

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.

[0061] 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.

[0062] 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.

[0063] 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.

[0064] 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.

[0065] 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.

[0066] 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.

[0067] 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.

[0068] Moving to block 700, the conflicting revision is displayed on a conflict page. The conflict page resembles the corresponding master page except that the conflicting revision is highlighted and displayed in place of the synchronized revision.

[0069] Proceeding to block 710, a conflict indicator is displayed on the master page of the shared object. The conflict indicator may be a drop down menu, a tab, or any other mechanism that informs a user that a conflict page is available for the master page. The conflict indicator for a conflict page associated with a particular user may be distinct from the conflict indicator for conflict pages associated with other users such that a current user may quickly identify the conflict pages generated by the current user.

[0070] Advancing to block 720, the conflict page is displayed alongside the master page when the conflict indicator is selected. The user is presented with both the synchronized state of the master page and the corresponding conflict page.

[0071] Transitioning to block 730, the user reconciles and merges the conflicting revisions into the master page. In one embodiment, the user may select the content container such that the content container is merged with the master page. In another embodiment, the user may directly implement revisions onto the master page. In yet another embodiment, the user may identify conflicting revisions as irrelevant.

[0072] Continuing to block 740, conflicting revisions that are identified as irrelevant are purged. In one embodiment, conflicting revisions may be identified as irrelevant by a user. In another embodiment, conflicting revisions may be automatically identified as irrelevant. For example, a user may have synchronized several revisions with the master version of the shared object located on a server while ignoring any corresponding conflict pages. The older conflict pages that the user did not reconcile are identified as irrelevant after a predetermined time period has elapsed. Processing then terminates at an end block.

[0073] FIG. 8 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 different versions of shared object are stored in different locations throughout a system. Moving to block 800, the shared object is downloaded from a store to a client.

[0074] Proceeding to decision block 810, a determination is made whether the shared object is the current version of the shared object. If the shared object is the current version of the shared object, processing terminates at an end block. If the shared object is not the current version of the shared object, processing continues at block 820. The shared object