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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 May 2014 (08.05.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/070342 A1

(51) International Patent Classification:
G06F 13/10 (2006.01) *G06F 9/44* (2006.01)
G06F 13/38 (2006.01)

(21) International Application Number:
PCT/US2013/061796

(22) International Filing Date:
26 September 2013 (26.09.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/663,617 30 October 2012 (30.10.2012) US

(71) Applicant: **GOOGLE INC.** [US/US]; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US).

(72) Inventor: **PARK, Keun, Young**; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US).

(74) Agent: **ANDERSON, Michael, D.**; McDonnell Boehnen Hulbert & Berghoff, 300 S. Wacker Dr., Ste. 3200, Chicago, IL 60606 (US).

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

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

Published:

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

(54) Title: DRIVE EMULATION FOR DEVICES WITH MASS STORAGE FEATURE

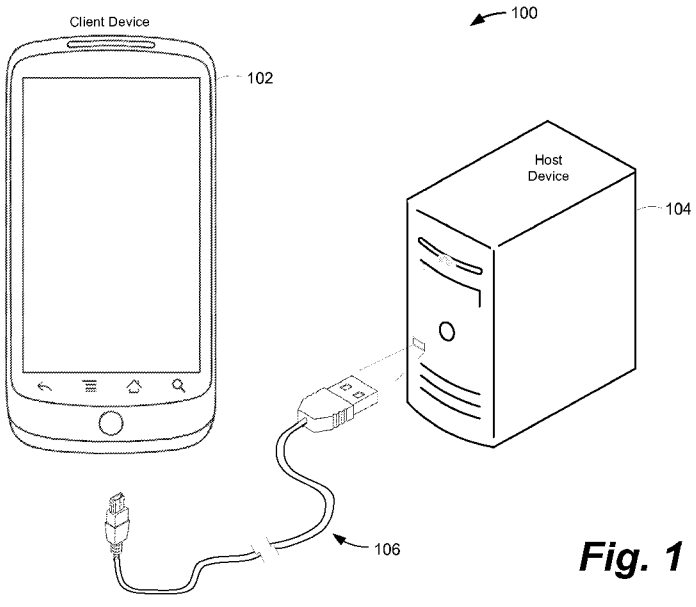


Fig. 1

(57) Abstract: Systems, methods, routines and/or techniques for drive emulation for devices with a mass storage feature are described. The present disclosure describes a drive emulator module that emulates the behavior of the mass storage drive inside a client device (e.g., a mobile device). The drive emulator module may emulate the various communication protocols used by a storage drive, for example, used to access various clusters of a storage drive. The drive emulator module may emulate the formatting of a drive, for example, the file system (e.g., FAT file system or other file system) formatting of a drive. The systems, methods, routines and/or techniques of the present disclosure may allow a client device to access its mass storage drive concurrently as an external host device accesses the storage drive, i.e., in "mass storage mode".

WO 2014/070342 A1

DRIVE EMULATION FOR DEVICES WITH MASS STORAGE FEATURE

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority from U.S. Patent Application No. 13/663,617 filed on October 30, 2012. The disclosure of this provisional application is incorporated by reference herein in its entirety.

FIELD

[0002] The present disclosure relates to devices that offer a mass storage feature (e.g., according to the USB mass storage standard), and more particularly to one or more systems, methods, routines and/or techniques for drive emulation for devices with a mass storage feature.

BACKGROUND

[0003] Computing devices have long used storage devices such as hard drives and the like. In recent years, it has become common for computing devices to not only use their own internal storage devices, but also to access external storage devices. Because of advancements in storage technology, large amounts of data can be stored in small storage devices such as portable hard drives and USB flash drives. Additionally, various portable digital devices (e.g., digital cameras, smartphones, etc.) now include small storage drives or devices that offer significant amounts of storage, sometimes referred to as "mass storage". A computing device (e.g., a personal computer) may connect (e.g., via USB cord) to one of these portable digital devices to either use the storage capacity of the storage drive inside the portable digital device or to transfer files between the computing device and the portable digital device. The storage drives or devices in these portable digital devices typically use the USB mass storage standard, which is supported natively by modern operating systems.

[0004] Further limitations and disadvantages of conventional and traditional approaches will become apparent to one of skill in the art, through comparison of such systems with some aspects of the present invention as set forth in the remainder of the present application and with reference to the drawings.

SUMMARY

[0005] The present disclosure describes one or more systems, methods, routines and/or techniques for drive emulation for devices with a mass storage feature. The present disclosure describes a drive emulator module that emulates the behavior of the mass storage drive inside a client device (e.g., a mobile device). The drive emulator module may emulate the various communication protocols used by a drive, for example, used to access various clusters of a drive. The drive emulator module may emulate the formatting of a drive, for example, the file system (e.g., FAT file system or other file system) formatting of a drive. The systems, methods, routines and/or techniques of the present disclosure may allow an external host device to access the storage drive of a client device using the same methods (e.g., the same USB cord and the same mass storage driver) that the host device may use to communicate with various mass storage devices. The systems, methods, routines and/or techniques of the present disclosure may allow a client device to access its mass storage drive concurrently as an external host device accesses the storage drive, i.e., in "mass storage mode". In this respect, the client device may maintain full functionality (e.g., may run its applications and features that require access to the storage drive) while in mass storage mode.

[0006] These and other advantages, aspects and novel features of the present disclosure, as well as details of an illustrated embodiment thereof, will be more fully understood from the following description and drawings. It is to be understood that the foregoing general descriptions are exemplary and explanatory only and are not restrictive of the disclosure as claimed.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] Several features and advantages are described in the following disclosure, in which several embodiments are explained, using the following drawings as examples.

[0008] FIG. 1 depicts an illustration of a block diagram showing example devices and connections of a network setup, where one or more embodiments of the present disclosure may be useful in such a network setup.

[0009] FIG. 2 depicts an illustration of a block diagram showing example components, connections and interactions of a network setup between a client device and a host device.

[00010] FIG. 3 depicts an illustration of a block diagram showing example components, connections and interactions of a network setup between a client device and a host device, according to one or more embodiments of the present disclosure.

[00011] FIG. 4A depicts an illustration of a block diagram showing a conceptual representation of a drive emulator module, according to one or more embodiments of the present disclosure.

[00012] FIG. 4B depicts an illustration of a block diagram showing a components, routines, connections and/or communications included in an example drive emulator module, according to one or more embodiments of the present disclosure.

[00013] FIG. 5 depicts a flow diagram that shows example steps in a method for drive emulation for devices with a mass storage feature, in accordance with one or more embodiments of the present disclosure.

[00014] FIG. 6 depicts a flow diagram that shows example steps in a method for drive emulation for devices with a mass storage feature, in accordance with one or more embodiments of the present disclosure.

[00015] FIG. 7 depicts a block diagram of an example data processing system that may be included within a mobile device or smartphone, according to one or more embodiments of the present disclosure.

DETAILED DESCRIPTION

[00016] Various mobile devices offer a mass storage feature (e.g., USB mass storage). For example, a mobile device may include a storage drive or device that may be accessible not only locally by the mobile device, but also by external devices that connect to the mobile device. The storage drives or devices in these mobile devices may use the USB mass storage standard, sometimes referred to as USB, MSC or UMS. The USB mass storage (UMS) standard is a protocol that allows a Universal Serial Bus (USB) device (e.g., flash drive or a mobile device with a mass storage feature) to provide its storage capabilities to an external computing device (e.g., a personal computer), to enable the external device to view the storage drive of the USB device, and/or to enable transfer of files between the external device and the USB device. The UMS standard may provide the external device with direct (i.e., low-level) access to the storage drive or device of the mobile device, for example, as though the external device was connected to a portable hard drive. The UMS standard comprises a set of computing communications protocols, and the standard provides an interface to allow a variety of storage devices to transfer data. In some situations, a mobile device that offers a mass storage feature may be referred to as a client device, and an external computing device that connects to the client device may be referred to as a host device.

[00017] FIG. 1 depicts an illustration of a block diagram showing example devices and connections of a network setup 100, where one or more embodiments of the present disclosure may

be useful in such a network setup. More specifically, FIG. 1 depicts a client device 102 connected to a host device 104. The client device 102 may be depicted as a smartphone, but the client device 102 may be one of various other mobile devices that offer a mass storage feature. The client device 102 may include a storage drive or device, for example, a hard drive, solid state drive, flash memory, non-volatile memory card or the like. The present disclosure may refer to a storage drive of a client device, but it should be understood that the term "drive" may refer to any storage drive or device that may be used to offer a mass storage feature. The client device 102 may implement and/or accommodate various mass storage standards (e.g., USB mass storage standard) in order to offer a mass storage feature. Although the USB mass storage standard is used in various mobile devices, and thus may be referred to in the present disclosure, it should be understood that the descriptions provided herein may apply to other mass storage standards such as FireWire, Blootooth or any other mass storage standard that provides the host device with direct access to the internal storage drive of the client device.

[00018] Client device 102 may be connected (e.g., via connection 106) to at least one host device 104. The connection 106 may be a wired or wireless connection. For example, the connection 106 may be a USB cord, a FireWire cord, a wireless Bluetooth connection or similar wired or wireless connection. The example of FIG. 1 shows connection 106 as a USB cord, where one end of the USB cord may be plugged into the client device 102 and the other end of the USB cord may be plugged into the host device 104. Each of the client device 102 and the host device 104 may include appropriate ports, circuitry and the like that may be required to accept the connection 106 and to communicate via the connection 106. Host device 104 may be any computing device, for example, a personal computer, a laptop computer, a server or any other type of computing device. Host device 104 may connect to client device 102 to utilize the mass storage drive inside the client device 104 and/or to transfer files to and/or pull files from the client device 104.

[00019] Various mass storage standards (e.g., the UMS standard) may operate such that when the host device 104 is connected to the client device 102, the host device 104 expects to receive (e.g., via the connection 106 and a mass storage driver in the client device) direct or low-level access to the storage drive of the client 102. In various mobile devices, only a single device may be allowed direct or low-level access to the storage drive of the mobile device at a time. For example, an operating system running on the mobile device may either allow the mobile device (e.g., its applications, processes and the like) to access the storage drive or it may allow the external host device to access the storage drive. Thus, if a host device is connected to the storage drive of a client device (i.e., referred to as "mass storage mode"), the client device may have to refrain from

accessing its own storage drive. This may prevent the client device from running various applications, processes and/or features that require access to the storage drive, thereby limiting the functionality of the client device while it is in mass storage mode.

[00020] FIG. 2 depicts an illustration of a block diagram showing example components, connections and interactions of a network setup between a client device 202 and a host device 204. The host device 204 may include a host file system 206. The host device 204 may include an application 208 that may interact with the host file system 206 and may provide to a user a view of the data stored on the storage drive 210 of the client device 206, or may allow a user to transfer data to or retrieve data from the storage drive 210 of the client device 202. The host file system 206 and/or application 208 may operate according to one or more mass storage standards, for example, the UMS standard.

[00021] The host file system 206 may be part of a mass storage feature and/or routine included in the host device. The host file system 206 may adapt the host device to communicate with a storage drive of an external device, for example, a mobile device. The host file system 206 may include information about how to access (e.g., at directly or at a low-level) a storage drive, where to look for particular information within the storage drive and how to interpret information received from the storage drive. For example, the host file system 206 may implement a storage drive protocol that is used to format low-level requests that may be sent directly (e.g., via a storage device driver) to the storage drive, and may be interpreted by the storage drive. The term "directly" should not be taken literally. As can be seen in FIG. 2, various components may facilitate the communication between the host file system 206 and the storage drive 210 of the client device 202, for example, a storage device driver 212 and an operating system 214. The term "directly" (or "low level") in this respect may refer to host requests being sent more or less directly to the storage drive 210 in a low-level format such that the storage drive may more or less directly interpret the requests without significantly reformatting the requests, and information being sent more or less directly from the storage drive 210 back to the host device 204 in a "raw data" format such that the host device may interpret the raw data of the storage drive 210. For example, the host device 204 may access particular clusters (e.g., by specifying a cluster number or address) of the storage device 210.

[00022] Client device 202 may include a storage device driver 212. The storage device driver may accommodate one or more mass storage standards, for example, the UMS standard. The storage device driver 212 may include various routines, processes and the like that run on the client device to facilitate communication between a host device (e.g., connected to the client device via a USB cord) and the storage drive 210 of the client device. The storage device driver may

communicate with a particular piece of communication hardware, for example, a USB port (and related circuitry) or other type of fast data transfer port. The storage device driver 212 may receive requests, information and the like that are transmitted (from the host device) according to a protocol associated with the communication hardware and translate them into requests, information and the like that are adapted to interface directly (e.g., at a low level) with the storage drive 210. The storage device driver 212 may also translate requests in the opposite direction, from the storage drive 210 to the host device 204. An operating system 214 may facilitate communication between the storage device driver 212 and the storage drive 210.

[00023] The client device 202 may include a storage drive 210. In some embodiments, the general term drive may refer to a mechanical hard drive with one or more magnetic platters and one or more actuator arms. As a person of skill in the art would understand, a mechanical hard drive may include a number of physical storage segments called sectors. Multiple sectors may together constitute a cluster, which may be the smallest storage unit of measurement that an operating system can access on the hard drive. Therefore, for example, an operating system may provide a cluster number or address to the hard drive and the hard drive may return data stored in that cluster. The specifics of how a cluster address is provided to the hard drive and how the hard drive returns associated data may be referred to as a communication or access protocol of the hard drive. In other embodiments, the general term drive may refer to a solid state type drive or other type of non-mechanical type drive, for example, a solid state drive that mimics the communication or access protocol of a mechanical hard drive. For example, an operating system may provide a cluster number or address to a solid state drive, and the solid state drive may return data associated with that cluster address, even if the data on the solid state drive is not necessarily organized according to physical sectors, clusters and the like. It should be understood that, although the visual representations of storage drives depicted in the figures may resemble mechanical hard drives, these are just conceptual illustrations of a drive. For example, although drive 210 of FIG. 2 and drive 310 of FIG. 3 show a circular platter with multiple sectors, the drive 210 and/or the drive 310 may be a solid state drive.

[00024] Storage drive 210 may store various pieces of information or data. The data stored on the storage drive 210 may be organized and/or formatted according to a file system, as can be seen generally by number 216 of FIG. 2. A file system may generally refer to a way to organize data stored on a storage drive. The file system may dictate specific locations (e.g., clusters) on the storage drive to store pieces of files. The file system may specify how to retrieve, delete and/or update data, as well as how to manage the available space on the storage drive. A tight coupling may exist between an operating system (e.g., operating system 214) and a file

system. For example, data may be organized (e.g., as generally indicated by number 216) on a storage drive 210 according to the file system protocols, and the operating system 214 may implement various file system routines (e.g., generally indicated by client file system 218) to store, retrieve and manage data on the storage drive 210. Therefore, the term file system should be understood as a flexible term to refer to file system routines and the organization of data on the storage drive. For the sake of clarity, various descriptions in this disclosure may refer to file system routines (e.g., client file system 218 and host file system 206) as simply file systems.

[00025] Client device 202 may include a file system 218. File system 218 may facilitate communication between the one or more applications (e.g., application 220) running on the client device 202 and the storage drive 210 of the client device. As explained above, file system 218 may include various file system routines (e.g., generally indicated by client file system 218) to store, retrieve and manage data on the storage drive 210. For clarity, these routines may be generally referred to as the client file system 218. As one example of how an application 220 may use the file system 218 to communicate with drive 210, application 220 may provide storage command to the operating system 214 where the storage command includes an address and data. The operating system and client file system may translate that address into one or more cluster numbers or addresses in order to store the data in the appropriate location on the drive 210.

[00026] The network setup of FIG. 2 may result in various limitations. According to various mass storage standards (e.g., USB mass storage standards), the host device 204 that is connected to the client device 202 may expect to communicate (e.g., via a host file system 206, a wired connection, communication ports and a storage device driver 212) with the storage drive 210 directly or at low-level. In various mobile devices, the operating system 214 running on the client device (and perhaps other components in the mobile device) may provide a single device with direct or low-level access to the storage drive 210 of the client device at a time. For example, the operating system 214 may either allow the client device 202 (e.g., its applications 220, processes and the like) to access the storage drive 210 or it may allow the external host device 204 (via storage device driver 212) to access the storage drive 210. Thus, if host device 204 is connected to the storage drive 210 of a client device 202 (i.e., referred to as "mass storage mode"), the client device 202 may have to refrain from accessing its own storage drive 210. This may prevent the client device 202 from running various applications 220, processes and/or features that require access to the storage drive 210, thereby limiting the functionality of the client device while it is in mass storage mode.

[00027] Another limitation of the network setup of FIG. 2 is that the storage drive 210 may have to be formatted according to the host file system 206 instead of a different file system that

may be preferable to the developers of the client device and/or the operating system 214. Because the host device (e.g., via host file system 206) may communicate with the storage drive 210 at a low level (e.g., sending and receiving raw data to/from the storage drive 210), the storage drive may need to be formatted such that the host device knows how to access and/or interpret the data. Various host devices 204 may use a relatively old or outdated file system 206 to access external storage devices. For example, various host devices may use a FAT (File Allocation Table) file system. The host device 204 may communicate requests to the client device to access the storage drive 210, and the requests may be transmitted directly to the storage drive 210, and therefore, in this situation, the data stored on the storage drive 210 must be organized according to the host file system. In other words, in this situation, the host file system expects to see data from the storage drive 210 organized in a particular way, so the drive must be organized in this way. Developers of the client device 202 and/or the operating system 214 may prefer to use a different file system (e.g., not a FAT file system), for example, a newer file system and/or a file system that is more appropriate for mobile devices. For example, various developers for mobile devices may prefer to use the EXT4 (fourth extended) file system. However, in the situation of FIG. 2, the developers may be forced to format the storage drive 210 according to the host file system, for example, because it may be impossible or inefficient to dynamically and continuously reorganize the data on storage drive 210 from one file system (e.g., FAT) to another (e.g., EXT4) and vice versa. Therefore, various mobile devices may be required to format their storage drives according old, outdated, or non-optimal file systems if they offer a mass storage feature.

[00028] The present disclosure describes one or more systems, methods, routines and/or techniques for drive emulation for devices with a mass storage feature. The present disclosure describes a drive emulator module that emulates the behavior of the mass storage drive inside a client device (e.g., a mobile device). The drive emulator module may emulate the various communication protocols used by a drive, for example, used to access various clusters of a drive. The drive emulator module may emulate the formatting of a drive, for example, the file system (e.g., FAT file system or other file system) formatting of a drive. The systems, methods, routines and/or techniques of the present disclosure may allow an external host device to access the storage drive of a client device using the same methods (e.g., the same USB cord and the same mass storage driver) that the host device may use to communicate with various mass storage devices. The systems, methods, routines and/or techniques of the present disclosure may allow a client device to access its mass storage drive concurrently as an external host device accesses the storage drive, i.e., in "mass storage mode". In this respect, the client device may maintain full

functionality (e.g., may run its applications and features that require access to the storage drive) while in mass storage mode.

[00029] FIG. 3 depicts an illustration of a block diagram showing example components, connections and interactions of a network setup between a client device 302 and a host device 304, according to one or more embodiments of the present disclosure. Several of the components of the host device 304 may be similar to the components of host device 204 of FIG. 2. For example, host file system 306 and host application 308 may be similar to the host file system 206 and 208 respectively. One goal of the present disclosure may be to provide a mass storage access solution that does not require any change in the operation of the components and/or routines of the host device 304. Several of the components of the client device 302 may be similar to the components of client device 202 of FIG. 2. For example, storage device driver 312 and application 320 may be similar to storage device driver 212 and application 220 respectively. One goal of the present disclosure may be to provide a mass storage access solution that does not require any change in the operation of some components and/or routines (e.g., storage device driver 312 and application 320) of the client device 302. In some embodiments of the present disclosure, it may be assume that a component depicted in FIG. 3 is constructed and operates the same or in a similar manner as a similar component discussed in relation to FIG. 2, unless an alternate construction and/or operation is discussed below.

[00030] In the embodiment of FIG. 3, various components and/or routines of the client device 302 may be added and/or altered when compared to the embodiment of FIG. 2. For example, storage drive 310 may be structurally or physically similar to the storage drive 210 as described with regard to FIG. 2, but the data on storage drive 310 may be organized or formatted (see generally number 316) according to a different file system, as explained further below. As another example, operating system 314 may be similar in many respects to operating system 214 as described with respect to FIG. 2, but operating system 314 may be adapted to include (or be in communication with) a drive emulator module 322. To accommodate drive emulator module 322, operating system 314 may route requests from the storage device driver 312 to the drive emulator module 322 instead of routing the requests directly to the storage drive. Operating system 314 may be adapted to facilitate communication between the drive emulator module 322 and the storage drive 310 via the client file system 318. Client file system 318 may be similar in many respects to client file system 218 as described with respect to FIG. 2, but it may be adapted to communicate with a drive emulator 322. Client file system 318 may be a newer file system and/or a file system that is more appropriate for mobile devices, for example, the EXT4 (fourth extended) file system. Data on storage drive 310 may be organized or formatted according to this client file system 318,

for example, EXT4. The client file system 318 and the data organization 316 of the storage drive may accommodate a new file system (e.g., EXT4) even if the host file system 306 uses an older, outdated, or non-optimal file systems (e.g., FAT).

[00031] Drive emulator module 322 emulates the behavior of a mass storage drive. The drive emulator module 322 may emulate the various communication protocols used to access a storage drive, for example, used to access various clusters of a storage drive. The drive emulator module may emulate the formatting of a drive, for example, the file system (e.g., FAT file system or other file system) according to which a storage drive may be formatted. In this respect, the drive emulator module 322 may appear to a storage device driver 312 as though it is a storage drive, in particular, a storage drive formatted according to a particular file system (e.g., FAT). The storage device driver 312 may communicate the same type of low-level requests (e.g., cluster number or address) to the drive emulator and may receive back what appears to be raw data. In this respect, the drive emulator module 322 may behave to the storage device driver like a storage drive. It should be understood that the storage device driver may emulate storage devices that are formatted according to various file systems. The FAT file system is a common file system used by host devices to access external storage devices, but similar principles, components, routines and/or techniques as are described in this disclosure may be used to emulate storage drives that are formatted according to other file systems. The drive emulator module 322 may communicate with the client file system to access data or files on the storage drive 310. The drive emulator module 322 may then format the data and/or files to appear as though the data and/or files were transmitted from a storage drive (and file system formatting) of the type that the host device is expecting to see.

[00032] FIG. 4A depicts an illustration of a block diagram showing a conceptual representation of a drive emulator module 422, according to one or more embodiments of the present disclosure. Drive emulator module 422 may be similar to drive emulator module 322 of FIG. 3, for example. As can be seen in FIG. 4A, drive emulator module 422 may appear to a host device (e.g., via a storage device driver 412) to be a storage drive (see "emulated drive" 411). Additionally, drive emulator module 422 may appear to a host device to be a storage drive that is formatted (see generally number 416) according to a particular file system, for example, the file system (e.g., FAT) used by the host device.

[00033] In one or more embodiments of the present disclosure, the drive emulator module 422 may emulate a drive that is formatted according to a FAT file system. A drive that is formatted according to a FAT file system may be divided into three main data areas: the header, the File Allocation Table (FAT) and the cluster area. Each data area may be designated to one or more clusters of the storage drive. The header area may occupy the first sector (or one of the first

sectors) of the storage drive. As an example, a device, driver or operating system that is accessing the storage drive may know (e.g., as part of the file system protocol) to look for the header at the first sector. The header area contains various pieces of information (e.g., static information) about the layout of the storage drive. The header area may contain information about where the File Allocation Table is located, for example, which cluster.

[00034] The File Allocation Table is a table that centralizes information about which storage drive areas (e.g., which clusters) belong to files or directories, which areas are free or possibly unusable, and where each file is stored on the drive. A cluster is the smallest unit that an operating system utilizes when it assigns space on the partition. Each file on the storage drive may occupy one or more of these clusters depending on its size; thus, a file is represented by a chain of these clusters, sometimes referred to as a linked list. The File Allocation Table may include a list of entries that map to each cluster of the drive. Each entry includes various pieces of information, for example, an indication of whether the cluster is used by a file or directory, the cluster number of the next cluster in a chain if this cluster is not the end of the chain, an "end of cluster chain" indications that indicates whether this cluster is the end of a chain and indications of whether the cluster is unused or bad. The cluster area includes the numerous clusters on the storage drive. Clusters are where the data or files are physically stored on the storage drive. Each cluster may be occupied by a file or a directory.

[00035] FIG. 4B depicts an illustration of a block diagram showing a components, routines, connections and/or communications included in an example drive emulator module 422, according to one or more embodiments of the present disclosure. Drive emulator module 422 may be similar to drive emulator module 322 of FIG. 3, for example. FIG. 4B also shows how a driver emulator module 422 may handle various requests communicated by a host device (e.g., via a storage device driver 412). As can be seen in FIG. 4B, driver emulator module 422 may include a cluster emulator 451, a header emulator module 452, a File Allocation Table (FAT) Emulator module 454 and a cluster map 456. The drive emulator module 422 may emulate a drive that is formatted according to a FAT file system, but it should be understood that the various principles, techniques and/or routines described herein may also apply to various other file systems. Therefore, it should be understood that various descriptions of example emulator modules that emulate a FAT file system should be understood as examples used to clearly describe the various principles, techniques and/or routines and should not be understood to limit the scope of the present disclosure.

[00036] Cluster emulator 451 may emulate an interface of a storage drive that may accept requests for data and may return data. For example, cluster emulator 451 may accept a request

(e.g., from a storage device driver 412) that specifies a particular cluster number / address, and the cluster emulator 451 may return data associated with the particular cluster number / address. Cluster emulator 451 may communicate with various other modules of the drive emulator 422 in order to receive data related to a particular cluster number. For example, if the cluster emulator 451 sees a cluster number associated with the header storage area, the cluster emulator 451 may communicate with a header emulator module 452 to receive header information. As another example, if the cluster emulator 451 sees a cluster number associated with the File Allocation Table (FAT) storage area, the cluster emulator 451 may communicate with a FAT emulator module 454 to receive FAT information. As another example, if the cluster emulator 451 sees a cluster number associated with a file storage area, the cluster emulator 451 may communicate with a cluster map 456 to receive file information.

[00037] Header emulator module 452 may emulate the data area of a storage drive that is designated to the header. As one example, a host device (e.g., via a storage device driver 412) may communicate a request to the client device to access information at the expected storage drive location of the header. The drive emulator module 422, using the cluster emulator 451 and header emulator module 452, may detect such requests and may return header information to the host device. The header emulator module 452 may construct header information based on information stored in the storage drive 410 of the client device. For example, the header emulator module may communicate with the storage drive 410 of the client device via an operating system and the client file system 418. In some embodiments, the storage drive 410 may not be formatted according to the same file system used by the host device (e.g., not a FAT file system). In this situation, the header emulator module 452 may request information (e.g., organization information) stored on the storage drive 410 that is similar to a FAT header and may translate this information to a FAT header or construct a FAT header from this information, and may return the FAT header to the cluster emulator, and then to the host device. For example, the header emulator may return various pieces of static header data and may return an indication of a cluster (e.g., an emulated cluster) where the File Allocation Table is located.

[00038] File Allocation Table (FAT) emulator module 452 may emulate the data area of a storage drive that is designated to the FAT. As one example, a host device (e.g., via a storage device driver 412) may communicate a request to the client device to access information at the expected storage drive location (e.g., cluster) of the FAT. The host device may know which cluster to access to receive the FAT based on the header information it received from the header emulator 452. The drive emulator module 422, using the cluster emulator 451 and the File Allocation Table emulator module 452, may detect such requests and may return File Allocation Table information

to the host device. The File Allocation Table emulator module 452 may construct FAT information based on information stored in the storage drive 410 of the client device. For example, the File Allocation Table emulator module 452 may scan or analyze (e.g., via the operating system and the client file system 418) the storage drive 410 to determine the files and/or data that is stored on the storage drive 410. The File Allocation Table emulator module 452 may then construct a list of the files and folders and may construct emulated clusters that the files are assigned to. A file may be assigned to one or more emulated clusters, as they would be in a File Allocation Table found on a real storage drive. The File Allocation Table emulator module 452 may return information to the host device (e.g., via the cluster emulator 451) that is formatted like a File Allocation Table, where the information details which clusters (emulated clusters) are occupied by which files. The File Allocation Table emulator module 452 may also return an indication of the amount of file storage space (e.g., total and free space).

[00039] In some embodiments of the present disclosure, the File Allocation Table emulator module 452 may return to the host device (e.g., via the cluster emulator 451) a subset of all the files and folders on the storage drive 410. In this respect, the client device may prevent some files from being visible and/or accessible by the host device. In these embodiments, the File Allocation Table emulator module 452 may also return a modified indication of the amount of file storage space (e.g., total and free space). In this respect, it may appear to a host device that there is less space on the storage drive of the client device than is actually the case. One or more routines in the drive emulator module 422 and/or in the operating system and/or in a related set of code may create the subset of files and folders. The File Allocation Table emulator module 452 may then scan or analyze the list of files and folders to construct an emulated FAT. In some embodiments of the present disclosure, the File Allocation Table emulator module 452 may return to the host device a list of files and folders on multiple storage drives, for example, the storage drive 410 and a second storage drive (e.g., located in the same client device). In this respect, the drive emulator module 422 may provide a single drive view of multiple drives. In other words, the drive emulator module 422 may appear to a host device like single drive while the represented files and/or folders may actually be located on multiple drives. In these embodiments, the FAT emulator module 454 may construct an emulated file table that represents clusters associated with files from multiple storage drives. Additionally, the cluster map 456 may map various emulated clusters to real files and/or folders located on multiple storage drives.

[00040] Cluster map 456 may emulate the data area of a storage drive that is designated to the clusters that are available for data storage (e.g., storage of files and/or folders). As one example, a host device (e.g., via a storage device driver 412) may communicate a request to the

client device to access a particular cluster, and the host device may expect to receive in return data stored at that cluster (and perhaps related/linked clusters). The host device may know which cluster to access to receive data related to a particular file or folder based on the FAT information it received from the FAT emulator module 454. The drive emulator module 422, using the cluster emulator 451 and the cluster map 452, may detect such requests and may return data (e.g., information related to a particular file or folder) to the host device.

[00041] Cluster map 456 may include a mapping between emulated clusters / cluster numbers and real files and/or folders stored on the storage drive 410. For example, the cluster map may construct a list of emulated clusters / cluster numbers that are associated with or linked to a particular file; however, because the clusters are emulated (i.e., not real physical clusters on a drive), the file may not actually exist directly in the drive emulator module 422 where the emulated cluster is located. Thus, the cluster map 456 may include a mapping or translation from the emulated clusters to the location of the real file/folder in the storage drive. When the cluster map 456 detects a request for a particular cluster, it may reference the cluster map to determine whether the cluster is allocated to a file or folder, or whether the cluster is empty. The cluster map 456 may then request the real file/folder from the storage drive 410, for example, by interacting with the operating system and the client file system 418. The real file/folder may be returned to the cluster map 456, and in turn, the cluster map 456 may return the file/folder to the host device. A similar routine may be used when the host device sends a request to change a file or create a new file. The cluster map 456 may reference the cluster map to determine whether the cluster is allocated to a file or folder, or whether the cluster is empty. The cluster map 456 may then request that the real file/folder from the storage drive 410 be changed or created, for example, by interacting with the operating system and the client file system 418.

[00042] The cluster map may provide access to various files and/or folder on the storage drive 410 without having to copy all of the files and/or folders to a location that is closer to the drive emulator module 422. This may save storage space and/or memory space on the client device. Alternate embodiments of the present disclosure may include a drive emulator that copies various files and/or folders to a location that is closer to the drive emulator module 422, for example, to a volatile memory of the client device. This local storage of files and/or folders may increase the speed of handling requests from a host device for various files and/or folders.

[00043] Referring again to FIG. 3, drive emulator module 322 (and perhaps related code such as the operating system 314 code) may include various routines to maintain consistency between the data emulated in the drive emulator module 322 and the data in the storage drive 310. As can be seen in FIG. 3, various applications (e.g., application 320) and the drive emulator 322

may attempt to read and write data from and to the storage drive 310. Various embodiments of the present disclosure allow for concurrent access by various applications and a host device 304 (e.g., via a storage device driver 312 and a drive emulator 322) to the same storage drive 310. In order to support concurrent access, various consistency routines must ensure that, for example, new changes to a file written to the storage drive 310 by the drive emulator 322 do not get overwritten by new changes to the same file subsequently written by an application 320. By performing these various consistency routines, the present disclosure allows for concurrent access to a storage drive that may not otherwise be supported by some file systems, such as FAT.

[00044] Drive emulator module 322 (and perhaps related code such as the operating system 314 code) may include various routines to maintain consistency of reads (i.e., requesting data/files) from the storage drive 310. For example, the drive emulator module 322 may maintain, for each directory / folder, an area in the emulated/constructed File Allocation Table (FAT) associated with the files in that directory / folder. The drive emulator module 322 may also maintain a directory entry for each directory / folder. The term directory entry may refer to information or "metadata" about all the files and/or sub-folders in the directory. The drive emulator module 322 may determine which files/directories are affected by the read. The drive emulator module 322 may compare its maintained directory entry and FAT area for the affected file/folder to the directory entry and file information of the directory that contains the real file on the storage drive. To perform this comparison, the drive emulator module 322 may access information maintained in the client file system 318 or in the storage drive 310. In this respect, the drive emulator module may acquire a degree of confidence as to whether the data in the storage drive 310 is updated or stale related to the information about the storage drive as maintained in the drive emulator module 322. If the data appears to be stale, the drive emulator may update its information about the storage drive, for example, by updating its FAT emulator module and its cluster emulator module. The drive emulator may read or request the data or file from the storage drive 310 and return the data or file to the storage device driver 312.

[00045] Drive emulator module 322 (and perhaps related code such as the operating system 314 code) may include various routines to maintain consistency of writes (i.e., changing or creating data/files) to the storage drive 310. For example, the drive emulator module 322 may maintain, for each directory / folder, an area in the emulated/constructed File Allocation Table (FAT) associated with the files in that directory / folder. The drive emulator module 322 may also maintain a directory entry for each directory / folder. When the emulator module 322 receives a request form the storage device driver 312 to change or create a file, the emulator module 322 may cause a temporary file (e.g., by temporarily storing the data in volatile memory) to be stored. The drive

emulator module 322 may determine which files/directories are affected by the write. The drive emulator module 322 may then compare its maintained directory entry and FAT area for the affected file/folder to the directory entry and file information of the directory that contains the real file on the storage drive. To perform this comparison, the drive emulator module 322 may access information maintained in the client file system 318 or in the storage drive 310. In this respect, the drive emulator module may acquire a degree of confidence as to whether the data in the storage drive 310 is updated or stale related to the information about the storage drive as maintained in the drive emulator module 322. If the data appears to be stale, the drive emulator may update its information about the storage drive, for example, by updating its FAT emulator module and its cluster emulator module.

[00046] When the corresponding storage drive 310 information and drive emulator module 322 information regarding the file to be written are consistent, the emulator may write the temporarily stored data to the storage drive 310 via local file system 318. In some instances, if information about a file / folder maintained in the drive emulator is not yet consistent with information about the file / folder in the client file system 318 or storage drive 310, the drive emulator may indicate to the host device that the file/folder is "incomplete." In some instances, the drive emulator 322 may indicate to the host device that a write could not be completed (e.g., an error was encountered), for example, if the write could not be completed because the information about the affected files was inconsistent.

[00047] Certain embodiments of the present disclosure may be found in one or more methods for drive emulation for devices with a mass storage feature. With respect to the various methods described herein and depicted in associated figures, it should be understood that, in some embodiments, one or more of the steps described and/or depicted may be performed in a different order. Additionally, in some embodiments, a method may include more or less steps than are described and/or depicted.

[00048] FIG. 5 depicts a flow diagram 500 that shows example steps in a method for drive emulation for devices with a mass storage feature, in accordance with one or more embodiments of the present disclosure. More specifically, FIG. 5 depicts example steps that may be included in a method to initialize a drive emulator module. At step 502, the drive emulator module may construct multiple emulated clusters. For example, each emulated cluster may mimic the behavior of a real cluster on a storage drive by returning information associated with a cluster number or address. At step 504, the drive emulator module may construct an emulated header. The emulated header may mimic the behavior of a cluster on a storage drive that stores information related to the file system header. At step 506, the drive emulator module may determine and/or receive (e.g.,

from the operating system) a list of files and/or folders that will be visible and/or accessible by a host device. The drive emulator module may determine and/or receive one or more associated storage space values, for example, free space and total space. At step 508, the drive emulator module may construct a File Allocation Table (FAT), for example, using various piece of information such as the emulated clusters and the list of files/folders. The constructed FAT may include a list that matches various files/folders to one or more emulated clusters. At step 510, the drive emulator module may construct a cluster map, for example, using various pieces of information such as the list of file/folders and the constructed FAT. The cluster map may map various emulated clusters to real files/folders stored on the client storage drive.

[00049] FIG. 6 depicts a flow diagram 600 that shows example steps in a method for drive emulation for devices with a mass storage feature, in accordance with one or more embodiments of the present disclosure. More specifically, FIG. 6 depicts example steps that may be used to handle requests from a host device to access data and/or files on a client storage drive. At step 602, the drive emulator module may receive (e.g., from a host device via a storage device driver) a request to access a cluster (e.g., an emulated cluster) that stores header information. At step 604, the drive emulator module may reference its emulated header and may return (e.g., to a host device via a storage device driver) header information to the host device. The header information may include the cluster number/address of the File Allocation Table (FAT). At step 606, the drive emulator module may receive a request to access a cluster (e.g., an emulated cluster) that stores the FAT. At step 608, the drive emulator module may reference the constructed FAT and return it to the host device. The FAT may include information about which clusters include information related to particular files and/or folders.

[00050] At step 610, the drive emulator module may receive a request to access a cluster that stores a particular file/folder (or a cluster that stores the start of a file/folder). At step 612, the drive emulator module may reference its cluster map to determine real files/folders on the storage drive that are associated with particular emulated clusters. The drive emulator module may access (e.g., read or write) a real file/folder via the client file system. At step 614, if the host device requested to read a particular file/folder, the drive emulator module may request and receive the real file/folder from the storage drive via the client file system, and in turn, may return the file/folder to the host device. At step 616, if the information maintained by the drive emulator module about the client storage drive has become out of date or inconsistent, the drive emulator module may update its maintained information (e.g., areas in the constructed FAT and/or maintained directory entries). The drive emulator module may return updated FAT information to the host device. As can be seen in FIG. 6, generally at number 618, various steps of the method

600 may repeat such that the host device may access various files and/or folders of the client storage drive.

[00051] Various embodiments of the present disclosure describe one or more systems, methods, routines and/or techniques for drive emulation for devices with a mass storage feature. In one or more embodiments, a method may be executed on a data processing system, computer, computing device, mobile device, smartphone or the like. The method may include establishing a mass storage mode for the data processing system. In some embodiments, the mass storage mode follows the USB mass storage standard. In the mass storage mode, the data processing system may be adapted to allow a connected host device to access, via a storage device driver, storage capabilities of the data processing system. The method may include receiving a low-level request from the storage device driver to access file information, where the request may be formatted according to a storage drive protocol operable to communicate with a storage drive directly at a low level. The method may include translating the low-level request into a translated request that is formatted to interact with a client file system that is used to access a client storage drive in the data processing system. The method may include sending the translated request, via the client file system, to the client storage drive to access file information related to the low-level request. The method may include returning to the storage device driver a response related to the low-level request. The response may be formatted according to the storage drive protocol such that the response appears to the storage device driver as though it came directly from a storage drive.

[00052] In some embodiments, the method may include allowing access, via the client file system, to the client storage drive by one or more applications running on the data processing system. The access by the one or more applications may occur while the data processing system is in the mass storage mode. In some embodiments, the client storage drive may be formatted according to the client file system and the storage drive protocol may be operable to allow communication with a storage drive that is formatted according to a different file system than the client file system. In some embodiments, the client file system is an EXT4 file system and the different file system is a FAT file system. In some embodiments, the low-level request includes a cluster address and translating the low-level request into the translated request may include referencing a cluster map that associates cluster addresses with files stored on the client storage drive.

[00053] In some embodiments, the method may include checking for drive consistency before sending the translated request. Checking for drive consistency may include determining which stored files and folders stored on the client storage drive will be affected by the translated request and determining whether cached information related to the affected stored files and folders

is up to date with the actual status of the stored files and folders on the client storage drive. The method may include, when the low-level request is a write command, causing temporary storage in a volatile memory unit of data related to the write command. The method may include, when it is determined that the cached information related to the affected stored files and folders is up to date, sending the translated request including the temporarily stored data to the client storage drive.

[00054] In one or more embodiments of the present disclosure, a method may be executed on a data processing system, computer, computing device, mobile device, smartphone or the like. The method may include constructing a plurality of emulated clusters, where each emulated cluster mimics the behavior of a cluster of a storage drive. The method may include accessing, via a client file system, a client storage drive in the data processing system to receive organization information that includes information about the organization of data on the client storage drive. The method may include constructing an emulated header that mimics header information that would be stored on a storage drive, where the emulated head is constructed using the organization information. The method may include accessing, via the client file system, the client storage drive to receive a list of files that are stored on the client storage drive of the data processing system. The method may include constructing an emulated file allocation table that indicates for each emulated cluster whether the emulated cluster is associated with a listed file from the list of files. The method may include constructing a cluster map that, for each listed file in the list of files, links the listed file to an associated filed stored on the client storage drive. In some embodiments, list of files constitutes a subset of the total set of files that is stored on the client storage drive of the data processing system.

[00055] In some embodiments, the method may include receiving a low-level request from a host device to access header information. The method may include returning to the host device information from the emulated header. The method may include receiving a low-level request from the host device to access file allocation table information. The method may include returning to the host device information from the emulated file allocation table. The method may include receiving a low-level request from the host device to receive file information associated with a starting cluster number. The method may include accessing one or more of the emulated clusters and the cluster map to determine a stored filed stored on the client storage drive. The method may include retrieving, via the client file system, the stored file from the client storage drive and returning the stored file to the host device.

[00056] In some embodiments, the method may include checking for drive consistency before retrieving the stored file from the client storage drive. Checking for drive consistency may include determining which stored files and folders stored on the client storage drive will be affected

by retrieving the stored file from the client storage drive and determining whether cached information related to the affected stored files and folders is up to date with the actual status of the stored files and folders on the client storage drive.

[00057] In one or more embodiments of the present disclosure, a data processing system may include one or more memory units that store computer code, a client storage drive, a data transfer port adapted to couple to a host device that is adapted to access storage capabilities of the data processing system, and one or more processor units coupled to the one or more memory units, the client storage drive and the data transfer port. The one or more processor units may execute the computer code stored in the one or more memory units to establish a mass storage mode for the data processing system when the data processing system is coupled to the host device. The one or more processor units may execute the computer code stored in the one or more memory units to implement a drive emulator module that emulates, from the perspective of the host device, the behavior of a storage drive. The drive emulator module being implemented to receive a low-level request from the host device to access file information, the request being formatted according to a storage drive protocol operable to communicate with a storage drive directly at a low level. The drive emulator module being implemented to translate the low-level request into a translated request that is formatted to interact with a client file system that is used to access the client storage drive. The drive emulator module being implemented to send the translated request, via the client file system, to the client storage drive to access file information related to the low-level request. The drive emulator module being implemented to return to the host device a response related to the low-level request. The response may be formatted according to the storage drive protocol such that the response appears to the host device as though it came directly from a storage drive. In some embodiments, the data transfer port may be a USB port and the mass storage mode may follow the USB mass storage standard.

[00058] In some embodiments, the one or more processor units may execute the computer code stored in the one or more memory units to allow access, via the client file system, to the client storage drive by one or more applications running on the data processing system. The access by the one or more applications may occur while the data processing system is in the mass storage mode. In some embodiments, the client storage drive may be formatted according to the client file system, and \ the storage drive protocol may be operable to allow communication with a storage drive that is formatted according to a different file system than the client file system. In some embodiments, the client file system is an EXT4 file system and the different file system is a FAT file system. In some embodiments, the low-level request may include a cluster address, and

translating the low-level request into the translated request may include referencing a cluster map that associates cluster addresses with files stored on the client storage drive.

[00059] In some embodiments, the drive emulator module may be implemented to check for drive consistency before sending the translated request. Checking for drive consistency may include determining which stored files and folders stored on the client storage drive will be affected by the translated request, and determining whether cached information related to the affected stored files and folders is up to date with the actual status of the stored files and folders on the client storage drive.

[00060] The methods, routines and solutions of the present disclosure, including the example methods and routines illustrated in the flowcharts and block diagrams of the different depicted embodiments may be implemented as software executed by a data processing system that is programmed such that the data processing system is adapted to perform and/or execute the methods, routines, techniques and solutions described herein. Each block or symbol in a block diagram or flowchart diagram referenced herein may represent a module, segment or portion of computer usable or readable program code which comprises one or more executable instructions for implementing, by one or more data processing systems, the specified function or functions. In some alterative implementations of the present disclosure, the function or functions illustrated in the blocks or symbols of a block diagram or flowchart may occur out of the order noted in the figures. For example in some cases two blocks or symbols shown in succession may be executed substantially concurrently or the blocks may sometimes be executed in the reverse order depending upon the functionality involved. Part or all of the computer code may be loaded into the memory of a data processing system before the data processing system executes the code.

[00061] FIG. 7 depicts a block diagram of an example data processing system 700 that may be included within a mobile device or smartphone 701, according to one or more embodiments of the present disclosure. The data processing system 700 may be used to execute, either partially or wholly, one or more of the methods, routines and/or solutions of the present disclosure. In some embodiments of the present disclosure, more than one data processing system, for example similar to data processing systems 700, may be used to implement the methods, routines, techniques and/or solutions described herein. In the example of FIG. 7, data processing system 700 may include a communications fabric 702 which provides communications between components, for example a processor unit 704, a memory 706, a persistent storage 708, a communications unit 710, an input/output (I/O) unit 712 and a display 714. A bus system may be used to implement communications fabric 702 and may be comprised of one or more buses such as a system bus or an

input/output bus. The bus system may be implemented using any suitable type of architecture that provides for a transfer of data between different components or devices attached to the bus system.

[00062] Processor unit 704 may serve to execute instructions (for example, a software program, an application, SDK code, native OS code and the like) that may be loaded into the data processing system 700, for example, into memory 706. Processor unit 704 may be a set of one or more processors or may be a multiprocessor core depending on the particular implementation. Processor unit 704 may be implemented using one or more heterogeneous processor systems in which a main processor is present with secondary processors on a single chip. As another illustrative example, processor unit 704 may be a symmetric multi-processor system containing multiple processors of the same type.

[00063] Memory 706 may be, for example, a random access memory or any other suitable volatile or nonvolatile storage device. Memory 706 may include one or more layers of cache memory. Persistent storage 708 may take various forms depending on the particular implementation. For example, persistent storage 708 may contain one or more components or devices. For example, persistent storage 708 may be a hard drive, a solid-state drive, a flash memory or some combination of the above.

[00064] Instructions for an operating system may be located on persistent storage 708. In one specific embodiment, the operating system may be some version of a number of known operating systems for mobile devices or smartphones (e.g, Android, iOS, etc.). Instructions for applications and/or programs may also be located on persistent storage 708. These instructions may be loaded into memory 706 for execution by processor unit 704. For example, the methods and/or processes of the different embodiments described in this disclosure may be performed by processor unit 704 using computer implemented instructions which may be loaded into a memory such as memory 706. These instructions are referred to as program code, computer usable program code or computer readable program code that may be read and executed by a processor in processor unit 704.

[00065] Display 714 may provide a mechanism to display information to a user, for example, via a LCD or LED screen or monitor, or other type of display. It should be understood, throughout this disclosure, that the term "display" may be used in a flexible manner to refer to either a physical display such as a physical screen, or to the image that a user sees on the screen of a physical device. Input/output (I/O) unit 712 allows for input and output of data with other devices that may be connected to data processing system 700. Input/output devices can be coupled to the system either directly or through intervening I/O controllers.

[00066] Communications unit 710 may provide for communications with other data processing systems or devices, for example, via one or more networks. Communications unit 710 may be a network interface card. Communications unit 710 may provide communications through the use of wired and/or wireless communications links. In some embodiments, the communications unit may include circuitry that communicates according to various wireless communication standards, for example, cellular standards, WIFI standards, BlueTooth standards and the like.

[00067] The different components illustrated for data processing system 700 are not meant to provide architectural limitations to the manner in which different embodiments may be implemented. The different illustrative embodiments may be implemented in a data processing system including components in addition to or in place of those illustrated for data processing system 700. Other components shown in FIG. 7 can be varied from the illustrative examples shown.

[00068] The description of the different advantageous embodiments has been presented for purposes of illustration and the description and is not intended to be exhaustive or limited to the embodiments in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. Further different advantageous embodiments may provide different advantages as compared to other advantageous embodiments. The embodiment or embodiments selected are chosen and described in order to best explain the principles of the embodiments of the practical application and to enable others of ordinary skill in the art to understand the disclosure for various embodiments with various modifications as are suited to the particular use contemplated.

CLAIMS

1. A method executed by a data processing system having one or more processors, the method comprising:

establishing a mass storage mode for the data processing system,

wherein, in the mass storage mode, the data processing system is adapted to allow a connected host device to access, via a storage device driver, storage capabilities of the data processing system;

receiving a low-level request from the storage device driver to access file information, the request being formatted according to a storage drive protocol operable to communicate with a storage drive directly at a low level;

translating the low-level request into a translated request that is formatted to interact with a client file system that is used to access a client storage drive in the data processing system;

sending the translated request, via the client file system, to the client storage drive to access file information related to the low-level request; and

returning to the storage device driver a response related to the low-level request, wherein the response is formatted according to the storage drive protocol such that the response appears to the storage device driver as though it came directly from a storage drive.

2. The method of claim 1, wherein the mass storage mode follows the USB mass storage standard, and wherein the low-level request is formatted such that a storage drive can directly interpret the request without significantly reformatting the request.

3. The method of claim 1, further comprising allowing access, via the client file system, to the client storage drive by one or more applications running on the data processing system, wherein the access by the one or more applications may occur while the data processing system is in the mass storage mode.

4. The method of claim 1, wherein the client storage drive is formatted according to the client file system, and wherein the storage drive protocol is operable to allow communication with a storage drive that is formatted according to a different file system than the client file system.

5. The method of claim 4, wherein the client file system is an EXT4 file system and the different file system is a FAT file system.

6. The method of claim 1, wherein the low-level request includes a cluster address, and wherein translating the low-level request into the translated request includes referencing a cluster map that associates cluster addresses with files stored on the client storage drive.

7. The method of claim 1, further comprising checking for drive consistency before sending the translated request, wherein checking for drive consistency includes:

determining which stored files and folders stored on the client storage drive will be affected by the translated request; and

determining whether cached information related to the affected stored files and folders is up to date with the actual status of the stored files and folders on the client storage drive.

8. The method of claim 7, further comprising:

when the low-level request is a write command, causing temporary storage in a volatile memory unit of data related to the write command; and

when it is determined that the cached information related to the affected stored files and folders is up to date, sending the translated request including the temporarily stored data to the client storage drive.

9. A method executed by a data processing system having one or more processors, the method comprising:

constructing a plurality of emulated clusters, where each emulated cluster mimics the behavior of a cluster of a storage drive;

accessing, via a client file system, a client storage drive in the data processing system to receive organization information that includes information about the organization of data on the client storage drive;

constructing an emulated header that mimics header information that would be stored on a storage drive, wherein the emulated head is constructed using the organization information;

accessing, via the client file system, the client storage drive to receive a list of files that are stored on the client storage drive of the data processing system;

constructing an emulated file allocation table that indicates for each emulated cluster whether the emulated cluster is associated with a listed file from the list of files; and

constructing a cluster map that, for each listed file in the list of files, links the listed file to an associated filed stored on the client storage drive.

10. The method of claim 9, wherein the list of files constitutes a subset of the total set of files that is stored on the client storage drive of the data processing system.

11. The method of claim 9, further comprising:

receiving a low-level request from a host device to access header information;

returning to the host device information from the emulated header;

receiving a low-level request from the host device to access file allocation table information; and

returning to the host device information from the emulated file allocation table.

12. The method of claim 9, further comprising:

receiving a low-level request from the host device to receive file information associated with a starting cluster number;

accessing one or more of the emulated clusters and the cluster map to determine a stored filed stored on the client storage drive;

retrieving, via the client file system, the stored file from the client storage drive; and

returning the stored file to the host device.

13. The method of claim 12, further comprising checking for drive consistency before retrieving the stored file from the client storage drive, wherein checking for drive consistency includes:

determining which stored files and folders stored on the client storage drive will be affected by retrieving the stored file from the client storage drive; and

determining whether cached information related to the affected stored files and folders is up to date with the actual status of the stored files and folders on the client storage drive.

14. A data processing system, comprising:

one or more memory units that store computer code;

a client storage drive;

a data transfer port adapted to couple to a host device that is adapted to access storage capabilities of the data processing system; and

one or more processor units coupled to the one or more memory units, the client storage drive and the data transfer port, wherein the one or more processor units execute the computer code stored in the one or more memory units to:

establish a mass storage mode for the data processing system when the data processing system is coupled to the host device; and

implement a drive emulator module that emulates, from the perspective of the host device, the behavior of a storage drive, the drive emulator module being implemented to:

receive a low-level request from the host device to access file information, the request being formatted according to a storage drive protocol operable to communicate with a storage drive directly at a low level;

translate the low-level request into a translated request that is formatted to interact with a client file system that is used to access the client storage drive;

send the translated request, via the client file system, to the client storage drive to access file information related to the low-level request; and

return to the host device a response related to the low-level request, wherein the response is formatted according to the storage drive protocol such that the response appears to the host device as though it came directly from a storage drive.

15. The data processing system of claim 14, wherein the data transfer port is a USB port and the mass storage mode follows the USB mass storage standard.

16. The data processing system of claim 14, wherein the one or more processor units execute the computer code stored in the one or more memory units to allow access, via the client file system, to the client storage drive by one or more applications running on the data processing system, wherein the access by the one or more applications may occur while the data processing system is in the mass storage mode.

17. The data processing system of claim 14, wherein the client storage drive is formatted according to the client file system, and wherein the storage drive protocol is operable to allow communication with a storage drive that is formatted according to a different file system than the client file system.

18. The data processing system of claim 17, wherein the client file system is an EXT4 file system and the different file system is a FAT file system.

19. The data processing system of claim 14, wherein the low-level request includes a cluster address, and wherein translating the low-level request into the translated request includes

referencing a cluster map that associates cluster addresses with files stored on the client storage drive.

20. The data processing system of claim 14, wherein the drive emulator module is implemented to check for drive consistency before sending the translated request, wherein checking for drive consistency includes:

determining which stored files and folders stored on the client storage drive will be affected by the translated request; and

determining whether cached information related to the affected stored files and folders is up to date with the actual status of the stored files and folders on the client storage drive.

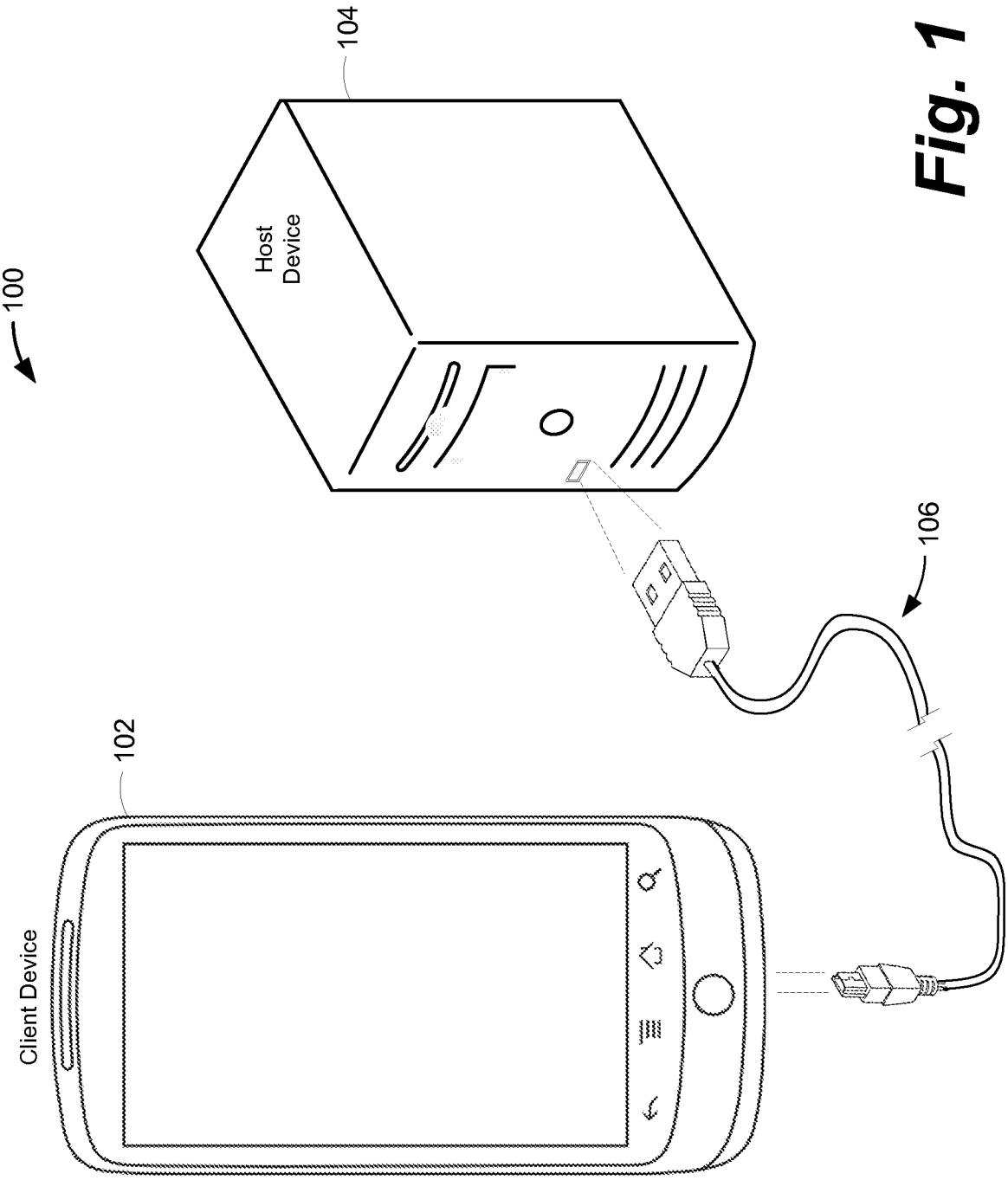


Fig. 1

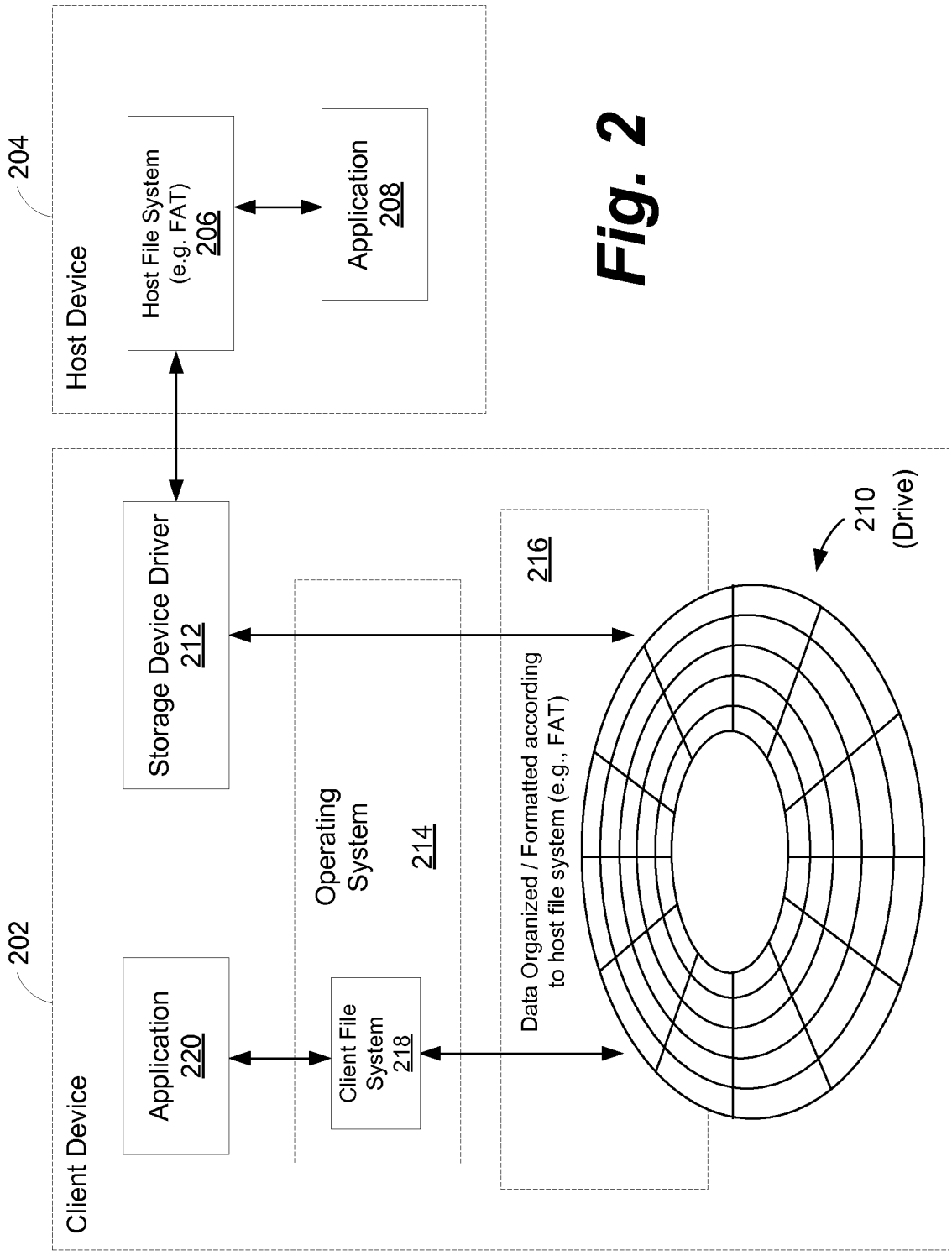

204
Host Device
Host File System
(e.g. FAT)
206
Application
208
202
Client Device
Application
220
Storage Device Driver
212
Operating
System
214
Client File
System
218
216
Data Organized / Formatted according
to host file system (e.g., FAT)
210
(Drive)

Fig. 2

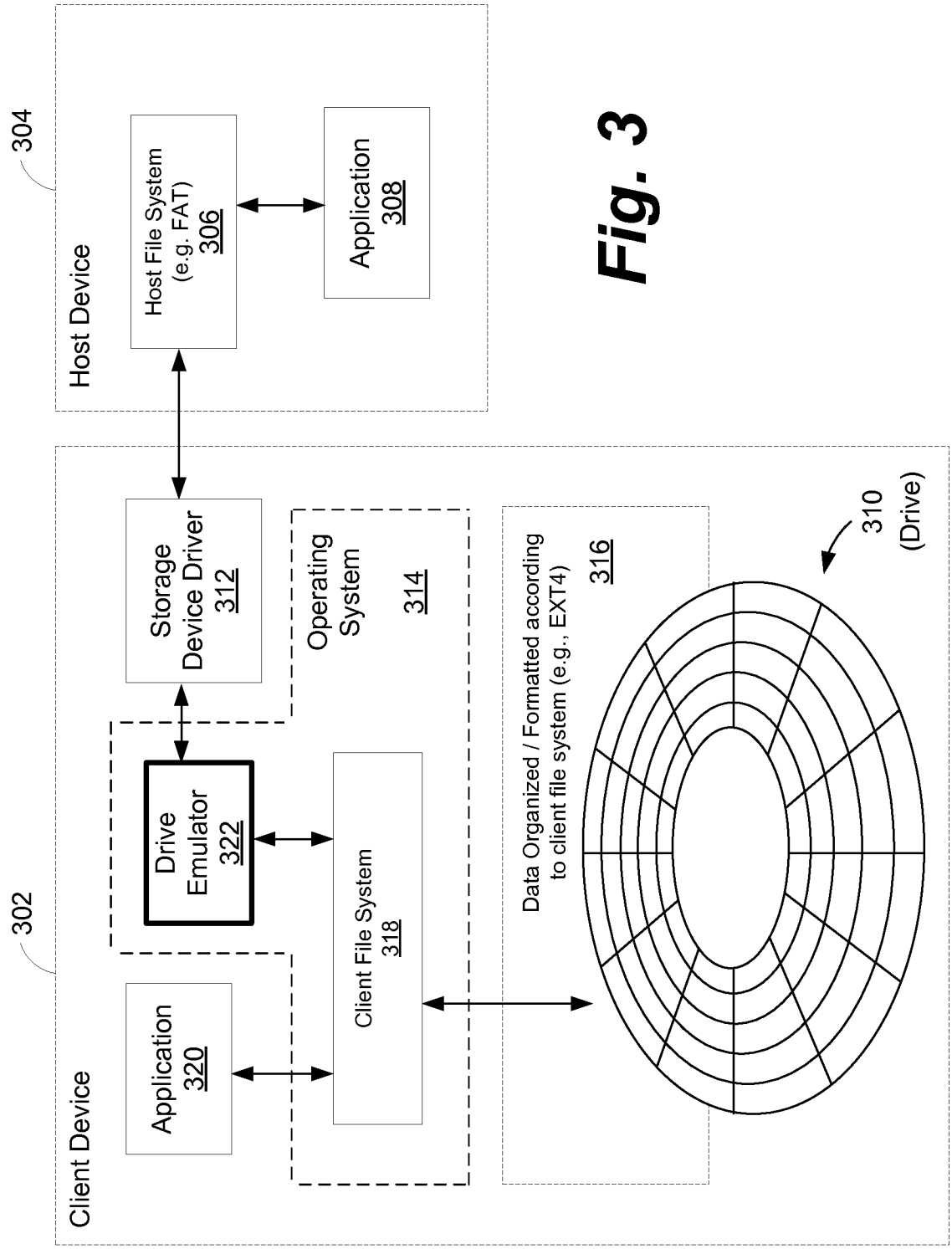

304
Host Device
Host File System (e.g. FAT) 306
Application 308
302
Client Device
Storage Device Driver 312
Operating System 314
Drive Emulator 322
Client File System 318
Application 320
Data Organized / Formatted according to client file system (e.g., EXT4) 316
310 (Drive)

Fig. 3

Fig. 4A

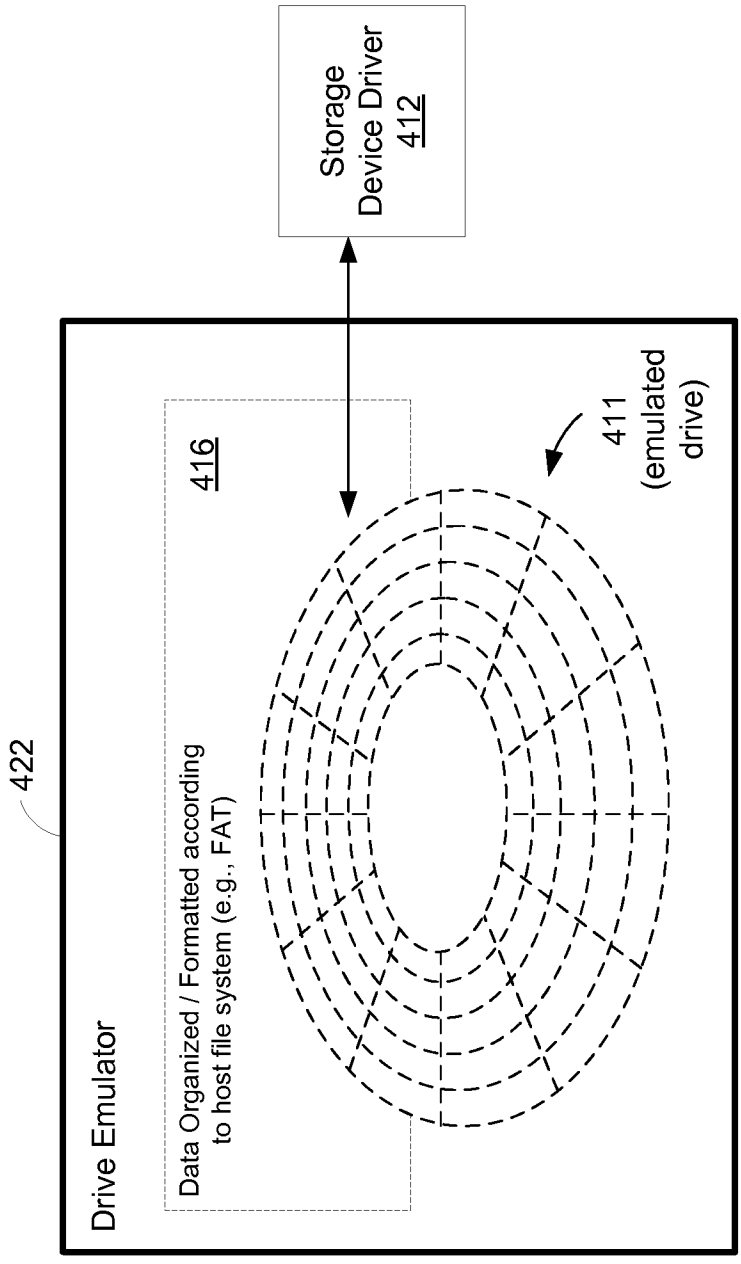

Storage
Device Driver
412
422
Drive Emulator
416
Data Organized / Formatted according
to host file system (e.g., FAT)
411
(emulated
drive)

Fig. 4B

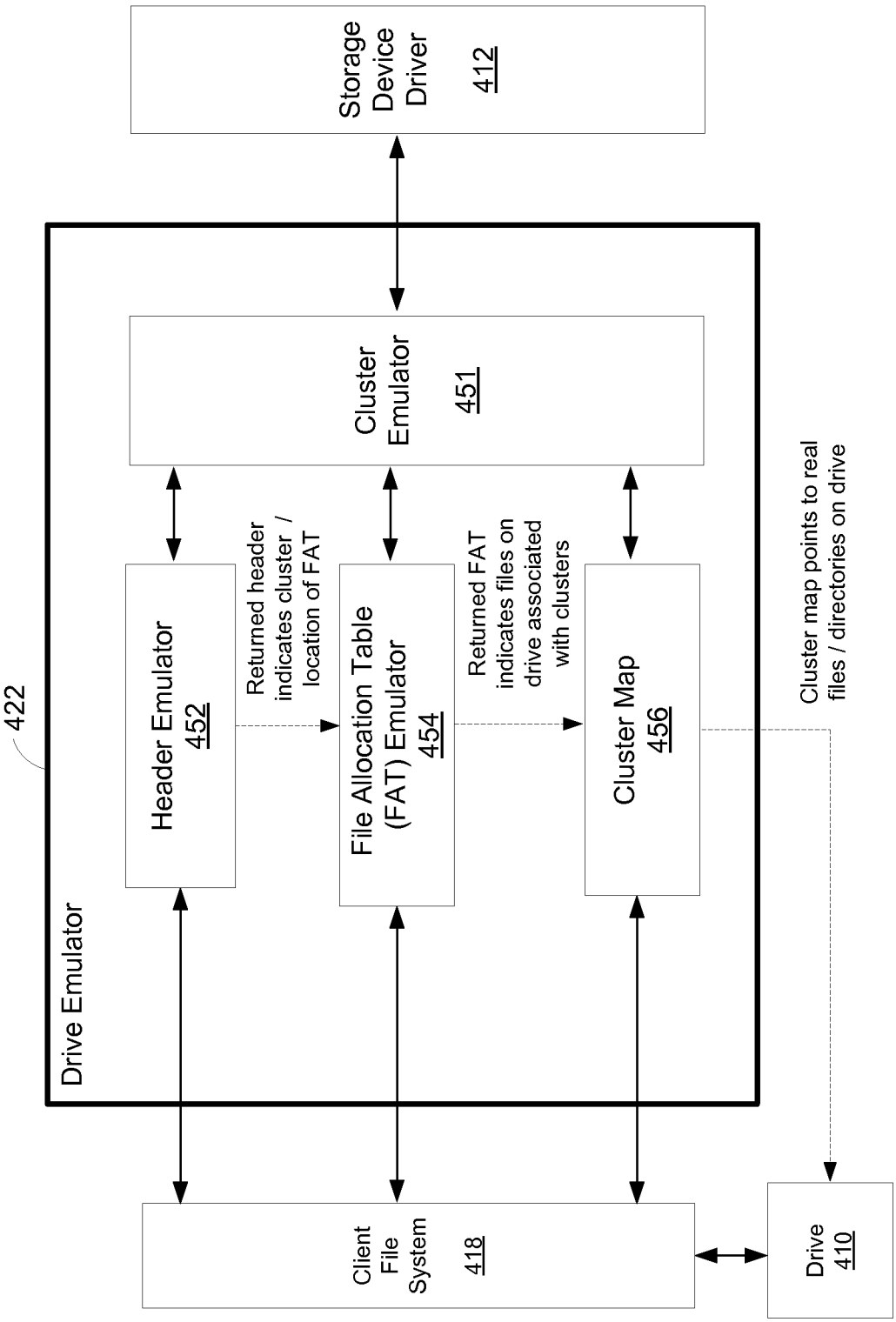

Storage Device Driver 412
Drive Emulator
422
Cluster Emulator 451
Header Emulator 452
Returned header indicates cluster / location of FAT
File Allocation Table (FAT) Emulator 454
Returned FAT indicates files on drive associated with clusters
Cluster Map 456
Cluster map points to real files / directories on drive
Client File System 418
Drive 410

Fig. 5

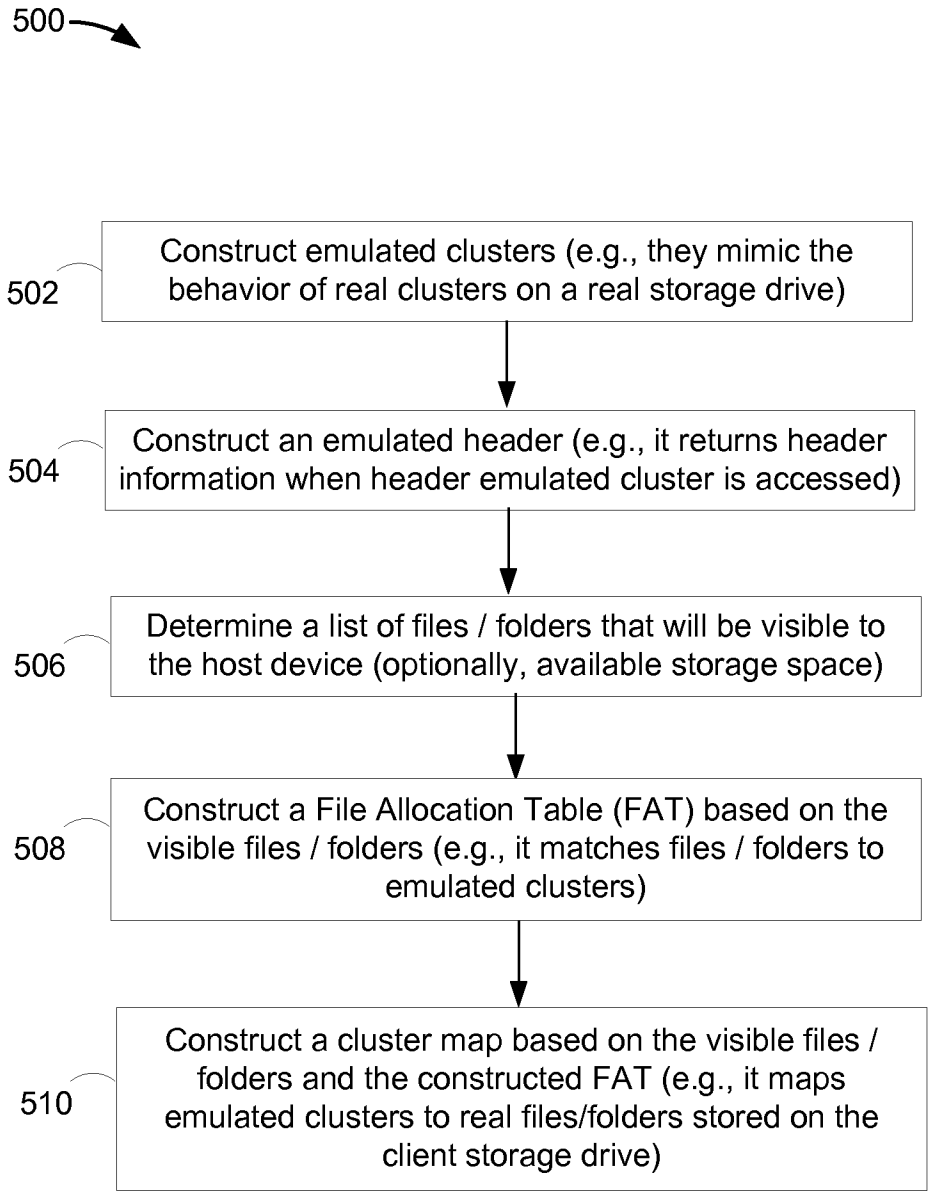

500
502
Construct emulated clusters (e.g., they mimic the behavior of real clusters on a real storage drive)
504
Construct an emulated header (e.g., it returns header information when header emulated cluster is accessed)
506
Determine a list of files / folders that will be visible to the host device (optionally, available storage space)
508
Construct a File Allocation Table (FAT) based on the visible files / folders (e.g., it matches files / folders to emulated clusters)
510
Construct a cluster map based on the visible files / folders and the constructed FAT (e.g., it maps emulated clusters to real files/folders stored on the client storage drive)

Fig. 6

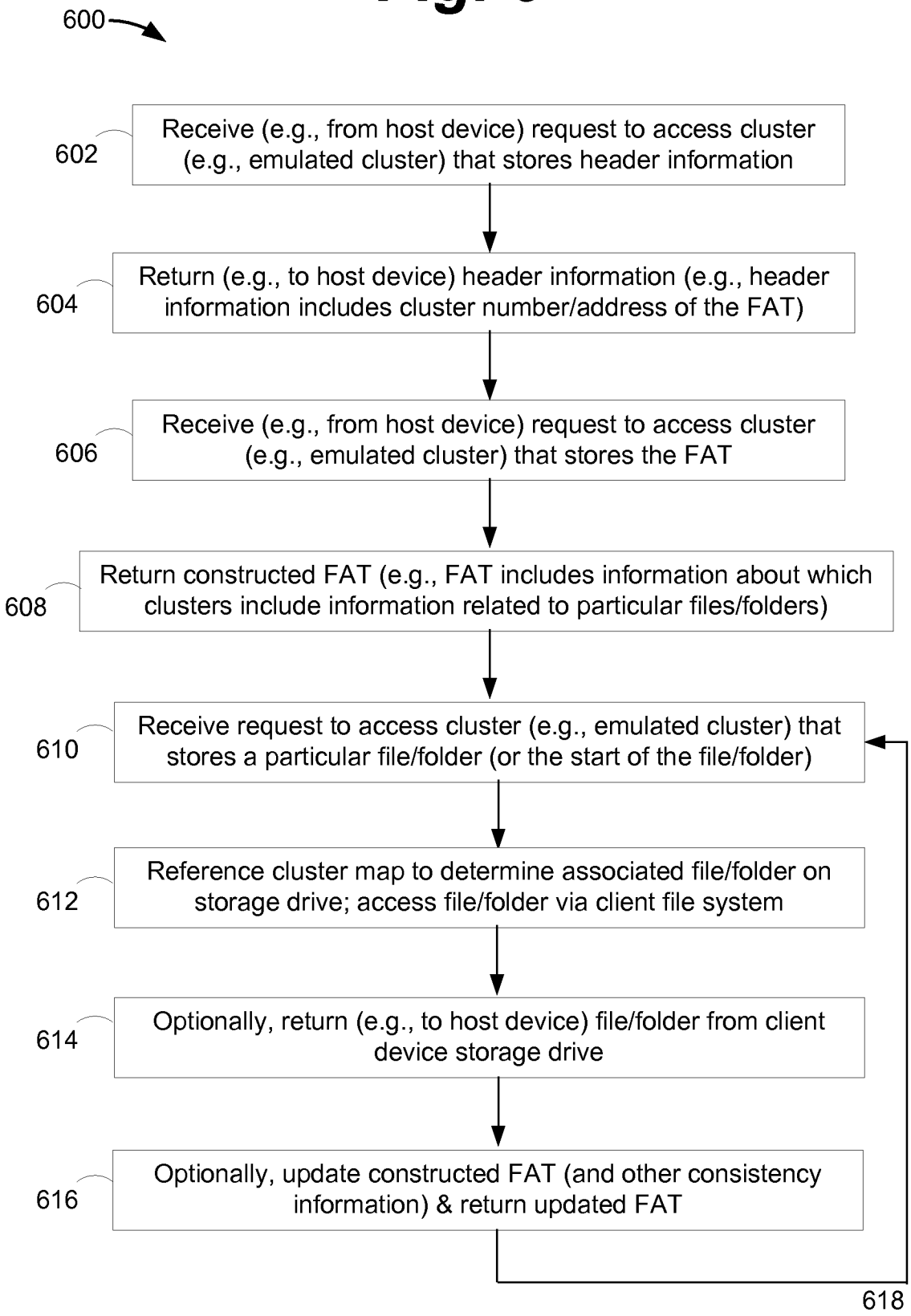
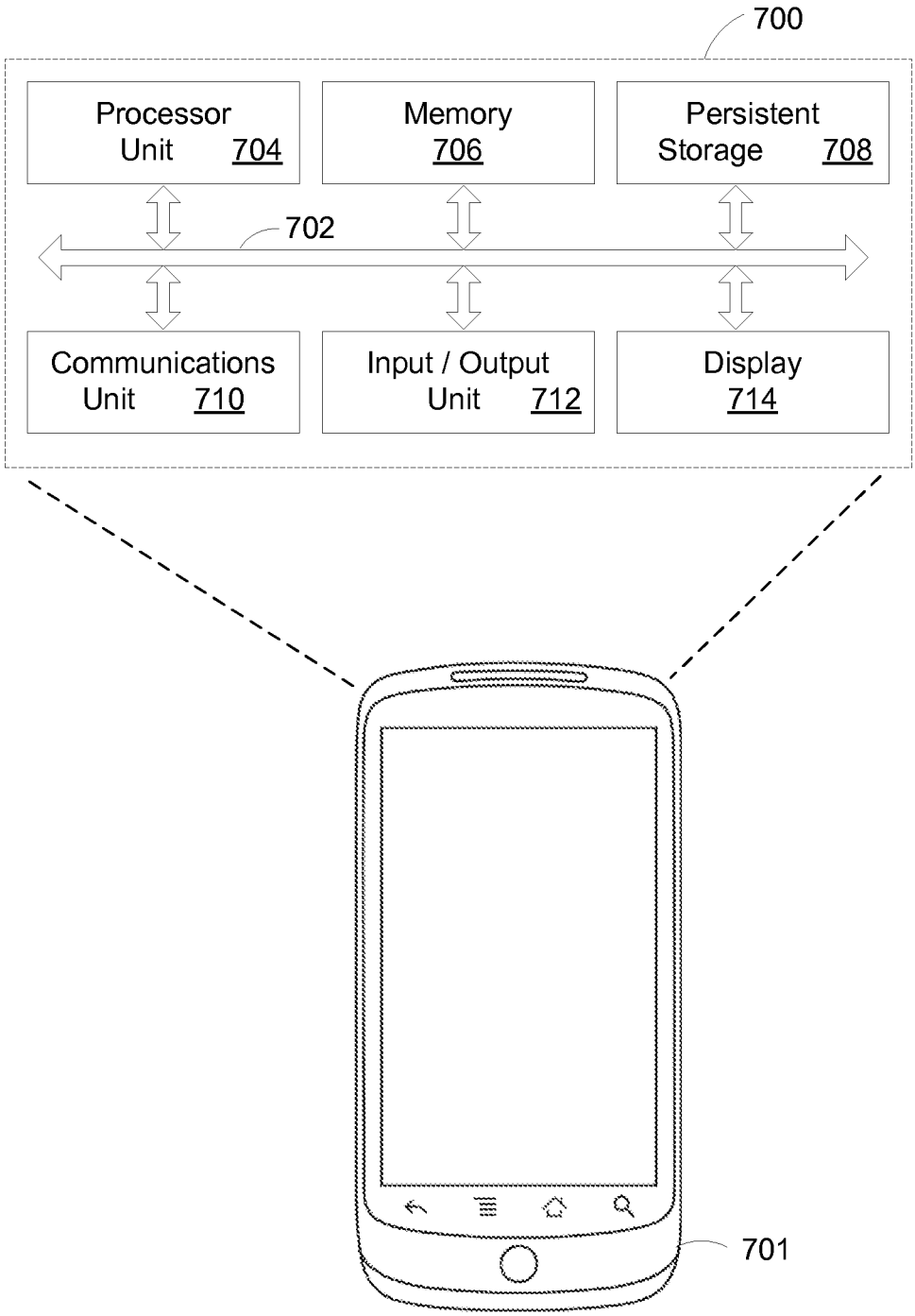

600
602
Receive (e.g., from host device) request to access cluster (e.g., emulated cluster) that stores header information
604
Return (e.g., to host device) header information (e.g., header information includes cluster number/address of the FAT)
606
Receive (e.g., from host device) request to access cluster (e.g., emulated cluster) that stores the FAT
608
Return constructed FAT (e.g., FAT includes information about which clusters include information related to particular files/folders)
610
Receive request to access cluster (e.g., emulated cluster) that stores a particular file/folder (or the start of the file/folder)
612
Reference cluster map to determine associated file/folder on storage drive; access file/folder via client file system
614
Optionally, return (e.g., to host device) file/folder from client device storage drive
616
Optionally, update constructed FAT (and other consistency information) & return updated FAT
618

Fig. 7


700
Processor Unit 704
Memory 706
Persistent Storage 708
702
Communications Unit 710
Input / Output Unit 712
Display 714
701

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/061796

A. CLASSIFICATION OF SUBJECT MATTER

G06F 13/10(2006.01)i, G06F 13/38(2006.01)i, G06F 9/44(2006.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 13/10; G06F 15/16; G06F 3/048; G06F 15/173; H04L 9/32; H04L 9/00; G06F 13/38; G06F 9/44

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: host, client, connect, emulate, protocol, format, translate, file, access

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2011-0307616 A1 (JIAN LIN et al.) 15 December 2011 See paragraphs 14-44; figures 1-5; and claim 1.	1-20
A	US 2011-0055407 A1 (GREGORY T. LYDON et al.) 03 March 2011 See paragraphs 20-42; claims 13, 20; and figure 4.	1-20
A	US 2009-0292995 A1 (GOWTHAM ANNE et al.) 26 November 2009 See paragraphs 24-56; and figures 1-2.	1-20
A	US 2010-0251352 A1 (RICHARD R. ZARCHY et al.) 30 September 2010 See paragraphs 30-85; and figures 2-9.	1-20
A	US 2009-0037594 A1 (GIL SEVER et al.) 05 February 2009 See paragraphs 23-63; and figures 1-3.	1-20

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

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 January 2014 (27.01.2014)	**28 January 2014 (28.01.2014)**

Name and mailing address of the ISA/KR
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
CHOI, Sung Hee
Telephone No. +82-42-481-8740

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2013/061796

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2011-0307616 A1	15/12/2011	US 2010-121959 A1	13/05/2010
		US 8037191 B2	11/10/2011
		US 8275892 B2	25/09/2012
US 2011-0055407 A1	03/03/2011	CN 102483728 A	30/05/2012
		EP 2473921 A1	11/07/2012
		GB 201014178 D	06/10/2010
		GB 201118925 D	14/12/2011
		GB 201118926 D	14/12/2011
		GB 2473123 A	02/03/2011
		GB 2473123 B	11/09/2013
		GB 2484832 A	25/04/2012
		GB 2484832 B	27/03/2013
		GB 2484833 A	25/04/2012
		GB 2484833 B	21/11/2012
		US 8626932 B2	07/01/2014
		WO 2011-028388 A1	10/03/2011
US 2009-0292995 A1	26/11/2009	None	
US 2010-0251352 A1	30/09/2010	WO 2010-111165 A1	30/09/2010
US 2009-0037594 A1	05/02/2009	AU 2004-295851 A1	16/06/2005
		AU 2004-295851 B2	11/03/2010
		EP 1690363 A2	16/08/2006
		EP 1690363 A4	08/02/2012
		US 8544062 B2	24/09/2013
		WO 2005-054973 A2	16/06/2005
		WO 2005-054973 A3	06/10/2005