

location and any available data sharing transports (e.g., peer-to-peer networks, servers) are automatically identified. Thus, the user may access a shared object and collaborate with other authorized users through different mechanisms.

[0031] Each shared object is associated with a manifest file. The manifest file identifies the locations where other versions and instances of the shared object are stored within the system. In one embodiment, the manifest file is an extensible markup language (XML) file. In another embodiment, the manifest file identifies multiple shared objects. In another embodiment, the manifest file may be associated with any object that may be shared between clients. For example, the manifest file may be associated with an entire shared object or any portion of the shared object (e.g., a content container, a section, a page, an outline, etc.).

[0032] The manifest file may be stored anywhere within the system. As shown in the figure, manifest file 254 is associated with shared object 252. Both shared object 252 and manifest file 254 are stored on web server 250. In another embodiment, the manifest file is stored in the shared object. In yet another embodiment, the manifest file is stored in an active directory. In still yet another embodiment, the manifest file is stored in multiple locations within the system. The manifest file is stored in a location identified by a unique location identifier. The unique location identifier may identify a file server, a shared area of a server, a web server, or a peer group.

[0033] The shared object may be accessed locally from a cache, through a server, or through a peer-to-peer network. The client retrieves the manifest file from the location identified by the unique location identifier in the corresponding shared object. In one embodiment, the client may store the manifest file locally for future reference. The manifest file indicates the location of any other versions and instances of the shared object within the system (e.g., in a substore or a peer group). If another version/instance of the shared object is stored in a peer group, the manifest file may include the corresponding peer group identifier.

[0034] In one embodiment, client 220 accesses shared object 252 on web server 250. Client 220 is automatically connected to other clients that are also accessing shared object 252 (e.g., the peer group). Client 220 retrieves manifest file 254 associated with shared object 252. Manifest file 254 identifies the locations of different versions and instances of shared object 252. Thus, client 220 may establish a peer-to-peer network with any other client in the peer group when any client in the peer group accesses a version/instance of shared object 252 identified by manifest file 254. Client 220 may then disconnect from web server 250 and continue to access shared object 252 on the peer-to-peer network.

[0035] In another embodiment, client 210 may access shared object 264 from peer-to-peer network 260. Client 210 retrieves manifest file 266 associated with shared object 264. Client 210 may connect to a server and determine which clients are also connected to the server. The connected clients may be accessed through the server when peer-to-peer network 260 is not available. Shared object 264 (or 252) and associated manifest file 264 (or 254) allow client 210 (or client 220) to transition automatically and seamlessly between asynchronous and synchronous communication modes.

[0036] Users are not blocked from accessing and revising a shared object when another user has access to the shared object. Any authorized users may simultaneously revise the shared object. In one embodiment, a brief instance of blocking may occur to ensure the integrity of the revision transaction. For example, a user may extensively revise the shared document while disconnected from the server. When the user reconnects to the server, other clients may be briefly blocked from accessing the shared object until all of the user's revision are implemented in the shared object.

[0037] FIG. 3 illustrates a hierarchical graph of linked nodes that indicate different portions of a shared object. In one embodiment, the shared object is a notebook that is shared among several users. Notebook node 300 symbolizes the entire shared object. Folder node 310 is included within notebook node 300. Section node 320 is included within folder node 310. Page nodes 330, 335 are included within section node 310. Table node 340, ink node 342, outline node 344, and image node 346 are included within page node 330. Outline element node 350 is included within outline node 344. Text node 360 is included within outline element node 350. Different nodes may be grouped together in a content container. For example, outline node 344, outline element node 350, and text node 360 may be grouped together as content container R0. Content container R0 is assigned a GUID (e.g., GUID-0). The GUID uniquely identifies content container R0.

[0038] A content container includes shared object content (e.g., a word, a sentence, a paragraph, a page, a table, a picture, handwriting, a uniform resource locator, or any combination of data included in the shared object). Content containers provide a dimension for object content that is grouped together. For example, a content container may correspond to a line, a paragraph, a page, or specific page elements (e.g., only the tables on a particular page).

[0039] The shared object stores an initial version of the graph. Specific operations may then be performed on individual content containers. For example, a user may revise the data of a content container. The revision to the shared object may be identified as a state of the content container. The shared object stores the revised content containers of the graph. A current state of the content container is compared to a previous state using GUIDs and time stamps such that a determination may be made whether the content container has been revised.

[0040] For example, two different users may each access the shared document and modify content container R0. One user may revise content container R0 by deleting text node 360 (as shown in revision R1). Revision R1 is stored in the shared object. Revision R1 is assigned a GUID (e.g., GUID-1) to uniquely identify the revised container and a timestamp that identifies the time and date when revision R1 is written to the shared object. Another user may revise content container R0 by adding text node 380 to outline element node 350 (as shown in revision R2). Revision R2 is stored in the shared object. Revision R2 is assigned a time stamp and a GUID (e.g., GUID-2) to uniquely identify the revised content container.

[0041] Different users may revise a shared object at different times such that multiple versions of the shared object may coexist. However, there is only one latest version of the shared object. In one embodiment, the latest version of the