

# Exploring Communication Overheads and Locking Policies in a Peer-to-Peer Synchronous Collaborative Editing System

Jon A. Preston  
PhD Student

Department of Computer Science  
Georgia State University  
P.O. Box 3994  
Atlanta, GA 30302-3994  
770 456 9681

Advisor: Dr. Sushil K. Prasad  
Category: Graduate

jonpreston@mail.clayton.edu

## ABSTRACT

In this paper, we describe recent work in developing a peer-to-peer collaborative environment. The study examines various locking mechanisms/policies by adjusting the granularity of the locked space within the shared document. Additionally, as the interface is highly dependent upon user input/interaction rather than CPU computation, the communication overheads dominate the scalability analysis. Consequently, this paper examines the communication overheads associated with five different events within the system. The paper concludes with a discussion of future work in further examining various locking granularities and related work in distributed memory systems and cache coherency models.

## Categories and Subject Descriptors

D.2.6 [Software Engineering]: Programming Environments – *integrated environments, interactive environments.*

## General Terms

Management, Measurement, Performance, Design, Experimentation, Human Factors.

## Keywords

Computer-supported collaborative work, CSCW, Software Engineering, Synchronous collaboration, Peer-to-peer, Mutual exclusion, Multi-granularity locking, Extreme programming.

## 1. PROBLEM AND MOTIVATION

Collaboration usually occurs asynchronously when a deadline for a deliverable is in the distant future and quick turnaround of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

43<sup>rd</sup> ACM Southeast Conference, March 18-20, 2005, Kennesaw, GA, USA. Copyright 2004 ACM 1-58113-000-0/00/0004...\$5.00.

changes is not critical; synchronous collaboration often occurs when a deadline for a deliverable is impending and quick turnaround of changes is essential.

This study is motivated by the exploration of synchronous collaboration for software development and document development. This study focuses on general document editing and on how a synchronous collaboration tool can benefit such interactions, but future work will focus on software development collaboration (a more specific domain).

## 2. BACKGROUND AND RELATED WORK

All computer-supported cooperative work (CSCW) systems involve some level of interaction among users; consequently communication protocols to update peers within the system (notification algorithms) are vital to the success of any CSCW system [3].

Locking an entire file (or subsystem) is too costly and potentially blocks other users from being able to view the document [1]. Magnusson et al. define a fine-granularity approach to revision control that focuses on language elements (classes, methods, attributes, functions, etc.) and merges these smaller elements; since the elements are smaller in nature, it is posited that edit collisions will be reduced, and merge operations will become more manageable [2].

## 3. APPROACH AND UNIQUENESS

This study focuses on analyzing the network communication costs [3] of a collaborative system and the impact of various locking granularities [2] on usability, i.e. how does locking granularity affect the ability for users to collaborate effectively? Rather than lock the entire file, why not allow users to edit the file synchronously? The system designed in this study allows for two locking policies: no locking and line-level locking.

The first implementation does not ensure locking at all; any user could edit any portion of the shared collaborative window at any time. We hypothesize that edit collision rate is inversely proportional to document size and proportional to number of users; consequently, we assume optimistically in this model that

locking is not needed (for large documents and small number of users). When collisions do occur, document integrity is not guaranteed.

The second implementation can lock at a line level. Thus only one user can “own” any given line in the system, and a first-come first-served policy is enforced. We hypothesize that this model is scalable to any number of users and for small documents.

The system was implemented in C# with DirectX 9 using peer-to-peer networking. The visual interface provides the users the ability to edit the collaborative space, send text chat messages, and log all interactions with the shared space.

The peer-to-peer aspect of the system is particularly novel; no centralized server acts as a single point of communication bottleneck or failure, and in this system, the host is able to migrate if the original peer host leaves the session.

#### 4. RESULTS AND CONTRIBUTIONS

Though other messages are sent and managed by the DirectX 9 code for establishing connections and joining, there are only five types of data packets that are sent in this system:

- (1) A peer has joined and is added to each existing peers’ local list of peers (i.e., the peers now “know” about the new user/peer).
- (2) A peer requests a SYNCH (i.e., requests the current state of the shared document), and the host responds with the current state of the shared document.
- (3) A peer has placed content into the chat window and sends this content. The chat content is sent to all peers.
- (4) A peer has updated its position (line owned) in the shared content window, and the new position is sent to all peers. This update changes the mutex for each peer (i.e. each peer tracks what other peers “own”).
- (5) A peer has made a modification to the shared content, and the modification event is sent to all peers. Each peer must then update its local copy to ensure all copies are synchronized to include the modification.

Table 1 summarizes the network communication overhead for a simulation using three peers. High (100mbps) and low bandwidth (33.6kbps with 2% packet loss) simulations were executed with reasonable/usable communication latency due to the system’s low communication overhead.

This preliminary study demonstrate that the peer-to-peer synchronous collaborative system offers acceptable network

communication latencies/overhead and warrants that various locking models should be further studied – such as the granularity level of block, method, and class, within the domain of software engineering. Additionally, the system motivates further study into the HCI issues of locking granularities.

Our future work will include caching modifications and sending updates to peers on blocks rather than immediate updates; we expect that the packet overhead in sending immediate update messages is significant relative to the small change data, and such caching will further reduce the communication overhead. Such future research will thus incorporate concepts from previous work in cache coherent NUMA, distributed shared memory, and causal memory systems.

The system naturally lends itself to the exploration of how extreme programming could be achieved in geographically-dispersed environments as an application of the peer-to-peer collaborative system.

For more information about this project, please visit <http://cims.clayton.edu/jpreston/se>.

#### 5. ACKNOWLEDGMENTS

The author greatly acknowledges the support and comments of Vernard Martin.

#### 6. REFERENCES

- [1] Harrison W.H., Ossher H., and Sweeney P.F. *Coordinating Concurrent Development*. In Proceedings of CSCW’90, 157-168, October 1990.
- [2] Magnusson B., Asklund U., and Minör S. *Fine-Grained Revision Control for Collaborative Software Development*. In Proceedings of the 1st ACM SIGSOFT symposium on Foundations of software engineering, vol. 18, issue 5, pp. 33-41, December 1993.
- [3] Marques J. M. and Navarro L. *WWG: a Wide-Area Infrastructure to Support Groups*. In Proceedings of GROUP’01, Boulder CO, pp. 179-187, September 2001.
- [4] Shen H. and Sun C. *Flexible Notification for Collaborative Systems*. In Proceedings of CSCW’02, New Orleans Louisiana, pp. 77-86, November 2002.
- [5] Tam, J., McCaffrey, L, Maurer, F. and Greenberg, S. *Change Awareness in Software Engineering Using Two Dimensional Graphical Design and Development Tools*. Report 2000-670-22, Department of Computer Science, University of Calgary, Alberta, Canada, October 2000.

**Table 1: Communication among peers for various activities (number of processors=3)**

Activity	Packets	Bytes	Complexity
Peer joins (DirectX overhead)	24	1770	O(p)
Peer SYNCH and response (68 bytes of content)	5	159	O(p) + O(n)
Chat input (14 bytes of content)	4	94	O(np)
Position update	4	68	O(p)
Modification of shared content	4	76	O(p)