



[About DSO](#) [About the Editors](#) [Mailing List](#) [Contact Us](#) [IEEE Computer Society](#)

FROM THE TRENCHES

Editor: Maarten van Steen, steen@cs.vu.nl

Fault-Tolerant Support in Real-Time Collaborative Editing Systems

Xiao Qin, Griffith University

Groupware systems let physically dispersed teams collaborate on common tasks over distance and time. A real-time groupware system requires all users to be present at their respective sites at the same time. A non-real-time groupware system, however, lets users work on common tasks at different times. Real-time collaborative editing systems that enable groups of geographically distributed users to simultaneously view and edit shared documents, make groupware applications more practical. This usefulness becomes even more pronounced when users can use real-time collaborative editing systems on the Internet.

The basics

Developers of real-time collaborative editing systems focus on good responsiveness, support for unconstrained collaboration, and tolerance of failed processes. When improving fault tolerance, two important approaches take precedence: replication and persistence.

Replication lets a component fail, permitting other replicas to take over. This leads to high availability. *Persistence* is needed for *checkpointing*, which enables failure recovery by restoring a previously saved state.

In a real-time collaborative editing system, the clients should be able to rejoin the system in the presence of failures. Current clients must also be able to continue their work while a previously crashed client is joining the group again. The server of the collaborative editing system transfers the group's current state to the recovering client.

Crash recovery

Our research group at Griffith University has devised a new efficient approach for our current project, [Reduce](#) (*real-time distributed unconstrained cooperative editing*), to support crash recovery. Reduce's topology is a star-like network, in which the server is the central part of the system and each client connects to the server through a separate channel. The server does not generate editing operations, but receives operations from client sites, maintains a remote final system state, and then propagates the operations to other client sites. The server also manages the memberships of the sites in the

collaborative editing systems. We have developed a primary–backup server that can tolerate a single server failure. If the primary server crashes, a backup server automatically takes over without restarting the primary.

The server starts by loading remote final states from persistent storage and sets the *remote history list* to empty. After the initialization phase, the server will wait for messages from client sites. If the server receives a join message, it sends the *remote final state* to the joining client. Then the client is added as a member of the collaborative editing system, so that it can receive editing operations forwarded by the server.

If a client sends a finish request, the server deletes it from the group. If the server receives a recover message from a client site C, it means that C crashed and is now rejoining the collaborative system by loading its final state from local storage. After the creation of the *local final state*, existing clients may generate operations, making the *local final state* inconsistent with the state at the server. To maintain the collaborative editing system's consistency, the server must send all the operations that were generated after the creation of this *local final state* to the rejoining client.

Normally, the server transmits the system's state to the client sites. Very large shared documents have an unacceptably high communication delay for the state transmission. Therefore, it is important to minimize delay for recovering shared state when a client rejoins the system. In our approach, clients persistently record a *state* to local storage. If a client is disconnected, it can rejoin by reloading this state. This provides an efficient approach to reduce the state transmission delay.

Most research work in real-time collaborative editing systems focus on [consistency maintenance](#), [user-intention preservation](#), [group undo](#), and [group awareness](#). [Fault tolerance](#), however, has not been studied in depth. Our work represents a first and preliminary attempt to study fault tolerance in a real-time collaborative system. We are now working on two issues. First, we plan to develop a simulation program to evaluate our new approach's performance. Second, we are investigating a fault-tolerant mechanism to provide basic fault-tolerant services in collaborative systems.

Xiao Qin is a PhD candidate in computer science at Griffith University. He also maintains the *DS Online Collaborative Computing* subarea. His research interests include real-time systems, distributed computing, fault-tolerant computing, and operating systems. He has a BS and masters of engineering in computer science, both from the Huazhong University of Science and Technology, China. Contact him at School of Computing and Information Technology, Griffith University, Brisbane, Queensland 4111, Australia; xqin@cit.gu.edu.au; www.cit.gu.edu.au/~xqin.

Related Research Resources on the Internet

- ACM Group'99 Workshop on [Consistency Maintenance and Group Undo in Real-Time Group Editors](#)

- [Collaborative Writing Project at the University of Toronto](#)
- [Informal Group Awareness Research](#)
- [Requirements Definition Collaborative HTML Editor](#)
- [Supporting Group Awareness in Multi-User Environments through Perceptualization](#)
- [User Intention Preservation](#)



Feedback? Send comments to dsonline@computer.org.

This site and all contents (unless otherwise noted) are [Copyright](#) ©2001, Institute of Electrical and Electronics Engineers, Inc. All rights reserved.