Innovation, Sciences et Développement économique Canada
Office de la Propriété Intellectuelle du Canada

Innovation, Science and Economic Development Canada
Canadian Intellectual Property Office

CA 2953031 C 2023/08/08

(11)(21) 2 953 031

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 2015/08/25
(87) Date publication PCT/PCT Publication Date: 2016/03/03
(45) Date de délivrance/Issue Date: 2023/08/08
(85) Entrée phase nationale/National Entry: 2016/12/20
(86) N° demande PCT/PCT Application No.: US 2015/046711
(87) N° publication PCT/PCT Publication No.: 2016/033056
(30) Priorité/Priority: 2014/08/26 (US62/041,941)

(51) Cl.Int./Int.Cl. *G06F 3/06* (2006.01),
G06F 12/08 (2016.01)

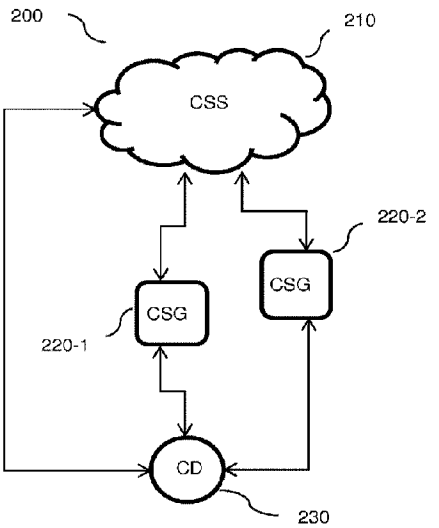
(72) Inventeur/Inventor:
BRAND, ARON, IL

(73) Propriétaire/Owner:
CTERA NETWORKS, LTD., IL

(74) Agent: AGENCE DE BREVETS FOURNIER

(54) Titre : PROCEDE ET DISPOSITIF INFORMATIQUE POUR PERMETTRE L'ACCES SYNCHRONISE AUX SYSTEME DE STOCKAGE EN NUAGE SUR LA BASE D'UN SUIVI D'ELEMENT DE REMPLACEMENT

(54) Title: A METHOD AND COMPUTING DEVICE FOR ALLOWING SYNCHRONIZED ACCESS TO CLOUD STORAGE SYSTEMS BASED ON STUB TRACKING



(57) Abrégé/Abstract:
A method and computing device for allowing synchronized access to a cloud storage system (CSS) based on stub tracking. The computing device comprises: a processing unit; a storage system, wherein the storage system is configured to maintain objects, each object having a state, wherein each state is any one of: a synchronized state, and a stub state; and a memory, the memory containing instructions that, when executed by the processing unit, configure the computing device to: receive a request from a client to access an object maintained in the storage system; change the state of the requested object from a stub state to a synchronized state upon receiving the request; and synchronize data of the requested object having a synchronized state with the CSS.

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 March 2016 (03.03.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/033056 A1

(51) International Patent Classification:
G06F 3/06 (2006.01) *G06F 12/08* (2016.01)

(21) International Application Number:
PCT/US2015/046711

(22) International Filing Date:
25 August 2015 (25.08.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/041,941 26 August 2014 (26.08.2014) US

(71) Applicant: **CTERA NETWORKS, LTD.** [IL/IL]; 25 Efal Street, 49511-25 Petach-tikva (IL).

(71) Applicant *(for BZ only)*: **M&B IP ANALYSTS, LLC** [US/US]; 45 S. Park Place #262, Morristown, NJ 07960 (US).

(72) Inventor: **BRAND, Aron**; Ctera Networks, Ltd., 25 Efal Street, 49511-25 Petach-tikva (IL).

(74) Agents: **BEN-SHIMON, Michael** et al.; M&B IP Analysts, LLC, 45 S. Park Place #262, Morristown, NJ 07960 (US).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK,

[Continued on next page]

(54) Title: A METHOD AND COMPUTING DEVICE FOR ALLOWING SYNCHRONIZED ACCESS TO CLOUD

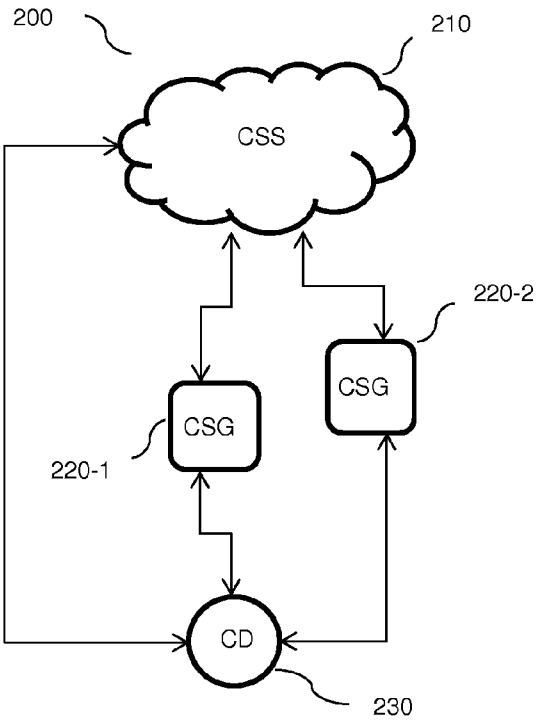


FIG. 2

(57) Abstract: A method and computing device for allowing synchronized access to a cloud storage system (CSS) based on stub tracking. The computing device comprises: a processing unit; a storage system, wherein the storage system is configured to maintain objects, each object having a state, wherein each state is any one of: a synchronized state, and a stub state; and a memory, the memory containing instructions that, when executed by the processing unit, configure the computing device to: receive a request from a client to access an object maintained in the storage system; change the state of the requested object from a stub state to a synchronized state upon receiving the request; and synchronize data of the requested object having a synchronized state with the CSS.

WO 2016/033056 A1

WO 2016/033056 A1

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))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

A METHOD AND COMPUTING DEVICE FOR ALLOWING SYNCHRONIZED ACCESS TO CLOUD STORAGE SYSTEMS BASED ON STUB TRACKING

TECHNICAL FIELD

[001] The present invention relates generally to file sharing and synchronization in cloud storage networks and, more particularly, to multilevel systems for file sharing and synchronization.

BACKGROUND

[002] The demand for data storage has been rapidly escalating because, as the amount of data such as digital media stored by users grows, so does their need to store digital media reliably over extended periods of time.

[003] Storage systems to store digital media range from a local storage media (e.g., CDs and backup tapes) and network storage systems (such as NAS or NAT) to cloud-based storage system.

[004] Network storage systems such as NAS and NAT provide access to files to users connected in a local area network though standard file sharing protocols (e.g., common internet file system (CIFS) or network file system (NFS)).

[005] Cloud-based storage systems, also referred to as cloud storage services (CSS), provide mass storage through a web service interface available through the Internet. The storage infrastructure includes a distributed array of geographically distributed data centers connected to a plurality of clients through a wide area network (WAN).

[006] Fig. 1 illustrates a storage system 100 designed to provide cloud storage services. The system 100 includes a distributed array of geographically distributed data centers 110-1 to 110-M (hereinafter referred to collectively as data centers 110 or individually as a data center 110, merely for simplicity purposes) connected to a plurality of clients 120-1 to 120-N (hereinafter referred to collectively as clients 120 or individually as a client 120, merely for simplicity purposes) through a wide area network (WAN) 130.

Date Reçue/Date Received 2023-06-07

[007] A data center 110 typically includes servers and mass-storage-facilitating cloud storage services to the clients 120. Such services enable applications including, for example, backup and restore of data, data migration, data sharing, data collaboration, and so on. Cloud storage services are accessible from anywhere in the world. To this end, each client 120 implements a web services interface designed to at least synchronize data with the data centers 110. Applications enabled by the cloud storage services are not typically aware of the specifics of the services and the underlying data synchronization operations. The disadvantage of commercially available cloud storage services is that such services do not implement standard file sharing protocols (e.g., common internet file system (CIFS) or network file system (NFS)). Furthermore, accessing files stored in the cloud storage is typically slower than accessing files stored in local storage devices.

[008] Although not shown in Fig. 1, the storage system 100 may include a plurality of cache servers to accelerate data storage and retrieval as well as cloud agents allowing access to files remotely stored in the data centers 110. A cloud agent may be a hardware component, a software component, or a combination thereof, which is connected to or associated with a specific workstation, server, or other computing device. For example, a workstation agent may be software installed on a personal computer, such as to integrate this workstation with the CSS and/or cloud integrated storage devices. As another example, a mobile device agent may be an application installed on a mobile device, such as a smartphone, acting to integrate the mobile device with the cloud storage system.

[009] The cloud storage system can be utilized to share content between users. For example, in enterprises, data can often be shared between different departments, branches, and individual users. Each such entity that can save or share files is typically assigned, e.g., different permission rules. Furthermore, each user may use a different type of device (node), each of which may be, but is not limited to, a PC, a smartphone, a storage appliance, a file server, and so on. Thus, a folder stored in the cloud storage (a data center 110) can be accessed by multiple different users from different geographical locations. In addition, a user can access the cloud storage from different locations and/or different devices associated with the user.

[0010] An essential requirement of a cloud storage system is to synchronize data between local devices and remote storage, between different devices of the same user, and among users that share the same content. Another essential requirement is to provide sufficient data throughout for storage and retrieval of data from any device and/or geographical location accessing the system.

SUMMARY

[0011] A summary of several example embodiments of the disclosure follows. This summary is provided for the convenience of the reader to provide a basic understanding of such embodiments and does not wholly define the breadth of the disclosure. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither identify key or critical elements of all embodiments nor to delineate the scope of any or all aspects. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later. For convenience, the term "some embodiments" may be used herein to refer to a single embodiment or multiple embodiments of the disclosure.

[0012] The disclosed embodiments include a computing device for allowing synchronized access to a cloud storage system (CSS). The computing device comprises: a processing unit; a storage system, wherein the storage system is configured to maintain objects, each object having a state, wherein each state is any one of: a synchronized state, and a stub state; and a memory, the memory containing instructions that, when executed by the processing unit, configure the computing device to: receive a request from a client to access an object maintained in the storage system; change the state of the requested object from a stub state to a synchronized state upon receiving the request; and synchronize data of the requested object having a synchronized state with the CSS.

[0013] The disclosed embodiments also include a method for allowing synchronized access to a cloud storage system (CSS). The method comprises: receiving a request from a client to access an object maintained in a storage system, wherein the storage system is configured to maintain objects, each object having a state, wherein each state is any one of: a synchronized state, and a stub state; changing the state of the requested

object from a stub state to a synchronized state upon receiving the request; and synchronizing data of the requested object having a synchronized state with the CSS.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] The subject matter disclosed herein is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the invention will be apparent from the following detailed description taken in conjunction with the accompanying drawings.

[0015] Figure 1 is a diagram of a storage network system.

[0016] Figure 2 is a schematic diagram illustrating a multilevel file system configured according to an embodiment.

[0017] Figure 3 is a schematic diagram illustrating a multilevel cloud storage file system according to an embodiment.

[0018] Figure 4 is a flowchart illustrating a multilevel cloud storage network including more than one cloud storage systems according to an embodiment.

[0019] Figure 5 an exemplary and non-limiting flowchart illustrating stub file tracking and synchronization according to an embodiment.

[0020] Figure 6 is an exemplary and non-limiting flowchart illustrating synchronization of regions of a file according to an embodiment.

[0021] Figure 7 illustrating the synchronization of metadata on-demand according to an embodiment.

[0022] Figure 8 is an exemplary and non-limiting block diagram of a centralized data flow management system operative in file collaboration networks according to an embodiment.

[0023] Figure 9 is a diagram of file collaboration network having a tree topology.

[0024] Figure 10 is a diagram illustrating a file collaboration network having a forest topology.

[0025] Figure 11 is a diagram illustrating a collaboration network having a graph-like topology.

[0026] Figure 12 is a flowchart illustrating a method for establishing data flows using templates in a file collaboration network.

[0027] Figure 13 is a screenshot of a user interface for a template editor according to an embodiment.

[0028] Figure 14 is a screenshot of a user interface allowing control over settings for a sync relation data flow according to an embodiment.

[0029] Figure 15 is a screenshot illustrating a user interface utilized for defining a template assignment protocol according to an embodiment.

[0030] Figure 16 is a schematic diagram of a system where the disclosed embodiments can be applicable.

[0031] Figure 17 is an exemplary and non-limiting flowchart illustrating a method for creation and utilization of data flows using data flow templates according to an embodiment.

DETAILED DESCRIPTION

[0032] It is important to note that the embodiments disclosed herein are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claimed inventions. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality. In the drawings, like numerals refer to like parts through several views.

[0033] Fig. 2 shows an exemplary and non-limiting schematic diagram of a multilevel file sharing and storage network 200 according to an embodiment. In the embodiment, a cloud storage system (CSS) 210 is communicatively connected to one or more cloud storage gateways (CSGs) 220-1 and 220-2 (hereinafter referred to individually as a CSG 220 and collectively as CSGs 220, merely for simplicity purposes).

[0034] A CSG 220 may be a hardware component, a software component, or a combination thereof. The CSG 220 may provide caching, WAN acceleration, storage acceleration and the like to an endpoint 230. The endpoint 230 may be, but is not limited to, a client device executing software agents that can remotely access files stored in the CSS 210 or in a cloud-enabled storage device, for example using the method described in the co-pending patent application 13/205,238, assigned to common assignee.

Date Reçue/Date Received 2023-06-07

In further embodiments, an agent may be installed on an endpoint 230. The agent permits the endpoint to access and synchronize data with the CSS 210 and/or one of the CSGs 220. For example, a workstation agent may be software installed on a personal computer that permits integration of the workstation with the CSS 210. As another example, a mobile device agent may be an application installed on an endpoint 230, such as a smartphone, acting to integrate the endpoint 230 with the CSS 210, another client, and/or a CSG 220.

[0035] Each of the endpoint 230 and the CSGs 220 may be a computing device that includes, but is not limited to, a storage, a processing unit, and a memory. The memory contains instructions that, when executed by the processing unit, configure the computing device to perform the methods described herein.

[0036] Some CSGs 220 may reside in close proximity to each other with respect to geographical locations, network topology access delay, bandwidth criteria, cost criteria, administrative configuration, and so on. In one embodiment, a plurality of CSGs 220 may be part of a trusted network, thereby allowing an endpoint 230 to access each CSG 220 in a trusted network. In another embodiment, one or more CSGs, e.g., the CSG 220-2, can provide shared, large capacity block cache services to increase the probability that block reads can be satisfied locally from the regional cache and without requiring access to the CSS 210. This shared block capacity is advantageous because the CSS 210 is often located at a distant datacenter and such a technique may reduce latency, reduce load on the WAN link, and increase the overall responsiveness and throughput of the cloud storage service. It should be noted that, in certain embodiments, the CSGs 220 can be configured with a local cache (e.g., a layer1-cache) as well.

[0037] The endpoint 230 is connected to the CSS 210 either through a direct connection or through one or more of the CSGs 220. It should be noted that only one endpoint 230 is shown in Fig. 2 merely for the sake of simplicity and without limitation on the various disclosed embodiments. If the endpoint 230 is connected to the CSS 210 both through a direct connection and through one of more of the CSGs 220, upon an attempt to retrieve data from the CSS 210 or to store data in the CSS 210, the endpoint 230 may be prompted to provide a selection of a preferred route, wherein the preferred route is either a direct connection or a connection through the CSGs. If a user of the endpoint 230 selects the

Date Reçue/Date Received 2023-06-07

connection through the CSGs 220, a best route for data flow through the CSGs 220 may be determined.

[0038] The CSGs 220 may provide data synchronization services for data flows in the incoming direction (clients to CSS), outgoing direction (CSS to clients), or both. A data flow may be, but is not limited to, file synchronization ("sync relation) between the CSS 210 and a computing device, caching ("cached access relation") of files in the CSS 210, backup ("backup relation") of files to the CSS 210, and remote access ("remote access relation") to files in the CSS 210 or in a computing device. For example, a data flow may be synchronizing a file obtained from a CSS in an endpoint. In an embodiment, data flows can be set through data flow templates. The data flow templates are typically maintained in a centralized system as described further herein below with respect to Fig. 8.

[0039] In sync relation data flows, selected contents, e.g., folders or files are synchronized, either in one direction across a data flow link, or across both directions of the data flow link. Sync relation data flows are discussed further in co-pending US Patent Application No. 2010/0161759, entitled "STORAGE DEVICE AND METHOD THEREOF FOR INTEGRATING NETWORK ATTACHED STORAGE WITH CLOUD STORAGE SERVICES" (hereinafter, the '759 Application), assigned to common assignee.

[0040] In backup access relation data flows, the content of the file or folder on one entity is continuously or periodically copied on the other file or folder. In such flows, updates are unidirectional. In remote access relation data flows (also known as direct access data flows), read data is retrieved directly on demand from a peer node (as a non-limiting example, a peer node may be an endpoint within the same level of a hierarchical file collaboration network). Written data may then be stored directly to the peer node.

[0041] In cached access data flows, read data is retrieved directly from a peer node and cached locally. Written data is stored to the peer node. Such storage may be performed asynchronously, and may be performed with an optimized delta-encoding protocol or WAN acceleration. Cached access data flows are discussed further in the above-referenced '759 Application.

Date Reçue/Date Received 2023-06-07

[0042] In addition to the aforementioned relation types, relations may include various configurable attributes. Such attributes may include, but are not limited to, the content that will be synchronized or cached as part of the relation, the content to be excluded from synchronization or caching (for example, temporary files may not be synchronized), Quality of Service (QoS) settings that may be either permanent or based on specific schedules (e.g., bandwidth limits, Diffserv tags, and so on), Digital Rights Management (DRM) instructions relating to content obtained over the relation that may be utilized to control use of such content after the content has been received (e.g., controlling the user's ability to view, copy, print, and/or share the received contents), and locking policies (e.g., manually or automatically locking files or folders after access). Support for locking policies may be desirable, particularly in massive shared folders, where automatic locking may be advantageous.

[0043] In various embodiments, a plurality of relations of data flows may be defined for a single file or folder between two entities. In some embodiments, such relations may be defined by users with administrative privileges and/or by users of individual entities.

[0044] In an embodiment, the relation for a given file or folder may be changed dynamically. When a relation is changed, different actions may be taken depending on which type the relation began or ended as. As a non-limiting example, when a relation changes from a cached relation to a sync relation or to a backup relation, the entity that cached the data may utilize the cached copy for a backed up or synchronized copy of the file or folder and may only request the portions of the file or folder that have not been cached from the matching entity. As another non-limiting example, if a relation changed from sync access or backup access to cached access, the entity may only need to change the labeling for the synchronized or backed up copy to mark it as cached. In this case, the data may be discarded based on cache management policy.

[0045] In one example, the endpoint 230 or a CSG 220 may be configured to regularly synchronize files that are frequently accessed locally so that the latest version of such files can be accessed immediately at local speed, while less frequently accessed files may not be stored locally and, instead, may be fetched from the cloud upon read or write request (such direct read and write requests may still be accelerated by the block level cache). As usage patterns change, specific files or folders may be transitioned

automatically from cached to direct access or vice versa. Information regarding which files are stubs is persistently tracked, and such stub files are synchronized. Files that are synchronized are kept up to date using the cloud file synchronization protocol. Stub tracking and synchronization are described in further detail herein below with respect to Fig. 5.

[0046] To support very large files, the synchronization may be on a sub-file granularity such that, for example, frequently accessed regions of the file may be regularly synchronized, while other regions of the file are fetched only on demand. The regions of each file to be synchronized may be persistently tracked.

[0047] In a preferred embodiment, the synced and cached files are all exposed by a virtual file system driver (running either in user level or in kernel level) as a unified virtual file system, thereby providing the user seamless read/write access to all files, whether they are available locally (synced) or are cached. In an embodiment, the files in the virtual file system are arranged in a multi-level namespace. An exemplary implementation of the multi-level namespace is disclosed in US Patent Application No. 14/709,902 assigned to the common assignee.

[0048] In an embodiment, the metadata of all files are synchronized to the endpoint 230 or to the CSG 220. The system stores the metadata locally in the endpoint 230 so that directory listing requests can be satisfied quickly and even without internet connectivity. In one possible implementation, files are stored as stub files in the local filesystem, containing only the synchronized regions of the file (if there are any). In this implementation, when a directory listing request is received from a user, the listing is provided directly by the underlying filesystem of the endpoint 230 by simply passing through the requests. When the endpoint 230 is not connected to the CSS, the endpoint 230 may still provide full access to browse the entire global directory structure. However, attempting to access stub files which are not synchronized locally will result in the client receiving an error result. Synchronization of portions of files is described further herein below with respect to Fig. 6.

[0049] In another embodiment, the entire metadata is not synchronized to every endpoint. Instead, each endpoint synchronizes only selective portions of the cloud storage

Date Reçue/Date Received 2023-06-07

directory structure. This may be dynamic based on usage patterns, wherein folders which are frequently accessed are synchronized automatically, while folders which were not accessed recently are synchronized only on demand. At times when the endpoint is not connected to the CSS due to connectivity problems, it may still provide access to browse the directory structure. Attempting to list the contents of folders which were not synchronized locally will result in the client receiving an error result or an empty listing. Selective synchronization of portions of cloud storage directory structure is described further herein below with respect to Fig. 7.

[0050] In another embodiment, cached relations may be generated automatically. In such an embodiment, an element may establish a cached relation for a specific file or folder if such a relation will assist with overall system performance by, e.g., reducing access latency and/or internet traffic. Such automatically generated relations may not require any "handshake" between the two entities, and may be established and/or cancelled by an entity. A "handshake" between two entities occurs when one entity indicates to another entity that it desires to engage in communication.

[0051] According to various embodiments, when the endpoint 230 attempts to store or obtain data in the CSS 210, it may do so through any of the CSGs 220 and/or through a direct connection to the CSS 210. A route is a path from the endpoint 230 to the CSS 210. A route may be a direct connection from an endpoint to a CSS. A route may also include a connection to the CSS 210 through one or more CSGs 220. In such embodiments, determination of which route to use is typically based on which CSG 220 provides the best connection. The best connection may be considered based on one or more of the following criteria: bandwidth, response time, price, data flow type, and the like. Determination of best connection is discussed further herein below with respect to Fig. 4.

[0052] As a non-limiting example according to the embodiment shown in Fig. 2, a user operating a mobile device (i.e., an endpoint 230) attempts to save a file in the CSS 210. Upon analysis of the available routes, it is determined that CSG 220-1 and CSG 220-2 are demonstrating slow response times and, therefore, that the best route for saving the data is via a direct connection between the endpoint 230 and CSS 210. Thus, in this example, the endpoint may save data to the CSS 210 through a direct route.

[0053] As another non-limiting example according to the embodiment shown in Fig. 2, a user operating a workstation (i.e., an endpoint 230) attempts to retrieve a file from the CSS 210. Upon analysis of the available routes, the direct connection between the endpoint 230 and the CSS 210 is determined to be unavailable. Additionally, CSG 220-1 has higher available bandwidth and/or lower latency than CSG 220-2. Consequently, the best route is determined to be the flow of data that passes through CSG 220-1. As a result, the user retrieves the data from the CSS 210 using the endpoint 230 via the CSG 220-1.

[0054] Fig. 3 shows an exemplary and non-limiting schematic diagram 300 of a multilevel file sharing and storage network featuring multiple cloud storage systems according to an embodiment. In the embodiment, cloud storage systems (CSSs) 310-1 and 310-2 are communicatively connected to one or more endpoints 330-1 through 330-3 (hereinafter referred to individually as a CSS 330 and collectively as CSSs 330, merely for simplicity purposes) and one or more cloud storage gateways (CSGs) 320. The endpoints 330 may be, but are not limited to, client devices executing software agents that can remotely access files stored in the CSS 310 or in a cloud-enabled storage device, for example using the method described in the co-pending patent application 13/205,238, assigned to common assignee. The endpoints 330 are typically installed with an agent as defined above. In various embodiments, more than two CSSs 310 may be implemented. Each endpoint 330 and CSG 320 may be, but is not limited to, a computing device including a processor and a memory, the memory containing instructions that, when executed by the processor, configure the computing device to perform the methods described herein.

[0055] In an embodiment, upon detecting an attempt to store or retrieve data in a CSS 310, the endpoint 330 is configured to select a preferred route through which to deliver the data. The preferred route is a path for data flow between the endpoint 330 and the CSS 310, and may be either direct or synchronization-based (indirect). A direct route is a direct connection between the CSS 310 and the endpoint 330. A synchronization-based route is an indirect connection wherein data passes through another CSS 310 and/or at least one CSG 320 during data flow. The endpoint 330 may select a preferred route

Date Reçue/Date Received 2023-06-07

based on, but not limited to, a user selection, predefined user preferences, and so on. If there is no direct route, the synchronization-based route may be determined to be the preferred route.

[0056] The CSGs 320 may provide caching, wide area network (WAN) acceleration, and/or storage acceleration to the endpoints 330. Each CSG 320 is further connected to one or more of the endpoints 330. In an embodiment, some or all of the endpoints 330 may be directly connected to a CSS (e.g., the CSS 310-1 or the CSS 310-2).

[0057] As a non-limiting example according to the embodiment illustrated in Fig. 3, a client attempts to save data to the CSS 310-2 via the endpoint 330-1. In this example, the endpoint 330-1 does not have any direct connection with the CSS 310-2. Thus, any data flow between the endpoint 330-1 and the CSS 310-2 must pass through the CSS 310-1. Thus, a synchronization-based route is determined to be the preferred route. The best route is determined to be the route that includes data flow from the endpoint 330-1 to CSS 310-1 directly rather than the route that includes data flow from the endpoint 330-1 to a CSG 320. Thus, in this example, the user may save data using the endpoint 330-1 to CSS 310-2 via CSS 310-1.

[0058] Fig. 4 shows an exemplary and non-limiting flowchart 400 illustrating a method for storing data in a multilevel file sharing network according to an embodiment. In S410, an initial connection is established through a default route. The initial connection is typically established to one or more default connection entities, such as one or more CSGs and/or agents. Alternatively, the initial connection may be a direct route as discussed above. The default route may be set through factory default settings or manual entry by a user. Each default route may be configured with a list of allowed entities through which to connect. Allowed entities are entities that the default connection entity may establish connections to.

[0059] As a non-limiting example, a client may allow a user to enter a URL address of the CSG for its default route. When an initial connection is established, the endpoint is connected to the default CSG which provides the client with a list of other entities that the endpoint may connect to through the URL. These other entities may include, but are not limited to, other CSGs and/or CSSs. In an embodiment, if an endpoint loses connection to its default connection point, the endpoint may be connected to an

alternative available route. In a further embodiment, if no connection points are available, the endpoint may be directly connected to a CSS.

[0060] In S420, the best route for a connection is selected. A route typically includes a link, which may be direct or indirect. A direct link is a direct connection between an endpoint and a CSS. A route may also include, but is not limited to, a local connection point, a CSS, a CSG included in another CSS, an agent, and so on. The selection may be global at a given time such that all interactions related to every maintained relation are performed with another single entity. Alternatively, the selection may be performed for each relation separately such that a different best route is selected for different files and/or folders. Such selection may be based on the data flows defined for the different files and/or folders. In an embodiment, multiple alternative routes for a single relation may be held. In another embodiment, multiple CSSs may operate together to provide entities and clients with optimized service. In such an embodiment, one CSS may cache data for other CSSs and/or serve as a gateway for an endpoint and CSG that are proximally located (e.g., physically or topologically close).

[0061] Determination of which route to select as the best route for connection may be based on various criteria including, but not limited to, the relative topological distance (e.g., whether or not the elements are located on the same LAN, the number of routers between two entities, and so on), the type of relation maintained for the specific file or folder with the specific entity, the availability of access to the specified element, the latency in accessing each of the elements, inclusion in or exclusion from a list of allowed elements, and a DNS query resolution.

[0062] In an embodiment, if the connection is lost before the request has been completed, a new connection may be established with other available entities. In a further embodiment, selection of the best route for the new connection utilizes the same criteria described herein above with respect to S430.

[0063] In S440, it is determined whether the entity on the other end of the selected best route has sufficient storage capacity to fulfill the request to store data. If so, execution continues with S460; otherwise, execution continues with S450. In S450, remote access is established to one or more connection points of another cloud storage system. In

S460, data is stored according to the selected best route and/or in any remotely accessed elements.

[0064] In an embodiment, additional CSGs may be included as part of a different cloud storage system. In such embodiments, the CSS or CSG may serve as a proxy for the requested relation. In various embodiments, the requesting endpoint or CSG may not be aware of the proxying and may be provided with service that is of similar quality to local service. In a further embodiment, the proxy may be canceled once a local or other proximate storage space is available.

[0065] In another embodiment, CSSs and CSGs may use other CSSs and connection points as cache services. This may occur, e.g., when the other CSS or connection point is proximately closer (physically or topologically), or when the other CSS or connection point demonstrates less latency. In such an embodiment, a CSS or CSG that needs to gain access to a specific file or folder may obtain access via a cache of one of the other CSSs or connections rather than by accessing the other CSS through the selected best connection.

[0066] Fig. 5 is an exemplary and non-limiting flowchart 500 illustrating stub file tracking and synchronization according to an embodiment. In S510, a request to read a byte range from a file is received. In an embodiment, such a request may come from an endpoint (e.g., an endpoint 330). In S520, it is checked whether the requested file is a stub file. If so, execution continues with S530; otherwise, execution continues with S550. In S530, file synchronization of the stub file begins. In S540, execution ceases until the requested byte range becomes available for delivery. In S550, the requested byte range is returned.

[0067] Fig. 6 is an exemplary and non-limiting flowchart 600 illustrating synchronization of regions of a file according to an embodiment. In S610, a request to read a byte range from a file is received. In S620, it is checked whether the requested byte range is available in a stub file. If so, execution continues with S630; otherwise, execution terminates. In an embodiment, if the requested byte range is not available in a stub file, the entire file may be synchronized. In S630, synchronization of the region in the file including the byte range begins. In S640, execution ceases until the requested byte range becomes available. In S650, the requested byte range is returned.

[0068] Fig. 7 is an exemplary and non-limiting flowchart 700 illustrating the synchronization of metadata on-demand according to an embodiment. In S710, it is determined whether directory metadata is synchronized. If so, execution continues with S720; otherwise, execution terminates. In S720, directory metadata is synchronized. In S730, the directory listing from the local file system is returned.

[0069] Fig. 8 shows an exemplary and non-limiting block diagram of a centralized data flow management system 800 operative in file collaboration networks according to an embodiment. The system 800 includes an endpoint database 810 configured to maintain metadata of each endpoint (endpoints not shown) connected to the network (network not shown), a data flow templates database 820, and a template assignment unit 830. The database 820 is configured to maintain data flow templates 825 to be associated with endpoints. The management system 800 also includes a user override policer 840 and an endpoint connector 850.

[0070] In an embodiment, the template assignment unit 830 is configured to evaluate criteria related to each endpoint to determine which data flow template or templates to be applied to each endpoint. In a further embodiment, the evaluated criteria may be, but is not limited to, an endpoint's metadata, templates predefined by a user of the endpoint, templates defined by an administrator, security settings, availability of appliances or cache servers in the network, and so on. A metadata of an endpoint may include, for example, a type of endpoint (workstation, server, appliance, mobile device, etc.), an operating system type and version, an owner group, an endpoint host name, and so on.

[0071] In an embodiment, a data flow template can be defined by the user (or by a system administrator) and automatically assigned to multiple endpoints. Defining template assignment policies is discussed further herein below with respect to Fig. 8. In an embodiment, templates may be manually assigned to some or all of the endpoints, regardless of criteria.

[0072] In one embodiment, the data flow templates may be prioritized according to the order the templates should be assigned to the endpoints. Such data flows may embed smart variables. In an embodiment, templates may further contain information related to configuration of additional device settings such as, for example, backup settings,

software updates, and so on. Smart variables are variables that are defined in a template but are evaluated by the endpoint according to its local environment such that they may have different values when interacting with different endpoints.

[0073] Endpoints perform substitution of smart variables with their locally applicable values to achieve configuration of the endpoints according to a respective data flow template. If the endpoints are arranged in a hierarchy, the locally applicable values may depend on information relating to the local endpoint and/or to other endpoints in the hierarchy. Additionally, smart variables may be globally defined such that all endpoint connectors can evaluate all smart variables, or may be defined as limited in scope such that only certain endpoint connectors are capable of evaluating any or all of the smart variables.

[0074] As a non-limiting example of smart variable substitution, a smart variable named "{MYDOCUMENTS}" may be substituted by the correct file extension of the device owner's "Documents" folder on his or her endpoint device. The correct value may then be used regardless of the user's operating system or configuration.

[0075] The template assignment unit 830 is configured to provide the selected templates and their designated endpoint connections to the endpoint connector 850. The endpoint connector 850 applies selected templates and a user override policy of the user override policer 840 to each endpoint via endpoint connections 815. In an embodiment, the template assignment unit 830 may select and apply templates to each endpoint directly rather than through the endpoint connector 850. The user override policy 840 is used by the endpoint to determine whether and how the owner of the endpoint can override the selected template. If the user is permitted to override a selected template, specific data flows may be added or altered locally by that owner. In an embodiment, the management system 800 is communicatively connected to an administrator console 860, which allows to users to override or define errant templates, and/or to control template selection.

[0076] Fig. 9 shows an exemplary diagram of a file collaboration network 900 having a tree topology. In the network 900, a single cloud storage system (CSS) 910 is communicatively connected to one or more endpoints 930 via connections 920. In this network, the endpoints 930 are arranged hierarchically. Consequently, some endpoints may interact with the CSS 910 only through connections to other endpoints. In such tree

collaboration networks, data flows are restricted to one data flow path. For example, if an endpoint 930 is connected to the CSS 910 only through another endpoint 930, data being sent to or retrieved from the CSS 910 must flow through the other endpoint 930. In contrast, file collaboration networks having forest and graph-like topologies, discussed further herein below with respect to Figs. 10 and 11, respectively, may have more than one potential data flow path available to any given endpoint.

[0077] Fig. 10 shows a diagram of a network 1000 illustrating a file collaboration network having a forest topology. In the network 1000, cloud storage systems (CSS) 1010-1 and 1010-2 are communicatively connected to various endpoints 1030 via connections 1020. Additionally, in this forest type collaboration network topology, the CSS 1010-1 and the CSS 1010-2 are communicatively connected via an interlink 1040. The interlink 1040 may be a bidirectional trust relationship that permits communication of data both to and from each CSS 1010. Alternatively, the interlink 1040 may be one way such that only one CSS 1010 communicates with the other.

[0078] The link between CSS 1010-1 and CSS 1010-2 permits each cloud to act as a cloud storage router for the other cloud. As a result, the endpoints 1030 connected to one cloud storage device 1010 may establish data flows to the other linked CSS 1010, either directly (not shown) or indirectly through connections 1020, other endpoints 1030, and/or through a CSS 1010.

[0079] Data flows may also be applied to links between portals such as through, for example, caching, selective synching, or direct access. Further, each CSS 1010 may enforce controls over which content can be accessed by users of interlinked cloud storage services. Each endpoint is configured with the correct data flow by means of a respective data flow template selected by a management system (e.g., the management system 800 (Fig. 8)).

[0080] Fig. 11 shows an exemplary diagram illustrating a collaboration network 1100 having a graph-like topology according to an embodiment. In the network 1100, like in the forest network topology, each CSS 1110 may be connected to multiple endpoints 1130 either directly or indirectly via connections 1120. In this embodiment, the endpoints 1130 may further be interconnected via connections 1120 such that any or all of endpoints 1130 may possess more than one direct and/or indirect connection to its

respective CSS 1110. As an example, as shown, various endpoints may interact with CSS 1110-1, both directly through a connection 1120 between the endpoint 1130 and the CSS 1110-1, and indirectly through a data flow 1120 between two endpoints 1130.

[0081] In an embodiment, this interaction may be used to permit dynamic flow selection so as to select the optimal data flow for the transfer of information depending on flow efficiency. Flow efficiency may be determined based on factors such as, e.g., connection 1120 availability and latency. Selecting optimal flows will typically lead to optimized performance and reduced energy consumption. The dynamic selection of data flows is performed through setting of data flow templates

[0082] Fig. 12 shows a non-limiting and exemplary flowchart 1200 illustrating a method for establishing data flows using templates in a file collaboration network. In an embodiment, the method is performed by a data flow management system, e.g., system 800.

[0083] At S1210, an indication that an endpoint is initialized is received. The endpoint is initialized when, for example, a new setting, software, firmware, and/or hardware is installed. Such initialization would likely modify the metadata associated with the endpoint. At S1220, the endpoint's metadata are retrieved from the endpoint, analyzed, and compared against selection criteria for templates to select pertinent templates for each analyzed endpoint. Any changes in the metadata or metadata of new endpoints are saved in an endpoint database. In an embodiment, this analysis is performed by a template assignment unit (e.g., template assignment unit 130) as discussed further herein above with respect to Fig. 1. As noted above, metadata of an endpoint may be, but is not limited to, endpoint type, operating system type and version, owner group, endpoint host name, and so on.

[0084] At S1230, it is determined whether a user (e.g., a system administrator) may override the template and manually create data flows. This determination may be based on a user override policy enforced by a user override policer (e.g., user override policer 840). At S1240, responsive to the metadata analysis or the override, an appropriate data flow template is selected and sent to the endpoint. Subsequent utilization of templates and creation of data flow is discussed further herein below with respect to Fig. 17.

[0085] As a non-limiting example, a first endpoint attempting to save data to a CSS may belong to a collaboration network topology that is a forest type hierarchical network topology. In such an example, two or more cloud storage services would be communicatively connected, thereby enabling access to any linked cloud storage service either directly or indirectly through any endpoint device in the network, subject to any restrictions on access imposed by the management system. Such topology may further include connections among endpoint devices. In an exemplary embodiment, a template may be selected by a management system and provided to an endpoint.

[0086] The first endpoint (e.g., a personal computer running a Windows® 7 operating system) is connected to a CSS indirectly through a connection to a second endpoint (e.g., a cloud storage gateway), wherein the second endpoint is connected to the CSS through a connection (e.g., an Internet connection). In this example, the first endpoint is also connected to a CSS indirectly through a connection to a third endpoint (e.g., another cloud storage gateway), wherein the third endpoint is connected directly to the CSS through a connection (e.g., an Internet connection). The metadata of the first endpoint is analyzed to determine which template(s) to select. In this example, a particular template is associated with Windows® operating systems. Thus, since this endpoint utilizes the Windows® 7 operating system, that template is selected.

[0087] Further, in this example, the condition set in the user override policy to determine whether a user may override the selected template is that the user must be an administrator. It is determined that the user is not an administrator, so the results of this determination, along with the user override policy and selected template, are provided to the endpoint.

[0088] Fig. 13 shows an exemplary screenshot 1300 of a user interface for a template editor according to an embodiment. In this embodiment, the user interface includes a list of folders 1310 that are synchronized with an endpoint device. The user may click on add button 1320 to add folders to the list of synchronized content.

[0089] Fig. 14 shows an exemplary screenshot 1400 of a user interface allowing control over settings for a sync relation data flow according to an embodiment. In the embodiment, a user may select one of two options 1410: synchronizing the local folder with a Cloud Drive folder, or synchronizing the local folder with another local folder. If

the user elects to sync the folder to another local folder, he or she would then provide the file extension of that other folder in file extension box 1420. In various embodiments, the user may elect to exclude one or more sub-folders from the synchronization by noting such folders in exclusion box 1430.

[0090] Fig. 15 shows an exemplary screenshot 1500 illustrating a user interface utilized for defining a template assignment protocol according to an embodiment. In the embodiment, various conditions that determine whether a template will be appropriately applied to an endpoint are established. Condition management controls 1510 permit a user to add a new condition or select an existing condition to modify. Each condition's relevance is defined by selecting one or more parameter types and defining conditions for relevance. In the embodiment shown in Fig. 15, two parameter types 1520 have been selected: the owner of the endpoint device, and the operating system of the endpoint device. Each parameter type 1520 is associated with a conditional statement 1530. The conditional statement defines the required association between the endpoint device and a comparison parameter 1540. If the conditional statement 1530 is true, then the template noted in template list 1550 is selected for that endpoint. In this embodiment, multiple conditional statements are applied to determine one or more relevant templates. In further embodiments, if multiple templates are selected, either all selected templates may be applied, or a determination may be made based on, e.g., predefined conditions. In various embodiments, if all conditional statements are false, then a default template may be used via, e.g., default check box 1560.

[0091] As a non-limiting example, in this embodiment, the condition "equals" as applied to the parameter type "owner" means that the owner name must match the comparison parameter "example" for the template "beta" to be selected. Similarly, according to the second condition, if the operating system name begins with "Windows®," template "TEMP-WIN7" would be selected. Thus, an endpoint device having Windows 7® would qualify, and the template "eyal" would be selected for that endpoint device. Alternatively, if a user did not have the name "example" and was operating an end user device using, e.g., the Mac OS X operating system, the default template "beta" would be selected according to this embodiment because the user does not qualify for any of the conditional statements associated with the templates.

[0092] Fig. 16 shows an exemplary and non-limiting schematic diagram of a system 1600 in which the disclosed embodiments can be applicable. The system 1600 includes a plurality of endpoints 1610-1 through 1610-N and one or more network attached storage devices 1620 connected to a local area network (LAN) 1630, which may be either a wireless or wired network. The device 1620 is connected to a cloud storage service (CSS) 1640 through the network 1650. The CSS 1640 includes a plurality of cloud connectors 1641 that facilitate connections between the device 1620 and storage devices 1642 (e.g., disk drives and/or SAN/NAS devices), local object storage systems (not shown) such as EMC Atmos or OpenStack, as well as connections to public cloud storage providers 1643 (e.g., Amazon Web Services, IBM SmartCloud, etc.).

[0093] The device 1620 is a cloud-enabled storage device that is integrated with cloud storage services. An example of such a device is disclosed in the co-pending patent application 12/641,559, which is assigned to common assignee. The device 1620 is also an endpoint of the CSS 1640.

[0094] The endpoints 1610 may be endpoints executing software agents that can remotely access files stored in the CSS 1640 or in a cloud-enabled storage device 1620, for example using the method described in the co-pending patent application 13/205,238, assigned to common assignee. A folder in the cloud storage can be accessed by many different users of clients.

[0095] In an embodiment, a centralized data flow management system 1644 is incorporated into the CSS 1640. In another embodiment, the centralized data flow management system 1644 is integrated in one more of the cloud connectors 1641.

[0096] Fig. 17 is an exemplary and non-limiting flowchart 1700 illustrating a method for creation and utilization of data flows using data flow templates according to an embodiment. At S1710, a data flow template is received from a management system (e.g., the management system 1644). At S1720, it is checked whether the user has satisfied the criteria for overriding the received template. If so, execution continues with S1730; otherwise, execution continues with S1740. In an embodiment, this check may be made based on a determination made by the management system.

Date Reçue/Date Received 2023-06-07

[0097] At S1730, upon determination that a user may override the selected template, a user is prompted for manual data flow entry. In an embodiment, a user who is permitted to override the template based on the user override policy may nevertheless elect to use the received template. In another embodiment, the user may view the template before deciding whether to manually enter a data flow.

[0098] At S1740, upon application of the template(s) or manual data flow entry, a data flow is created via connections between endpoints as defined in each applicable template or entry among all relevant endpoint devices. At S1750, any smart variables associated with the created data flow are substituted with locally applicable values that are relevant to each endpoint device, as appropriate. At S1760, the requested data is saved or retrieved via the established data flow.

[0099] As a non-limiting example, a template respective of the first endpoint's metadata may be provided along with a user override policy and a request to save the data to a CSS. Upon determining that the user may not override the provided template, a data flow is created according to the template. Any smart variables are replaced with relevant local variables, and the endpoint saves the data in the CSS via the established data flows.

[00100] The various embodiments disclosed herein can be implemented as hardware, firmware, software, or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or computer readable medium consisting of parts, or of certain devices and/or a combination of devices. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units ("CPUs"), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such a computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform such as an additional

data storage unit and a printing unit. Furthermore, a non-transitory computer readable medium is any computer readable medium except for a transitory propagating signal.

[00101] All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the invention and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and embodiments of the invention, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

CLAIMS

What is claimed is:

1. A computing device for allowing synchronized access to a cloud storage system (CSS), comprising:

a processing unit;

a storage system, wherein the storage system is configured to maintain objects, wherein each object is a file, each object having a content, a corresponding object stored in the CSS, and a state, wherein each state is any one of: a synchronized state, in which an object in the computing device has its content at least automatically updated in accordance with changes to content of its corresponding object in the CSS and changes to the content of the object in the computing device are automatically propagated to the content of the corresponding object in the CSS so as to continuously maintain the content of the object in the computing device in synchronization with the content of the corresponding object in the CSS so long as the object is in the synchronized state, and a stub state, in which an object's content is not automatically updated to be maintained in synchronization with the content of the corresponding object in the object in the CSS; and

a memory, the memory containing instructions that, when executed by the processing unit, configure the computing device to:

receive a request from a client to access data in a byte range within at least one object maintained in the storage system in the stub state;

change the state of the requested at least one object from the stub state to a synchronized state upon receiving the request;

synchronize data of the requested at least one object having a synchronized state with the CSS; and

upon the byte range becoming available, provide the requested data to the client.

Date reçue / Date received 2021-12-13

2. The computing device of claim 1, wherein the state of the requested at least one object is changed from a stub state to a synchronized state based on a cache management policy, wherein the computing device is further configured to:

disable synchronization of data of the at least one object in the stub state with the CSS.

3. The computing device of claim 1, wherein the computing device is further configured to:

synchronize metadata of objects that are files to the storage system, wherein the metadata of each file includes at least a filename of the file; and

selectively synchronize metadata of a set of files residing in a predefined set of directories to the storage system, wherein the selective synchronization is performed on-demand.

4. The computing device of claim 1, wherein the requested at least one object has a stub state and wherein the computing device is further configured to:

change a state of at least the requested byte range to a synchronized state.

5. The computing device of claim 1, wherein the objects that are files that are maintained in the storage system are exposed through a virtual file system or arranged in a multi-level namespace.

6. The computing device of claim 1, wherein the computing device is any one of: a cloud storage gateway (CSG) and an endpoint.

7. A method for allowing synchronized access to a cloud storage system (CSS), comprising:

receiving a request from a client to access data in a byte range within at least one object maintained in a storage system, wherein the storage system is configured to maintain objects, wherein each object is any one of a file and a byte range within a file, each at least one object having a content, a corresponding object stored in the CSS,

Date reçue / Date received 2021-12-13

and a state, wherein each state is any one of: a synchronized state, in which an object in the computing device has its content at least automatically updated in accordance with changes to content of its corresponding object in the CSS and changes to the content of the object in the computing device are automatically propagated to the content of the corresponding object in the CSS so as to continuously maintain the content of the object in the computing device in synchronization with the content of the corresponding object in the CSS so long as the object is in the synchronized state, and a stub state, in which an object's content is not automatically updated to be maintained in synchronization with the content of the corresponding object in the object in the CSS;

changing the state of the requested at least one object from a stub state to a synchronized state upon receiving the request;

synchronizing data of the requested at least one object having a synchronized state with the CSS; and

upon the byte range becoming available, providing the requested data to the client.

8. The method of claim 7, wherein the state of the requested at least one object is changed from a stub state to a synchronized state based on a cache management policy, wherein the method further comprises:

disabling synchronization of data of the at least one object in the stub state with the CSS.

9. The method of claim 7, further comprising any one of:

synchronizing metadata of objects that are files to the storage system, wherein the metadata of each file includes at least a filename of the file; and

selectively synchronizing metadata of a set of files residing in a predefined set of directories to the storage system, wherein the selective synchronization is performed on-demand.

10. The method of claim 7, wherein the requested at least one object has a stub state, the method further comprising:

Date reçue / Date received 2021-12-13

changing a state of at least the requested byte range to a synchronized state.

11. The method of claim 7, wherein the objects that are files that are maintained in the storage system are exposed through a virtual file system or arranged in a multi-level namespace.

12. The method of claim 7, wherein the method is performed by a computing device that is any one of: a cloud storage gateway (CSG) and an endpoint.

13. A non-transitory computer readable medium having stored thereon instructions for causing one or more processing units to execute a process for allowing synchronized access to a cloud storage system (CSS), the process comprising:

receiving a request from a client to access data in a byte range within at least one object maintained in a storage system, wherein the storage system is configured to maintain objects, wherein each object is any one of a file and a byte range within a file, each at least one object having a content, a corresponding object stored in the CSS, and a state, wherein each state is any one of: a synchronized state, in which an object in the computing device has its content at least automatically updated in accordance with changes to content of its corresponding object in the CSS and changes to the content of the object in the computing device are automatically propagated to the content of the corresponding object in the CSS so as to continuously maintain the content of the object in the computing device in synchronization with the content of the corresponding object in the CSS so long as the object is in the synchronized state, and a stub state, in which an object's content is not automatically updated to be maintained in synchronization with the content of the corresponding object in the object in the CSS;

changing the state of the requested at least one object from a stub state to a synchronized state upon receiving the request;

synchronizing data of the requested at least one object having a synchronized state with the CSS; and

upon the byte range becoming available, providing the requested data to the client.

Date reçue / Date received 2021-12-13

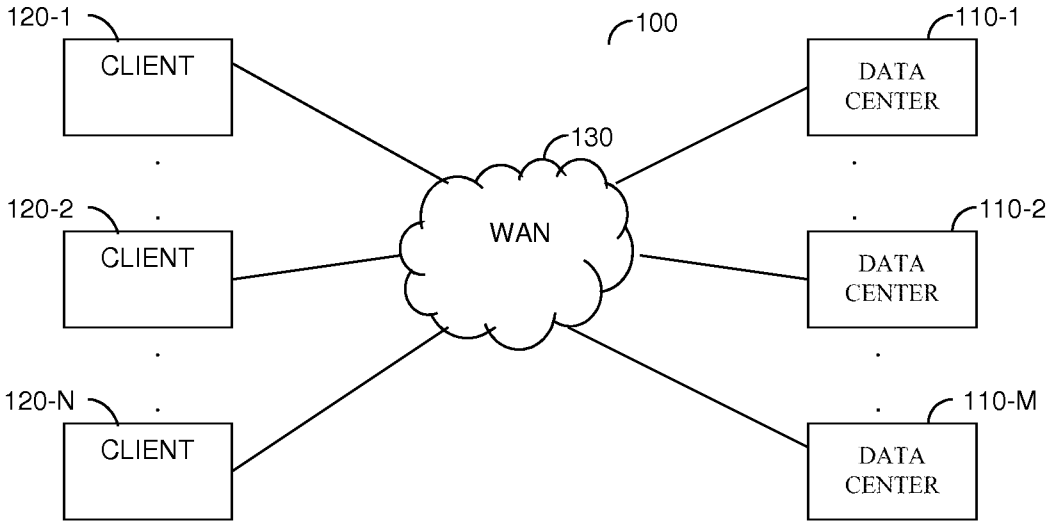

120-1
CLIENT
120-2
CLIENT
120-N
CLIENT
100
130
WAN
110-1
DATA CENTER
110-2
DATA CENTER
110-M
DATA CENTER

FIG. 1

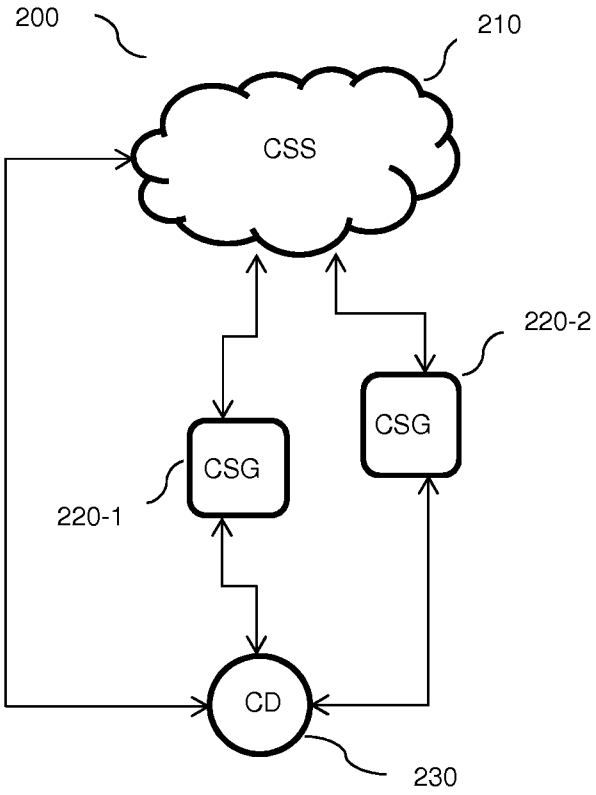


FIG. 2

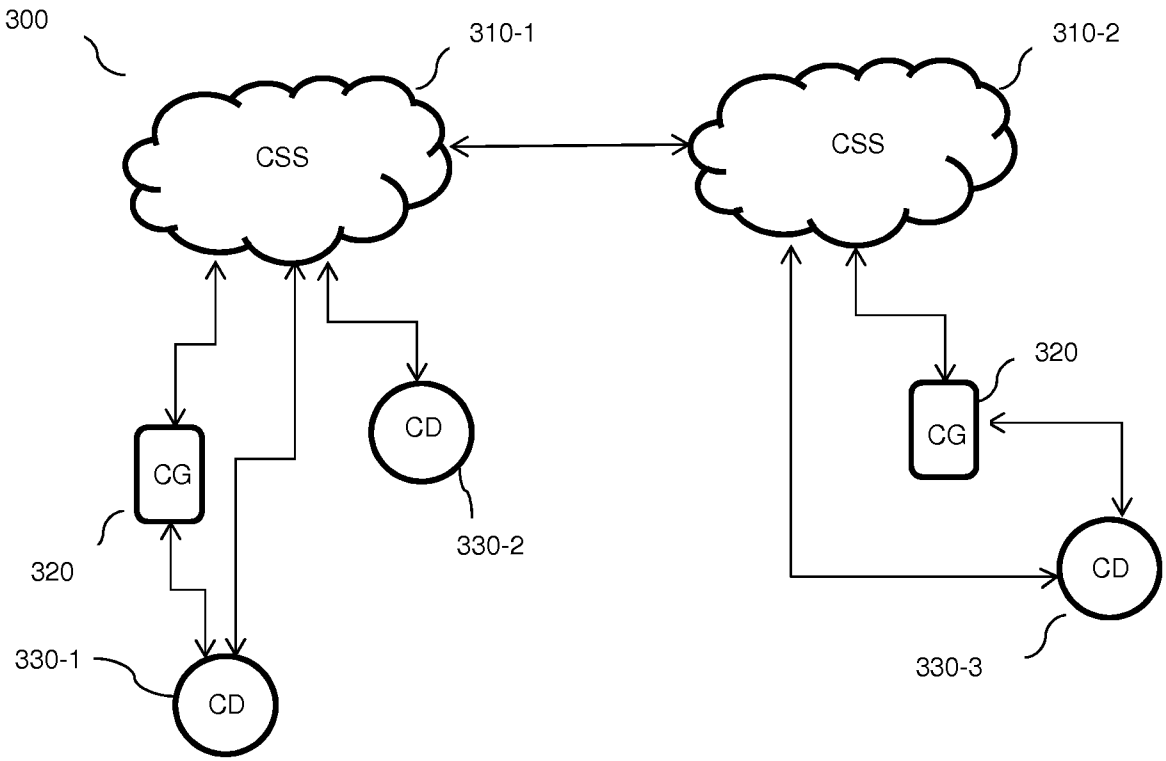


FIG. 3

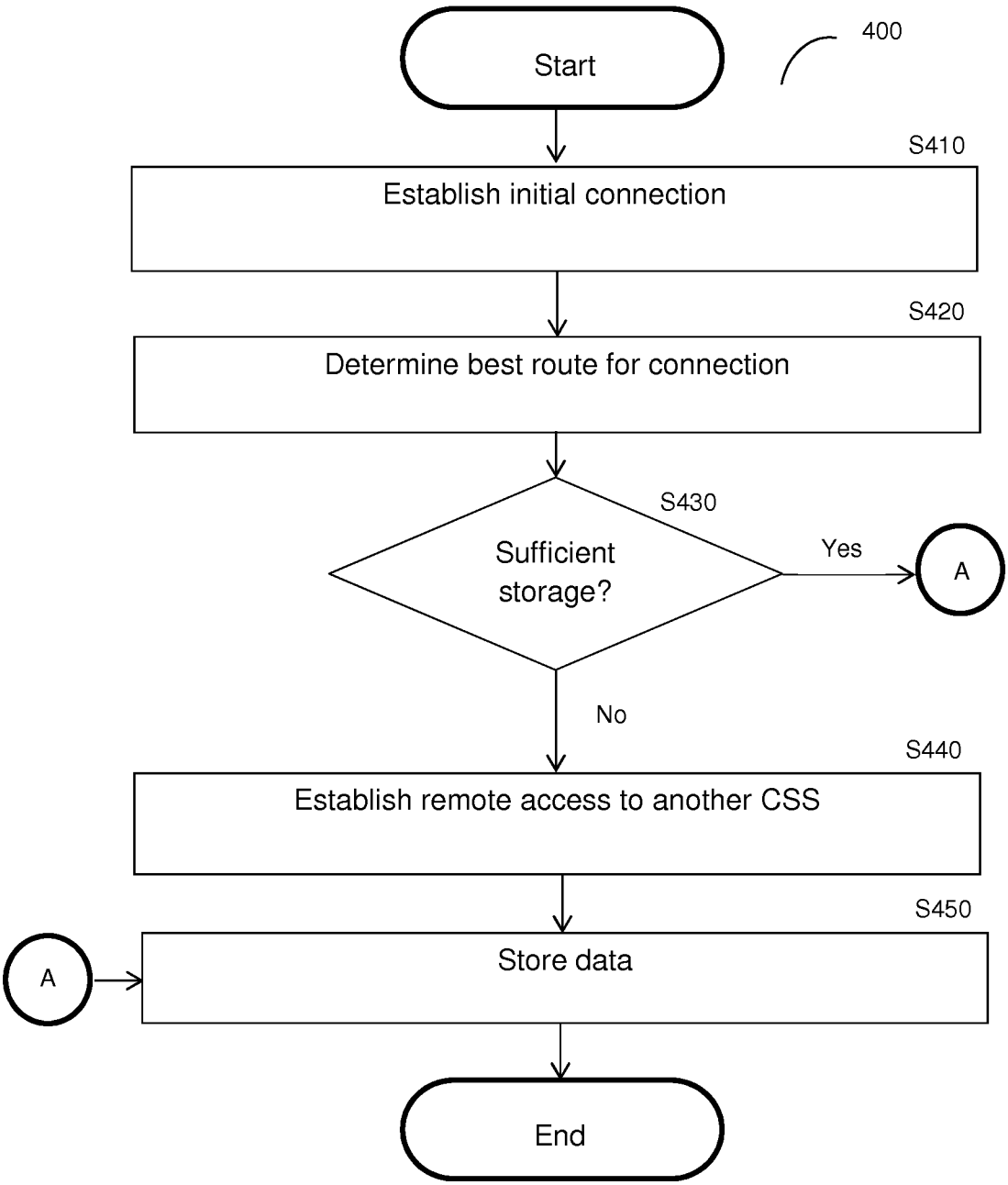

Start
400
S410
Establish initial connection
S420
Determine best route for connection
S430
Sufficient storage?
Yes
A
No
S440
Establish remote access to another CSS
S450
A
Store data
End

FIG. 4

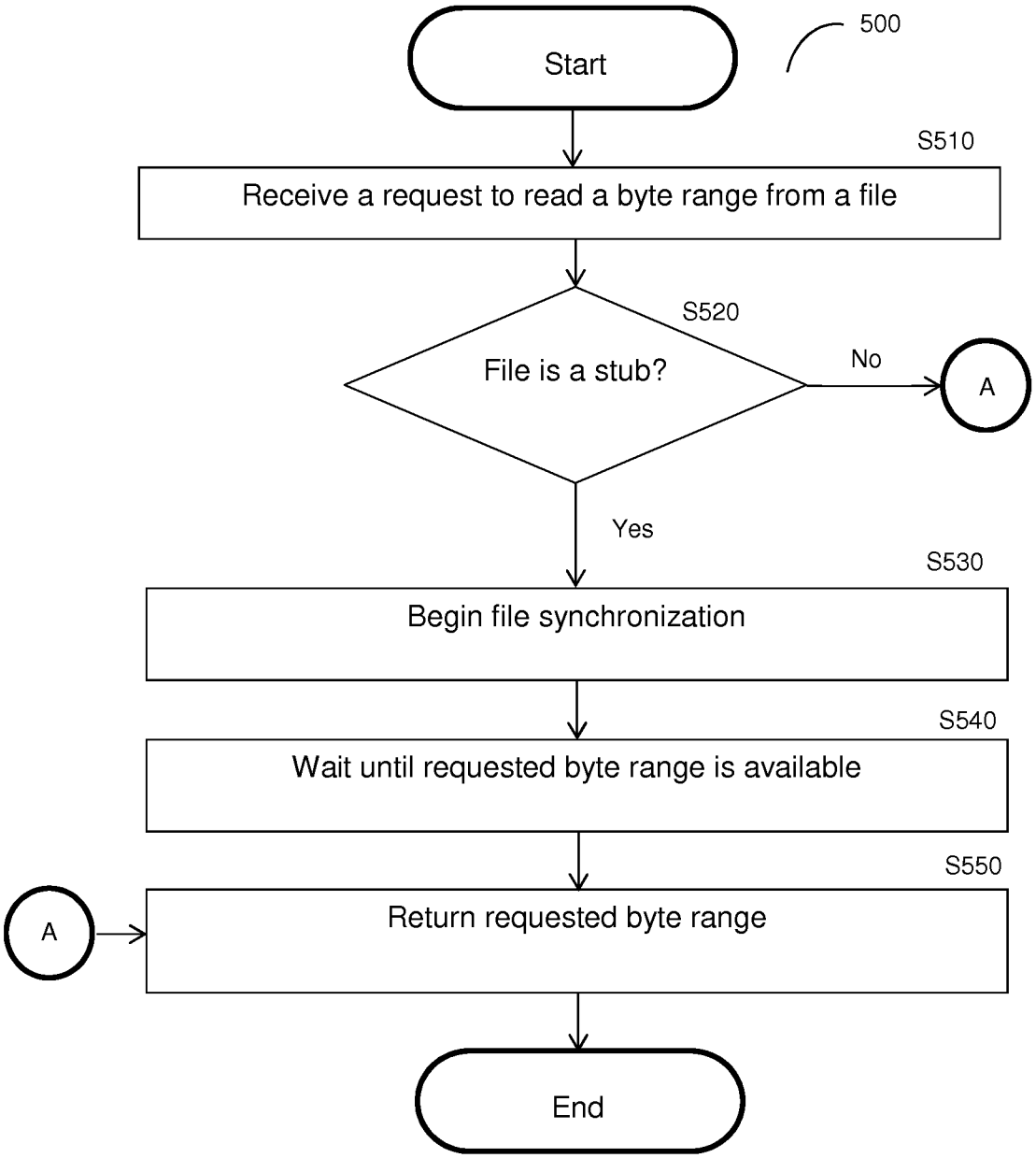

Start
500
S510
Receive a request to read a byte range from a file
S520
File is a stub?
No
A
Yes
S530
Begin file synchronization
S540
Wait until requested byte range is available
S550
A
Return requested byte range
End

FIG. 5

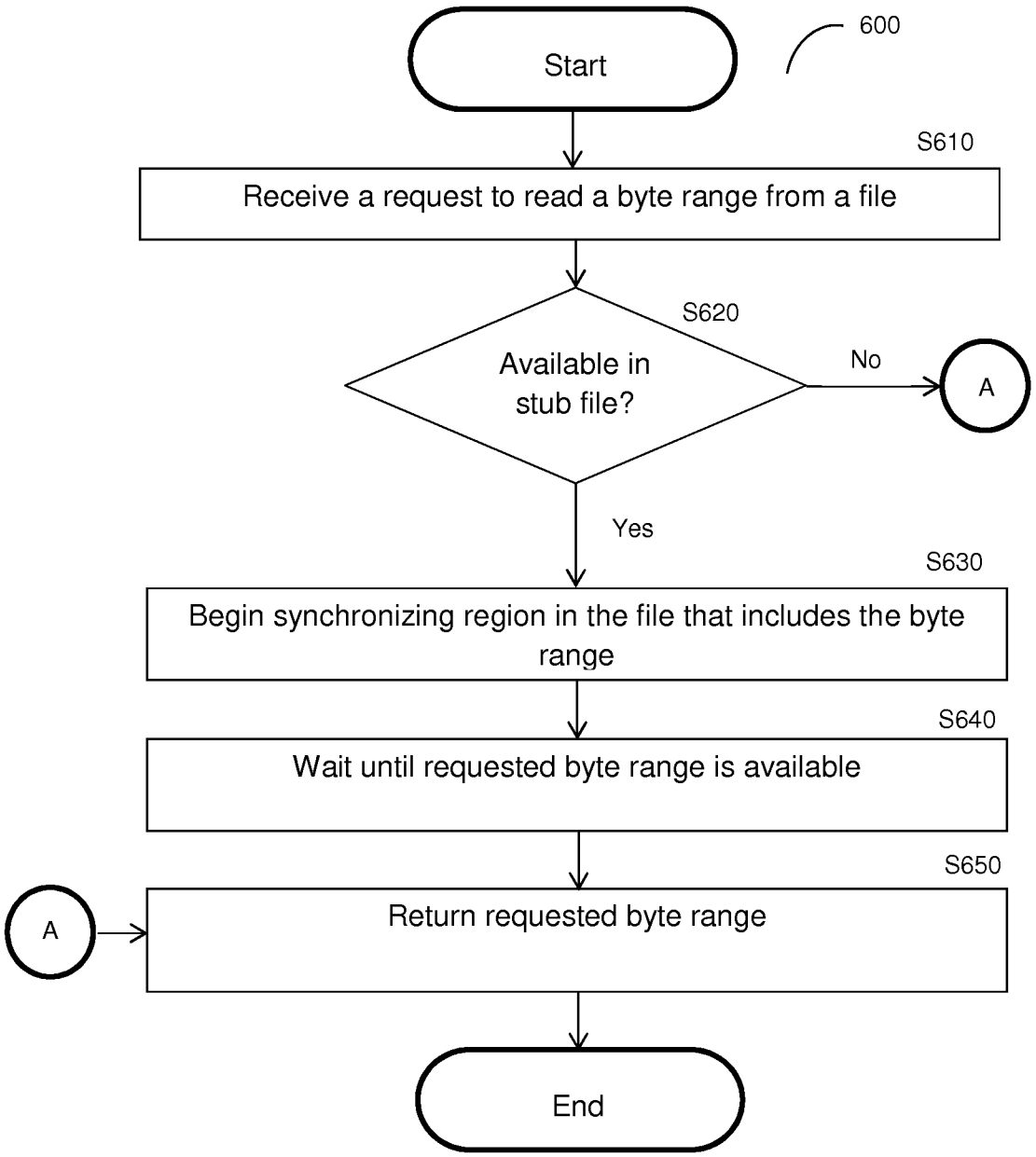

Start
600
S610
Receive a request to read a byte range from a file
S620
Available in stub file?
No
A
Yes
S630
Begin synchronizing region in the file that includes the byte range
S640
Wait until requested byte range is available
S650
A
Return requested byte range
End

FIG. 6

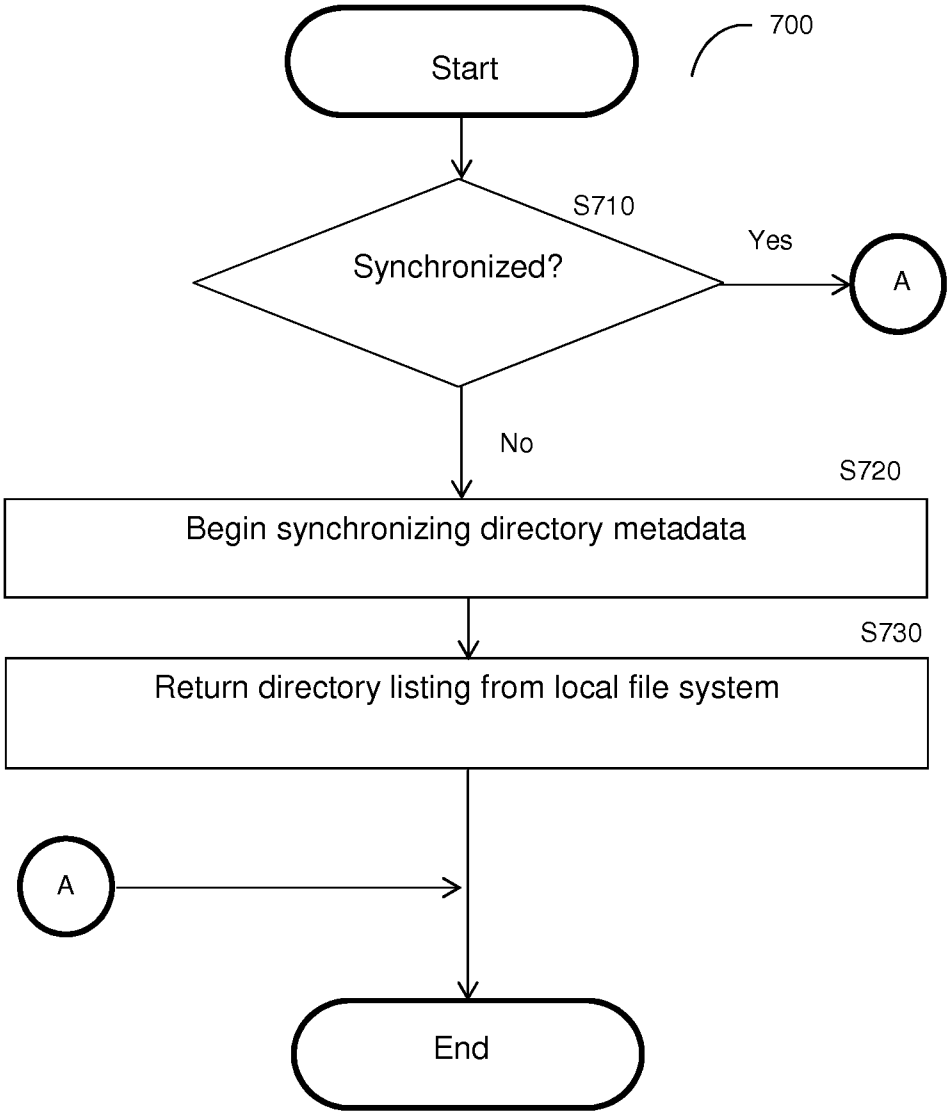

Start
700
S710
Synchronized?
Yes
A
No
S720
Begin synchronizing directory metadata
S730
Return directory listing from local file system
A
End

FIG. 7

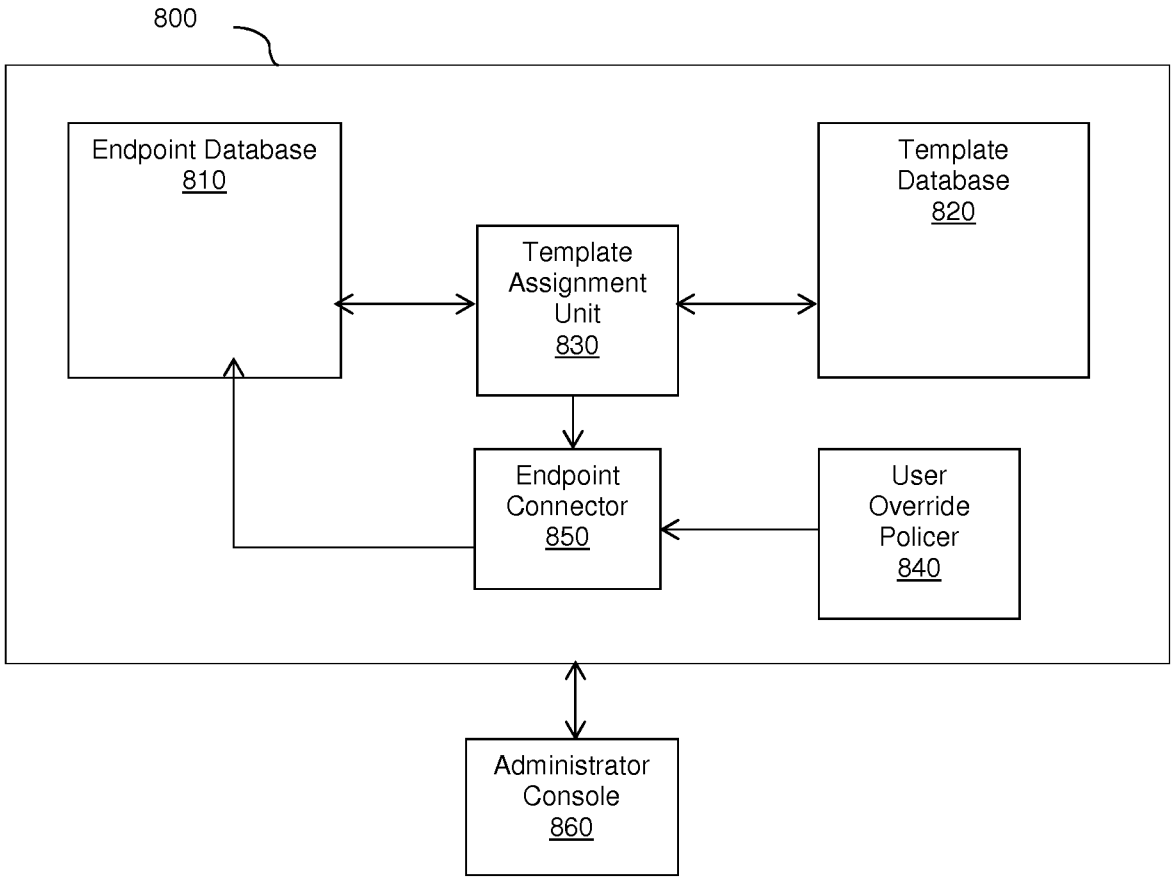

800
Endpoint Database
810
Template Assignment Unit
830
Template Database
820
Endpoint Connector
850
User Override Policer
840
Administrator Console
860

FIG. 8

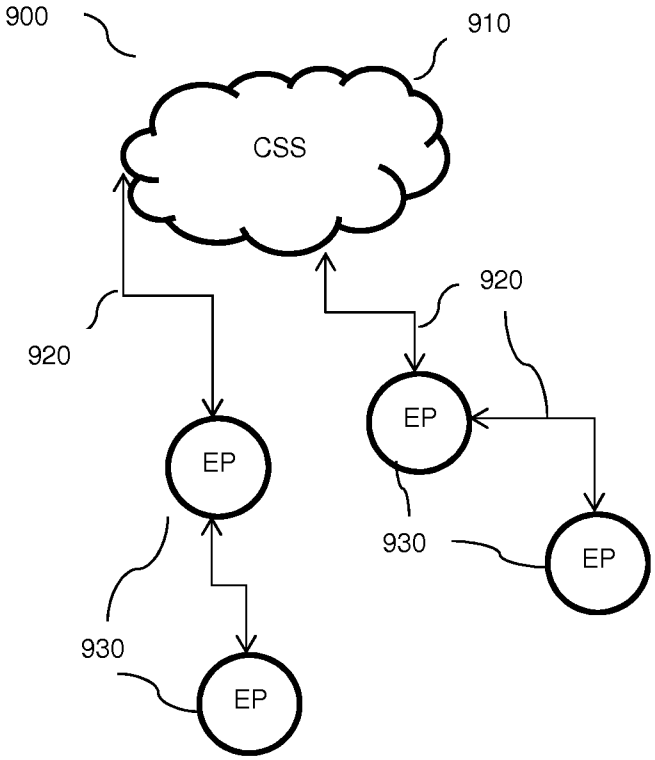

900
910
CSS
920
920
EP
EP
930
EP
930
EP

FIG. 9

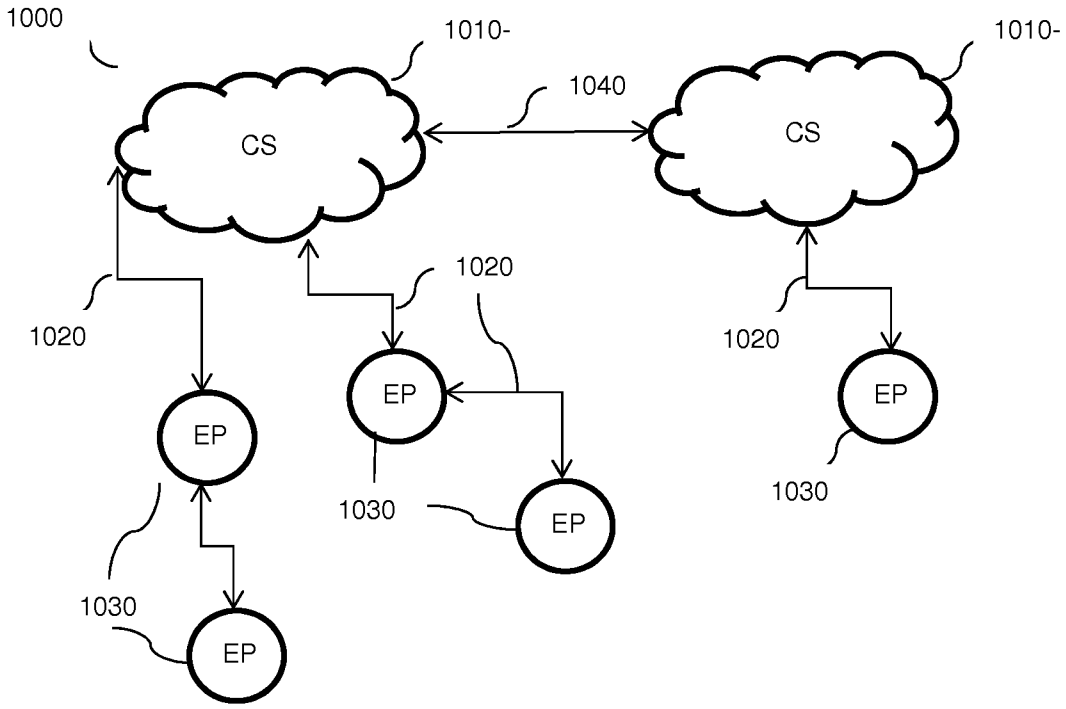

1000
1010-
1040
1010-
CS
CS
1020
1020
1020
EP
EP
EP
1030
EP
1030
1030
EP

FIG. 10

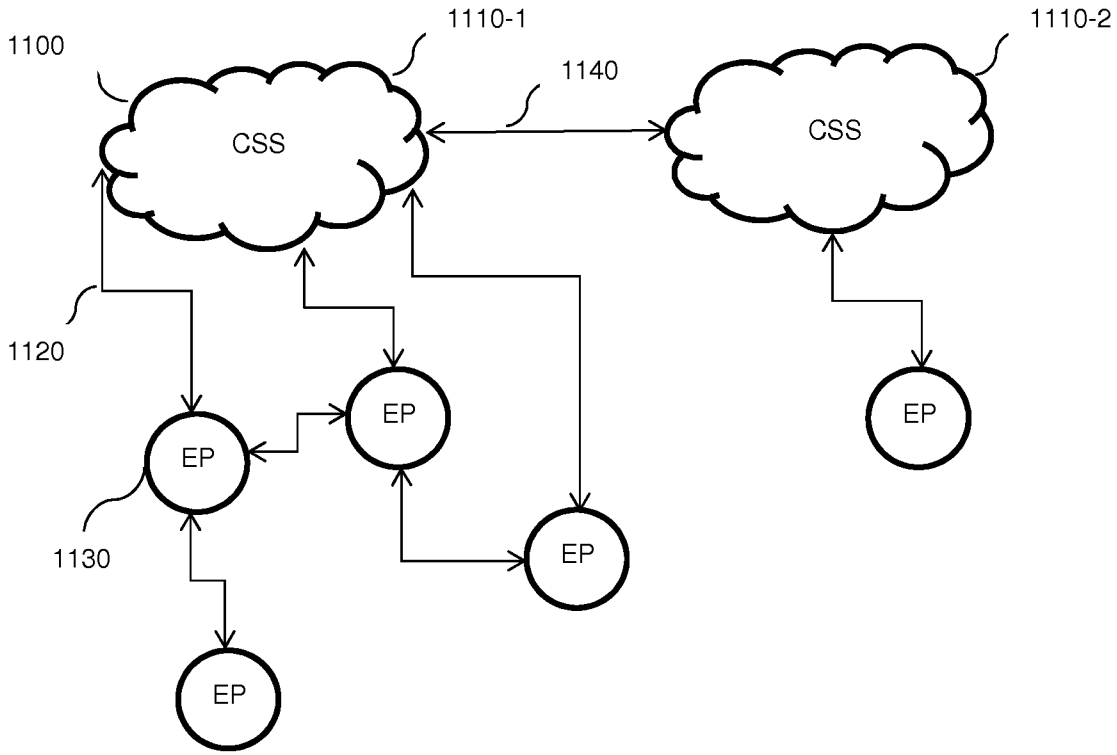


FIG. 11

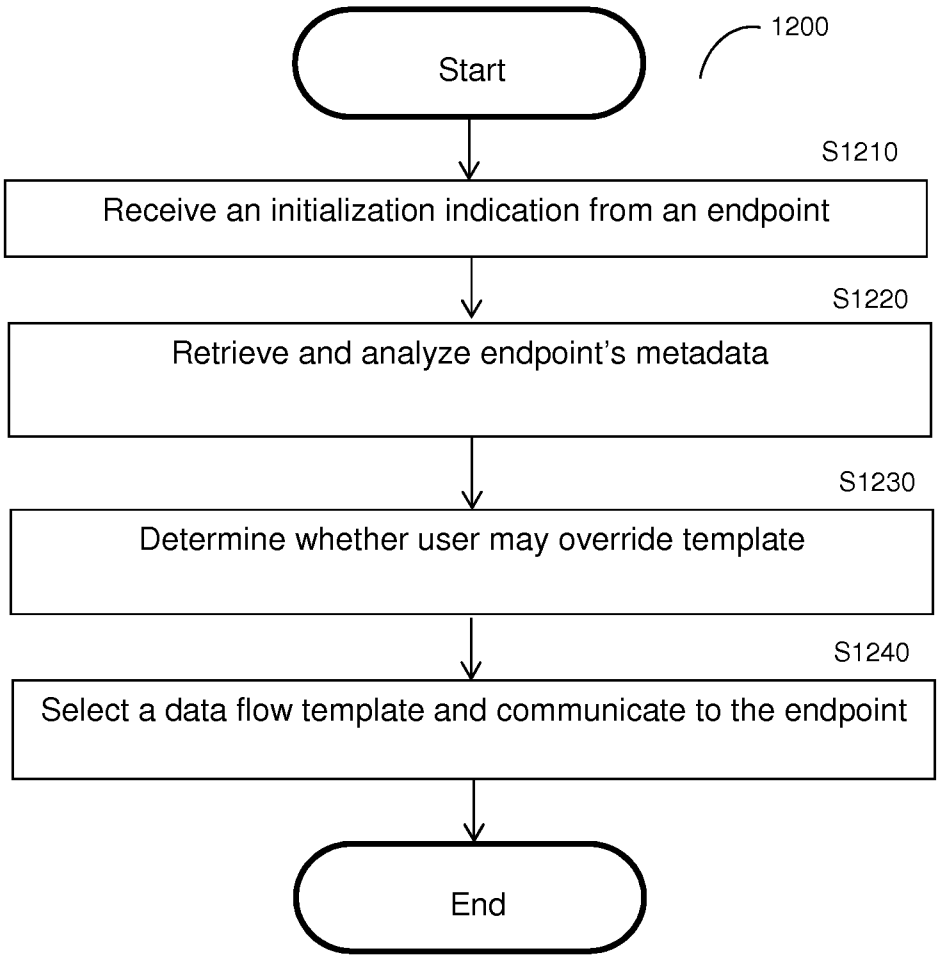

Start
1200
S1210
Receive an initialization indication from an endpoint
S1220
Retrieve and analyze endpoint's metadata
S1230
Determine whether user may override template
S1240
Select a data flow template and communicate to the endpoint
End

FIG. 12

1300
1310
1320
Template11
Type is managed
Sync home folder
Edit...
Sync additional folders:
Add...
Quick Search
Name
Owner
Size
Settings...
Cancel
Save
Delete
Applications
Backup Schedule
Backup Throughput
Cloud Drive
Software Updates
Update Schedule

FIG. 13

13/16

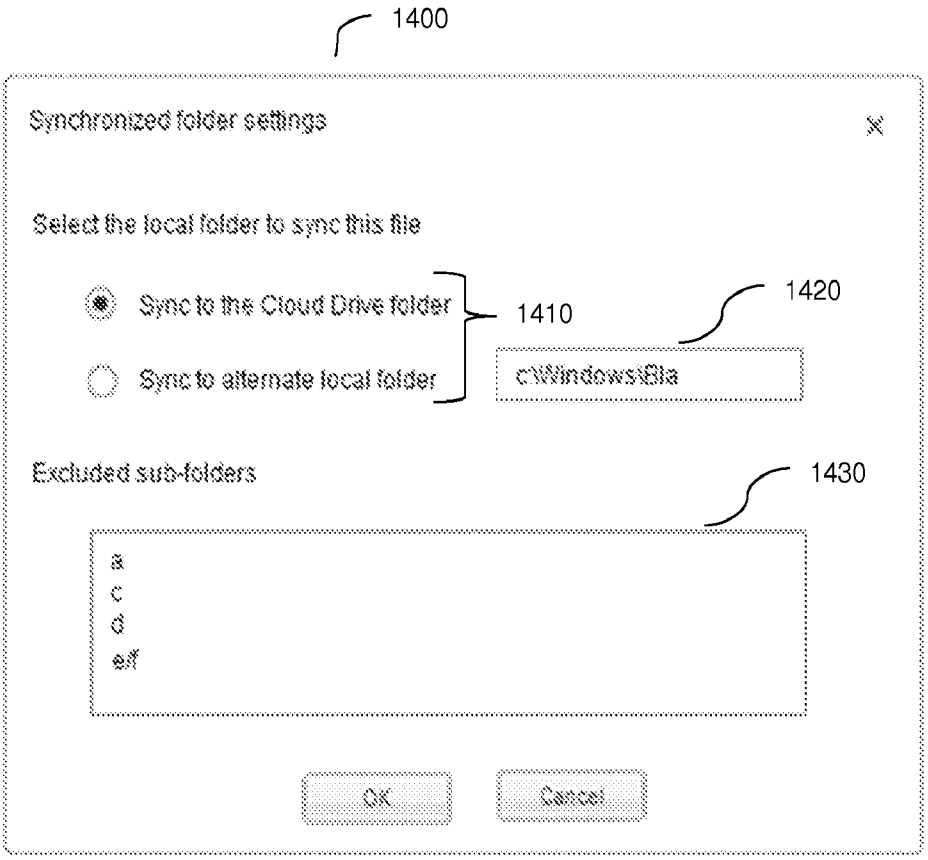

1400
Synchronized folder settings
Select the local folder to sync this file
Sync to the Cloud Drive folder
Sync to alternate local folder
1410
1420
c:\Windows\Bla
Excluded sub-folders
1430
a
c
d
e:\
OK
Cancel

FIG. 14

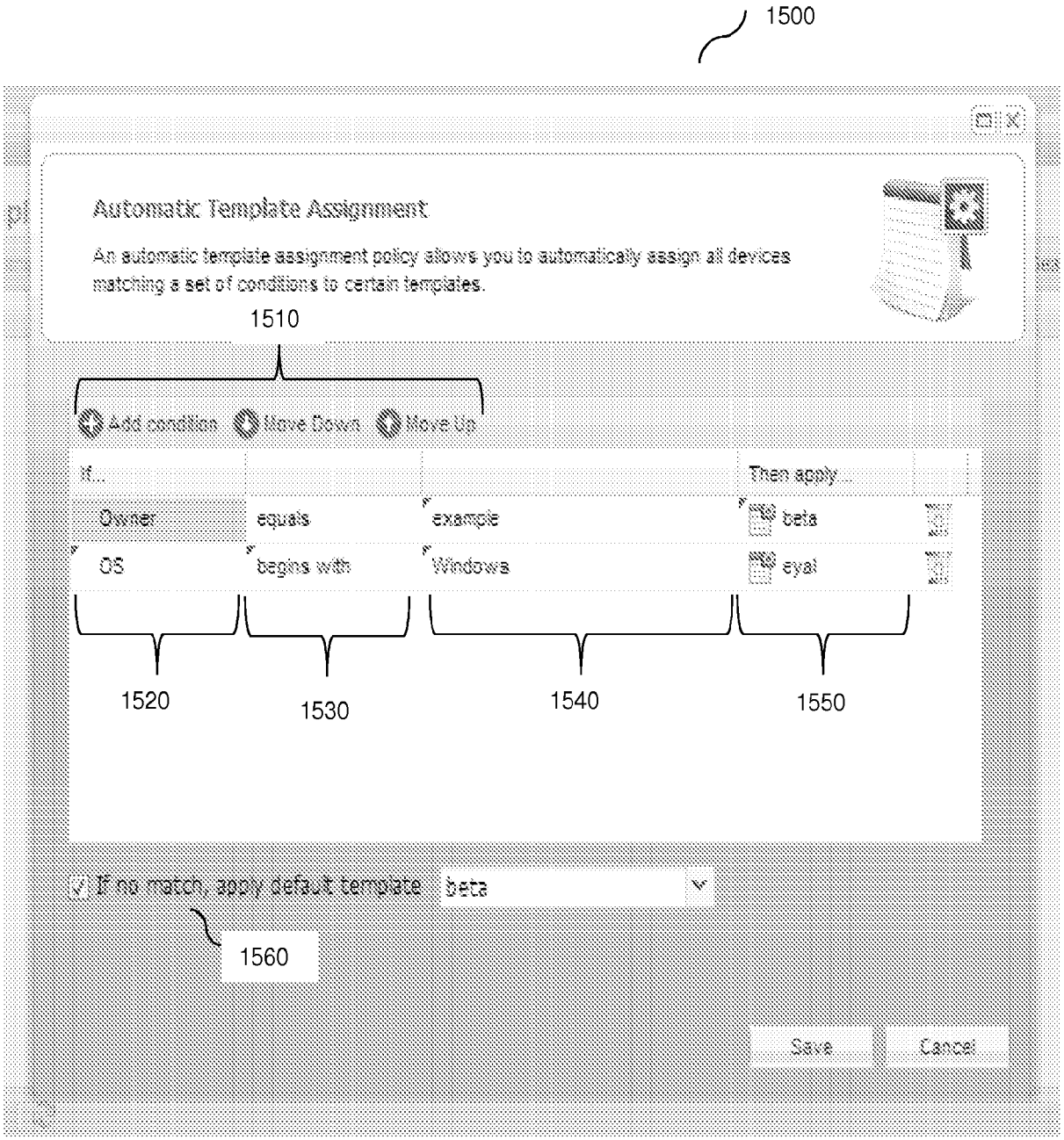

1500
Automatic Template Assignment
An automatic template assignment policy allows you to automatically assign all devices matching a set of conditions to certain templates.
1510
Add condition
Move Down
Move Up
If...
Then apply...
Owner
equals
example
beta
OS
begins with
Windows
eyal
1520
1530
1540
1550
If no match, apply default template
beta
1560
Save
Cancel

FIG. 15

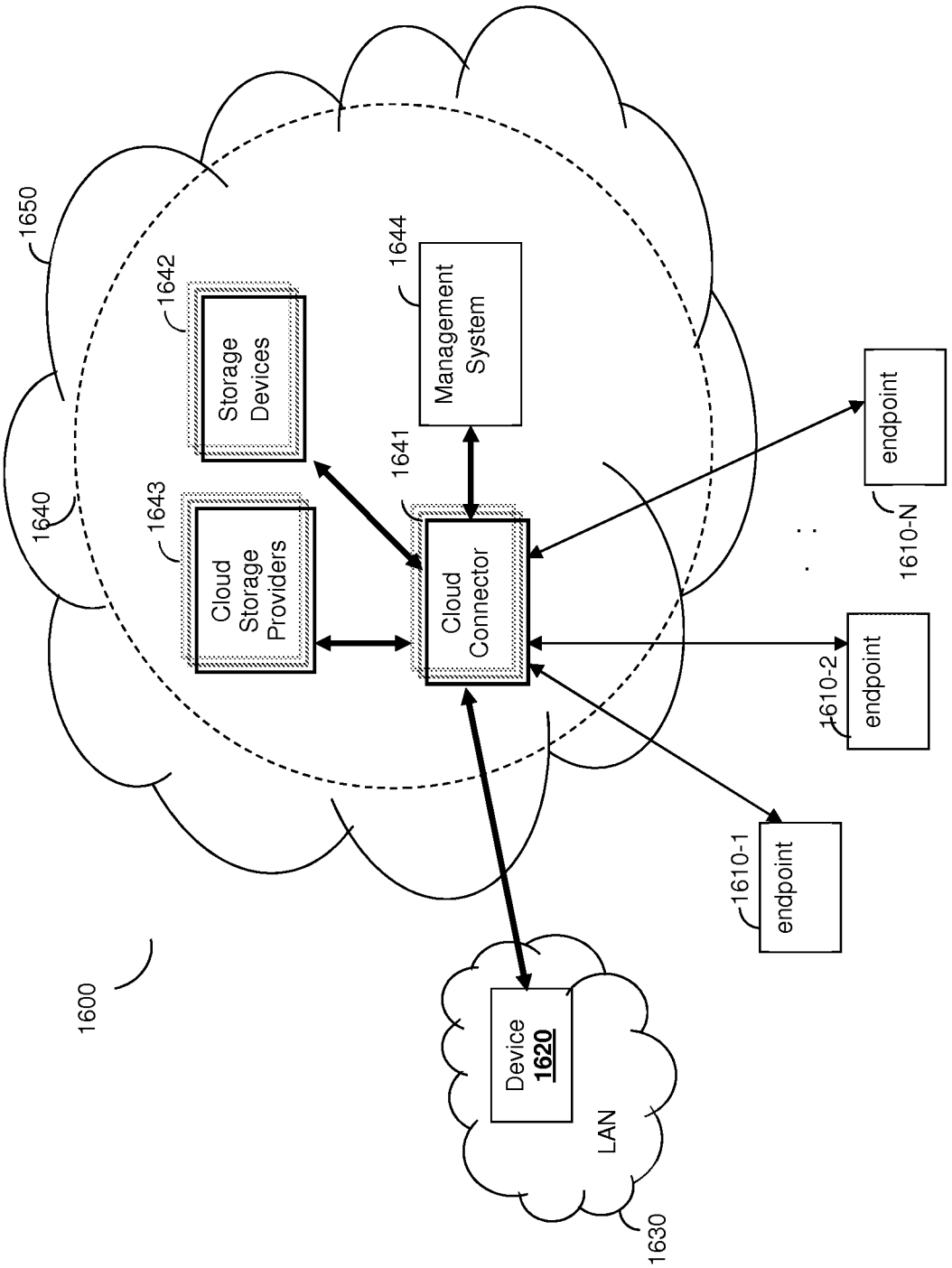

1650
1642
Storage Devices
1644
Management System
1641
1643
Cloud Storage Providers
Cloud Connector
1640
endpoint
1610-N
endpoint
1610-2
endpoint
1610-1
1600
Device 1620
LAN
1630

FIG. 16

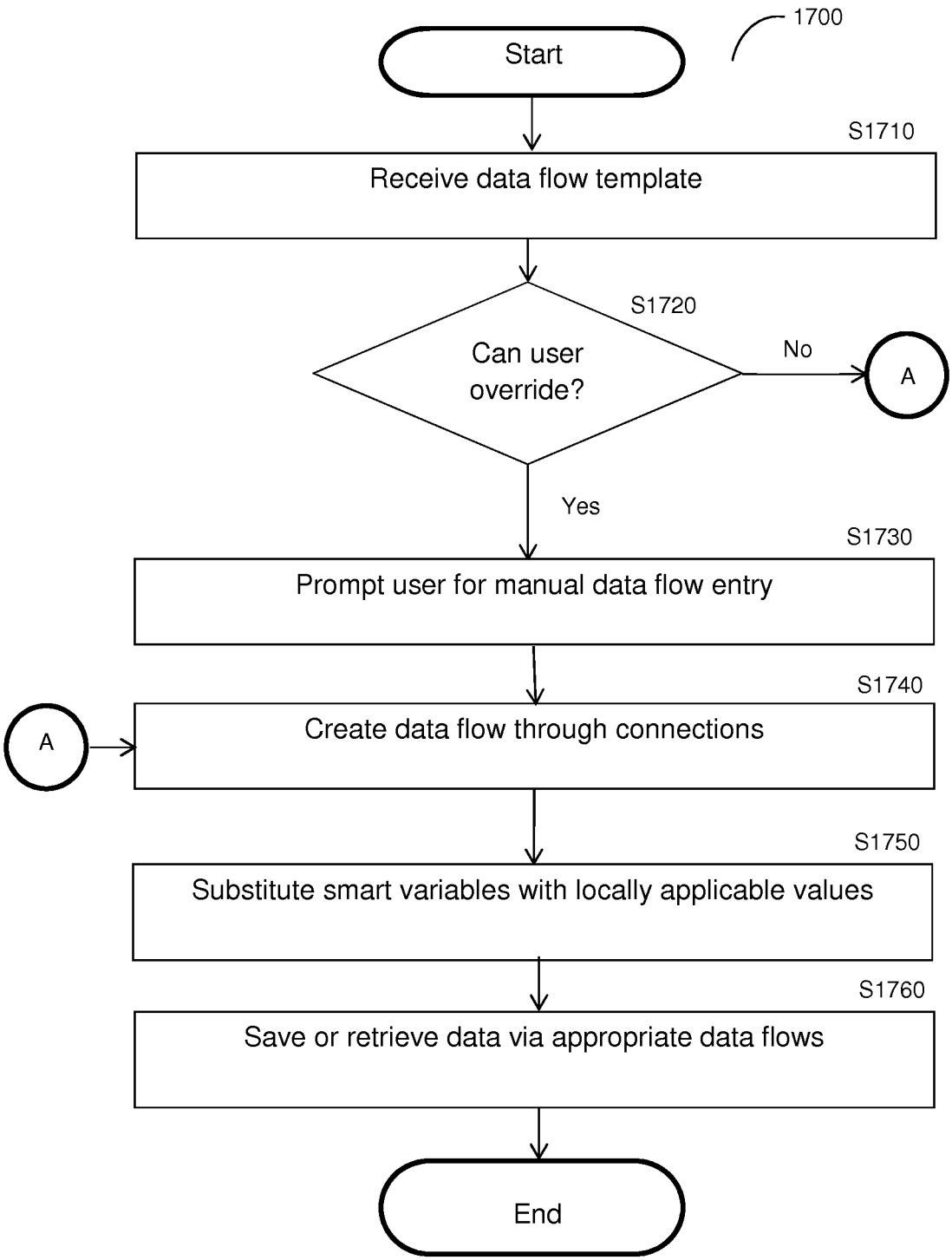
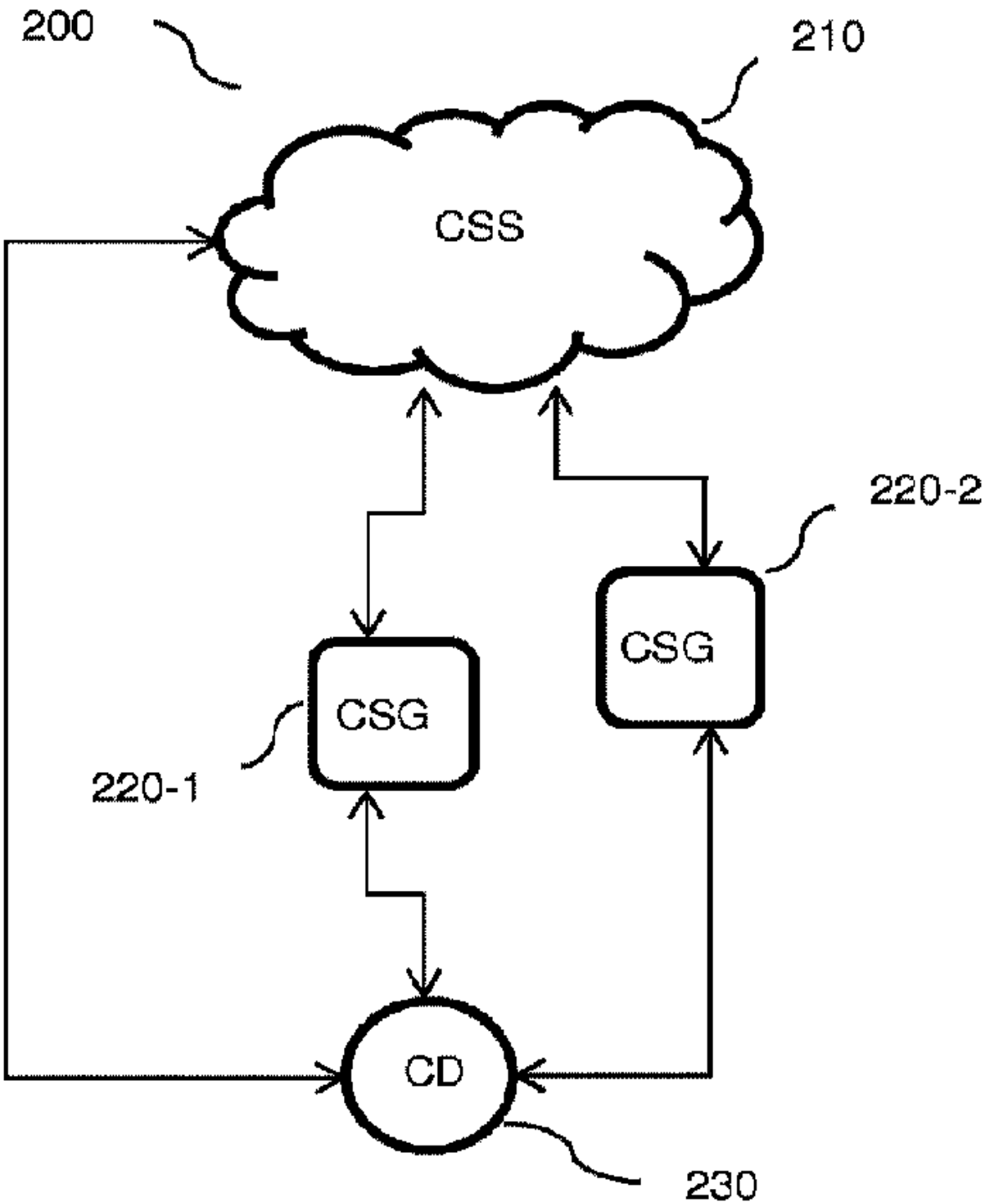

1700
Start
S1710
Receive data flow template
S1720
Can user override?
No
A
Yes
S1730
Prompt user for manual data flow entry
S1740
A
Create data flow through connections
S1750
Substitute smart variables with locally applicable values
S1760
Save or retrieve data via appropriate data flows
End

FIG. 17


200
210
CSS
220-2
CSG
CSG
220-1
CD
230