

another user is revising a particular content container. The user may be encouraged to revise a different content container until the other user's revisions are complete.

FIG. 5 illustrates a block diagram of a system for synchronizing multiple user revisions to a shared object. The system includes clients 500, 510, 540, 550 and servers 520, 530. Client 500 is coupled to servers 520, 530. Client 510 is coupled to server 520. Clients 540, 550 are coupled to server 530. Client 540 includes store 542 and child store 544. Server 520 includes store 522 and child stores 524, 526. Server 530 includes store 532. Store 532 includes substores 534, 536.

Stores 522, 532, child stores 524, 526, and substores 534, 536 may store revisions associated with a shared object. Store 522, 532, child stores 524, 526 and substores 534, 536 are hierarchical. For example, store 522 may be associated with an entire shared notebook document. Child store 524 may be associated with a section of the shared notebook document. Child store 526 may be associated with a page of the shared notebook document. In one embodiment, only the most recent version of the top-level shared object is included in store 522. Store 532 may store an entire top-level shared object. Substores 534, 536 are associated with portions of the shared object. For example, substore 534 may be associated with a section of the shared object, and substore 536 may be associated with a different section of the shared object.

An application may load a shared object from server 520 or server 530 to client 500 without a current version of a particular content container of the shared object. For example, client 500 requests a shared object from store 522. The most recent available version of the shared object is presented at client 500. The most recent available version of the shared object may not correspond to the current version of the shared object because data of the most recent revision is not available in the corresponding child store 526. A request tag is assigned to child store 526 to indicate that client 500 requires the most recent revision data to update the shared object. Child store 526 may also be assigned a time stamp that identifies the time and date when client 500 requested the revision data from child store 526. Child store may also be assigned a GUID that identifies the client that requested the data (e.g., client 500). The request tag, time stamp, and GUID are used to inform client 500 when another client accesses child store 526. For example, client 510 may access child store 526 with the most current revision data. Thus, client 500 is informed that the most current revision data of the shared object is available in child store 526.

Client 500 may be a user's home computer and client 540 may be a user's work computer. Server 530 may be an exchange server that transfers a revision file between clients 500, 540. The revision file may be used to update a shared object stored on clients 500, 550. In one embodiment, client 500 is restricted from handling files larger than a predetermined size (e.g., 2 megabytes). For example, client 500 may include an email application that limits the size of email messages that may be received. Store 542 includes revisions associated with a top-level shared object. Child store 544 includes revisions associated with a content container of the shared object.

Client 540 may poll server 530 to determine whether another client has submitted a data revision request. Client 540 may satisfy the request when the latest version of the requested data revision is available in store 542 or child store 544. Client 540 may transfer the entire requested revision to client 500 if the size of the revision file is less than the limit that can be handled by client 500. If the size of the revision file is greater than the limit, the file may be divided into smaller files that are less than the limit. Alternatively, the size of the

revision file may be reduced by deleting previous requests. The smaller files are then transferred from client 540 to client 500 through server 530.

Multiple requests for revision data may be waiting on a server. In one embodiment, the requests may be made from different clients (e.g., clients 500, 550). Each requesting client may be associated with a different file size limit. For example, client 500 is limited to files less than 2 megabytes and client 550 may handle files up to 20 megabytes. Therefore, both requests cannot be satisfied through one transfer transaction when the revision file is greater than 2 megabytes. In one embodiment, a priority bit is associated with each requesting client to establish the order in which the requests are satisfied.

The requests are satisfied by synchronizing the revision file with clients 500, 550. The revision file may be synchronized with clients 500, 550 in one transaction or through a series of multiple transactions depending on the size of the revision file. Each client 500, 550 determines that the request is satisfied when the entire revision file is synchronized. Client 540 may purge the requested data because the requests are satisfied. Client 540 may later poll server 530 to determine if any additional requests are waiting to be satisfied.

FIG. 6 illustrates an operational flow diagram illustrating a process for synchronizing multiple user revisions to a shared object. The process begins at a start block where many users are authorized to access and revise a shared object simultaneously (i.e., the peer group). The object may be any entity capable of being shared such as a file. The peer group may be identified by a peer group identifier. Different versions of the shared object are identified by corresponding GUIDs and time stamps. The time stamp identifies the time when the shared object was last synchronized with a revision.

Moving to block 600, a user revises the shared object. The shared object may be revised on a server, in a local cache, or on a peer-to-peer network. In one embodiment, the revision is stored as a revision file. Proceeding to block 610, the revision is associated with a GUID and a time stamp. The time stamp identifies the time when the user revised the shared object.

Advancing to block 620, the latest version of the shared object is located. The latest version of the shared object is the version that includes the most recent revisions that are synchronized with the shared object and made available to other authorized users. The latest version of the shared object may be determined from the time stamps and GUIDs associated with different versions of the shared object.

Transitioning to decision block 630, a determination is made whether any conflicting revisions exist. Revisions may conflict when different users revise the same content container. The revision cannot be synchronized with the shared object if conflicting revisions exist. If conflicting revisions exist, processing continues at block 640 where the conflicting revisions are reconciled and merged (as discussed with reference to FIG. 7). If no conflicting revisions exist, processing continues at block 650 where the revision is synchronized with the shared object such that other users may view the revision. Processing then terminates at an end block.

FIG. 7 illustrates an operational flow diagram illustrating a process for reconciling and merging conflicting multiple user revisions to a shared object. The process begins at a start block where more than one user has revised the same content container in a shared object. A conflict results when one of the revised content containers is synchronized with the shared object such that any other revisions to the content container cannot be synchronized.

Moving to block 700, the conflicting revision is displayed on a conflict page. The conflict page resembles the corre-