

**SYSTEM AND METHOD FOR DISTRIBUTED  
CONFLICT RESOLUTION BETWEEN DATA  
OBJECTS REPLICATED ACROSS A  
COMPUTER NETWORK**

**BACKGROUND OF THE INVENTION**

**1. The Field of the Invention**

The present invention relates to systems and methods for replication of data, that is, transferring changes (e.g., creation of new data, modification of existing data or deletion of existing data) made locally at one server to a specified list of other remote or locally connected servers. More specifically, the present invention relates to systems and methods for resolving conflicts between different versions of the same replica object replicated on one or more other servers in a computer network.

**2. The Prior State of the Art**

Today, business and technology trends are changing the way we use computers and information. The personal computer or PC has become the standard business information tool as prices have decreased and computing power has increased. In record numbers, businesses are re-engineering their organizational structure and processes to become faster and more competitive, in addition to being better able to use the wealth of information resources available today. Never before has there been so much information so readily available nor such high expectations for how much the individual will be able to accomplish by utilizing this information. The result is that people today need access to information everywhere, anytime. In June 1994, Microsoft announced a new product designed to meet these needs, called Microsoft® Exchange.

The main concept behind Microsoft® Exchange is to provide a product that integrates E-mail, scheduling, electronic forms, document sharing, and other applications such as customer tracking to make it altogether easier to turn information into a business advantage. The result is that users can access, organize, and exchange a world of information, wherever they happen to be in the world—whether from the office, the home, or while traveling on the road. In essence, a main barrier to PC-based communication, namely, accessibility and sharing by multiple parties of up-to-the-minute information, has now been significantly reduced.

With the increased accessibility and sharing of information between multiple users, it is now more common than ever for such multiple users to simultaneously or in tandem work on shared data set objects, as, for example, word processing documents, spreadsheets, electronic forms, E-mail messages, graphics images or a host of other such data objects. With such shared use of data objects among multiple users of a computer network, there arises the need for each user to keep all other users of the same data object or the same set of data objects apprised of changes that are made locally by that user. This need gives rise to a process called replication of data, that is, transferring changes (e.g., creation of new data, modification of existing data or deletion of existing data) made locally at one server to a specified list of other remote or locally connected servers.

In a computer network where multiple copies of the same data object reside at several different servers, there arises the possibility of two users working with two different copies of the same data object at the same time. These users may each change the data object in such a way that the two copies of the data object are in conflict. A simple example of such a

conflict might be where two users are working on the same proposal document and one user deletes a paragraph while another user adds additional information to the same paragraph. Because of this eventuality, replication processes require systems and methods for discovering and resolving conflicts among multiple copies of the same data object.

In the past, many approaches to conflicts focused on simply preventing conflicts in the first place. For example, many computer networks are currently configured in a client/server topology, such as that illustrated in FIG. 1. In the client/server model, shown in FIG. 1 generally as 10, a single copy of the data object resides on the server machine, as for example server 12 of FIG. 1. Client machines 14 are only allowed to access data objects stored on server 12 in a manner that avoids conflicts. Methods such as file locking so that a data object can only be changed by one client at a time, object check out/check in procedures and other such techniques are used to avoid conflicts. Similar methods are also employed for other types of networks not configured in a client/server topology.

While the above methods avoid conflicts between multiple copies of the same data object, the above methods also severely limit the utility of shared data and hamper efforts of groups of people to work collaboratively on a single project. Other methods have thus been developed which try to overcome the shortfalls of methods which only attempt to prevent conflicts. One such method involves arbitration of conflicts between servers. For example, when two servers realize that their objects are in conflict, they can initiate a two-way communication dialog to arbitrate any conflicts between them. The arbitration process necessarily includes the exchange of multiple communication messages to resolve the conflict in a manner satisfactory to both servers. Such a process generates additional burdens that the communication links between servers must handle. Thus, such a process requires relatively high bandwidth communication links. Furthermore, because the communication dialog is two-way, the physical communication links between the servers must be able to transfer two-way communication messages with a relatively short elapsed time between sending a message and receiving a reply. This places additional burdens on the servers to respond in a timely fashion and may require dedication of more CPU power to the conflict resolution process.

Another problem with the arbitrated conflict resolution method is that if conflicts are multi-way (three or more servers possess objects in conflict), the difficulty of resolving the conflict, the communication bandwidth, CPU power, and the time needed to resolve the conflict increase at an alarming rate. Furthermore, protocols must be established to allow multi-way conflict resolution in an orderly fashion. When resolving multi-way conflicts, the order of resolution may be important, and negotiation of the order of resolution can add additional overhead to an already complex process. It can be readily seen that as the number of systems in conflict increases, the resources needed to resolve the conflict become prohibitively high for arbitrated conflict resolution.

In order to solve some of the problems with the above arbitrated conflict resolution method it may be possible, in some instances, to send all conflicts to a central system where the conflicts would be resolved. This method, however, also creates problems. First, the method increases the communication traffic on the network since all systems in conflict must send their data objects to the central location, and, after the conflict is resolved, receive the results of the conflict resolution process. Another problem