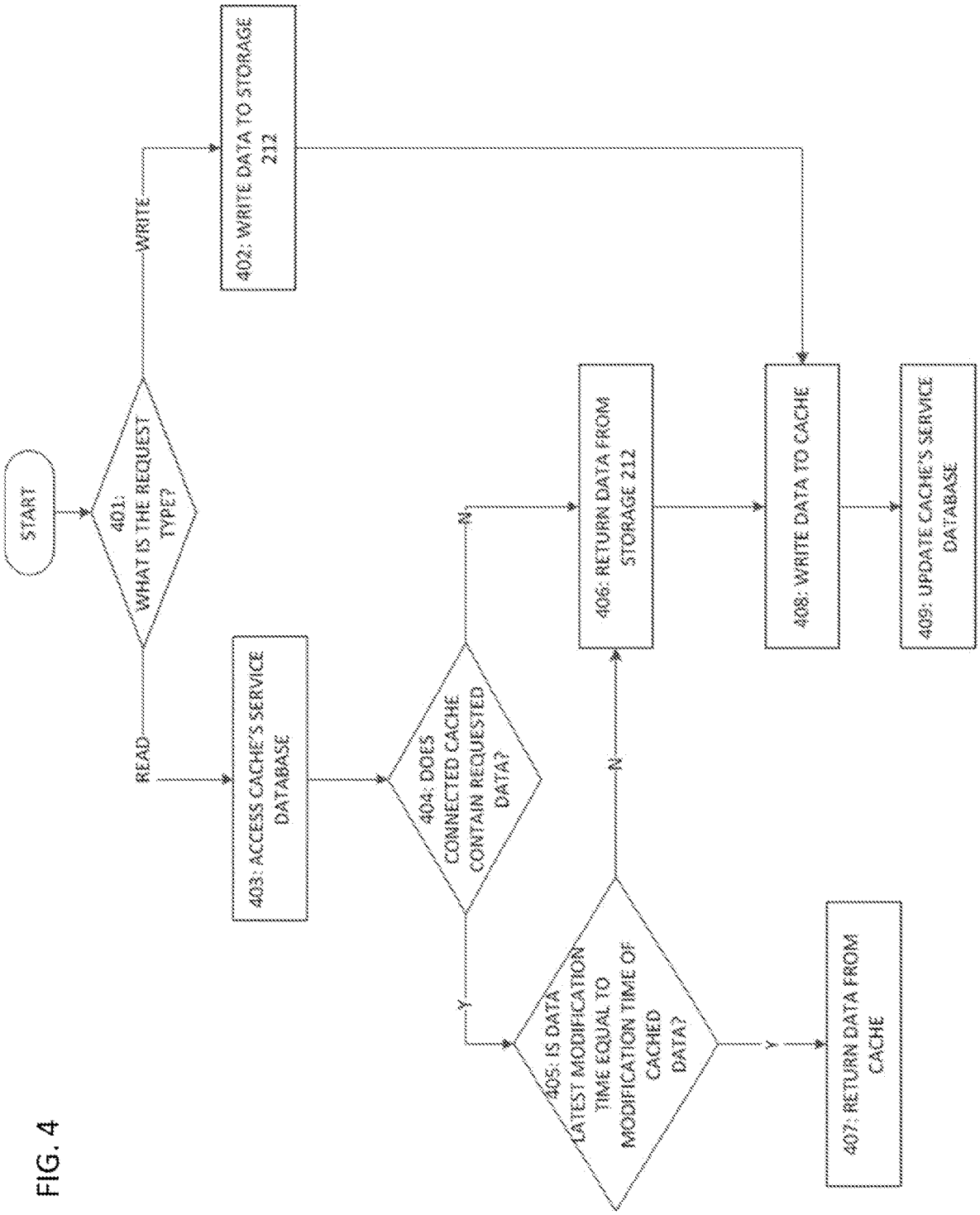

START
401: WHAT IS THE REQUEST TYPE?
WRITE
READ
402: WRITE DATA TO STORAGE 212
403: ACCESS CACHE'S SERVICE DATABASE
404: DOES CONNECTED CACHE CONTAIN REQUESTED DATA?
N
Y
405: IS DATA LATEST MODIFICATION TIME EQUAL TO MODIFICATION TIME OF CACHED DATA?
N
Y
406: RETURN DATA FROM STORAGE 212
407: RETURN DATA FROM CACHE
408: WRITE DATA TO CACHE
409: UPDATE CACHE'S SERVICE DATABASE

FIG. 4

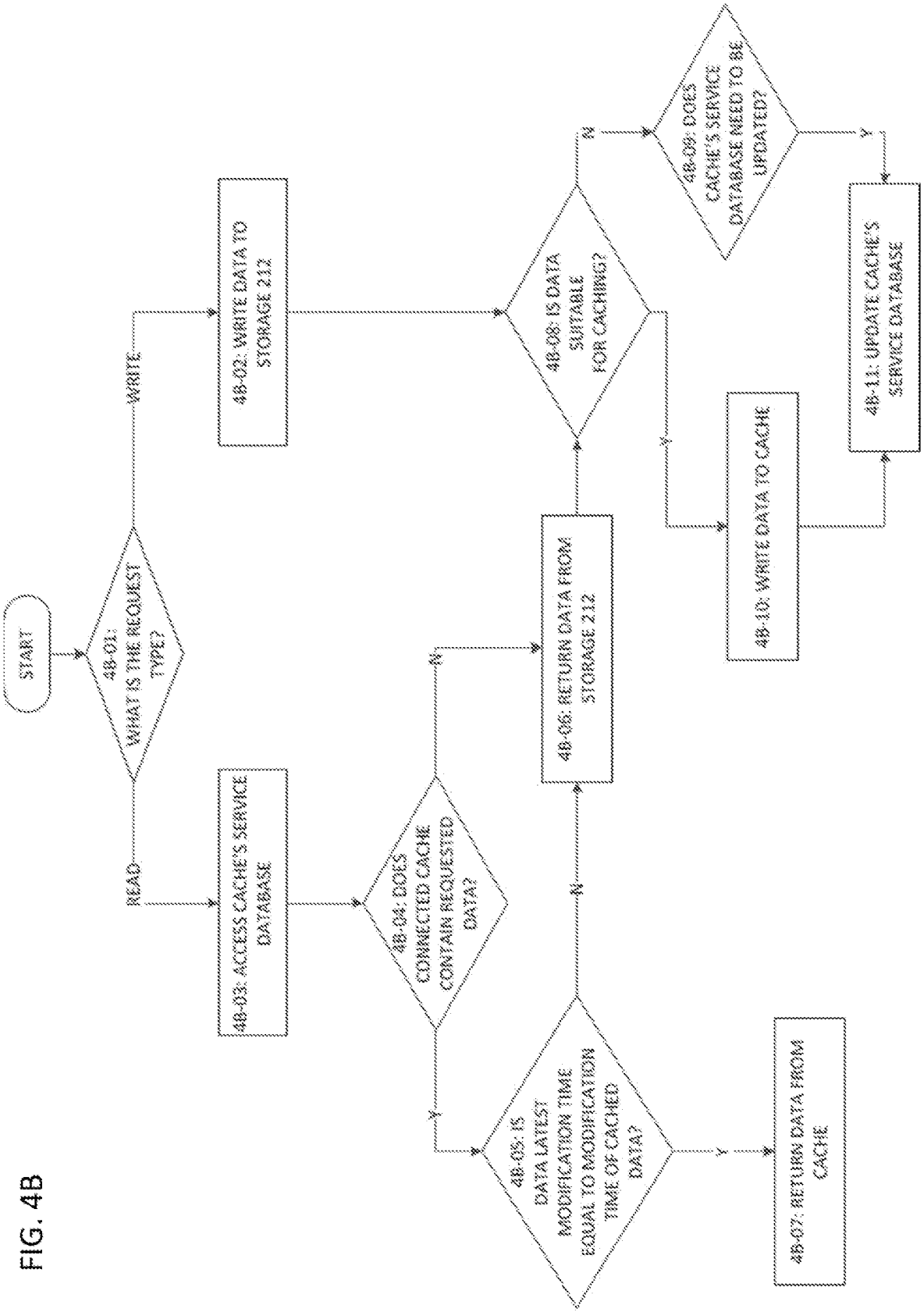

START
4B-01: WHAT IS THE REQUEST TYPE?
WRITE
READ
4B-02: WRITE DATA TO STORAGE 212
4B-03: ACCESS CACHE'S SERVICE DATABASE
4B-04: DOES CONNECTED CACHE CONTAIN REQUESTED DATA?
Y
N
4B-05: IS DATA LATEST MODIFICATION TIME EQUAL TO MODIFICATION TIME OF CACHED DATA?
Y
N
4B-06: RETURN DATA FROM STORAGE 212
4B-07: RETURN DATA FROM CACHE
4B-08: IS DATA SUITABLE FOR CACHING?
Y
N
4B-09: DOES CACHE'S SERVICE DATABASE NEED TO BE UPDATED?
Y
N
4B-10: WRITE DATA TO CACHE
4B-11: UPDATE CACHE'S SERVICE DATABASE

FIG. 4B

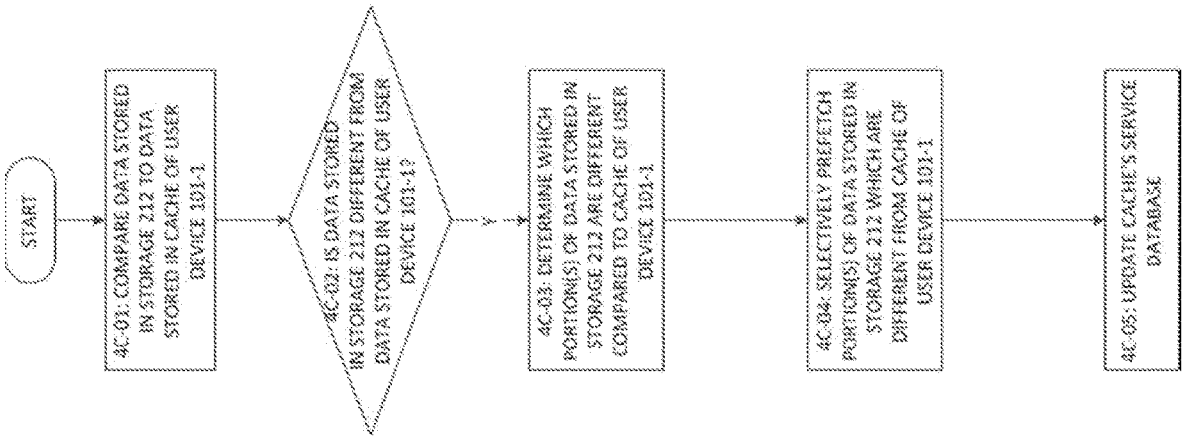

START
4C-01: COMPARE DATA STORED IN STORAGE 212 TO DATA STORED IN CACHE OF USER DEVICE 101-1
4C-02: IS DATA STORED IN STORAGE 212 DIFFERENT FROM DATA STORED IN CACHE OF USER DEVICE 101-1?
Y
4C-03: DETERMINE WHICH PORTION(S) OF DATA STORED IN STORAGE 212 ARE DIFFERENT COMPARED TO CACHE OF USER DEVICE 101-1
4C-04: SELECTIVELY PREFETCH PORTION(S) OF DATA STORED IN STORAGE 212 WHICH ARE DIFFERENT FROM CACHE OF USER DEVICE 101-1
4C-05: UPDATE CACHE'S SERVICE DATABASE

FIG. 4C

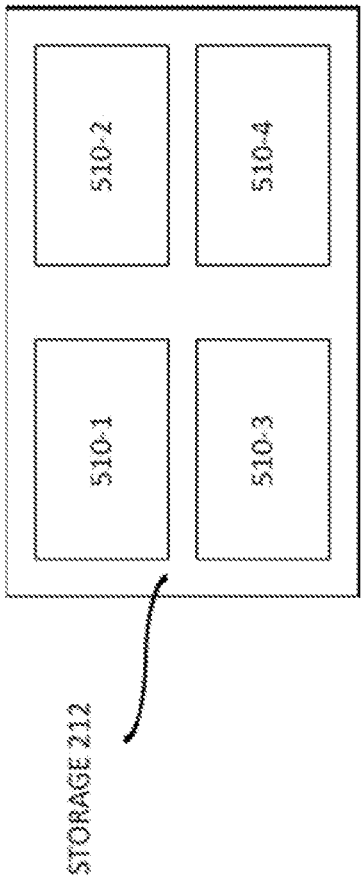



FIG. 5

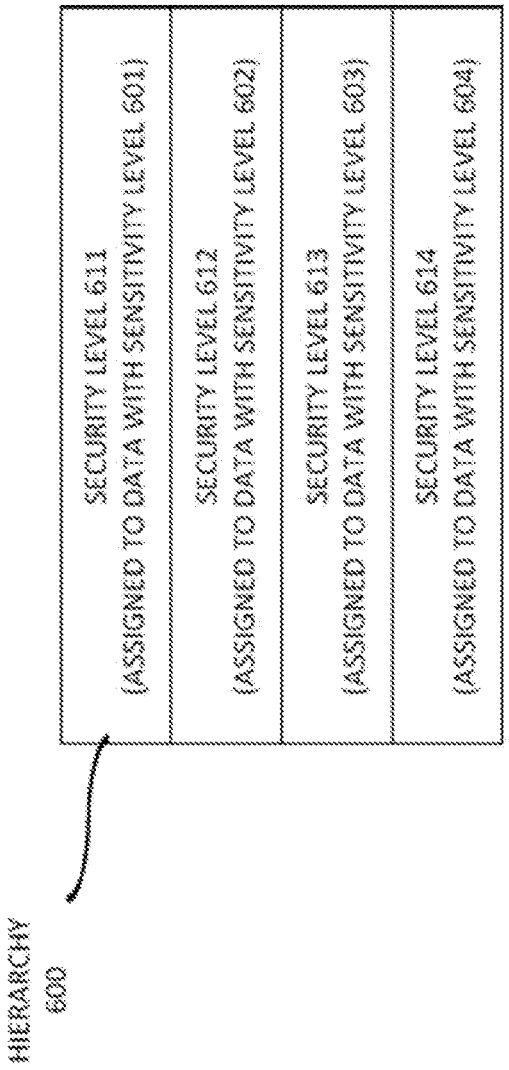
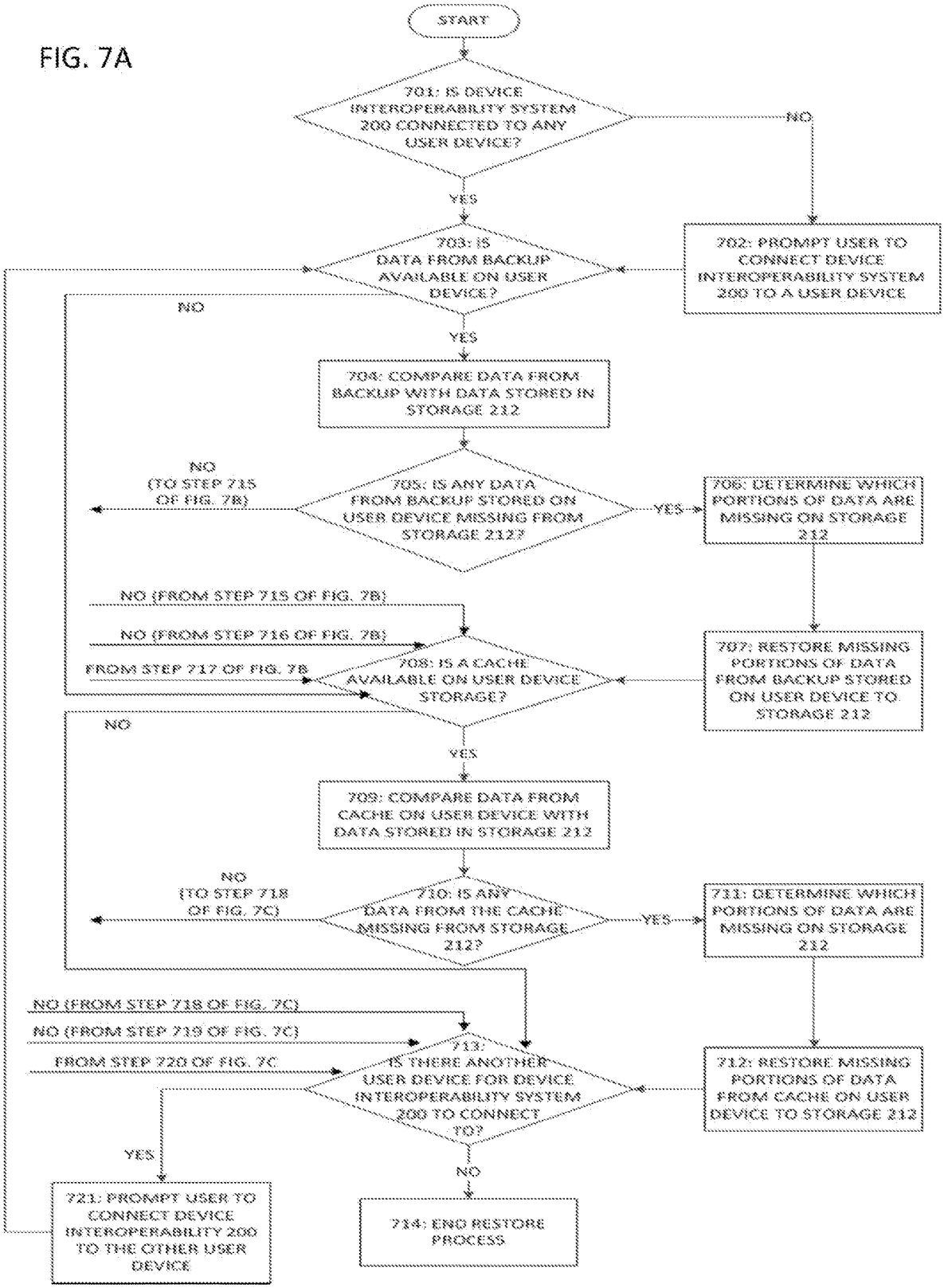

SECURITY LEVEL 611
(ASSIGNED TO DATA WITH SENSITIVITY LEVEL 601)
SECURITY LEVEL 612
(ASSIGNED TO DATA WITH SENSITIVITY LEVEL 602)
SECURITY LEVEL 613
(ASSIGNED TO DATA WITH SENSITIVITY LEVEL 603)
SECURITY LEVEL 614
(ASSIGNED TO DATA WITH SENSITIVITY LEVEL 604)
HIERARCHY
600

FIG. 6

FIG. 7A


START
701: IS DEVICE INTEROPERABILITY SYSTEM 200 CONNECTED TO ANY USER DEVICE?
NO
YES
702: PROMPT USER TO CONNECT DEVICE INTEROPERABILITY SYSTEM 200 TO A USER DEVICE
703: IS DATA FROM BACKUP AVAILABLE ON USER DEVICE?
NO
YES
704: COMPARE DATA FROM BACKUP WITH DATA STORED IN STORAGE 212
NO (TO STEP 715 OF FIG. 7B)
705: IS ANY DATA FROM BACKUP STORED ON USER DEVICE MISSING FROM STORAGE 212?
YES
706: DETERMINE WHICH PORTIONS OF DATA ARE MISSING ON STORAGE 212
NO (FROM STEP 715 OF FIG. 7B)
NO (FROM STEP 716 OF FIG. 7B)
FROM STEP 717 OF FIG. 7B
708: IS A CACHE AVAILABLE ON USER DEVICE STORAGE?
707: RESTORE MISSING PORTIONS OF DATA FROM BACKUP STORED ON USER DEVICE TO STORAGE 212
NO
YES
709: COMPARE DATA FROM CACHE ON USER DEVICE WITH DATA STORED IN STORAGE 212
NO (TO STEP 718 OF FIG. 7C)
710: IS ANY DATA FROM THE CACHE MISSING FROM STORAGE 212?
YES
711: DETERMINE WHICH PORTIONS OF DATA ARE MISSING ON STORAGE 212
NO (FROM STEP 718 OF FIG. 7C)
NO (FROM STEP 719 OF FIG. 7C)
FROM STEP 720 OF FIG. 7C
713: IS THERE ANOTHER USER DEVICE FOR DEVICE INTEROPERABILITY SYSTEM 200 TO CONNECT TO?
712: RESTORE MISSING PORTIONS OF DATA FROM CACHE ON USER DEVICE TO STORAGE 212
YES
NO
721: PROMPT USER TO CONNECT DEVICE INTEROPERABILITY 200 TO THE OTHER USER DEVICE
714: END RESTORE PROCESS

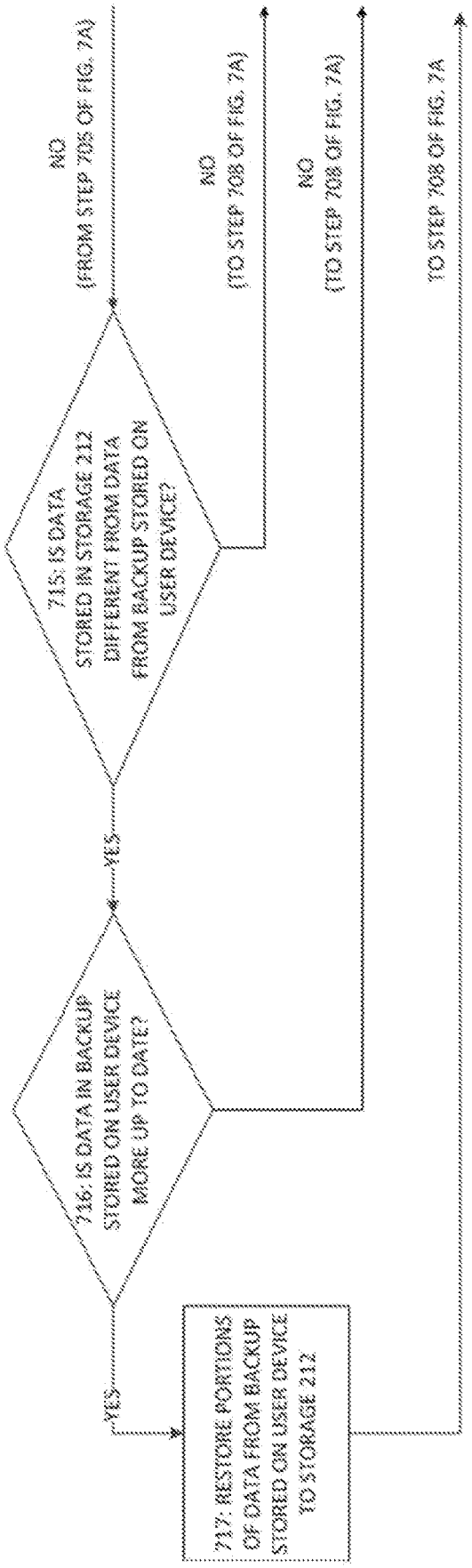

715: IS DATA STORED IN STORAGE 212 DIFFERENT FROM DATA FROM BACKUP STORED ON USER DEVICE?
NO (FROM STEP 705 OF FIG. 7A)
NO (TO STEP 708 OF FIG. 7A)
YES
716: IS DATA IN BACKUP STORED ON USER DEVICE MORE UP TO DATE?
NO (TO STEP 708 OF FIG. 7A)
YES
717: RESTORE PORTIONS OF DATA FROM BACKUP STORED ON USER DEVICE TO STORAGE 212
TO STEP 708 OF FIG. 7A

FIG. 7B

FIG. 7C

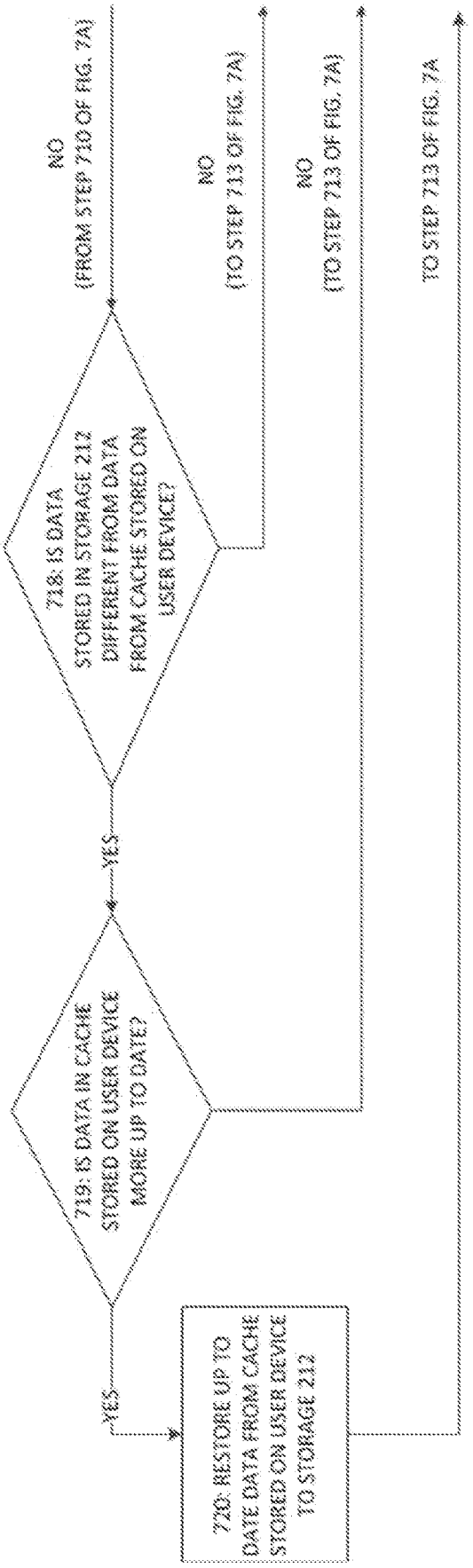

718: IS DATA STORED IN STORAGE 212 DIFFERENT FROM DATA FROM CACHE STORED ON USER DEVICE?
NO (FROM STEP 710 OF FIG. 7A)
NO (TO STEP 713 OF FIG. 7A)
YES
719: IS DATA IN CACHE STORED ON USER DEVICE MORE UP TO DATE?
NO (TO STEP 713 OF FIG. 7A)
YES
720: RESTORE UP TO DATE DATA FROM CACHE STORED ON USER DEVICE TO STORAGE 212
TO STEP 713 OF FIG. 7A

FIG. 8A

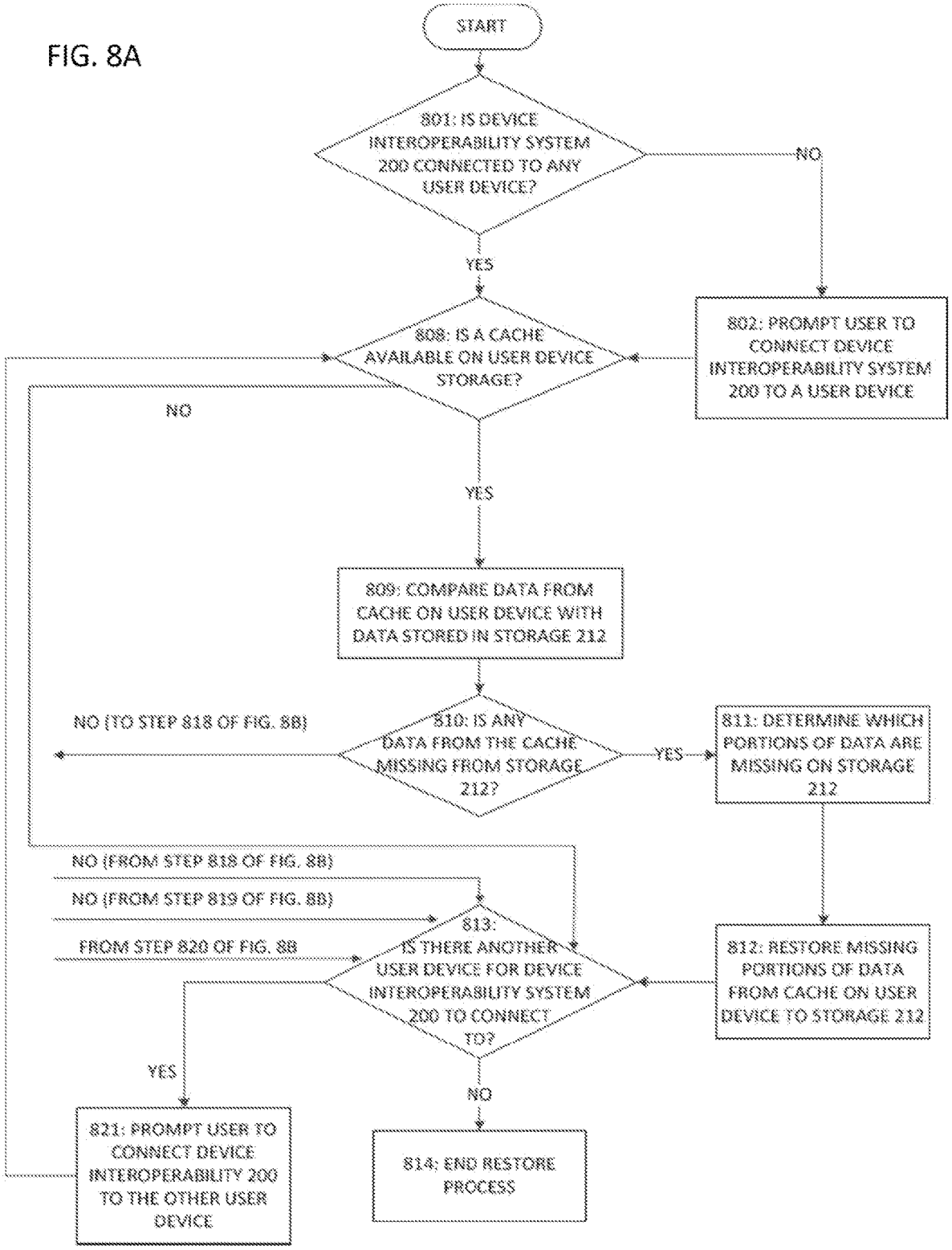
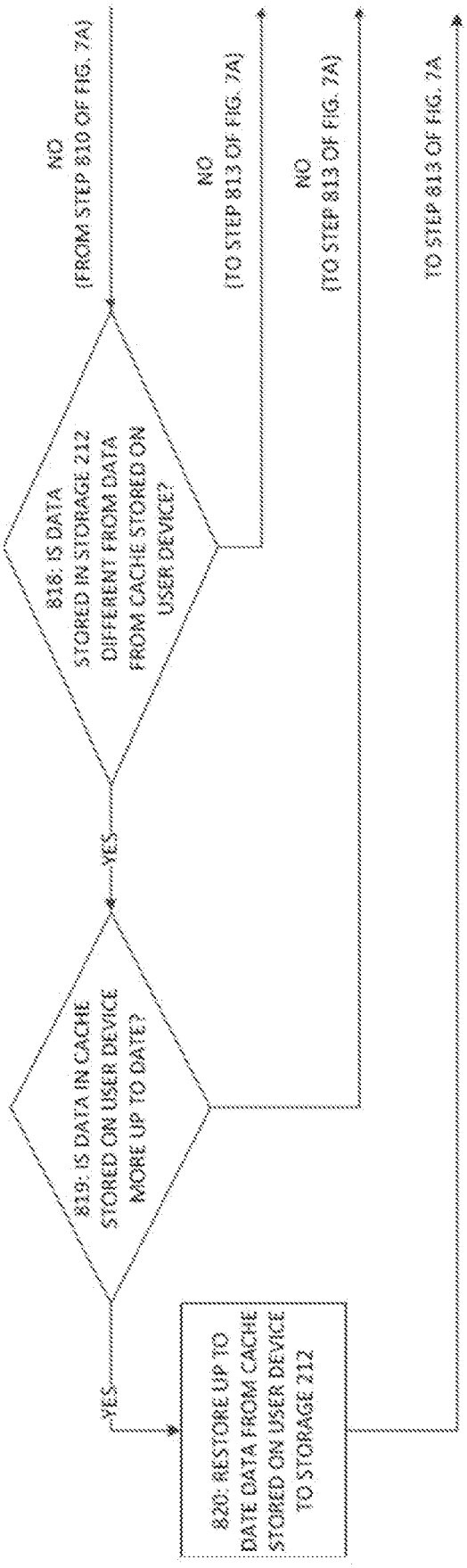

START
801: IS DEVICE INTEROPERABILITY SYSTEM 200 CONNECTED TO ANY USER DEVICE?
NO
YES
802: PROMPT USER TO CONNECT DEVICE INTEROPERABILITY SYSTEM 200 TO A USER DEVICE
808: IS A CACHE AVAILABLE ON USER DEVICE STORAGE?
NO
YES
809: COMPARE DATA FROM CACHE ON USER DEVICE WITH DATA STORED IN STORAGE 212
NO (TO STEP 818 OF FIG. 8B)
810: IS ANY DATA FROM THE CACHE MISSING FROM STORAGE 212?
YES
811: DETERMINE WHICH PORTIONS OF DATA ARE MISSING ON STORAGE 212
NO (FROM STEP 818 OF FIG. 8B)
NO (FROM STEP 819 OF FIG. 8B)
FROM STEP 820 OF FIG. 8B
813: IS THERE ANOTHER USER DEVICE FOR DEVICE INTEROPERABILITY SYSTEM 200 TO CONNECT TO?
812: RESTORE MISSING PORTIONS OF DATA FROM CACHE ON USER DEVICE TO STORAGE 212
YES
NO
821: PROMPT USER TO CONNECT DEVICE INTEROPERABILITY 200 TO THE OTHER USER DEVICE
814: END RESTORE PROCESS

FIG. 8B


818: IS DATA STORED IN STORAGE 212 DIFFERENT FROM DATA FROM CACHE STORED ON USER DEVICE?
NO (FROM STEP 810 OF FIG. 7A)
NO (TO STEP 813 OF FIG. 7A)
YES
819: IS DATA IN CACHE STORED ON USER DEVICE MORE UP TO DATE?
NO (TO STEP 813 OF FIG. 7A)
YES
820: RESTORE UP TO DATE DATA FROM CACHE STORED ON USER DEVICE TO STORAGE 212
TO STEP 813 OF FIG. 7A

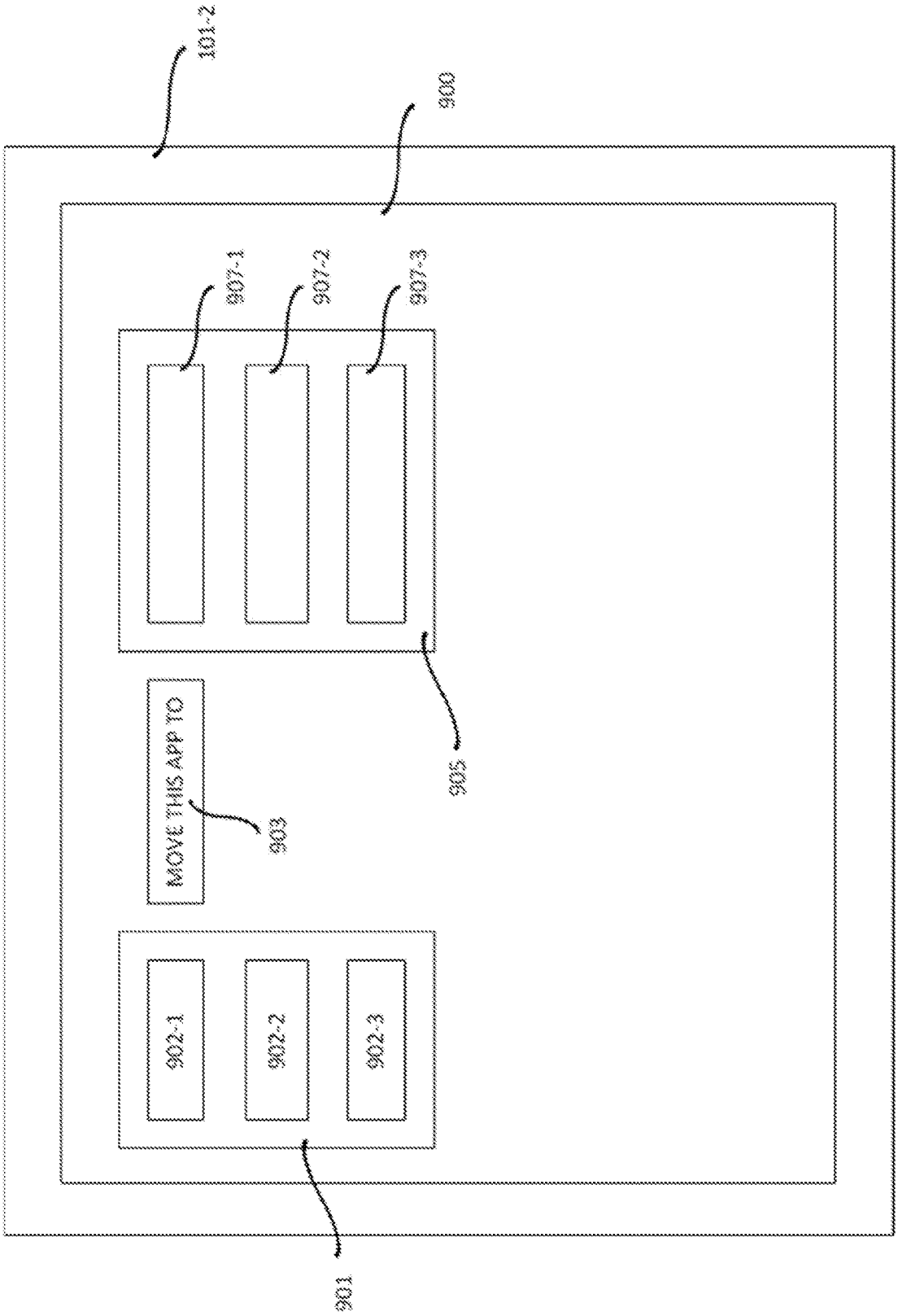

101-2
900
907-1
907-2
907-3
905
MOVE THIS APP TO
903
902-1
902-2
902-3
901

FIG. 9A

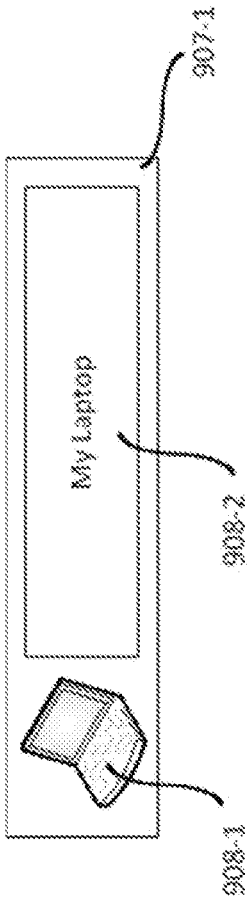



FIG. 9B

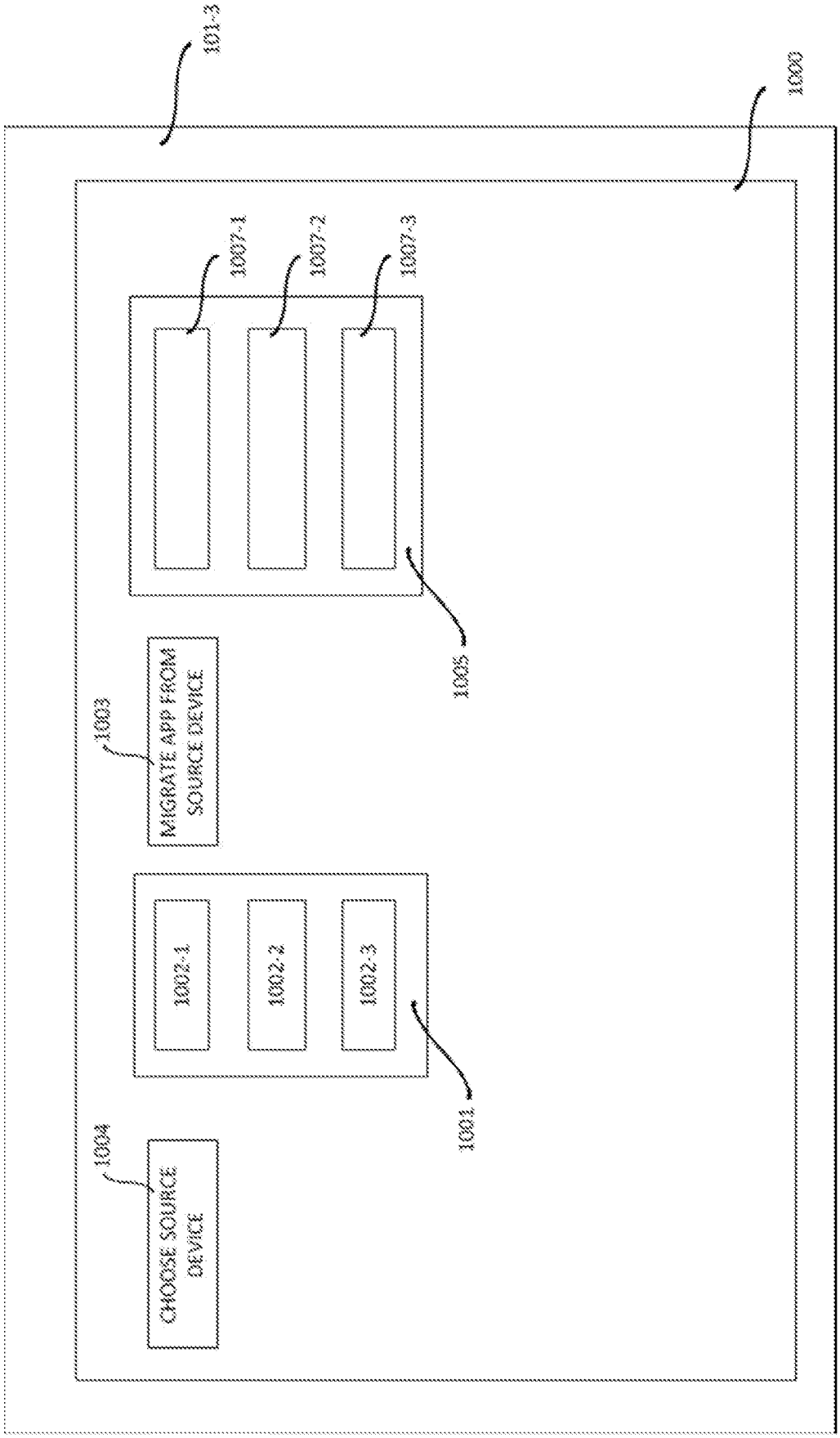

101-3
1000
1007-1
1007-2
1007-3
1005
1003
MIGRATE APP FROM SOURCE DEVICE
1002-1
1002-2
1002-3
1001
1004
CHOOSE SOURCE DEVICE

FIG. 10A

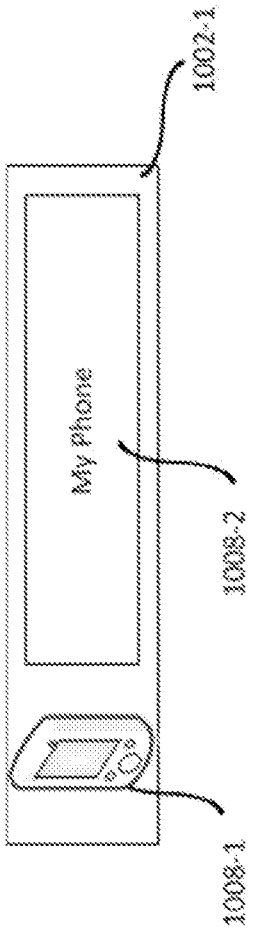
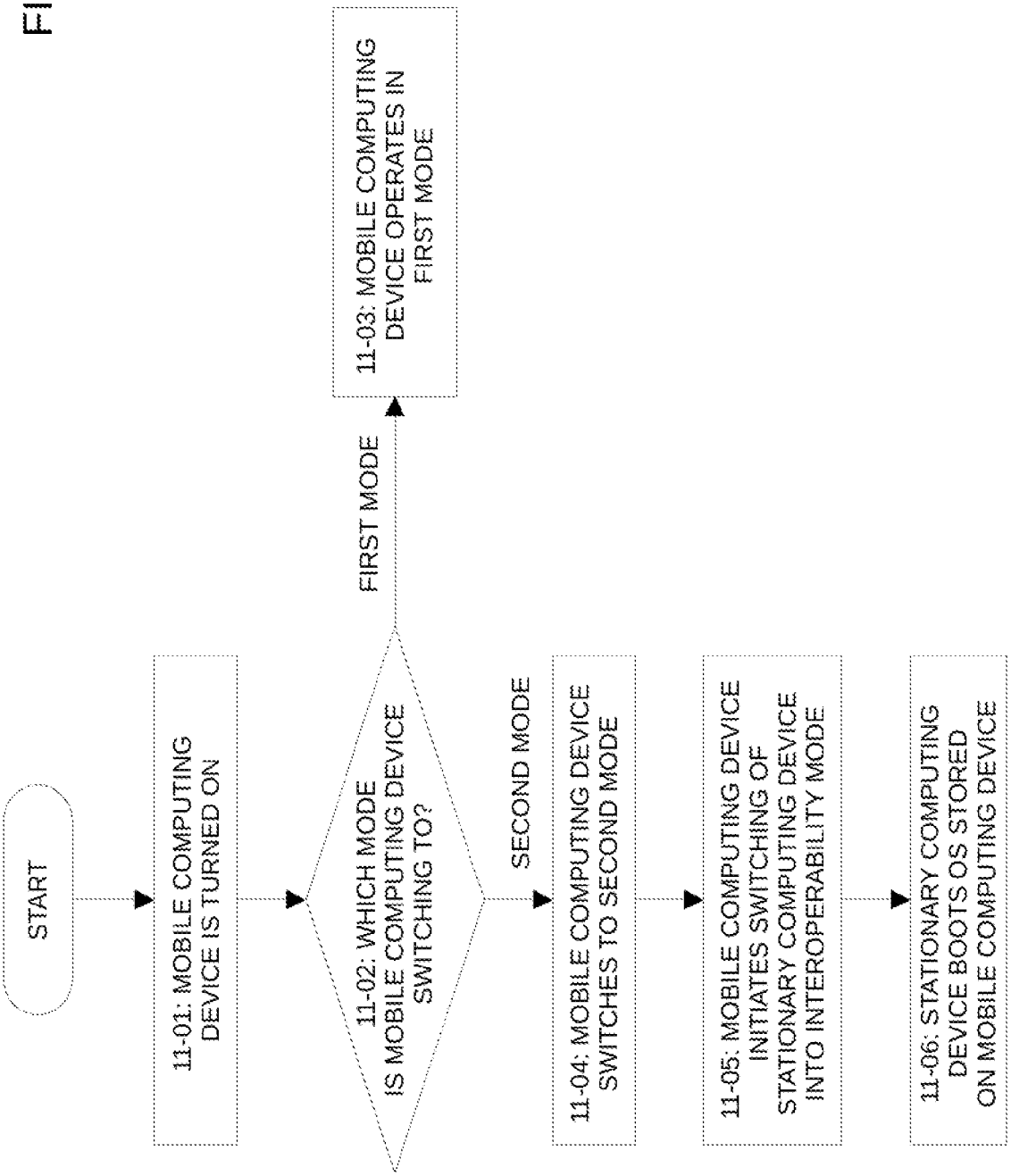



FIG. 10B

FIG. 11


START
11-01: MOBILE COMPUTING DEVICE IS TURNED ON
11-02: WHICH MODE IS MOBILE COMPUTING DEVICE SWITCHING TO?
FIRST MODE
11-03: MOBILE COMPUTING DEVICE OPERATES IN FIRST MODE
SECOND MODE
11-04: MOBILE COMPUTING DEVICE SWITCHES TO SECOND MODE
11-05: MOBILE COMPUTING DEVICE INITIATES SWITCHING OF STATIONARY COMPUTING DEVICE INTO INTEROPERABILITY MODE
11-06: STATIONARY COMPUTING DEVICE BOOTS OS STORED ON MOBILE COMPUTING DEVICE

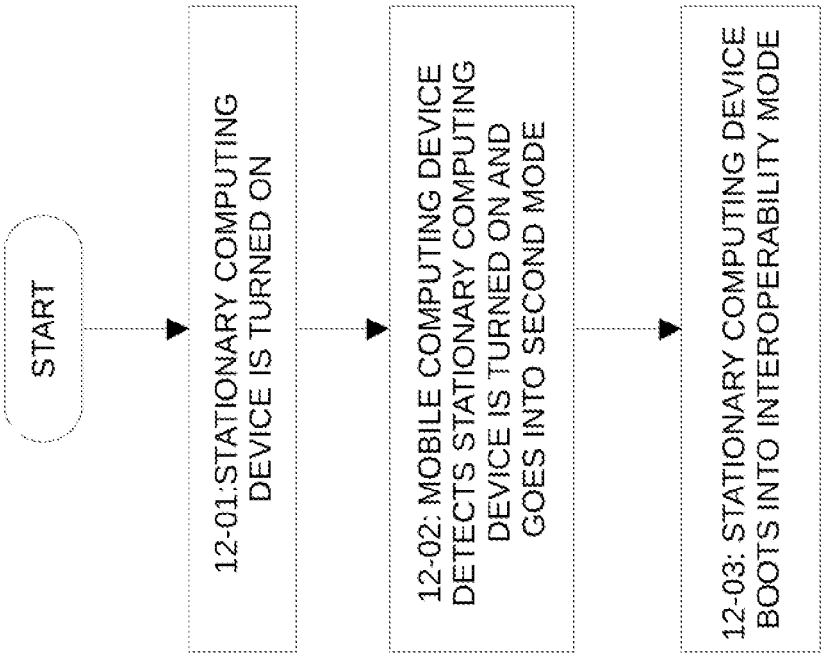

START
12-01:STATIONARY COMPUTING DEVICE IS TURNED ON
12-02: MOBILE COMPUTING DEVICE DETECTS STATIONARY COMPUTING DEVICE IS TURNED ON AND GOES INTO SECOND MODE
12-03: STATIONARY COMPUTING DEVICE BOOTS INTO INTEROPERABILITY MODE

FIG. 12

FIG. 13

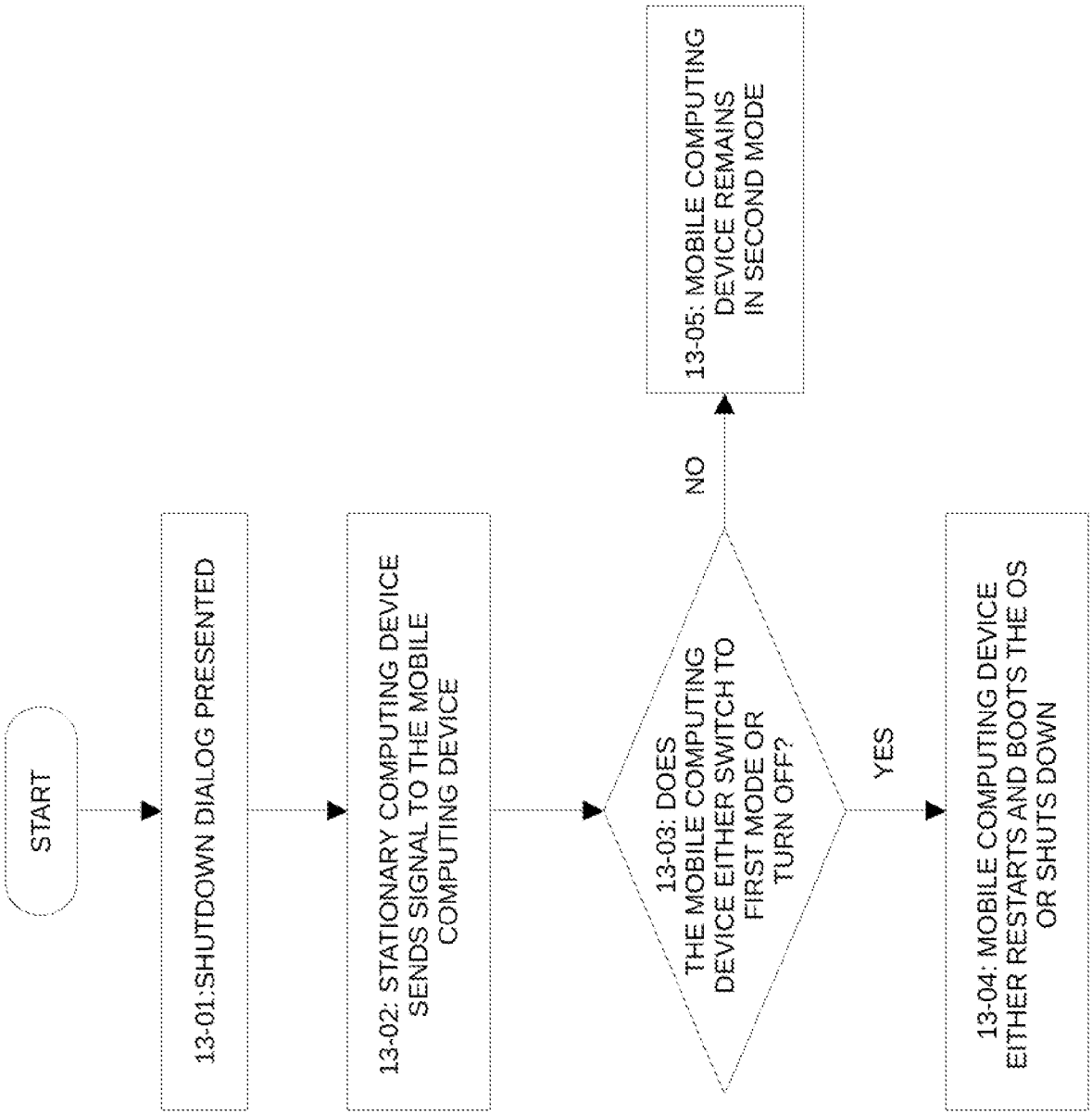
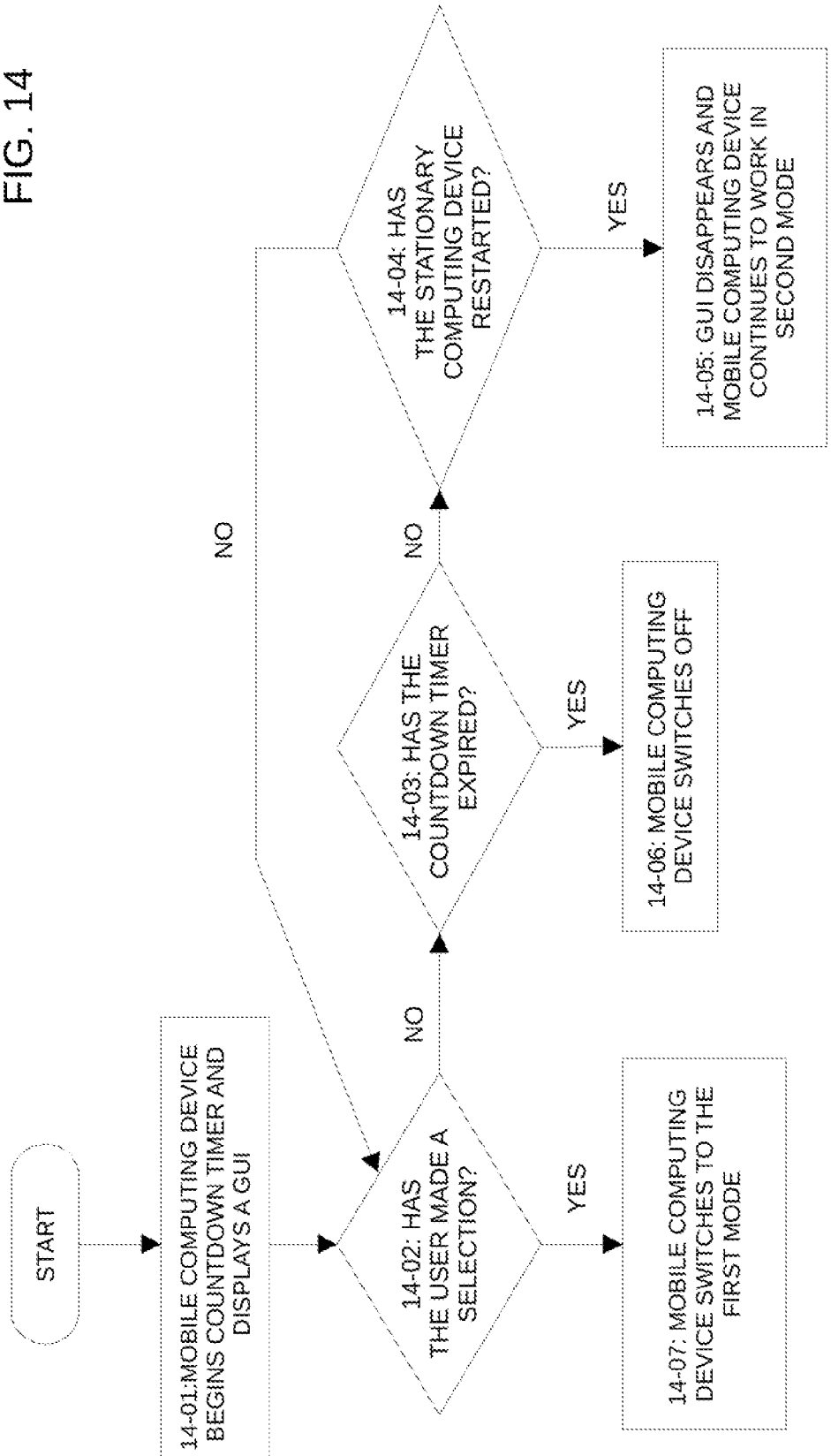

START
13-01:SHUTDOWN DIALOG PRESENTED
13-02: STATIONARY COMPUTING DEVICE SENDS SIGNAL TO THE MOBILE COMPUTING DEVICE
13-03: DOES THE MOBILE COMPUTING DEVICE EITHER SWITCH TO FIRST MODE OR TURN OFF?
NO
13-05: MOBILE COMPUTING DEVICE REMAINS IN SECOND MODE
YES
13-04: MOBILE COMPUTING DEVICE EITHER RESTARTS AND BOOTS THE OS OR SHUTS DOWN

FIG. 14


START
14-01:MOBILE COMPUTING DEVICE BEGINS COUNTDOWN TIMER AND DISPLAYS A GUI
14-02: HAS THE USER MADE A SELECTION?
YES
14-07: MOBILE COMPUTING DEVICE SWITCHES TO THE FIRST MODE
NO
14-03: HAS THE COUNTDOWN TIMER EXPIRED?
YES
14-06: MOBILE COMPUTING DEVICE SWITCHES OFF
NO
14-04: HAS THE STATIONARY COMPUTING DEVICE RESTARTED?
YES
14-05: GUI DISAPPEARS AND MOBILE COMPUTING DEVICE CONTINUES TO WORK IN SECOND MODE
NO

# SYSTEM AND METHOD FOR MOBILE AND STATIONARY COMPUTING DEVICE INTERWORKING

## FIELD OF THE INVENTION

**[0001]** The present disclosure relates to the integration of mobile and stationary computing.

## SUMMARY

**[0002]** A system for a mobile computing device and a stationary computing device associated with a user to interwork, wherein: the mobile computing device comprises a device interoperability system having a communications module, wherein a first connection is established between the stationary computing device and the communications module; storage coupled to said communications module, wherein the storage stores an operating system, one or more programs, data associated with the user, further wherein the operating system is booted by the stationary computing device via the first connection, the operating system runs on the stationary computing device, and the operating system uses one or more processing capabilities of the stationary computing device for operation; and one or more processors to support said device interoperability system.

**[0003]** A method for a mobile computing device and a stationary computing device associated with a user to interwork, the method comprising: providing a device interoperability system comprising a communications module, storage coupled to the communications module, and the device interoperability system is installed on the mobile computing device; enabling establishment of a first connection between the stationary computing device and the communications module; enabling the storage to store information comprising an operating system, one or more programs, and data associated with the user; enabling booting of the operating system by the stationary computing device via the first connection; and enabling the operating system to run on the stationary computing device and use one or more processing capabilities of the stationary computing device for operation.

**[0004]** The foregoing and additional aspects and embodiments of the present disclosure will be apparent to those of ordinary skill in the art in view of the detailed description of various embodiments and/or aspects, which is made with reference to the drawings, a brief description of which is provided next.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0005]** The foregoing and other advantages of the disclosure will become apparent upon reading the following detailed description and upon reference to the drawings.

**[0006]** FIG. **1** illustrates a situation for a user with one or more user devices.

**[0007]** FIG. **2** illustrates an example of a device interoperability system working in conjunction with the user devices.

**[0008]** FIG. **2**B illustrates an example architecture for the device interoperability system.

**[0009]** FIG. **2**C illustrates an example of a gadget to run the device interoperability system.

**[0010]** FIG. **2**D shows an example where the device interoperability system is integrated into a user device.

**[0011]** FIG. **2**E shows an example where the device interoperability system runs as an application on a user device.

**[0012]** FIG. **3** shows an example algorithm for user device switchability between "stand-alone" and "interoperability system" modes.

**[0013]** FIG. **3**B shows an example algorithm for switching the operating system between different hardware configurations on booting.

**[0014]** FIG. **4** shows an example embodiment of a caching operation.

**[0015]** FIG. **4**B illustrates an example embodiment of caching where additional checks are performed before writing data to the cache.

**[0016]** FIG. **4**C shows an example embodiment of prefetching to a cache.

**[0017]** FIG. **5** illustrates an example embodiment of different storage sub-areas, each having different associated security levels.

**[0018]** FIG. **6** illustrates an example embodiment of a hierarchy of secure storage for a user's data.

**[0019]** FIG. **7**A illustrates part of an example embodiment of a data restore process.

**[0020]** FIG. **7**B illustrates part of an example embodiment of a data restore process.

**[0021]** FIG. **7**C illustrates part of an example embodiment of a data restore process.

**[0022]** FIG. **8**A illustrates part of an example embodiment of a data restore process where only data stored in a cache is used for data restore.

**[0023]** FIG. **8**B illustrates part of an example embodiment of a data restore process where only data stored in a cache is used for data restore.

**[0024]** FIG. **9**A illustrates an example embodiment of a graphical user interface (GUI) for a push-based application migration.

**[0025]** FIG. **9**B illustrates an example embodiment of a GUI item corresponding to a user device for a push-based application migration.

**[0026]** FIG. **10**A illustrates an example embodiment of a GUI for a pull-based application migration.

**[0027]** FIG. **10**B illustrates an example embodiment of a GUI item corresponding to a user device for a pull-based application migration.

**[0028]** FIG. **11** illustrates an example embodiment of a process for a stationary computing device to boot an operating system (OS) stored in a mobile computing device when both the mobile computing device and the stationary computing device are turned off.

**[0029]** FIG. **12** illustrates an example embodiment of a process to turn on and switch both devices to interoperability modes by turning on the stationary computing device.

**[0030]** FIG. **13** illustrates an example embodiment of a process to enable a user to perform actions after the end of a working session.

**[0031]** FIG. **14** illustrates an example embodiment of a timer-based process to enable a user to perform actions after the end of a working session.

**[0032]** While the present disclosure is susceptible to various modifications and alternative forms, specific embodiments or implementations have been shown by way of example in the drawings and will be described in detail herein. It should be understood, however, that the disclosure is not intended to be limited to the particular forms dis-

closed. Rather, the disclosure is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of an invention as defined by the appended claims.

## DETAILED DESCRIPTION

**[0033]** The number of devices owned or operated by a person has grown tremendously.

**[0034]** Typically, a person has a plurality of computing devices, such as:

**[0035]** Smartphones,

**[0036]** Tablets,

**[0037]** Desktops,

**[0038]** Laptops,

**[0039]** Game consoles,

**[0040]** Smart watches/bands, and

**[0041]** Smart glasses.

**[0042]** In addition, many other devices and items have become "smart", that is, their computing capabilities and processing power have increased, and they have been network enabled. These include, for example:

**[0043]** Vehicles such as cars and trucks,

**[0044]** Television (TV) sets,

**[0045]** Kitchen appliances such as refrigerators and microwave ovens,

**[0046]** Cameras,

**[0047]** Fitness devices such as FITBIT®,

**[0048]** Medical devices such as blood pressure monitors and heart rate monitors,

**[0049]** Air-conditioning systems, and

**[0050]** Smart home systems.

**[0051]** Furthermore the "Internet of Things" (IoT) has also grown tremendously. The IoT refers to networks of consumer and industrial devices interconnected with each other and with other computing devices.

**[0052]** All of this means that the number of devices which have computing and network capability, and are associated with a particular user, is growing rapidly.

**[0053]** FIG. **1** illustrates a typical situation faced by a user. In FIG. **1**, user devices **101-1** to **101**-N comprise the devices associated with user **100**. These include, for example, the electronic computing devices and the other devices and items mentioned above.

**[0054]** Given the situation shown in FIG. **1**, users such as user **100** face many different challenges. Firstly, documents and data from different devices need to be synchronized with each other. Typically this is performed using, for example:

**[0055]** Portable data storage devices such as Universal Serial Bus (USB) flash drives, and removable hard drives, and

**[0056]** Network or "cloud"-based techniques.

**[0057]** These techniques of document and data synchronization have deficiencies. Cloud connectivity may not always be present. When it is, connectivity may be intermittent or slow. Privacy may also be an issue with cloud-based techniques. Furthermore, aggregating data from a lot of users at a central location, means that the central location poses an attractive target to cyber-criminals, which poses security challenges.

**[0058]** Secondly, synchronization may be imperfect or incomplete due to each computing device and consumer item running different operating systems (OSes) and different platforms. Referring to FIG. **1**, user device **101-1** to **101**-N have their own processing and memory capabilities and may run different OSes, platforms and software. As a consequence the user is forced to get used to different environments on different devices and also to repeat the same tasks for several devices, for example, installing applications, customizing settings or performing service tasks like software updates or antivirus scanning. Compatibility may also be an issue. As an example, if user **100** edits a file first with user device **101-1** and then with user device **101-2**, that file may end up being corrupted as a result due to the different versions of the editing software installed on user device **101-1** and **101-2**.

**[0059]** It is therefore necessary to address these deficiencies in device synchronization in order to ensure continued growth and adoption of "smart" technology; and interoperability of these user devices.

**[0060]** These problems are also faced by users who work in mobile and stationary work modes. Mobile computing devices such as laptops are often used in a combination of mobile and stationary work modes. Users use laptops in a stationary mode at home or in the office, and they switch to a mobile mode whereby they take their laptop with them, for example, to business meetings, cafes, and to travel.

**[0061]** Currently, mobile computing devices offer tradeoffs between mobility, battery runtime, performance and convenience. In stationary mode, the mobile computing device can be plugged into mains power supply, so battery availability is not such a problem. Then, computing performance and convenience take precedence. A user who wants computing performance and convenience can purchase a high performance laptop, such as, for example, a gaming laptop or a workstation laptop.

**[0062]** However, using a high performance laptop may pose problems to the user in mobile mode. High performance laptops have higher energy consumption, thereby reducing battery runtime. Heat dissipation during operation increases, requiring bigger cooling systems which are heavier and draw more energy, which lead to reduced mobility and further reduce battery runtime. High performance laptops tend to be more expensive as well.

**[0063]** Users can purchase laptops which are designed to be cheaper, smaller and lighter and therefore better suit the mobile mode. However these laptops tend to offer lower performance and less convenience to the user while in stationary mode. Currently users can take some steps to extend the capabilities of such lightweight models. For example, a user can attach a docking station while in stationary mode, so as to extend and simplify the connectivity of the laptop. The user can also connect an external display while in stationary mode to increase the size of the display available for viewing. Finally, a user can connect an external graphics processing unit (eGPU) to the laptop. Typically, this is a normal desktop-grade graphics card placed in a separate case with a power supply unit (PSU) and cooling system. The eGPU is connected to the laptop in a stationary mode via a high-speed interface port such as a Thunderbolt port, to provide additional graphics performance. However these steps can only extend the capabilities so much.

**[0064]** Another potential solution for the user who wants good performance and convenience in the stationary mode, and also good mobility and battery runtime while in mobile mode, is to purchase a combination of a lightweight and long battery runtime laptop for mobile mode; and a desktop which offers high performance and convenience for stationary mode. This laptop-desktop combination offers several

advantages over a single high-performance laptop: Typically desktops can deliver much more processing power than a high-performance laptop in stationary mode. The desktop is designed as a stationary mode device, where else the high-performance laptop designer must take mobility and battery runtime into account. Therefore, desktops do not face the energy, space and cooling limitations faced by a high performance laptop, meaning that a desktop designer can optimize for performance of tasks such as gaming and professional tasks in a stationary mode.

**[0065]** The desktop provides more flexibility in terms of hardware configuration, meaning that the user will be better able to customize to requirements in a more cost-efficient way.

**[0066]** When the user is mobile, the user can utilize the lightweight, highly mobile laptop which is also more energy efficient, and therefore offers better battery run-time.

**[0067]** While the laptop-desktop combination does offer a potential solution to improve user experience in mobile and stationary modes, it does face some of the general limitations detailed above. For example, a user still needs to synchronize data between the laptop and desktop, which may pose privacy and security issues if the user chooses to use cloud-based techniques. The OSes and installed applications on the laptop and desktop may be configured differently, leading to some switching friction between mobile and stationary modes. Together, this leads to a less than ideal user experience.

**[0068]** The remainder of this specification details a system and a method for device interoperability to address the above problems, and a specific application of such a system and method for the user who works in mobile and stationary modes.

**[0069]** The system and method for device interoperability is detailed first. An example architecture of such a device interoperability system **200** is shown in FIGS. **2** and **2**B. In FIG. **2**, one or more connections **201-1**, **201-2**, **201-3** to **201**-N between device interoperability system **200** and one or more user devices **101-1**, **101-2**, **101-3** to **101**-N are established as needed. In one embodiment, the one or more user devices **101-1**, **101-2**, **101-3** to **101**-N initiate the establishment of the connection. In another embodiment, the device interoperability system **200** initiates the establishment of the connection.

**[0070]** Device interoperability system **200** comprises several components necessary for its functioning. An illustration of one embodiment of device interoperability system **200** is shown in FIG. **2**B. As shown in FIG. **2**B, device interoperability system **200** comprises battery **211**, battery charging module **221**, storage **212**, one or more processors **215** and communications module **213**.

**[0071]** The one or more processors **215** perform the functions of supporting the other elements of device interoperability system **200**. This includes, for example:

**[0072]** Maintaining interconnection between the elements of device interoperability system **200**

**[0073]** Maintaining overall security of device interoperability system **200**, and

**[0074]** Service functions necessary for the operation of device interoperability system **200**.

**[0075]** Communications module **213** participates in the establishment of the one or more connections **201-1** to **201**-N. Communications module **213** also works to maintain the one or more connections **201-1** to **201**-N to the one or more user devices **101-1** to **101**-N.

**[0076]** Communications module **213** also works to perform operations necessary to secure connections **201-1** to **201**-N. These include, for example, encryption and access operations. In one embodiment, communications module **213** also manages and optimizes power consumption related to one or more connections **201-1** to **201**-N. For example, communications module **213** adjusts the transmission powers used for the one or more connections **201-1** to **201**-N based on distances from user devices such as user device **101-1**.

**[0077]** Battery **211** supplies power for the operation of device interoperability system **200**. Charging module **221** enables charging of battery **211** using an external power source. In one embodiment, charging module **221** enables wireless charging.

**[0078]** As shown in FIG. **2**B, storage **212** is coupled to communications module **213** and is used to store OS **214**, programmes and data **216** which are necessary for the functioning of device interoperability system **200**. For example, user preferences, applications and user documents and data may also be stored on storage **212**. The functioning of OS **214** will be discussed in detail below. In one embodiment, storage **212** is built using energy-efficient storage technology such as SSD (Solid State Drive) or embedded MultiMedia Controller (eMMC) flash memory technology. In one embodiment, the information stored in storage **212** is encrypted. This reduces the risk of a malicious party obtaining access to the stored information. In one embodiment, the Advanced Encryption Standard (AES) is used for encryption.

**[0079]** Referring to FIGS. **2** and **2**B, connection **201-1** between device interoperability system **200** and user device **101-1** is established before the native user device **101-1** OS loads. Once connection **201-1** is established with user device **101-1**, OS **214** boots and runs from storage **212** on user device **101-1**. Then, user device **101-1** is able to access data and program code stored on storage **212** as required. The program code of OS **214** and installed applications is run on the user device which device interoperability system **200** is connected to, and uses the processing capabilities of this user device for its operation. For example, referring to FIG. **2**, if device interoperability system **200** is connected to user device **101-1**, then the program code is run on user device **101-1** using the processing power and memory of user device **101-1** as needed. The establishment of connection **201-1** and subsequent booting of OS **214** is performed in a variety of ways, as will be detailed below.

**[0080]** In one embodiment, at least one of the connections **201-1** to **201**-N is a direct connection. This direct connection can be, for example, a direct wireless connection.

**[0081]** In some embodiments, user device **101-1** comprises firmware that provides the ability to support booting from device interoperability system **200** via a direct wireless connection. For example, in one embodiment, user device **101-1** comprises a Basic Input Output System (BIOS) or Unified Extensible Firmware Interface (UEFI) which supports the Media Agnostic USB specification. This allows the user device **101-1** to use the USB protocol over the direct wireless connection to facilitate the booting of OS **214** on the user device **101-1** and data transfer between device interoperability system **200** and user device **101-1**.

**[0082]** In some embodiments, user device **101-1** does not comprise firmware that provides the ability to support booting from device interoperability system **200** via a direct wireless connection. Then it is necessary to use an intermediary. For example, in one embodiment, the device interoperability system **200** is coupled wirelessly to a miniature USB dongle plugged into a USB port on user device **101-1**. Then the miniature USB dongle will simulate a USB flash drive connected to user device **101-1**. Then when user device **101-1** is switched on, the direct wireless connection is established between the miniature USB dongle and device interoperability system **200**. Then OS **214** is booted on the user device **101-1** from the storage **212** as though it is an ordinary USB flash drive connected to the USB port. In one embodiment, the user must change the BIOS or UEFI settings for user device **101-1** so that user device **101-1** will boot from the device interoperability system **200**.

**[0083]** In another embodiment, the direct connection is a direct wired connection. In a further embodiment, the at least one direct wired connection includes, for example, a USB connection. In further embodiments, the direct wired connection is a connection facilitated via docking. In yet another embodiment, at least one of the connections **201-1** to **201**-N are direct wireless and at least one of the connections **201-1** to **201**-N are direct wired.

**[0084]** When connection **201-1** between user device **101-1** and device interoperability system **200** is facilitated via docking, further embodiments are also possible. In one embodiment, both the device where device interoperability system **200** is installed, and user device **101-1** have direct docking capabilities including, for example, docking ports. Then, device interoperability system **200** interacts with the user device **101-1** via these direct docking capabilities. In yet another embodiment, the device where device interoperability system **200** is installed is coupled to a docking station which is connected to user device **101-1**. In a further embodiment, when the docking station is connected to user device **101-1** by, for example, USB cable, the user device **101-1** will recognize the docking station with the device where device interoperability system **200** is installed as a connected external USB drive. In one embodiment, the user must change the BIOS or UEFI settings for user device **101-1** so that user device **101-1** will boot from the USB connected device. In a further embodiment, the docking station provides charging for the device where device interoperability system **200** is installed.

**[0085]** While the above describes situations where connections **201-1** to **201**-N are direct connections between two devices, one of skill in the art would know that it is possible to use indirect connections as well. In another embodiment, at least one of the connections **201-1** to **201**-N are indirect connections. These indirect connections include, for example, one or more of:

**[0086]** connections facilitated via a Local Area Network (LAN), or

**[0087]** connections facilitated via a cloud-based service.

**[0088]** In a further embodiment, when at least two of the described above types of connection are available, the choice between connection types is performed automatically for at least one of the connections **201-1** to **201**-N. In one embodiment, the choice is based on the following factors:

**[0089]** connection speed,

**[0090]** connection latency,

**[0091]** data transmission costs, and

**[0092]** user preferences.

**[0093]** In a further embodiment, when the connectivity is lost, either a different type of direct or indirect connection is automatically selected.

**[0094]** In a further embodiment, the at least one connection is secured. The securing is performed by, for example:

**[0095]** Encryption using techniques such as Wi-Fi Protected Access (WPA2), and

**[0096]** Requiring access authentication on both endpoints of a connection when the connection is first established. This is performed using, for example passwords and techniques such as near field communication (NFC) or Wi-Fi Protected Setup (WPS)-like algorithms.

**[0097]** In embodiments where the at least one connection is secured, before establishing the connection authentication is performed at the end points, that is, between device interoperability system **200** and user device **101-1**.

**[0098]** Device interoperability system **200** can be implemented in a variety of ways. In one embodiment, device interoperability system **200** is implemented using a separate gadget, such as gadget **210** as shown in FIG. **2**C. Then, the one or more connections **201-1** to **201**-N are established with gadget **210**.

**[0099]** In another embodiment, device interoperability system **200** is installed via integration into one of user devices **101-1** to **101**-N. For example, as shown in FIG. **2**D, device interoperability system **200** is integrated into user device **101-1**. This is achieved by, for example, implementing device interoperability system **200** as a firmware module of user device **101-1**. Then, device interoperability system **200** uses one or more of the battery, storage, communications module, processors and other capabilities of user device **101-1** in a similar fashion to the above-described use of battery **211**, storage **212**, one or more processors **215** and communications module **213** for its operation. OS **214** is stored within the storage of user device **101-1**. Then the user device **101-1** runs OS **214** instead of its native OS. When, as shown in FIG. **2**D, at least one of connections **201-2** to **201**-N are established between user device **101-1** and at least one of the other devices **101-2** to **101**-N, device interoperability system **200** enables the connected user device to:

**[0100]** boot OS **214** which is stored in the storage of user device **101-1**, and

**[0101]** use programmes and data **216** which are stored in the storage of user device **101-1**.

**[0102]** In a further embodiment, some hardware components of the user device **101-1** with integrated device interoperability system **200** are recognised and used by the OS **214** as connected external devices, when OS **214** runs on a different device which is connected to user device **101-1**. For example, in the case where user device **101-1** is a mobile device: When OS **214** runs on user device **101-2** which is connected to user device **101-1**, the hardware components of user device **101-1** such as the microphone, sensors, mobile telecommunications module and display are used by OS **214** as external devices.

**[0103]** In yet another embodiment, device interoperability system **200** is implemented as an installed application or an "app" which runs on one of user devices **101-1** to **101**-N, for example user device **101-1**. For example, as shown in FIG. **2**E, device interoperability system **200** runs as an app on user device **101-1**. Then device interoperability system **200**

uses one or more of the battery, storage, communications module, processors and other capabilities of user device **101-1** for its operation, similar to the integrated case described above and in FIG. **2**D. Similar to as described above, when a connection is established with a user device, device interoperability system **200** gives the connected user device the ability to boot OS **214** which is stored in the storage of user device **101-1**. In another embodiment, in case the app is not able to provide the required level of access to data stored on storage **212** of user device **101-1** which is used to boot OS **214** on user device **101-2**, the interoperability system **200** also includes a separate image of

**[0104]** either a copy of the OS **214**, or

**[0105]** some of its components.

**[0106]** This image is used on its own or in conjunction with user device **101-1** OS's components stored on storage of user device **101-1** to boot the OS on user device **101-2**. Similar to the cases described above, in a further embodiment, some hardware components of the user device **101-1** are recognised and used by the OS **214** as connected external devices, when OS **214** runs on a different device which is connected to user device **101-1**.

**[0107]** In some of the embodiments where device interoperability system **200** is installed via integration into user device **101-1** or as an app on user device **101-1**, as part of communications module **213**, an external wireless adapter is added to user device **101-1** to provide additional communications capabilities not available on user device **101-1**, so as to improve performance and/or energy efficiency. This external wireless adapter works with, for example, an integrated controller which is already present on user device **101-1**. Then, the communications module **213** comprises the integrated controller and the external wireless adapter of user device **101-1**. For example, a USB wireless adapter based on WiGig or Li-Fi communication technology is plugged into a USB port of user device **101-1**. This plugged in wireless adapter will interact with an integrated USB controller already present on user device **101-1**. Then, the communications module **213** comprises this integrated USB controller and plugged in USB wireless adapter. These added components provide additional communication technology which is not initially available on user device **101-1** to improve performance and/or energy efficiency.

**[0108]** An example of the operation of device interoperability system **200** will be detailed below with reference to a user device, specifically user device **101-1**. The descriptions below are applicable to a variety of situations including, for example:

**[0109]** device interoperability system **200** installed on a gadget such as gadget **210**;

**[0110]** device interoperability system **200** is installed via integration into one of user devices different from user device **101-1**, for example user devices **101-2** to **101**-N; and

**[0111]** device interoperability system **200** is installed as an app on one of user devices different from user device **101-1**, for example user devices **101-2** to **101**-N.

**[0112]** Additionally, there is a need to determine if user device **101-1** will operate in either "stand-alone" or "device interoperability system" mode. In stand-alone mode, the user device **101-1** runs its native OS. In interoperability system mode, the user device **101-1** is connected to device interoperability system **200** and runs OS **214**. In a further embodiment, the user device **101-1** is switchable between stand-alone and interoperability system modes.

**[0113]** An example algorithm for switching between stand-alone and interoperability system modes comprising:

**[0114]** establishment of connection **201-1** in the embodiments where connection **201-1** is a secured connection, and

**[0115]** subsequent booting of the appropriate OS depending on whether stand-alone or interoperability modes is used,

is provided in FIG. **3**.

**[0116]** In FIG. **3**, in step **301**, user device **101-1** is switched on. In step **302**, prior to establishing connection **201-1**, user device **101-1** presents the user with the option of setting up for interoperability system mode.

**[0117]** If the user accepts the option of setting up for interoperability system mode within a predetermined period in step **303**, then in step **304** the user performs authentication. In one embodiment, in step **304** the user enters a unique string, password or passphrase specific to the device interoperability system **200**. In another embodiment, in step **304** the user enters a login name and a password specific to the OS **214**. In yet another embodiment, the user uses login details from another social media site, or web mail site, for example, Facebook®, LinkedIn®, Twitter®, Google®, Gmail®, or others. Additional steps are also possible for authentication. In another embodiment, the user is additionally asked to recognise a combination of letters, numbers and symbols in an image and enter the combination into a box. An example of such a test is the Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) test. In another embodiment, the user is asked a security question, to which the user only knows the answer. In yet another embodiment, the user may be asked additional personal information, such as date of birth and home address. In another embodiment, the user is asked to take a picture of himself or herself and the device interoperability system **200** will match the image to a pre-stored image. In yet another embodiment other biometric measures such as fingerprints scanning are used. The authentication data is used as a pre-shared key and authentication/encryption keys for encrypted connection are built.

**[0118]** In step **306**, the user device saves the authentication/encryption keys and connection parameters for future use.

**[0119]** In step **307**, connection **201-1** is established.

**[0120]** In step **308**, OS **214** boots on user device **101-1**.

**[0121]** In step **309**, the user device **101-1** works in device interoperability system mode.

**[0122]** If the user does not accept the option of setting up for interoperability system mode in step **303**, then in step **305**, user device **101-1** determines if it is already set up for interoperability system mode. If in step **305** the user device **101-1** is already set up, then in step **310** the user device tries to establish a connection with device interoperability system **200** using the stored authentication keys and parameters.

**[0123]** Following on from step **310**, if the connection establishment is successful in step **311** then the OS **214** boots on user device **101-1** (step **308**), and the user device **101-1** works in device interoperability system mode (step **309**).

**[0124]** If the connection is unsuccessful in step **311**, then user device **101-1** loads its own OS in step **312**. In step **313**, the user device **101-1** works in stand-alone mode.

**[0125]** If in step **305** the user device is not already set up, then the user device **101-1** loads its own OS (step **312**) and works in stand-alone mode (step **313**).

**[0126]** In one embodiment, in order to improve speed of operation and to reduce the amount of data transmitted through a connection such as connection **201-1**, the swap file or swap partition of OS **214** is placed on the storage of the user device **101-1**.

**[0127]** In one embodiment, in order to improve speed of operation, caching is performed by, for example, setting aside a portion of the storage of the connected user device for a cache. In one embodiment, when OS **214** is booted, it will determine if there is a cache on the connected user device. Caching operations will be discussed in detail further below.

**[0128]** In one embodiment, at least some portion of the local storage of a user device connected to device interoperability system **200** is used by OS **214** to store data intended for use only on this particular device. An example is where user device **101-1** is a desktop used by user **100** specifically for running high resource demand applications such as video games. Then, some of the data necessary for running the high resource demand application is stored on the local storage of user device **101-1** instead of storage **212**. In a further embodiment, the portion of the user device **101-1** storage which is to be used is recognized by OS **214** as a connected additional drive and presented accordingly.

**[0129]** In one embodiment, a portion of the local storage space of a user device connected to device interoperability system **200** is used by OS **214** to perform a backup of at least some of the data stored in storage **212**. The amount of data which is backed up depends on the available capacity of the local storage of the user device. In a further embodiment, the backup is performed using a plurality of user devices. That is, data is backed up from storage **212** to a portion of each local storage space corresponding to each of the plurality of user devices.

**[0130]** In yet another embodiment, a backup status is associated with each portion of data stored in storage **212**. Then when a backup operation is performed on that particular portion of data, the backup status is updated.

**[0131]** In a further embodiment, at least some of the data which is stored on the storage of user device **101-1** for either caching, swapping, expanding storage, backups or any combination of these purposes is placed in one or more partitions set up on the storage of user device **101-1**. In another embodiment, the data which is stored on the storage of user device **101-1** for either caching, expanding storage, backups or any combination of these purposes is placed in one or more file-containers created in an existing partition of user device **101-1**. This eliminates the need for repartitioning or erasing any data from the storage of user device **101-1**.

**[0132]** In a further embodiment, the data which is stored on the storage of user device **101-1** for either caching, swapping, expanding storage, backups or any combination of these purposes is encrypted. The decryption keys are stored and managed by OS **214** thus preventing unauthorized access to the data.

**[0133]** In one embodiment, the OS **214** is able to switch between different hardware configurations during booting, such as in step **308** of FIG. **3**. An example algorithm for switching between different hardware configurations is provided in FIG. **3**B.

**[0134]** When the OS **214** is booted on a user device such as user device **101-1**, then in step **3**B-**01** OS **214** identifies the user device. In step **3**B-**02**, OS **214** determines whether it has stored the configuration set corresponding to the identified user device in storage **212**. If yes, then in step **3**B-**03** OS **214** uses the correct set of drivers and settings for the identified user device. The user device then works in device interoperability system mode in step **3**B-**07**.

**[0135]** If in step **3**B-**02** OS **214** is unable to find the configuration set corresponding to the identified user device in storage **212**, then in step **3**B-**04** OS **214** will detect all hardware on this user device and install needed drivers automatically.

**[0136]** In step **3**B-**05**, OS **214** prompts the user to enter one or more answers to one or more questions to determine how the user device storage will be used for the functioning of OS **214**. Example questions include:

**[0137]** will the user device storage be used for caching?

**[0138]** will the user device storage be used for backups?

**[0139]** will the user device storage be used to provide additional storage space for OS **214** for its use?

**[0140]** how much space will be reserved for specified above purposes?

**[0141]** In step **3**B-**06**, OS **214** will save the configuration set and reboot if necessary, before proceeding to work in device interoperability system mode in step **3**B-**07**.

**[0142]** As mentioned previously, example embodiments of caching operations are discussed in detail below with reference to FIGS. **4**, **4**B and **4**C.

**[0143]** FIG. **4** shows an example flow when a received read or write operation request is processed by OS **214**. In this example OS **214** is stored on the device where device interoperability system **200** is installed. This device is connected to user device **101-1**. Also, a cache has been set up on user device **101-1**.

**[0144]** In step **401**, the request type is determined by OS **214**.

**[0145]** If the request is determined to be for a read operation, then in step **403** the OS **214** determines if the cache contains the requested data. In one embodiment, OS **214** accesses the cache service database to determine if the cache set up on user device **101-1** contains the requested data. The cache service database describes modification times of the versions of the files stored in at least one of the caches of the user devices **101-1** to **101**-N, and modification times of the original versions of those files stored in the storage **212**. Determination if the cache contains the requested data is performed by comparison of those modification times. The cache service database is stored in storage **212**.

**[0146]** Table 1 shows an example of a cache service database:

TABLE 1

Example of Cache Service Database

| File [1C-01] | Storage 212 [1C-02] | Cache #1 [1C-03] | Cache #2 [1C-04] | . . . |
|---|---|---|---|---|
| C:\path1\file1 [1R-01] | dd/mm/yy HH:MM:SS [1R-01, 1C-02] | dd/mm/yy HH:MM:SS [1R-01, 1C-03] | dd/mm/yy HH:MM:SS [1R-01, 1C-04] | |

TABLE 1-continued

Example of Cache Service Database

| File [1C-01] | Storage 212 [1C-02] | Cache #1 [1C-03] | Cache #2 [1C-04] | . . . |
|---|---|---|---|---|
| C:\path2\file2 [1R-02] | dd/mm/yy HH:MM:SS [1R-02, 1C-02] | n/a [1R-02, 1C-03] | dd/mm/yy HH:MM:SS [1R-02, 1C-04] | |
| . . . | | | | |

**[0147]** In Table 1, column 1C-01 represents the files. Each file corresponds to a separate row of Table 1. With reference to Table 1, file 1 is assigned to row 1R-01, file 2 is assigned to row 1R-02 and so on.

**[0148]** Column 1C-02 of Table 1 represents the modification times of the original versions of those files stored in the storage **212** and in at least one of the caches on the user devices **101-1** to **101**-N. Then, referring to Table 1:

**[0149]** Cell [1R-01, 1C-02] represents the modification time of the original version of file 1 in storage **212**,

**[0150]** Cell [1R-02, 1C-02] represents the modification time of the original version of file 2 in storage **212**,

**[0151]** Columns 1C-03 and 1C-04 represent the modification times of the versions of the file in the respective caches. For example, column 1C-03 corresponds to the cache 1 stored on user device **101-1**, 1C-04 corresponds to cache 2 stored on user device **101-2**, and so on. Then:

**[0152]** Cell [1R-01, 1C-03] represents the modification time of the version of file 1 in cache 1,

**[0153]** Cell [1R-01, 1C-04] represents the modification time of the version of file 1 in cache 2,

**[0154]** Cell [1R-02, 1C-03] represents the modification time of the version of file 2 in cache 1, and

**[0155]** Cell [1R-02, 1C-04] represents the modification time of the version of file 2 in cache 2.

**[0156]** There are a variety of formats which can be used to represent the times in the cache service database. One example format is a two-digit representation of day/month/year followed by hour:minute:second or "dd/mm/yy HH:MM:SS".

**[0157]** There is a variety of other information which can be also included in the cache service database. For example, the cache service database can also include file sizes and checksums for data integrity checks.

**[0158]** In another embodiment the cache service database is based on file checksums instead of file modification times. Then, the cache service database describes checksums of the versions of the files stored in at least one of the caches of the user devices **101-1** to **101**-N, and checksums of the original versions of those files stored in the storage **212**. Determination if the cache contains the requested data is performed by comparison of those checksums.

**[0159]** If the data cannot be found on the cache in step **404**, or the data on the cache has a modification time which is different from the modification time of the corresponding data on storage **212** (step **405**), or the checksums are different; then OS **214** retrieves the data from storage **212** (step **406**). In step **408**, the retrieved data is then written into the cache, so that subsequent data read operations are performed using the cache. This also has the advantage of reducing the power consumption of the device where device interoperability system **200** is installed, as data does not have to be transmitted from this device to user device **101-1** via connection **201-1**. In step **409**, the cache service database is also updated.

**[0160]** If, in step **404** the data is found on the cache, and the data on the cache matches the corresponding data on storage **212** (step **405**); then in step **407** the data is read from the cache.

**[0161]** In a further embodiment, if the request type is determined to be a write operation in step **401**, then in step **402** OS **214** performs a data write operation. In one embodiment, this data write operation is performed in write-through mode. Then, following from the OS **214** writing the data to storage **212** via connection **201-1** in step **402**, OS **214** also writes data to the user device **101-1** cache in step **408**. In step **409**, the cache service database is also updated.

**[0162]** In another embodiment, one or more additional checks are performed to determine whether the data should be written to the cache. FIG. **4**B illustrates an example of an embodiment. Steps **4**B-**01** to **4**B-**07** are similar to steps **401**-**407** in FIG. **4**. In step **4**B-**08**, additional checks are used to determine if the data is suitable for caching. Examples of the factors which are examined to determine if the data is suitable for caching include:

**[0163]** Maximum capacity of the cache,

**[0164]** Utilization of the cache capacity,

**[0165]** Size of data item,

**[0166]** Usage frequency of data item,

**[0167]** Time expiration of data item,

**[0168]** Currently running applications,

**[0169]** Previously collected data usage patterns, and

**[0170]** Device type of user device **101-1**.

**[0171]** If the data is determined to be suitable for caching in step **4**B-**08**, then in step **4**B-**10** the data is written to the cache and in step **4**B-**11** the cache service database is updated.

**[0172]** In one embodiment, the OS **214** performs additional cache servicing functions, for example:

**[0173]** cache defragmentation, or

**[0174]** deletion of less cache-suitable data to free up space.

**[0175]** Then, the above described factors used in step **4**B-**08** are used to optimize the performance of these additional cache servicing functions as well.

**[0176]** In a further embodiment, if the data is determined to not be suitable for caching in step **4**B-**08**, then in step **4**B-**09** an additional check to determine the necessity of updating of the cache service database is performed. For example, if the data is determined not to reside in any cache of any user device, then it is unnecessary to update the cache service database.

**[0177]** In one embodiment, when user device **101-1** runs OS **214**, data is prefetched from storage **212** and used to update the user device **101-1** cache. That is, data is fetched from storage **212** and transmitted to the user device **101-1** cache in readiness for future use.

**[0178]** FIG. **4**C shows an example embodiment of prefetching. In step **4**C-**01**, data stored in storage **212** is compared by the OS **214** to the data stored in the user device **101-1** cache. The OS **214** performs the data comparison by comparing the information from the cache service database corresponding to user device **101-1**, to the information stored in file system of the storage **212**.

**[0179]** If the data stored in the connected cache is determined not to match the data stored in storage **212** in step

4C-02, then in step 4C-03 the OS 214 determines which of the one or more portions of data stored in storage 212 are different compared to the data on the user device 101-1 cache.

[0180] In a further embodiment, in step 4C-04, OS 214 selectively prefetches one or more portions of data stored in storage 212 which are different from the data stored in the cache of user device 101-1.

[0181] The selection and prioritization of data depends on several factors:

[0182] Connection of the device where device interoperability system 200 is installed to a power source,

[0183] Charge level of battery 211,

[0184] Total capacity of battery 211,

[0185] Current utilization of connection 201-1,

[0186] Current user activity,

[0187] Current hardware utilization of user device 101-1,

[0188] Maximum capacity of the connected cache,

[0189] Utilization of the cache capacity,

[0190] Size of data portion,

[0191] Usage frequency of data item,

[0192] Time expiration of data item,

[0193] Currently running applications,

[0194] Previously collected data usage patterns, and

[0195] Device type of user device 101-1.

[0196] Then in step 4C-05, the cache service database is updated accordingly based on the data stored in the user device 101-1 cache.

[0197] In one embodiment, security measures are used so as to reduce the risk of a malicious party gaining access to device interoperability system 200. For example, in one embodiment access to device interoperability system 200 is secured using biometric measures such as fingerprints scanning or facial recognition.

[0198] In other embodiments, multi-factor authentication is used to secure device interoperability system 200. In some embodiments, a multi-factor authentication process is based on the following factors, or answering the following questions:

[0199] The knowledge factor, or what I know;

[0200] The possession factor, or what I have; and

[0201] The inherence factor, or what I am.

[0202] In one embodiment, the OS 214 is able to pause its operation if the connection 201-1 is lost, and resume operation immediately when the connection is reestablished.

[0203] In one embodiment, OS 214 includes one or more kernels corresponding to one or more architectures. For example, OS 214 includes kernels for the x86 and ARM architectures. Then depending on the architecture of the connected user device, the appropriate kernel is used automatically. This behavior is completely transparent for the user.

[0204] In one embodiment, a graphical user interface (GUI) is generated on user device 101-1 to enable the user to interact and interface with user device 101-1 including OS 214. In one embodiment, the OS 214 automatically optimizes and adapts the GUI according to the following factors:

[0205] physical form-factor of the user device 101-1. For example, what type of device is user device 101-1? Is it a laptop, tablet, TV set, game console or integrated in-car system?

[0206] number and size of screens associated with the user device 101-1;

[0207] screen resolution; and

[0208] input methods. For example, is the input device a keyboard and mouse, touchscreen, infrared remote control or gamepad?

[0209] Examples of GUI optimizations and adaptations include:

[0210] adjusting the size and placement of GUI control elements such as buttons and checkboxes;

[0211] adjusting the size and placement of windows;

[0212] enabling or disabling specific text input methods such as on-screen keyboard or voice text input; and

[0213] enabling or disabling GUI parts for device-specific features such as controls for in-car air conditioning system.

[0214] In one embodiment, OS 214 is only able to work on one connected user device at a time. An example is when OS 214 is running on user device 101-1. Then, to work on a different user device such as user device 101-2 after establishing connection 201-2, in one embodiment OS 214 must be shut down on user device 101-1, then booted on user device 101-2. In another embodiment, OS 214 operation on user device 101-1 is first paused. Then OS 214 is either booted or, if it was previously paused, resumed on user device 101-2.

[0215] In another embodiment, OS 214 is able to work with a plurality of user devices such as, for example, user devices 101-1, 101-2 and 101-3. In order to enable this, in an embodiment communications module 213 is able to establish and simultaneously maintain connections 201-1, 201-2 and 201-3 with user devices 101-1, 101-2 and 101-3 respectively. Then, user devices 101-1, 101-2 and 101-3 are simultaneously connected to the device interoperability system 200 and each one of these user devices runs its instance of OS 214 in parallel with each other. In one embodiment, the transmission capacity of communications module 213 is balanced between connections 201-1, 201-2 and 201-3 according to the current utilization of each connection.

[0216] In a further embodiment, different instances of OS 214 which are simultaneously running on user devices 101-1, 101-2 and 101-3 use the distributed lock management approach to coordinate concurrent access to the storage 212. For example, the lock managers of all three instances of OS 214 which are running on the user devices 101-1, 101-2 and 101-3, use the same lock database which is distributed among these instances by means of device interoperability system 200 and connections 201-1, 201-2 and 201-3.

[0217] In one embodiment, the interoperability system 200 is used by the different instances of OS 214 which are simultaneously running on different user devices to exchange some details about their current status. This, for example, includes:

[0218] number and device types of simultaneously working user devices,

[0219] status of important OS service functions, for example, an OS update process,

[0220] current user activity, and

[0221] currently running applications.

[0222] This data is used by every running instance of OS 214 to coordinate and optimize its service functions. For example, when three instances of OS 214 are running on user devices 101-1, 101-2 and 101-3, coordination is performed to ensure that the OS update process is not running simultaneously on all three devices. In a further embodiment, this data is used to prioritize the balancing of the

transmission capacity of communications module **213** between established connections **201-1** to **201-3**. For example, a higher priority is given to that user device which user **100** currently uses.

**[0223]** In yet another embodiment, the OS **214** supports migration of running applications between OS instances running on different user devices. With reference to the example above, OS **214** supports the ability to move a currently running application from user device **101-2** to user device **101-3**. After the migration, the application continues to have access to any previously opened files. In a further embodiment, the data described previously is used to present more details to a user if the user opts to migrate applications and the connections **201-2** and **201-3** are used to facilitate the migration process.

**[0224]** The use of device interoperability system **200** offers several other advantages. In some embodiments, device interoperability system **200** is used in conjunction with cloud-based data synchronization capabilities. For example, if cloud-based services are used for synchronization of data between different user devices, device interoperability system **200** reduces the necessity for user devices to connect to the cloud to perform data synchronization. Instead, the user devices use data from storage **212**. This reduces the utilization of the cloud connection with the user devices. Furthermore, in some embodiments, the device interoperability system **200** ensures data availability in case cloud connectivity is lost or not available, as the user devices can retrieve data from storage **212**. In some embodiments, intelligent approaches to ensuring availability of data which is most likely to be relevant to a user are employed. These include, for example, approaches based on:

**[0225]** Temporal locality: Data which was most recently used on a user device is stored on storage **212** as it is likely that the user device will use this data again in the near future.

**[0226]** Spatial locality: Data sets which occupy memory locations close to recently used data are stored on storage **212** as it is likely that the user device will use these data sets in the near future.

**[0227]** Branch locality: In cases where there are multiple possible outcomes from conditional branching instructions, then data related to each of these outcomes are stored on storage **212** as it is likely that the user device will use this data.

**[0228]** Probabilistic analysis of user interactions with user devices: For example, if there is a high probability that a user will use one or more data sets either in conjunction with or after using a particular program, then these data sets are stored on storage **212**.

**[0229]** In some embodiments, some user data is stored on storage **212** but not within the cloud. This capability is useful if, for example, users want to keep control of sensitive data such as private keys for blockchain-enabled mobile services. This feature enables users to keep control of their sensitive data without exposure to potential security breaches in the cloud.

**[0230]** The security of this feature is enhanced by the addition of device interoperability system **200**. In particular, this enhancement is achieved by running device interoperability system **200** on a device or gadget where storage **212** has these secure storage capabilities, and allowing only user devices that are connected to device interoperability system **200** and have been booted up using OS **214** to access the data stored within storage **212**. As will be seen below, this enables the creation of a more secure and private ecosystem.

**[0231]** For example: In the case where private keys for blockchain-enabled mobile services are stored on storage **212**, only user devices that are connected to device interoperability system **200** and have been booted up using OS **214** are able to access these private keys. This is useful, for example, to ensure that the user is able to securely perform cryptocurrency transactions.

**[0232]** Sensitive data can be further secured through other means. For example, in some embodiments, as explained previously, the sensitive user data is encrypted prior to being stored in storage **212**. In some of these embodiments, the user selects the type of encryption to be used.

**[0233]** In other embodiments, sensitive user data is not accessible directly to user devices that have been connected to device interoperability system **200** and have been booted up using OS **214**. Instead, only the results of processing or operations performed by, for example, one or more programmes which are part of programmes and data **216** residing on storage **212** and which uses the sensitive data, are made available to the user devices. For example, if the user using user device **101-3** wants to perform cryptocurrency transactions and needs to access private keys to sign transactions, then the transactions which require signing are transmitted from the user device **101-3** over connection **201-3** to the device interoperability system **200**. At device interoperability system **200**, the transactions are signed using one or more programmes resident on storage **212**. The signed transactions are then transmitted back over connection **201-3** to the user device **101-3**. This way, user device **101-3** does not access the sensitive user data at all.

**[0234]** In some embodiments, the availability of the sensitive user data or the results of processing or operations which use the sensitive data is based on a security level associated with the user device. This is illustrated with reference to user device **101-3**. For example, in some embodiments, the user device **101-3** is considered to have a low security level if it is publicly accessible. If user device **101-3** is only privately accessible and access is secured using, for example, two authentication factors as described previously, it is considered to have a very high security level.

**[0235]** In some embodiments, storage **212** comprises several storage sub-areas, wherein each sub-area has a different associated security level. An example embodiment is shown in FIG. **5**, where storage **212** comprises one or more sub-areas **510-1** to **510-4**. Then, for example:

**[0236]** Sub-area **510-1** has the highest level of security,

**[0237]** Sub-area **510-2** has the second highest level of security,

**[0238]** Sub-area **510-3** has the third highest level of security, and

**[0239]** Sub-area **510-4** has the lowest level of security.

**[0240]** The Samsung Galaxy S10 is an example of this, as it has a secure storage sub-area to store private keys for blockchain-enabled mobile services and a less secure storage sub-area. This can also be used to differentiate the level of access of a user device to data. For example:

**[0241]** the most sensitive data is stored in the most secure storage sub-area,

**[0242]** the next most sensitive data is stored in the next most secure storage sub-area, and

**[0243]** so on.

**[0244]** The implementation of sub-areas within storage **212** into sub-areas can be carried out in a variety of ways. In some embodiments, the implementation is performed physically, that is, each sub-area corresponds to a different physical storage area. In some embodiments, the implementation is performed virtually, that is, a physical storage area is partitioned into different sub-areas. In yet other embodiments, a combination of virtual and physical implementations is used.

**[0245]** In some embodiments, the above concepts are combined to create a hierarchical or differentiated system of secure storage for a user's data. This hierarchy has a plurality of levels, wherein each level of the hierarchy corresponds to a different level of data sensitivity and therefore required security.

**[0246]** FIG. **6** shows an example embodiment of such a hierarchy **600**. In hierarchy **600**,

**[0247]** Sensitivity level **601** corresponds to the user's most sensitive data, and the highest level of security **611** is assigned to data with sensitivity level **601**.

**[0248]** Sensitivity level **602** corresponds to the user's second most sensitive data, and the second highest level of security **612** is assigned to data with sensitivity level **602**.

**[0249]** Sensitivity level **603** corresponds to the user's third most sensitive data, and the third highest level of security **613** is assigned to data with sensitivity level **603**, and

**[0250]** Sensitivity level **604** corresponds to the user's least sensitive data, and the lowest level of security **614** is assigned to data with sensitivity level **604**.

**[0251]** Then the accessibility to the data is based on the security level which has been assigned to the data. An example of this is demonstrated below with reference to FIGS. **5** and **6**. For example:

**[0252]** Data with security level **611** is not accessible to user devices and is stored in storage sub-area **510-1**. The results of processing or operations which use the data are accessible to a user device, if

**[0253]** the user device has been connected to device interoperability system **200**,

**[0254]** the user device was booted by OS **214**, and

**[0255]** the security level associated with the user device is very high;

**[0256]** Data with security level **612** is stored in storage sub-area **510-2**. It is accessible to a user device, if

**[0257]** the user device has been connected to device interoperability system **200**,

**[0258]** the user device was booted by OS **214**, and

**[0259]** the security level associated with the user device is high;

**[0260]** Data with security level **613** is stored in storage sub-area **510-3**. It is accessible to a user device if

**[0261]** the user device has been connected to device interoperability system **200**,

**[0262]** the user device was booted by OS **214**, and

**[0263]** the security level associated with the user device is medium; and

**[0264]** Data with security level **614** is stored in the cloud.

**[0265]** User data can be assigned to one of sensitivity levels **601-604** using different techniques. In some embodiments, assignment is based on user inputs on a user interface presented to the user at a user device. The user interface is, for example, a GUI. An example embodiment is as follows: A user assigns data to one of the levels by selecting a sensitivity setting of Very High, High, Medium, Low corresponding to levels **601-604**. Then, based on this setting, one of the security levels described above is assigned to the data.

**[0266]** In other embodiments, user data is assigned to one of levels **601-604** based on the type of data. For example, sensitivity level **601** may be assigned to data related to an ultra-secure cryptocurrency "cold wallet" such as cryptocurrency, public and private keys as well as signing private keys for cryptocurrency transactions. Sensitivity level **602** may be assigned to user Personal Identification Numbers (PINs) and passwords for financial applications. Sensitivity level **603** is assigned to user media files that the user has indicated are sensitive. Finally, sensitivity level **604** is assigned to other user data which the user has allowed many cloud-based applications to use.

**[0267]** The above shows an embodiment where one level of security is assigned based on the sensitivity level of the data. In some embodiments, more than one level of security to be assigned based on the sensitivity level of the data. In some of these embodiments, based on the sensitivity level of the data, a minimum level of security is assigned. Then, any level of security either at or above that minimum level can be assigned to the data. For example, the minimum level for data with sensitivity level **602** is set to security level **612**. Therefore, security levels **611** and **612** can be assigned to the data. Similarly, the minimum level for data with sensitivity level **603** is set to security level **613**. Then security levels **611**, **612** and **613** can be assigned to that data.

**[0268]** The above embodiments enable the creation of a secure, private ecosystem where sensitive data is stored within storage **212** of device interoperability system **200**, and only devices which connect to device interoperability system **200** and are booted by OS **214** can access this data. Furthermore, the above details embodiments for a hierarchical or differentiated system of secure storage.

**[0269]** The use of device interoperability system **200** also offers advantages for IoT-enabled user devices. Similar to as with cloud-based services, device interoperability system **200** reduces the need to connect to the cloud to perform data synchronization. Furthermore it reduces the difficulty of having to maintain separate cloud credentials and device settings for user devices.

**[0270]** The hierarchical or differentiated system of secure storage mentioned above is of particular importance for IoT-enabled devices, as many of these devices are publicly accessible and may be difficult to monitor, thereby reducing the level of security associated with these devices. By using a hierarchical or differentiated system of secure storage, such devices can be used as part of a secure and private ecosystem without jeopardizing the overall level of security and privacy of the ecosystem.

**[0271]** It is also possible to perform data restore in the event of damage or loss of the device where device interoperability system **200** is installed. As previously described, in some embodiments OS **214** backs up data from storage **212** to a portion of each local storage space corresponding to each of the user devices connected to device interoperability system **200**. Embodiments to perform caching were also previously described above. Then, in some embodiments, OS **214** uses the data stored in the one or more caches corresponding to the user devices connected to device

interoperability system **200**, in conjunction with the data backed up on a portion of the local storage space of the user devices connected to device interoperability system **200**, to perform a data restore.

**[0272]** FIGS. **7**A-**7**C illustrate an exemplary embodiment of a data restore process performed by OS **214** which uses the data stored in the one or more caches and the data backed up on the user devices to perform a data restore. In step **701**, OS **214** checks to see if device interoperability system **200** is connected to any of the user devices. If no, then in step **702**, the user is prompted to connect device interoperability system **200** to a user device. If device interoperability system **200** is connected to a user device, or after connection to a user device in step **702**, then in step **703** OS **214** checks to see if data from a previous backup operation is available on the user device. If yes, then in step **704**, OS **214** compares data from the backup with data stored in storage **212**. If no, then OS **214** progresses to step **708** which will be explained further below.

**[0273]** Once step **704** is completed, in step **705** OS **214** determines if any data from the backup stored on the user device is missing from storage **212**. If yes, then in step **706** OS **214** determines the data which is missing from storage **212**. Once step **706** is completed, then data is restored from the backup on the user device to storage **212** in step **707**.

**[0274]** If in step **705** OS **214** determines that there is no data from the backup on the user device missing from storage **212**, then in step **715** of FIG. **7**B OS **214** determines if the data from the backup on the user device is different to the data stored on storage **212**. If in step **715**, OS **214** determines that there is no difference, then OS **214** progresses on to step **708** of FIG. **7**A. If OS **214** determines that there is a difference, then in step **716** OS **214** determines whether the data in the backup stored on the user device is more up to date than the data stored in storage **212**. If the data in the backup stored on the user device is less up to date, then OS **214** progresses on to step **708** of FIG. **7**A. If the OS **214** determines in step **716** that the data in the backup stored on the user device is more up to date, then OS **214** restores the up to date data from the backup to storage **212**. Once step **717** is completed, OS **214** progresses to step **708** of FIG. **7**A.

**[0275]** In step **708**, OS **214** determines if there is a cache available on the user device storage. If there is no cache available, then OS **214** progresses to step **713** which will be explained further below. If there is a cache available, then in step **709** OS **214** compares data from the cache with the data stored in storage **212** to see if there is data present in the cache which is missing from storage **212**. In one embodiment, the cache service database which is stored on the user device is used to determine if there is data present on the cache which is missing from storage **212** in step **709**. If OS **214** determines in step **710** that there is data missing, then in step **711**, OS **214** determines which portions of data are missing on storage **212**. Once step **711** is completed, then in step **712** OS **214** restores the missing portions of data from the cache to storage **212**. OS **214** then progresses to step **713**.

**[0276]** If OS **214** determines in step **710** that there is no data missing, then in step **718** of FIG. **7**C, OS **214** determines if the data stored in storage **212** is different from the data from the cache stored on the user device. In some embodiments, the cache service database which is stored on the user device is used to determine if the data present on the cache is different from the data stored on storage **212** in step **718**. If there is no difference, then OS **214** progresses to step **713** of FIG. **7**A. If there is a difference, then in step **719** OS **214** determines if the data stored in the cache is more up to date than the data stored in storage **212**. If the data stored in the cache is less up to date, then OS **214** progresses to step **713**. If it is more up to date, then in step **720** OS **214** restores the up to date data from the cache to storage **212**. Once step **720** is completed, OS **214** progresses to step **713** of FIG. **7**A.

**[0277]** In step **713** of FIG. **7**A, OS **214** determines if there is another user device for device interoperability system **200** to connect to. If there is another user device available for connection, then OS **214** prompts the user to connect device interoperability system **200** to the other user device in step **721**. After completing step **721**, OS **214** returns to perform step **703**. If there is no other user device available, then OS **214** stops the restore process in step **714**.

**[0278]** Variations on the above are also possible. For example, in some embodiments, only data stored in the cache is used by OS **214** for data restore. An exemplary embodiment is illustrated with reference to FIGS. **8**A and **8**B. In step **801**, similar to step **701**, OS **214** checks to see if device interoperability system **200** is connected to any of the user devices. If no, then in step **802**, similar to step **702**, OS **214** prompts the user to connect device interoperability system **200** to a user device. If device interoperability system **200** is connected to a user device, or after connection to a user device in step **802**, then in step **808** OS **214** determines if there is a cache available on the user device storage, similar to previously disclosed step **708**.

**[0279]** If there is no cache available, then OS **214** progresses to step **813** which is similar to step **713**. If there is a cache available, then in step **809** (similar to step **709**) OS **214** compares data from the cache with the data stored in storage **212** to see if there is data present in the cache which is missing from storage **212**. In one embodiment, the cache service database which is stored on the user device is used to determine if there is data present on the cache which is missing from storage **212** in step **809**. If OS **214** determines in step **810** (similar to step **710**) that there is data missing, then in step **811** (similar to step **711**), OS **214** determines which portions of data are missing on storage **212**. Once step **811** is completed, then in step **812** (similar to step **712**) OS **214** restores the missing portions of data from the cache to storage **212**. OS **214** then progresses to step **813**.

**[0280]** If OS **214** determines in step **810** that there is no data missing, then in step **818** (similar to step **718** of FIG. **7**C) of FIG. **8**B, OS **214** determines if the data stored in storage **212** is different from the data from the cache stored on the user device. In some embodiments, the cache service database which is stored on the user device is used to determine if the data present on the cache is different from the data stored on storage **212** in step **818**. If there is no difference, then OS **214** progresses to step **813** of FIG. **8**A. If there is a difference, then in step **819** (similar to step **719** of FIG. **7**C) OS **214** determines if the data stored in the cache is more up to date than the data stored in storage **212**. If the data stored in the cache is less up to date, then OS **214** progresses to step **813**. If it is more up to date, then in step **820** (similar to step **720** of FIG. **7**C) OS **214** restores the up to date data from the cache to storage **212**. Once step **820** is completed, OS **214** progresses to step **813** of FIG. **8**A.

**[0281]** In step **813** of FIG. **8**A, OS **214** determines if there is another user device for device interoperability system **200** to connect to. If there is another user device available for connection, then OS **214** prompts the user to connect device

interoperability system **200** to the other user device in step **821** (similar to step **721**). After completing step **821**, OS **214** returns to perform step **808**. If there is no other user device available, then OS **214** stops the restore process in step **814**, similar as in what would have happened in step **714**.

**[0282]** As previously explained and as shown in step **4**C-**04** of FIG. **4**C, OS **214** selectively pre-fetches one or more portions of data stored in storage **212** which are different from the data stored in the cache of a user device, and the selection and prioritization of data depends on several factors. In additional embodiments, these factors include the previously discussed backup status associated with each portion of data.

**[0283]** In one embodiment, when a portion of the data is about to be deleted from the cache, an additional check of its backup status is performed by OS **214**. For example, if there are no more copies of particular portion of the data stored on other user devices or at other backup locations then this portion of data will not be deleted from the cache.

**[0284]** As explained above, the OS **214** supports migration of running applications between OS instances running on different user devices. In some embodiments, migration is performed using a "push"-based technique, that is, where migration is initiated at a source user device, and an application is then pushed from the source device to a destination user device. In further embodiments, a GUI is generated on a source user device to allow the user to select an application for migration, and the destination user device to which the application will be migrated to. An exemplary illustration is shown in FIG. **9**A. In FIG. **9**A, GUI **900** shows a list of running applications **901** on the source user device **101-2**. List of running applications **901** has entries **902-1** to **902-3**, each corresponding to a running application. When any of these entries are selected, an option is presented to the user to enable the user to decide which destination device the application is to be migrated to. As shown in FIG. **9**A, when an entry such as entry **902-1** is selected, option **903** with the prompt "MOVE THIS APP TO" is presented to the user, along with a list of connected user devices **905**. List **905** comprises, for example, items **907-1** to **907-3**, wherein each item corresponds to one of other user devices **101-3**, **101-4** and **101-5** currently connected to the device interoperability system **200**. In some embodiments, each item consists of an icon representing the type of the user device and the device name. For example, item **907-1** corresponds to user device **101-3** which is a laptop. An exemplary illustration of item **907-1** is shown in FIG. **9**B. In FIG. **9**B item **907-1** comprises icon **908-1** to represent a laptop, and name label **908-2** comprising, for example, "My Laptop".

**[0285]** In some embodiments, migration is performed using a "pull"-based technique, that is, where migration is initiated at the destination user device, and an application is then pulled from a source user device for migration to the destination user device. In further embodiments, a GUI is generated on a destination user device to allow the user to select a source user device where an application will be migrated from, and an application for migration. An exemplary illustration is shown in FIG. **10**A. In FIG. **10**A, GUI **1000** presents an option **1004** with the prompt "CHOOSE A DEVICE", along with a list of source devices **1001**. List of source user devices **1001** has entries **1002-1** to **1002-3**, corresponding to source user devices **101-2**, **101-4** and **101-5** connected to device interoperability system **200**. In some embodiments, each entry comprises an icon representing the type of the user device and the device name. For example, entry **1002-1** corresponds to user device **101-2** which is a smartphone. An exemplary illustration of entry **1002-1** is shown in FIG. **10**B. In FIG. **10**B entry **1002-1** comprises icon **1008-1** to represent a phone, and name label **1008-2** comprising, for example, "My Phone". When any of these entries are selected, an option is presented to the user to enable the user to decide which running application should be migrated from the selected source user device. As shown in FIG. **10**A, when an entry such as entry **1002-1** is selected, option **1003** with the prompt "MIGRATE APP FROM OTHER DEVICE" is presented to the user, along with a list of applications **1005** running on user device **101-2**. List **1005** comprises, for example, items **1007-1** to **1007-3**, wherein each item corresponds to one of the applications which are running on source user device **101-2**.

**[0286]** A specific application of the above described solution for a system and method of mobile and stationary computing device interworking is now discussed. Such a system and method allows a user to work in mobile and stationary work modes.

**[0287]** Many users have mobile computing devices, such as laptops, and stationary computing devices such as desktops. These users often need to work in both mobile and stationary work modes. In some embodiments, a device interoperability system such as the one described above is implemented on the mobile computing device either via integration or as an app. Then, the mobile computing device plays the role of user device **101-1** as shown in FIGS. **2**D and **2**E, and the implemented device interoperability system is analogous to device interoperability system **200** shown in FIGS. **2**B, **2**D and **2**E. When the user connects to a stationary computing device, the stationary computing device plays the role of, for example, one of user devices **101-2** to **101**-N as shown in FIGS. **2**D and **2**E. The connection between the mobile computing device and the stationary computing device is analogous to, for example connection **201-2**.

**[0288]** Once the connection is established, the stationary computing device boots up an OS such as OS **214** from the storage of the device interoperability system, as described above.

**[0289]** In some embodiments, the OS uses the processing capabilities of the stationary computing device to run, as described above. The stationary computing device accesses the storage of the mobile computing device. In some embodiments, as explained above, the stationary computing device recognizes one or more of the hardware components of the mobile computing device such as the microphone, speakers, webcam, keyboard, touchpad, wireless adapter, sensors, and display as connected external devices.

**[0290]** In some embodiments, the stationary computing device operates in either stand-alone or device interoperability system mode, as described above. The stationary computing device operates in a stand-alone mode when a connection is not established between the mobile computing device and the stationary computing device. When a connection is established between the mobile computing device and the stationary computing device, then in some embodiments, the stationary computing device operates in device interoperability system mode. In some embodiments, authentication is performed using a modified version of the algorithm shown in FIG. **3**. This modified version comprises, for example, steps **304**, **306**, **307**, **308** and **309**.

**[0291]** In some embodiments, the OS switches between different hardware configurations when the stationary computing device boots up. To achieve this, the OS uses, for example, the algorithm detailed in FIG. **3**B. In some embodiments the OS maintains some user-defined settings separately for the mobile and stationary computing devices. These settings, for example, include speaker volume level and display scaling settings. Then in some of these embodiments, the OS uses a modified version of the algorithm shown in FIG. **3**B, wherein step **3**B-**03** comprises the OS using the setting values which correspond to either the stationary or mobile computing devices, depending on which device the OS is booted on.

**[0292]** In some embodiments, the connection between the mobile and stationary computing devices is a high-speed wired connection. In some embodiments, the type of technology which is used for this high-speed wired connection is based on one or more factors. These factors comprise, for example:

**[0293]** Measures related to speed of data transfer, such as throughput and latency;

**[0294]** Capability to supply sufficient energy to, for example, charge up the battery of the laptop;

**[0295]** Maximum allowable length of cable between the mobile and stationary computing device;

**[0296]** Commercial availability of the type of technology, for example cost considerations, levels of use in existing products and availability for licensing; and

**[0297]** Native support for connecting external data storage, for example, ability to couple to external solid state drives (SSD) or hard disk drives (HDD).

**[0298]** In some embodiments, the connection between the mobile and stationary computing devices is a Thunderbolt connection. In other embodiments, the connection is based on USB4.

**[0299]** In some embodiments, the stationary computing device requires an expansion card to facilitate the high speed wired connection. Then, the expansion card is installed on the desktop using one or more techniques known to those of skill in the art. For example, in some embodiments the installation comprises plugging in the expansion card to a free slot in a system board of the stationary computing device, such as a Peripheral Component Interconnect express (PCIe) slot.

**[0300]** In some embodiments, the mobile computing device is charged up when it is connected to the stationary computing device. The power for this charging process is delivered through the connection between the mobile and the stationary computing devices, using techniques known to those of skill in the art.

**[0301]** In some embodiments, the mobile computing device operates in a plurality of modes. In a first of these plurality of modes, the mobile computing device boots up the OS stored on its storage and runs using this OS. The user can then perform all the normal functions as necessary.

**[0302]** In a second of these plurality of modes, the mobile computing device operates in a device interoperability system host mode. In this mode, the mobile computing device does not run the OS. Instead, the stationary computing device boots up the OS via the connection between the mobile and stationary devices and uses it as described above. In further embodiments, in this second mode one or more programmes and data stored on the storage of the device interoperability system are used to translate read and write commands and queries from the stationary computing device, thereby enabling access to the storage of the mobile computing device. Then, when the stationary computing device transmits read and write commands and queries via the connection between the two devices, these translation programmes are used to fulfil these commands and queries.

**[0303]** Different methods are possible to enable these functionalities. In some embodiments, as explained above, the stationary computing device recognizes the storage on the mobile computing device as a connected external drive. This can be achieved in a number of ways. In some embodiments, the mobile computing device storage is simulated as an external drive using the mobile computing device's BIOS/UEFI, and therefore the translation program runs as part of mobile computing device's firmware. In other embodiments, the storage is simulated as an external drive by a second OS installed on the mobile computing device's hard drive, and the translation program runs as an app.

**[0304]** A process for the stationary computing device to boot the OS stored in the mobile computing device when both the mobile computing device and the stationary computing device are turned off, is outlined below and in FIG. **11**.

**[0305]** In step **11-01**, the mobile computing device is turned on.

**[0306]** In step **11-02** a mode of operation is selected for the mobile computing device to switch to. In some embodiments, prior to the selecting, one or more options are presented to the user to select a mode of operation. In some embodiments, the presenting of the one or more options comprises the mobile computing device displaying a GUI on its screen with the one or more options. For example:

**[0307]** For the option to switch to the first mode, an indication such as a visual, audible or haptic indication is sent to the user. For example, a visual indication comprising a textual message such as "Press F5 to start the OS on this laptop" is displayed. If the user chooses this option, then in step **11-03** the mobile computing device switches to the first mode.

**[0308]** For the option to switch to the second mode, or the device interoperability system host mode, an indication such as a visual, audible or haptic indication is sent to the user. For example, a visual indication comprising a textual message such as "Press F6 to switch to the device interoperability mode" is displayed. If the user chooses this option, then in step **11-03** the mobile computing device switches to the second mode.

**[0309]** Then, the user makes a selection using these options.

**[0310]** In other embodiments, the presenting of the one or more options in step **11-02** comprises the mobile computing device performing a timer-based process. In some of these embodiments, the timer-based process comprises the mobile computing device beginning a countdown timer and displaying a GUI on its screen. The GUI comprises, for example:

**[0311]** An indication of the time remaining on the countdown timer, which is, for example, visual, audible or haptic, and an indication such as a visual, audible or haptic indication that after a period of time the mobile computing device will be automatically placed in one mode. An example is a message such as "After 10 seconds the OS will be booted on this laptop"; and

[0312] An option for the user to switch to the other mode. For example, "Press any key to switch to the device interoperability mode".

[0313] In yet other embodiments, the selection of the mode of operation in step **11-02** is based on determining if a connection has been established between the mobile computing device and the stationary computing device. If no connection has been detected, then the mobile computing device goes into the first mode and boots the OS in step **11-03**. If the connection to the stationary device is detected, then the mobile computing device automatically switches to the second mode which is the device interoperability system host mode in step **11-04**.

[0314] After step **11-04** is completed, then in step **11-05** the mobile computing device initiates switching the stationary computing device into interoperability mode. In some embodiments, the initiating comprises performing at least one of the following processes:

[0315] establishing a connection between the mobile computing device and the stationary computing device;

[0316] switching, by the mobile computing device, the stationary computing device from a first power state to a second power state. For example, the mobile computing device switches the stationary computing device from a low power sleep state such as states S1, S2, S3, S4 or a "soft off" state S5, to a working state S0; as described in "System Power States" at https://docs.microsoft.com/en-us/windows/win32/power/system-power-states; retrieved Jun. 6, 2021. In some embodiments this switching is based on a determination of the power state of the stationary computing device as will be explained below;

[0317] sending an indication to the user to switch the stationary computing device from a first power state to a second power state. For example, this comprises sending an indication to the user to switch from one of the low power sleep states S1-S4 or soft off state S5 to a working state S0 as explained above. This indication is, for example, a visual, audible or haptic indication;

[0318] sending an indication to the user to turn on the stationary computing device, which is, for example, a visual, audible or haptic indication; or

[0319] sending an indication such as a visual, audible or haptic indication to the user to restart the stationary computing device.

[0320] In some embodiments the at least one or more processes performed as part of the initiating is based on a determination of the state of the connection between the mobile computing device and the stationary computing device. In some of these embodiments, when no connection is detected, or the state of the connection cannot be determined, the mobile computing device sends one or more indications such as visual, audible or haptic indications to the user to perform one or more of:

[0321] establishing a connection between the mobile computing device and the stationary computing device;

[0322] switching the stationary computing device from a first power state to a second power state as described above;

[0323] turning on the stationary computing device; or

[0324] restarting the stationary computing device.

[0325] An example of this is where the mobile computing device displays a message such as "Please make sure that the desktop is connected and turn it on to continue".

[0326] In some embodiments the at least one or more processes performed as part of the initiating is based on the mobile computing device determining the power state of the stationary computing device. For example, in some of these embodiments, based on this determination:

[0327] an indication such as a visual, audible or haptic indication is sent to the user to switch the power state of the stationary computing device, or

[0328] the mobile computing device switches the power state of the stationary computing device from a first state to a second state, as explained above.

[0329] In step **11-06** the stationary computing device boots into the interoperability system mode, that is, it boots the OS stored on the storage of the mobile computing device.

[0330] In some embodiments the mobile computing device:

[0331] is placed in one of the low power sleep states, for example, sleep states S1, S2, S3 or S4, or soft off state S5 as described above,

[0332] is connected to the stationary computing device,

[0333] is able to detect the event of turning on the stationary computing device through the connection between the mobile and stationary computing devices, and

[0334] is able to switch to a working state S0 as described above, after detection.

In these embodiments, there is a need for a process to turn on and switch both devices to interoperability modes by turning on the stationary computing device. A process for this is outlined in FIG. **12**.

[0335] In step **12-01** the user turns on the stationary computing device. In some embodiments this is done by pressing a power-on button.

[0336] Then in step **12-02** the mobile computing device detects that the stationary computing device has been turned on, via, for example, one or more signals sent through the connection between the mobile and stationary computer devices. Then the mobile computer device enters the second or device interoperability system host mode. In some embodiments this comprises switching from one of the sleep states such as states S1, S2, S3 or S4 to the working state S0 as explained previously. In other embodiments this comprises switching from the soft off state S5 to another state, for example the working state S0 as explained previously.

[0337] In step **12-03**, the stationary computing device continues the process of booting into the device interoperability system mode, that is, it boots the OS stored on the storage of the mobile computing device.

[0338] After a user wants to end a working session during which the mobile computing device is in device interoperability system host mode, and the stationary computing device is in interoperability mode, the user may want to follow different courses of action for the mobile computing device and the stationary computing device. Examples of these courses of action include:

[0339] Restarting the stationary computing device while keeping the interoperability system mode on;

[0340] Turning off the stationary computing device only; and

[0341] Turning off the stationary computing device and the mobile computing device.

[0342] An example process to enable a user to perform actions after the end of a working session is shown in FIG. **13**. In FIG. **13**, in step **13-01**, when the user acts to shut

down the OS, a shutdown dialog is presented. Presenting the dialog comprises presenting different options to the user for selection, for example via a GUI. Examples of these different options comprise:

**[0343]** Prompting the user to restart the stationary computing device;

**[0344]** Prompting the user to shutdown the stationary computing device and switch to the mobile computing device; and

**[0345]** Prompting the user to shutdown both the stationary computing device and the mobile computing device.

**[0346]** In step **13-02**, after the user selects one of these options, the stationary computing device sends one or more signals to the mobile computing device to perform one of:

**[0347]** remaining in second mode or the device interoperability system host mode and wait for the desktop to be restarted;

**[0348]** waiting for the desktop to be shut down and then switch to the first mode, that is, restart itself if needed and boot the OS; or

**[0349]** wait for the desktop to be shut down and then turn-off itself.

**[0350]** For the embodiments where the mobile computing device either switches to the first mode or turns off in step **13-02**, then in steps **13-03** and **13-04**, the mobile computing device either restarts and boots the OS or shuts down. When the stationary computing device sends the one or more signals to the mobile computing device to stay in the second mode, then in steps **13-03** and **13-05** the mobile computing device remains in the second mode.

**[0351]** In some embodiments the mobile computing device monitors whether the stationary computing device is turned off. Various methods are possible to achieve this. In some embodiments, the BIOS/UEFI of the mobile computing device monitors whether the stationary computing device is turned off. In other embodiments, the second OS on the mobile computing device monitors whether the stationary computing device is turned off. In some of the embodiments where the second OS monitors whether the stationary computing device is turned off, this functionality is performed by an app. Once the mobile computing device has determined that the stationary computing device is turned off, then the mobile computing device performs a process to either shut down or switch into the first mode as previously described. In some embodiments, the process comprises the mobile computing device displaying a GUI on its screen presenting one or more options to, for example:

**[0352]** Shut down the mobile computing device. This is performed by, for example, sending the user an indication such as a visual, audible or haptic indication. For example, a visual indication comprising a textual message such as "press F9 to shutdown this laptop" is displayed; and

**[0353]** Switch to the first mode. This is performed by, for example, sending the user an indication such as a visual, audible or haptic indication. For example, a visual indication comprising a textual message such as "press F5 to start the OS on this laptop" is displayed.

**[0354]** In some embodiments, the mobile computing device performs a timer-based process. An example is illustrated in FIG. **14**. In step **14-01**, the mobile computing device begins a countdown timer and displays a GUI on its screen which comprises:

**[0355]** An indication of the time remaining on the countdown timer and a visual indicator e.g. a message which states that after this period of time the laptop will be automatically turned-off; and

**[0356]** An option for the user to switch to the first mode. For example, "press any key to start the OS on this laptop".

**[0357]** In steps **14-02** and **14-03**, while the user has not made a selection and the timer is running, the mobile computing device monitors the state of the stationary computing device. If the stationary computing device is restarted into the interoperability system mode in step **14-04**, then in step **14-05** the GUI disappears and the mobile computing device continues to work in the second mode.

**[0358]** In steps **14-02** and **14-03**, if the user has not made a selection and the timer expires, the mobile computing device is automatically switched off in step **14-06**.

**[0359]** If in step **14-02** the user selects the option to switch to the first mode, then in some embodiments, in step **14-07** the mobile computing device switches to the first mode.

**[0360]** In some embodiments, in the interoperability system mode the OS disables and hides from its GUI the option to switch to power states which may be incompatible or unsupported by the interoperability system mode. For example, an option to enter the "Sleep" state is disabled to avoid issues caused by disconnecting the laptop, which contains the system drive, from the desktop.

**[0361]** In some embodiments, when the mobile computing device switches from the first mode to the second mode or vice versa, a process of saving the working session is performed. This process is performed by, for example, either the OS or a service app running on the OS.

**[0362]** In some embodiments, the process of saving the working session comprises saving information related to one or more of the following:

**[0363]** User identification (ID) or name;

**[0364]** Running apps, such as, for example, current running state;

**[0365]** Status of files, for example whether the files were open;

**[0366]** Last browser session, for example, which Uniform Resource Locators (URLs) were open;

**[0367]** Current viewing position for some file types, such as text or document files and video files. This is achieved by, for example, using particular viewing apps.

**[0368]** Window sizes and positions. In some embodiments, this comprises saving an absolute window position. In other embodiments, this comprises saving a relative window position since resolutions of computing devices can be different; and

**[0369]** Additional connections such as a virtual private network (VPN) connection.

This information is saved in case of, for example, future restoration.

**[0370]** In some embodiments, a functionality to restore the saved working session is provided. In some of these embodiments, the user is asked whether the user wants to restore the saved working session after the OS is restarted. If the user accepts, the saved working session is restored.

**[0371]** During a working session in the interoperability mode, a user may either accidentally or intentionally disconnect the mobile computing device from the stationary

computing device. This disconnection can lead to problems such as a system crash, data corruption or loss, and may require OS recovery.

**[0372]** This problem may be addressed in several ways. In some embodiments, one or more indicators are used to indicate when it is safe to disconnect the mobile computing device from the stationary computing device. In some embodiments, an indicator such as a visual, audible or haptic indicator is provided to the user to show that it is safe to disconnect the mobile computing device from the stationary computing device. Examples of such indicators comprise:

**[0373]** A message, for example, "Now you can safely disconnect the laptop", when a user selects an option corresponding to shutdown of the stationary computing device;

**[0374]** A light emitting diode (LED) indicator near the Thunderbolt/USB port on the laptop either changes colour or turns off when it is safe to disconnect the laptop.

**[0375]** Both a message is displayed and the LED indicator either changes colour or turns off.

**[0376]** In other embodiments, a physical lock is used to lock the port and prevent disconnection. This physical lock is, for example, an electro-mechanical lock. Then, the physical lock automatically locks the connector in the port on the mobile computing device when the user starts to work in the interoperability mode. So the user is prevented from disconnecting the mobile computing device from the stationary computing device without physically breaking the lock. In some embodiments, the physical lock comprises an electromechanically controlled mechanism with a "muscle wire" similar to the one described in U.S. Pat. No. 6,461,185 titled "Method and apparatus for an electromechanically controlled electronic interface plug" to James et al, filed Apr. 20, 2001 and issued on Oct. 8, 2002.

**[0377]** In some embodiments, when the connection between the mobile computing device and the stationary computing device is lost, the OS is paused, and resumed when the connection is reestablished. This mitigates the undesired effects of accidental disconnection, such as crashes and data loss. In some embodiments, pausing and resuming when the connection is lost and reestablished is implemented similar to that explained in "What happens if I remove my Windows To Go drive while it is running?", located at https://docs.microsoft.com/en-us/windows/deployment/planning/windows-to-go-frequently-asked-questions #what-happens-if-i-remove-my-windows-to-go-drive-while-it-is-running-, retrieved on Jun. 7, 2021. That is:

**[0378]** when the connection between the mobile computing device and the stationary computing device is lost, the OS is paused for a period of time within which the connection must be resumed; and

**[0379]** when the connection is reestablished within the period of time, the OS will resume at the point when the connection was lost.

**[0380]** In some embodiments, a direct wireless connection between the mobile computing device and the stationary computing device is set up as a secondary connection, in case the user physically disconnects the mobile computing device from the stationary computing device without shutting down the OS on the stationary computing device. The backup direct wireless connection is set up via, for example, a wireless adapter on the stationary computing device and direct wireless transmission and reception capabilities on the mobile computing device. In some embodiments, an external USB-based Wi-Fi dongle on the stationary mobile computing device is used to set up the direct wireless connection. In yet other embodiments, the installed extension card is used to set up the direct wireless connection through, for example, a Wi-Fi module. In some embodiments, the OS is paused when the main connection is interrupted and resumed after the reserved connection is established, using the pause-and-resume functionality described above. In other embodiments this is enabled via an indirect connection, for example, through a Wi-Fi router.

**[0381]** Although the algorithms described above including those with reference to the foregoing flow charts have been described separately, it should be understood that any two or more of the algorithms disclosed herein can be combined in any combination. Any of the methods, algorithms, implementations, or procedures described herein can include machine-readable instructions for execution by: (a) a processor, (b) a controller, and/or (c) any other suitable processing device. Any algorithm, software, or method disclosed herein can be embodied in software stored on a non-transitory tangible medium such as, for example, a flash memory, a CD-ROM, a floppy disk, a hard drive, a digital versatile disk (DVD), or other memory devices, but persons of ordinary skill in the art will readily appreciate that the entire algorithm and/or parts thereof could alternatively be executed by a device other than a controller and/or embodied in firmware or dedicated hardware in a well known manner (e.g., it may be implemented by an application specific integrated circuit (ASIC), a programmable logic device (PLD), a field programmable logic device (FPLD), discrete logic, etc.). Also, some or all of the machine-readable instructions represented in any flowchart depicted herein can be implemented manually as opposed to automatically by a controller, processor, or similar computing device or machine. Further, although specific algorithms are described with reference to flowcharts depicted herein, persons of ordinary skill in the art will readily appreciate that many other methods of implementing the example machine readable instructions may alternatively be used. For example, the order of execution of the blocks may be changed, and/or some of the blocks described may be changed, eliminated, or combined.

**[0382]** It should be noted that the algorithms illustrated and discussed herein as having various modules which perform particular functions and interact with one another. It should be understood that these modules are merely segregated based on their function for the sake of description and represent computer hardware and/or executable software code which is stored on a computer-readable medium for execution on appropriate computing hardware. The various functions of the different modules and units can be combined or segregated as hardware and/or software stored on a non-transitory computer-readable medium as above as modules in any manner, and can be used separately or in combination.

**[0383]** While particular implementations and applications of the present disclosure have been illustrated and described, it is to be understood that the present disclosure is not limited to the precise construction and compositions disclosed herein and that various modifications, changes, and variations can be apparent from the foregoing descriptions without departing from the spirit and scope of an invention as defined in the appended claims.

1. A system for a mobile computing device and a stationary computing device associated with a user to interwork, wherein:
the mobile computing device comprises a device interoperability system having
a communications module, wherein a first connection is established between the stationary computing device and the communications module;
storage coupled to said communications module, wherein the storage stores an operating system,
one or more programs,
data associated with the user,
further wherein
the operating system is booted by the stationary computing device via the first connection,
the operating system runs on the stationary computing device, and
the operating system uses one or more processing capabilities of the stationary computing device for operation; and
one or more processors to support said device interoperability system.

2. The system of claim 1, wherein the first connection is a Thunderbolt connection.

3. The system of claim 1, wherein the first connection is a USB connection.

4. The system of claim 1, wherein prior to the booting of the operating system, the mobile computing device is turned on, and
one or more options are presented to the user to select a mode of operation.

5. The system of claim 4, wherein the presenting of the one or more options comprises a timer-based process.

6. The system of claim 4, wherein the presenting of the one or more options comprises displaying a GUI with the one or more options.

7. The system of claim 1, wherein prior to the booting of the operating system, the mobile computing device is turned on, and
a selection of a mode of operation is made based on determination of the establishment of the first connection.

8. The system of claim 1, wherein prior to the booting of the operating system,
the mobile computing device is turned on,
the mobile computing device initiates switching the stationary computing device into interoperability mode,
the initiating comprising at least one of:
establishing the first connection,
switching the stationary computing device from a first power state to a second power state,
sending an indication to the user to switch the stationary computing device from the first power state to the second power state, turning on the stationary computing device, or
sending an indication to the user to restart the stationary computing device.

9. The system of claim 1, wherein
prior to the booting of the operating system by the stationary computing device, the stationary computing device is turned on, and
the mobile computing device enters a device interoperability system host mode.

10. The system of claim 9, wherein the entering of the device interoperability system host mode comprises switching from a sleep state to a working state.

11. The system of claim 9, wherein the entering of the device interoperability system host mode comprises switching from a soft off state to a working state.

12. The system of claim 1, wherein
the mobile computing device is in a device interoperability system host mode; based on an action performed by the user which is indicative of user intention to shut down the operating system,
a shutdown dialog is presented with one or more options for the user to select; the stationary computing device sends a signal to the mobile computing device based on the selection of the one or more options presented in the shutdown dialog; and
based on the sent signal, the mobile computing device performs one of
switching to a first mode of operation from the device interoperability system host mode,
remaining in the device interoperability system host mode, or turning off.

13. A method for a mobile computing device and a stationary computing device associated with a user to interwork, the method comprising:
providing a device interoperability system comprising
a communications module,
storage coupled to the communications module, and
the device interoperability system is installed on the mobile computing device; enabling establishment of a first connection between the stationary computing device and the communications module;
enabling the storage to store information comprising
an operating system,
one or more programs, and
data associated with the user;
enabling booting of the operating system by the stationary computing device via the first connection; and
enabling the operating system to run on the stationary computing device and use one or more processing capabilities of the stationary computing device for operation.

14. The method of claim 13, wherein prior to the enabling of the booting of the operating system,
initiating, by the mobile computing device, switching the stationary computing device into interoperability mode,
the initiating comprising performing at least one or more processes comprising:
establishing the first connection,
switching the stationary computing device from a first power state to a second power state, sending an indication to the user to switch the stationary computing device from the first power state to the second power state,
turning on the stationary computing device, or
sending an indication to the user to restart the stationary computing device.

15. The method of claim 14, wherein the performing of the at least one or more processes is based on a determination of a state of the first connection between the mobile computing device and the stationary computing device.

**16**. The method of claim **14**, wherein the performing of the at least one or more processes is based on a determination of a power state of the stationary computing device.

**17**. The system of claim **5**, wherein the presenting of the one or more options comprises displaying a GUI with the one or more options.

* * * * *