



Improving Concurrent Access in Collaborative Editing Systems

Jon A Preston

Committee

Dr. Sushil K. Prasad (Advisor)

Dr. Rajshekhar Sunderraman

Dr. Xiaolin Hu

Dr. Melody Moore Jackson (external)



Agenda

- Introduction
- Motivation
- Problem Statement
- Contributions
- Related Work
- Research Goals
- Current Research Summary
- Conclusion/Future Work
- Proposal Related Publications





Agenda

- Introduction
- Motivation
- Problem Statement
- Contributions
- Related Work
- Research Goals
- Current Research Summary
- Conclusion/Future Work
- Proposal Related Publications





Introduction

- Many modern computing activities are heavily interconnected
 - Stable, high-speed networks
 - Robust, single-user applications
 - Widespread consumer familiarity/comfort
- Globalization, cross-discipline and cross-organizational development teams increase need to collaborate efficiently
- Standard protocols and applications support the ability to collaborate/share documents





Introduction

- Computer Supported Collaborative Work (CSCW) relies upon foundational research
 - Database
 - Operating systems
 - Networking
 - Human-computer interaction
 - Software engineering
- Collaborative Editing Systems (CES)
 - Allow multiple users to synchronously collaborate on shared documents





CSCW and CES

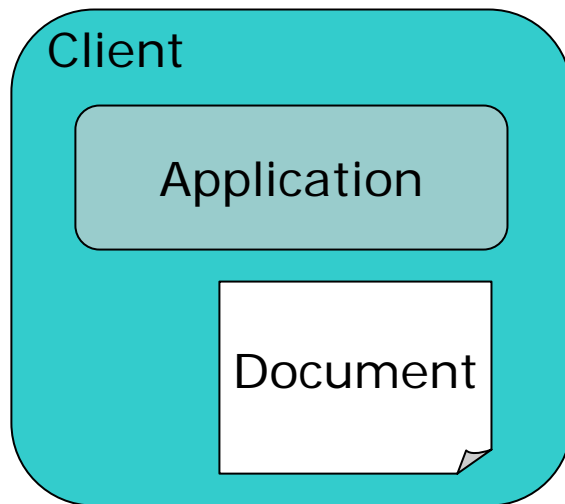
- A rich field with a history of research dating 25+ years
- Multiple foci of CSCW
 - Awareness/presence
 - Coordination
 - Integrating into existing applications
- CES a subfield of CSCW
 - Attention on consistency maintenance
 - Recent work focuses on optimistic concurrency control and solving consistency maintenance





Single-User Application

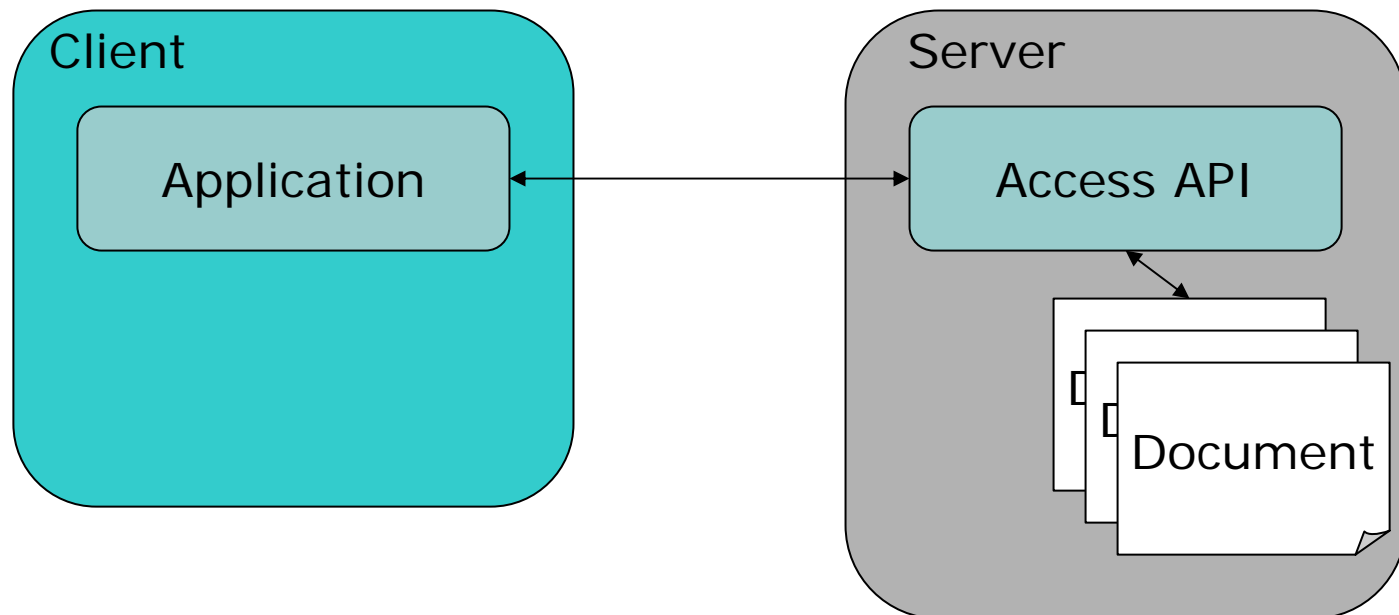
- Application and document reside on local machine





File-Server Single-User Application

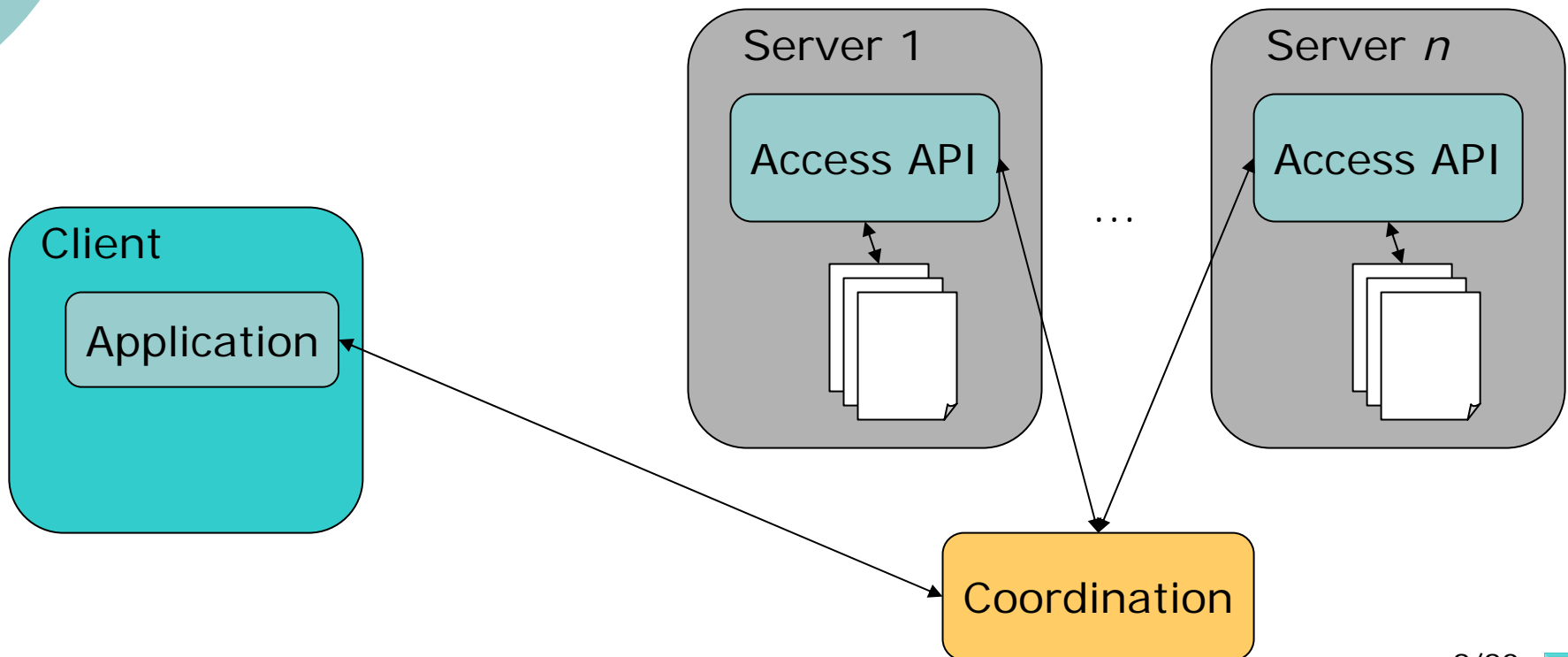
- Application resides on local machine
- Document resides on server





Distributed File-Server

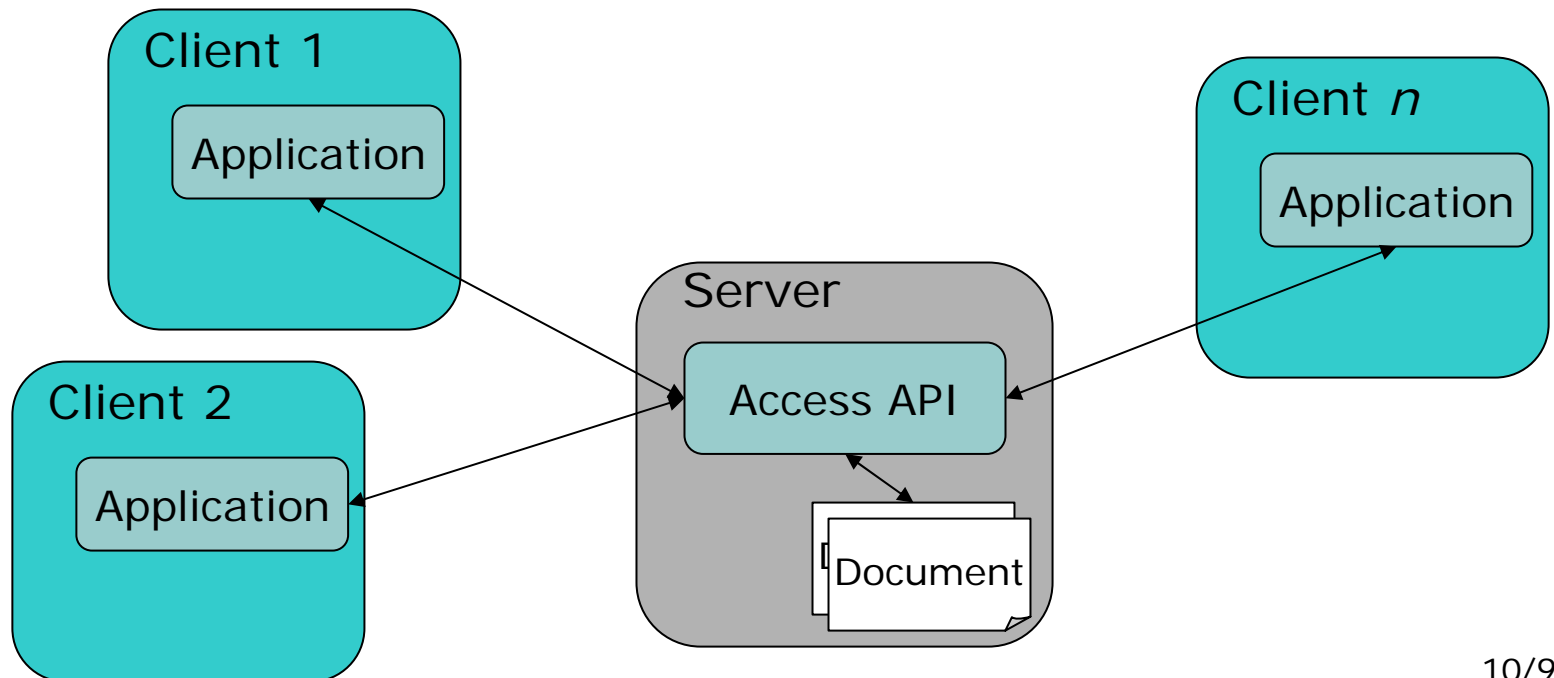
- Documents resides on multiple servers transparently to user





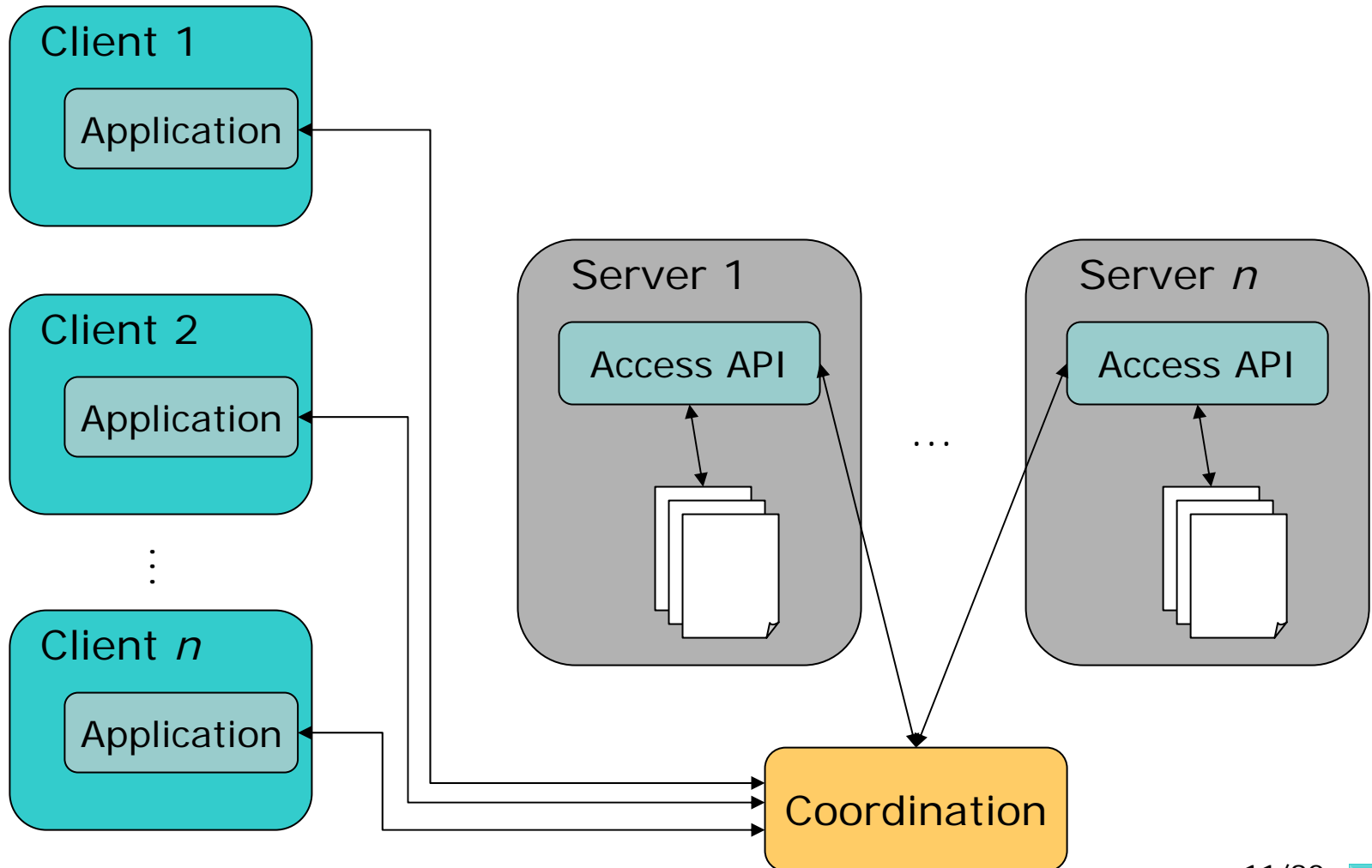
File-Server Multi-User Application

- Multiple clients access documents
- Documents reside on server





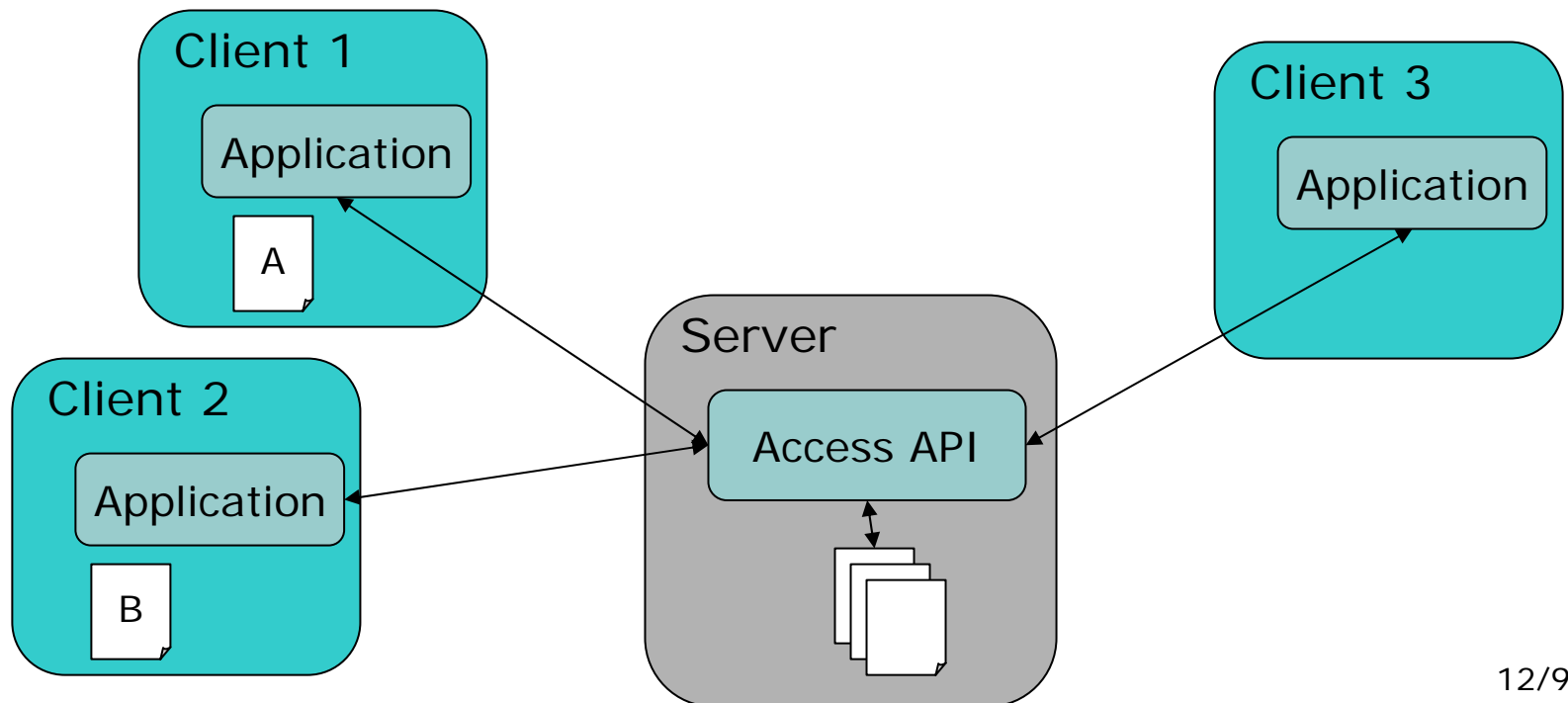
Distributed File-Server, Multi-User





Pessimistic Concurrency

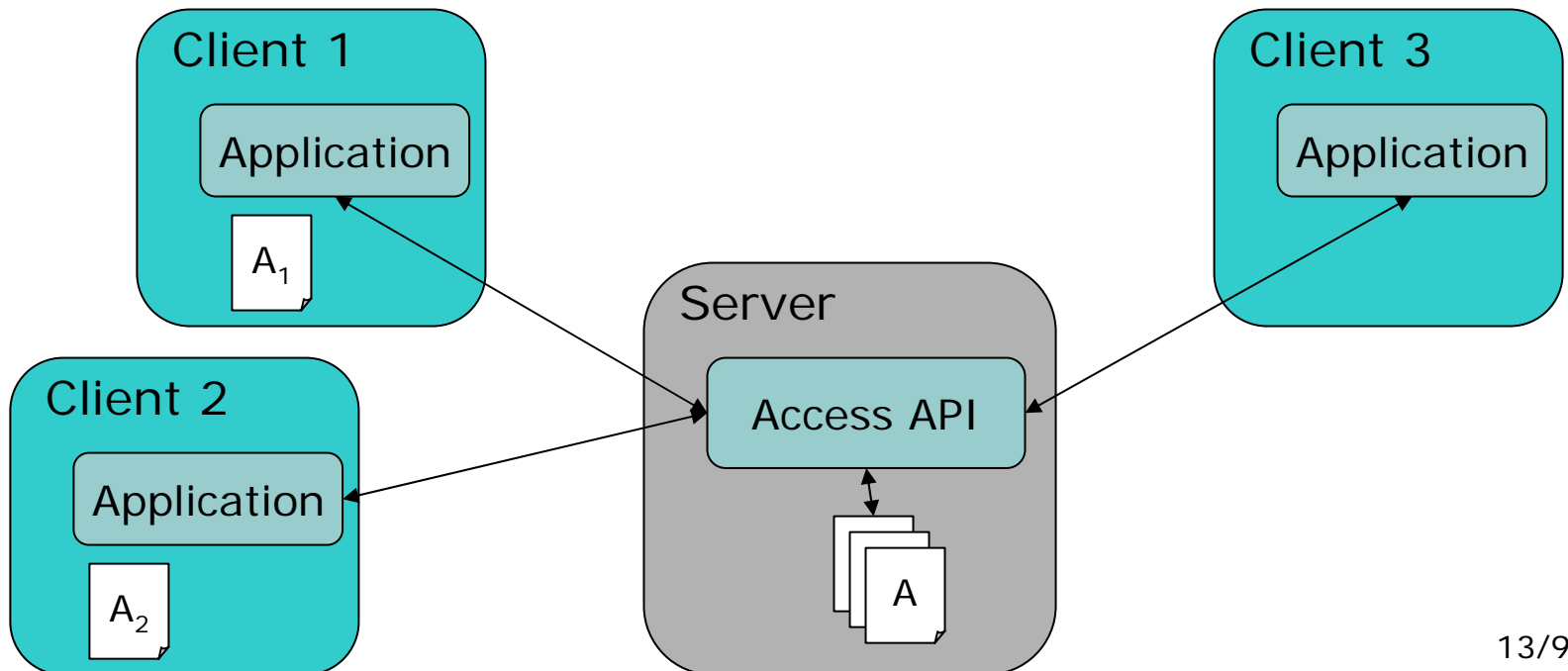
- Exclusive lock on each document (check in/out)
- Client 3 blocked from accessing documents A and B
- Reduces concurrent access





Optimistic Concurrency

- Each document copied and locally modified
- Increases concurrent access (non blocking)
- Injects consistency maintenance problem
 - Merge, multicast changes, etc.





Agenda

- Introduction
- **Motivation**
- Problem Statement
- Contributions
- Related Work
- Research Goals
- Current Research Summary
- Conclusion/Future Work
- Proposal Related Publications





Motivation

- Allow rapid response time for local changes
- Increase concurrent access (non-blocking when possible)
 - Avoid consistency maintenance problem
- Allow users to utilize their already-favored applications
 - Moving to a new, proprietary CES prohibitive
- Avoid high communication costs





Application Scenarios

- Software Engineering (Perry et al, 2001)
 - Process documents (requirements, plan, test cases) evolve
 - Various users accessing code base
 - Managers overseeing progress (reporting of activities – changes to files)
 - Testers accessing latest build
- Collaborative Document Generation
 - Multiple researchers publishing their work
 - Real-time editing that is highly interactive
 - Many revisions within a short time frame
- Computer-Aided Design (CAD)
 - Objects with layering
 - Not necessarily spatially structured





Agenda

- Introduction
- Motivation
- **Problem Statement**
- Contributions
- Related Work
- Research Goals
- Current Research Summary
- Conclusion/Future Work
- Proposal Related Publications





Problem Statement

- Clear need for collaboration
- Popular document editors remain single-user applications
- Most server repositories only support file-level access
 - Lack document format knowledge
- Lack of support for heterogeneous client/server tools





Problem Statement: Existing Editors

- MS Office, StarOffice, OpenOffice, etc. are designed to be single-user applications
- Collaborative tools (if present) merely
 - Track changes made to copies of the document (history)
 - Coordinate who has copies of the document (or when checked in/out)
 - But ultimately only offer serialized editing (not truly concurrent)
 - Result in redundant, wasted effort since they lack real-time capabilities





Problem Statement: Server Repositories

- Pessimistic Systems: RCS, VSS, etc.
 - Reduce concurrent access
 - Only view files as the atomic unit
 - Lack semantic knowledge of documents
- Optimistic Systems: CVS, etc.
 - Improve concurrency but have merge/OT problem
 - Still lack fine-granular approach and semantic knowledge of documents
- Fine-granular Systems: Coven, COOP/Orm, etc.
 - Reduce merge problem to smaller unit
 - Lack semantic knowledge of documents
 - Lock size is not dynamic (fixed at class/method)





Problem Statement: Heterogeneity

- Existing CES research typically addresses theoretical topics
 - Optimistic concurrency control assumed
 - Focus on OT or other
- In so doing, a special-purpose editor is typically utilized
 - Lacks commercial appeal
 - Application is the same among all clients
- Li (2004) address the heterogeneity issue, but only on the client side, not the server side





Agenda

- Introduction
- Motivation
- Problem Statement
- **Contributions**
- Related Work
- Research Goals
- Current Research Summary
- Conclusion/Future Work
- Proposal Related Publications





Contributions

- Explore algorithms and functionality necessary to maximize concurrent access to shared documents
- Develop an architecture that allows for a heterogeneous set of client editing software to connect with a heterogeneous set of server document repositories
- Develop a testbed prototype system of our architecture that is responsive to users' actions and minimizes communication costs





Contributions: Theory

- Managing the complexity of concurrent access to the shared documents, utilizing cache techniques to minimize communication costs and developing notification policies are essential.
- Efficient multi-granular locking algorithms (maximize concurrent access, minimize communication costs)
- Efficient data structures to store the document state and ownership information in the distributed system
- Efficiency analysis of the algorithms and data structures to ensure real-time responsiveness for users





Contributions: Architecture

- Definition of necessary and sufficient set of interfaces (API) that cover the core functionality of a technology-independent collaborative editing system; this will involve improving our existing version of the heterogeneous architecture
- Establishing a mapping protocol by which client components may hook/listen into existing client editing tools, capture edit and viewspace event changes
- Establishing a mapping protocol by which server components may hook into existing server document repository tools, sending check-in and check-out events via proxy to these tools





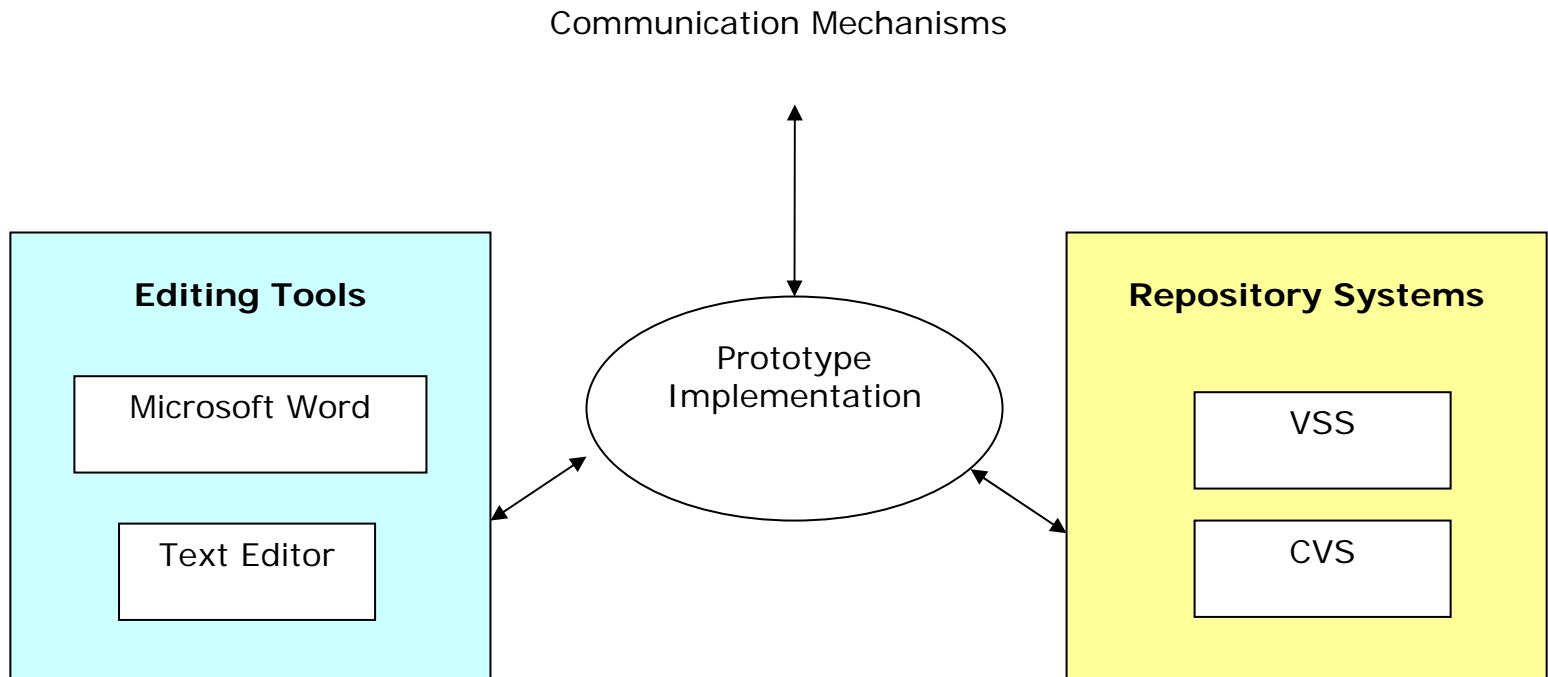
Contributions: Prototype

- The prototype developed will demonstrate the viability of the proposed algorithms and architecture and will
 - Implement client hooks into at least two existing client editing technologies
 - Implement server proxies into at least two existing server repository systems





Prototype: Proposed Technologies





Agenda

- Introduction
- Motivation
- Problem Statement
- Contributions
- **Related Work**
- Research Goals
- Current Research Summary
- Conclusion/Future Work
- Proposal Related Publications





Related Work: Theory

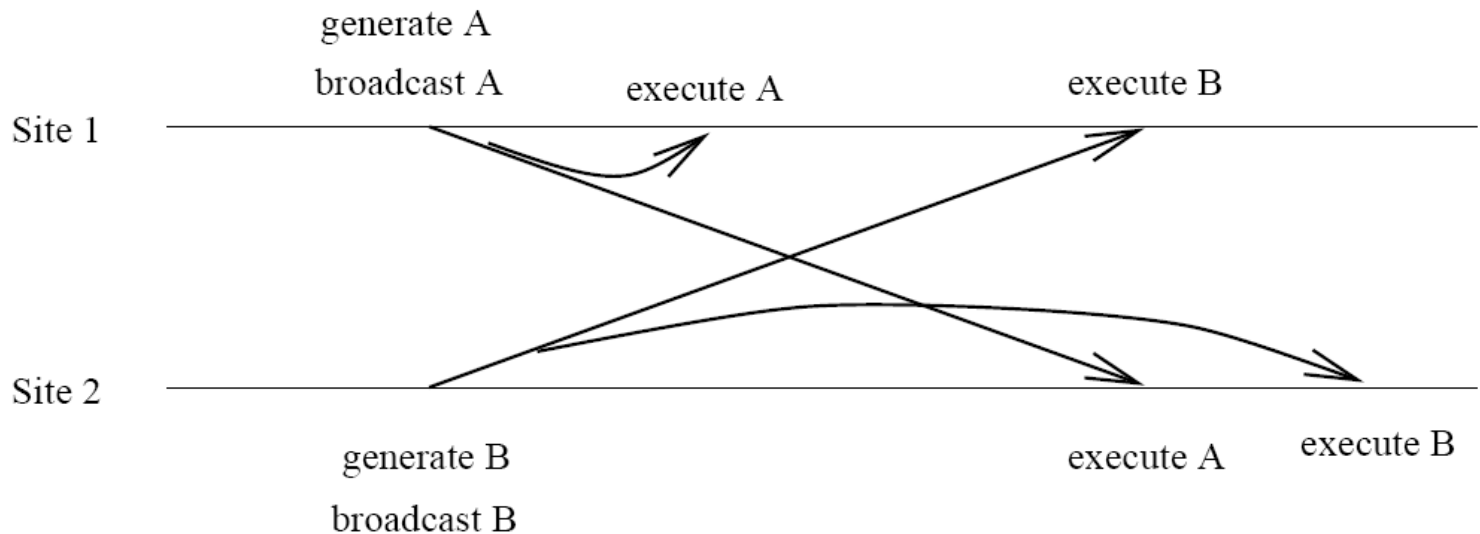
- The CCI model (Sun, 1998)
 - Convergence – if all events are enacted, then the copies should all be the same
 - Causality preservation – if O_1 precedes O_2 , then O_1 will be executed before O_2 on all copies
 - Intention preservation – operations will execute intended change locally and on all copies
- Achieving “Undo” also critically important





Ordered Broadcast

- Ensures proper ordering (causality preservation)
- But latency remains a problem
- And intention preservation not guaranteed





Concurrency Control

- Pessimistic (lock-based)
- Optimistic (OT, merge, etc.)
- Hybrid (Greenburg, 1994)

Level of Optimism	Can manipulate the object while waiting for its lock	Can release the changed object while waiting for its lock
Non-optimistic	No	No
Semi-optimistic	Yes	No
Fully-optimistic	Yes	Yes





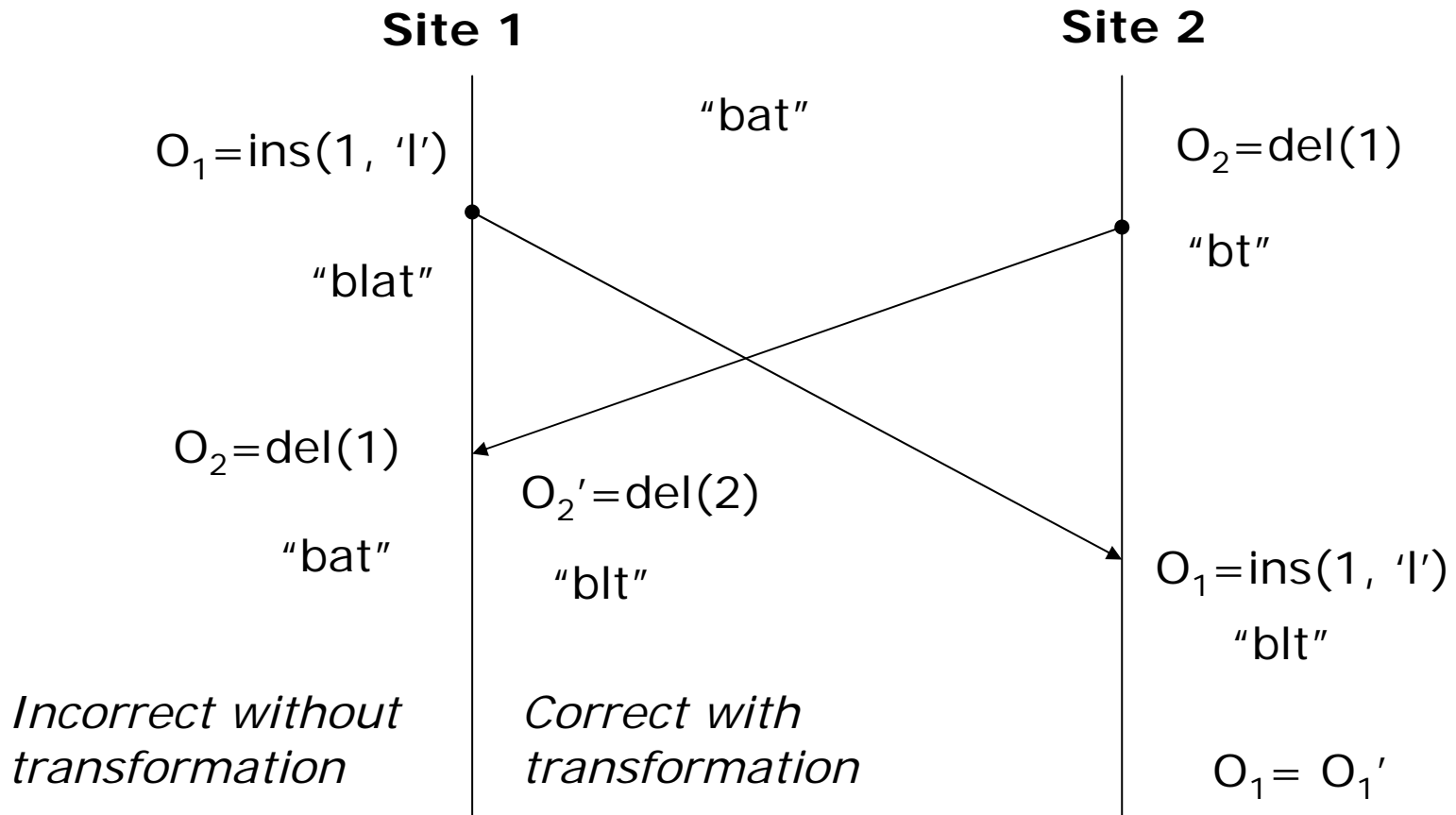
Operational Transformation

- Operational Transformation (OT)
 - Transforms events received from other clients into the document space
 - Attempts to ensure intention preservation





OT Example

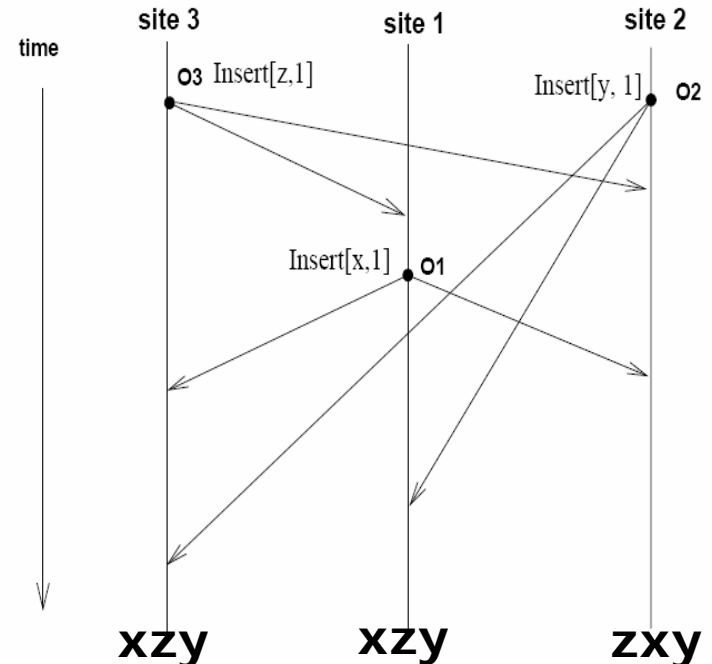




OT Problems

- There exist problems in preserving the CCI model using OT
- More complicated OT algorithms have been developed over the past 16+ years

- dOPT problem (Ellis 1989)
 - If tie on position, shift lower site's op
 - $O_2 \parallel O_3$ and $O_1 \parallel O_2$
 - x must precede y which must precede z which must precede x





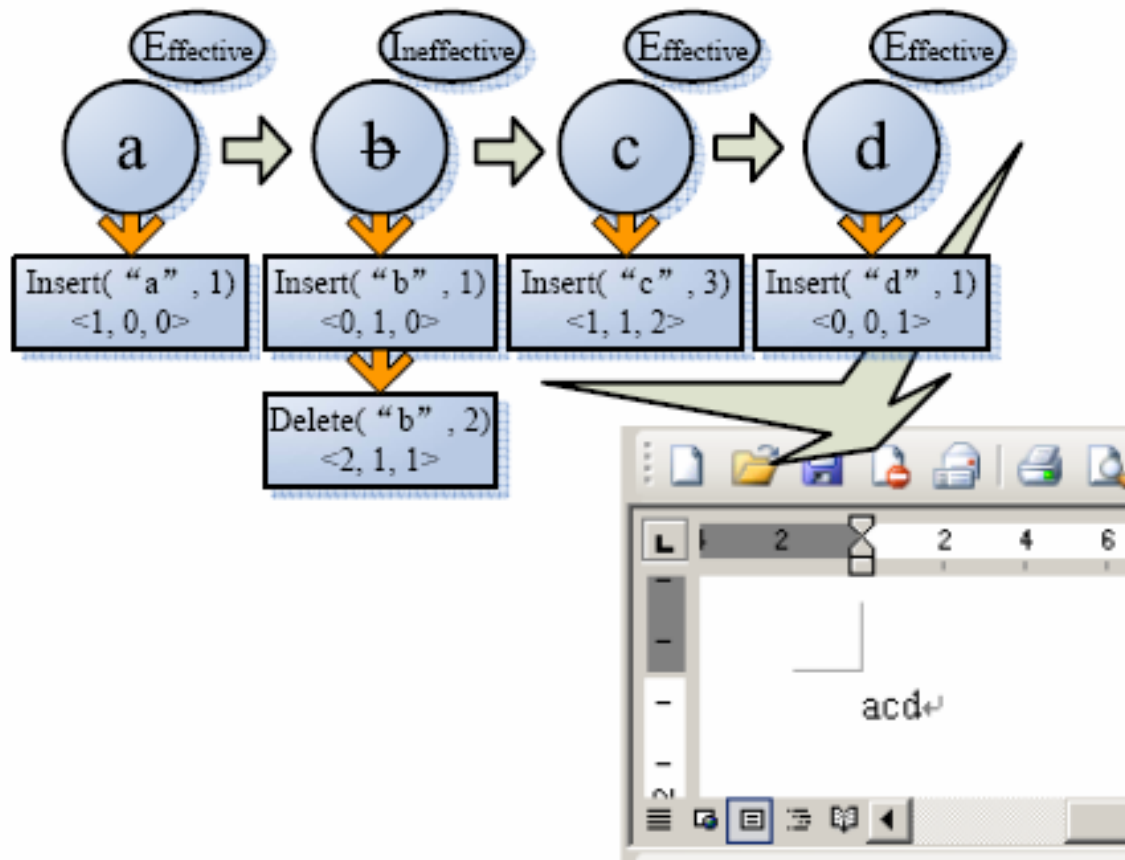
Other Solutions

- More elaborate work in OT
 - (Li and Li, 2005) propose “Landmark-Based Transformation” (LBT) to avoid the intention preservation problems of dOPT
 - Must *integrate* new operation into history buffer in $O(n)$ where n = size of history buffer
- Mark & Retrace technique (Gu et al, 2005)
 - Keep all operations in linear state list so as to transform state space (not operation)
 - Mark as Effective/Ineffective
 - Memory intensive
 - Reasonable runtime but does incur work to calculate effect of operation on state of the document
- Logging/Replay (Sun, 1998)





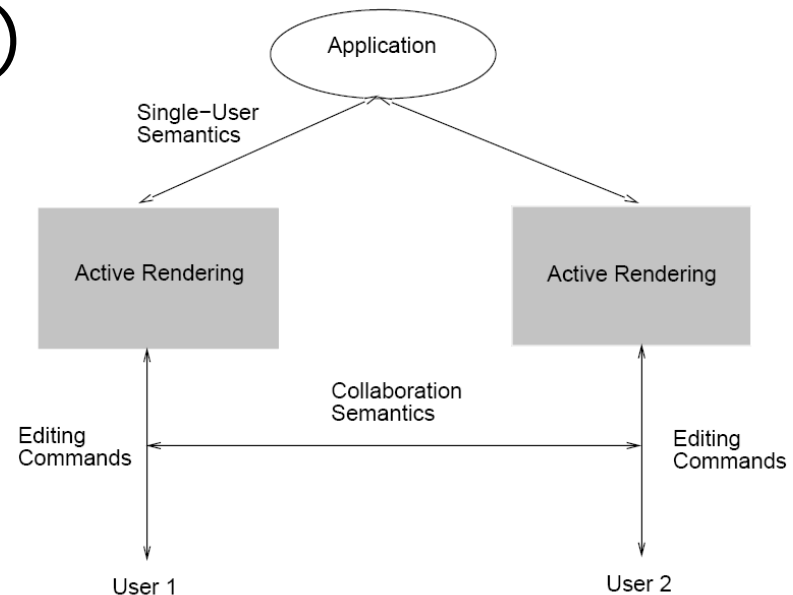
Mark and Retrace (Gu et al, 2005)





Related Work: Architectures

- Centralized, screen-sharing
- Replicated, broadcast
- Hybrid, dynamic (Chung, 2004)
- (Dewan, 1999)





Related Work: Architectures

- Fine-grain locking (Chu-Carroll et al, 2002)(Magnusson, 1993)
- Notification Mechanisms (Arregui et al, 2001)
- Awareness (Dewan, 1999)
 - Heterogeneous binding of awareness groupware (Bharadwaj, 2003)
- Web-Services (Younas, 2003)





Collaborative Coordination

○ Van der Hoek (2004)

	<i>Conceptual Visualization</i>	<i>Strengths</i>	<i>Weaknesses</i>
<i>Formal process-based coordination</i>		Scalable; Control; Insulation from other activities; Group-centric	Resynchronization problems; Insulation becomes isolation
<i>Informal, awareness-based coordination</i>		Flexible; Promotes synergy; Raises awareness; User-centric	Not scalable; Requires extensive human intermediation
<i>Continuous coordination</i>		<i>Expected to be the strengths of both formal and informal coordination</i>	<i>To be discovered by future research</i>





Related Work: Existing Systems

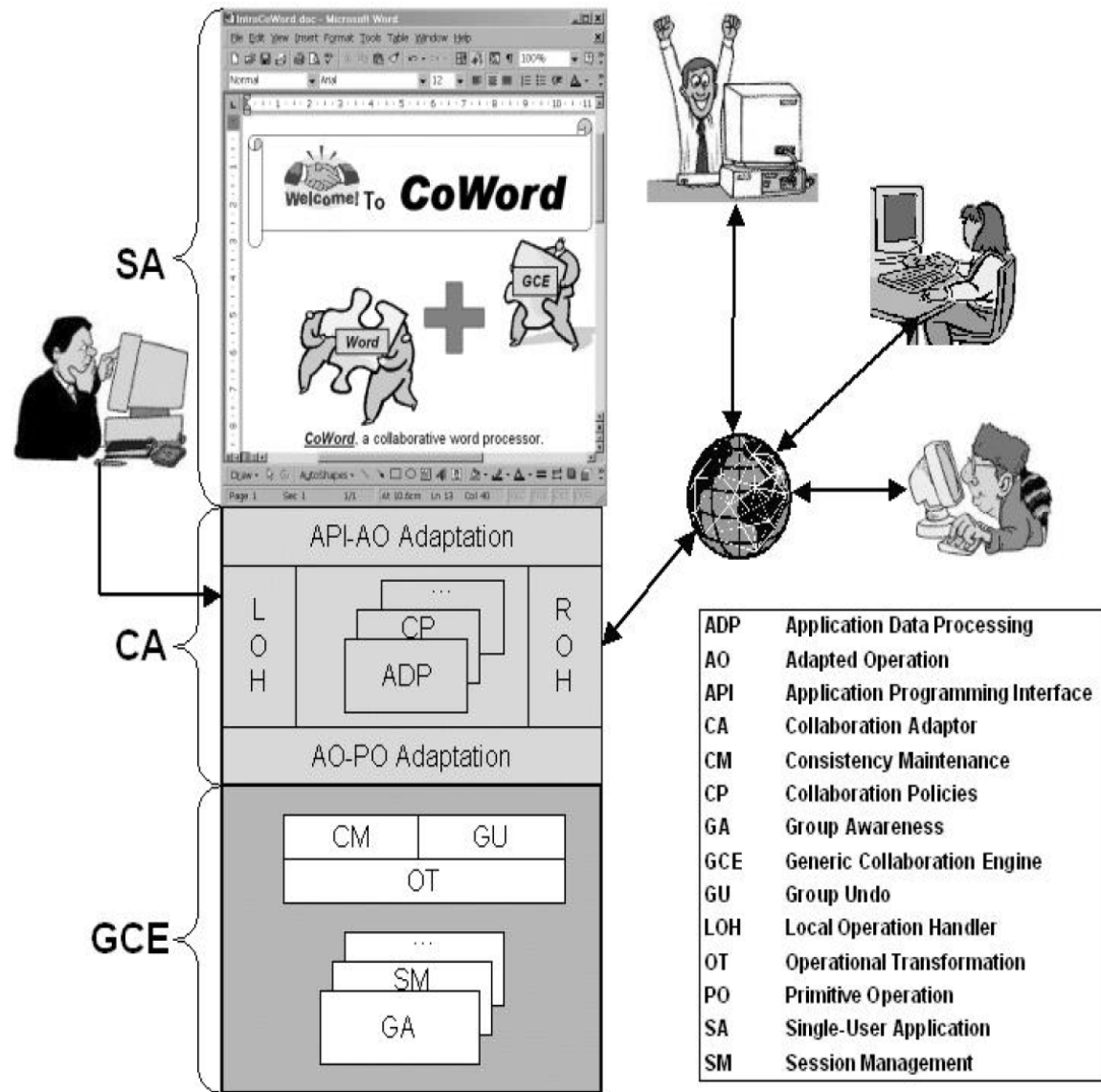
- Some commercial products exist
 - LiveMeeting (Microsoft)
 - Groove, SharePoint
 - IDEs – Visual Studio, Eclipse, JBuilder
- Research projects
 - DistEdit (Knister, 1990)
 - Coven (Chu-Carroll, 2000)
 - Grove/Reduce (Sun, 1998)
 - Palantir (Sarma, 2003)
 - CoWord, CoPPT (Xia, 2004)





CoWord

- Xia
2004





Agenda

- Introduction
- Motivation
- Problem Statement
- Contributions
- Related Work
- **Research Goals**
- Current Research Summary
- Conclusion/Future Work
- Proposal Related Publications





Research Goals

We propose the following three research goals

1. Develop a set of algorithms and services that maximize concurrent access to shared documents while minimizing communication costs. The outcomes of this goal will provide a solid theoretical foundation and framework for the development of the architecture and prototype (goals 2 and 3).
2. Develop an architecture wherein existing client editing systems can connect to existing server document repositories that efficiently support the algorithms (goal 1). This architecture will allow users to synchronously and asynchronously edit documents, subscribe for notification upon document change through varied communication mechanisms (email, IM, etc.), and reflect an open-systems approach whereby future server and client tools can be integrated easily into the architecture (flexibility in design).
3. Develop a prototype system that demonstrates the effectiveness of the architecture developed in goal 2. This prototype will connect a heterogeneous set of client editors residing on the user machine to a heterogeneous set of document repositories residing on distributed server machines. We will leverage existing systems if possible.





Goals: Details

○ Theoretical

- Extend our document to tree algorithm to support general trees (not limited to binary trees)
- Extend our deadlock-free lock management algorithms to support general trees (not limited to binary trees)
- Research how to best combine locking and OT solutions to maximize concurrent access and minimizing communication costs
- Generate data structures that support the efficient storage of ownership information in a distributed fashion (not necessarily at a central server) to minimize communication cost when updating this information
- Generate technology-independent event transformation algorithms that may be applied to specific client and server technology transformations

○ Architecture

- Define an architecture such that technology-specific events are propagated via technology-independent components to heterogeneous client and server technologies.
- Determine the proper location of the algorithms and data structures defined in the theoretical goals to minimize communication cost among the components of the architecture.

○ Prototype

- Implement the components defined in the architecture goals; Web services will be utilized to pass client-side events to the coordination server. This coordination server will then pass events to and from server-side repositories
- Implement two client-side technology-specific event transformations (MS Word and a text editor) and pass these events to the coordination server
- Implement two server-side technology-specific event transformations (MS Visual Source Safe and CVS)
- Analyze the communication and computation effectiveness of the system





Agenda

- Introduction
- Motivation
- Problem Statement
- Contributions
- Related Work
- Research Goals
- **Current Research Summary**
- Conclusion/Future Work
- Proposal Related Publications





Current Research Summary

- Theoretical foundations for maximizing concurrency while minimizing communication costs
- Heterogeneous architecture for CES
- Prototype system to validate architecture





Current Research Summary

- **Theoretical foundations for maximizing concurrency while minimizing communication costs**
- Heterogeneous architecture for CES
- Prototype system to validate architecture





Theoretical Foundation

- Document partitioning
- Deadlock-free, Multi-granular Locking Algorithms
 - Insert User
 - Remove User





Document Partitioning: Tree Structure

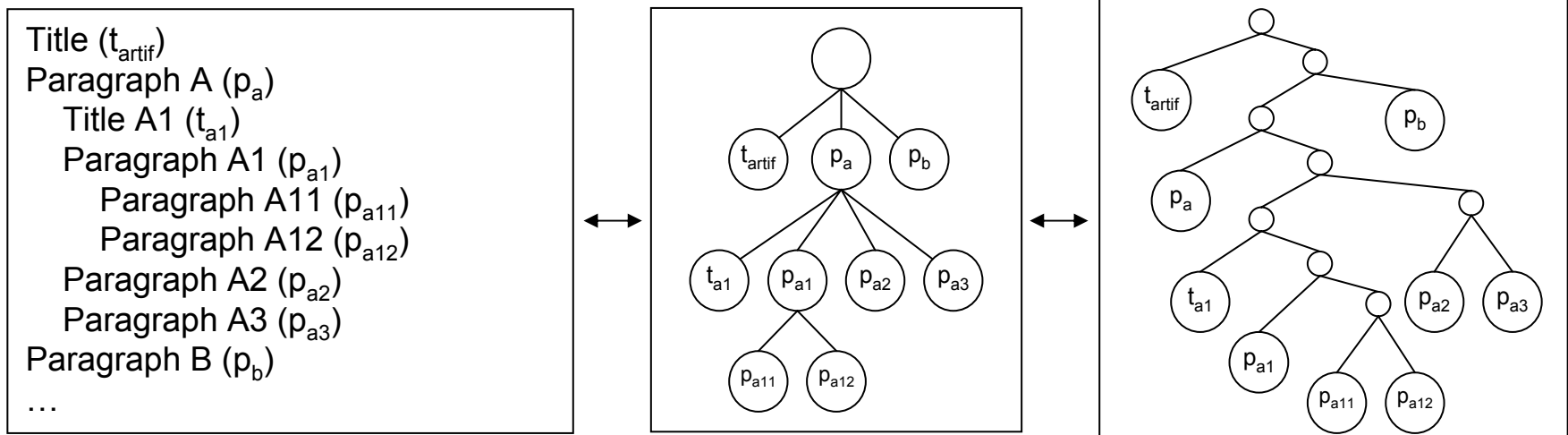
- Structure denotes sections and subsections
- Only leaf nodes contain satellite data
- All access requests from client will reference a specific leaf node (as the client knows only of document content, not tree)
- Nodes store ownership/lock information
 - If a section of the document is owned by a user u , then all subsections (i.e. child nodes in the tree) are also owned by the same user u .
 - Similarly, if a section of the document is not owned by any user, then all subsections (i.e. child nodes in the tree) are also not owned by any.





Document Partitioning: Mappings

- It is possible to map a document to a tree and from this tree to a binary tree
- Without loss of generality, our algorithms work on this binary tree





Dynamic Locking Algorithm

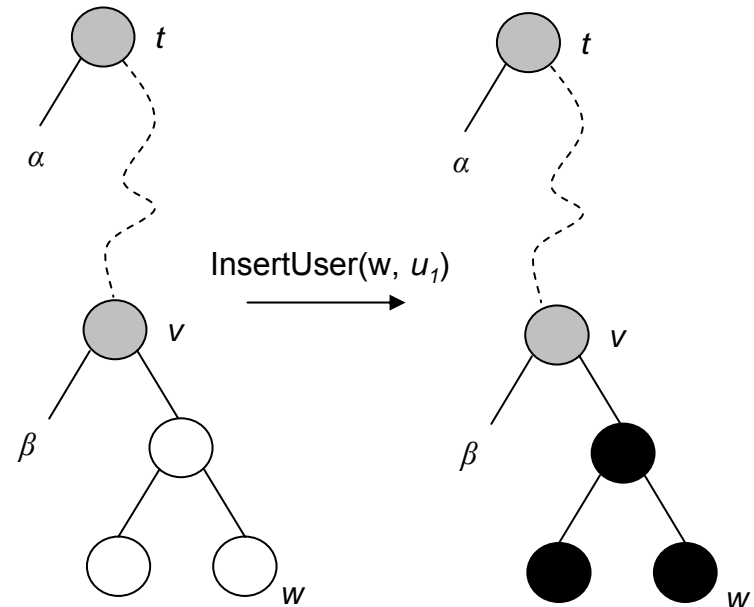
- Lock largest sub-tree possible
- Demote lock if request creates contention
- Promote lock if contention is removed
- Deadlock-free
 - Always traverse top to bottom
 - Handshake lock
 - Acquire node
 - Acquire node's child
 - Release node
- Can integrate OT to share larger subsections if desired





Insert User without Demotion

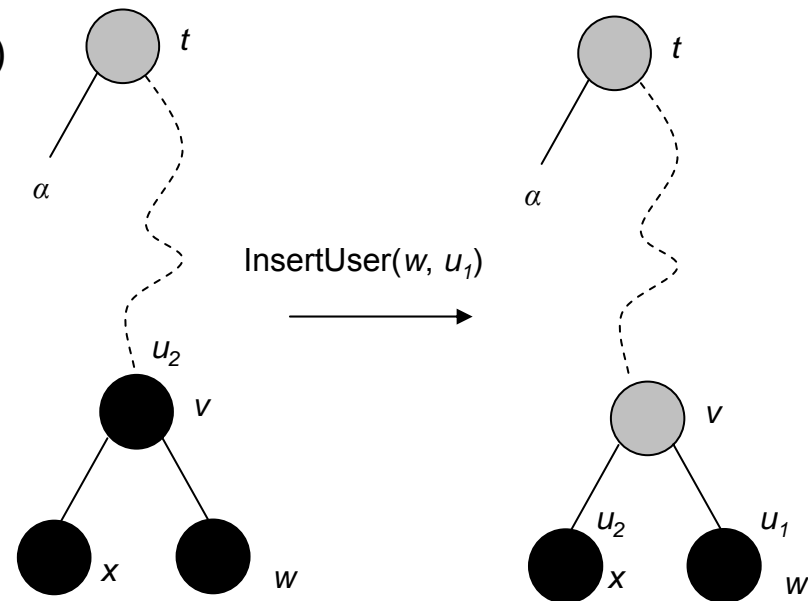
- User u_i requests lock on leaf node w
- All nodes in the path from t to v must be grey
 - If white, lock higher
 - If black, lock fails
- Increase grey-color in path





Insert User with Demotion

- At times, demotion is necessary
 - When the top-to-bottom access encounters a black node
 - Push the ownership “down” (if a leaf, fail)
 - Requires that we maintain original request (i.e. why this node black)
 - Paint node grey
 - Recursion if we push along the same path to requested node





InsertUser

```

InsertUser(w, ui)
  if w.owner ≠ ui
    RecurseInsert(root, w, ui)

RecurseInsert(n, w, ui)
  if n.color = white // ensures we always acquire largest lock
    then      SetOwner(n, ui, w)
  else if n is a leaf node // a leaf node but not white, so failed insert (i.e. we can't demote a leaf)
    then      RecurseRemove(root, w, ui) // undo the false insert's effect on greyColors
    return failure
  else if n.color = grey // keep looking down
    then      n.greyColor = n.greyColor + 1
    RecurseInsert(NextInPath(n, w), w, ui)
  else // color of node in path to w is black, so we must demote it
    b = NextInPath(n, w)
    a = NextInPath(n, n.originalRequest)
    SetOwner (a, n.owner, n.originalRequest) // demote n to a
    n.color = grey
    n.greyColor = 2
    if a ≠ b
      then      SetOwner (b, ui, w) // the nodes are in separate paths
      else      RecurseInsert(b, w, ui) // the nodes are in the same path

SetOwner(w, ui, r)
  w.color = black
  w.owner = ui
  w.originalRequest = r

```





Remove User

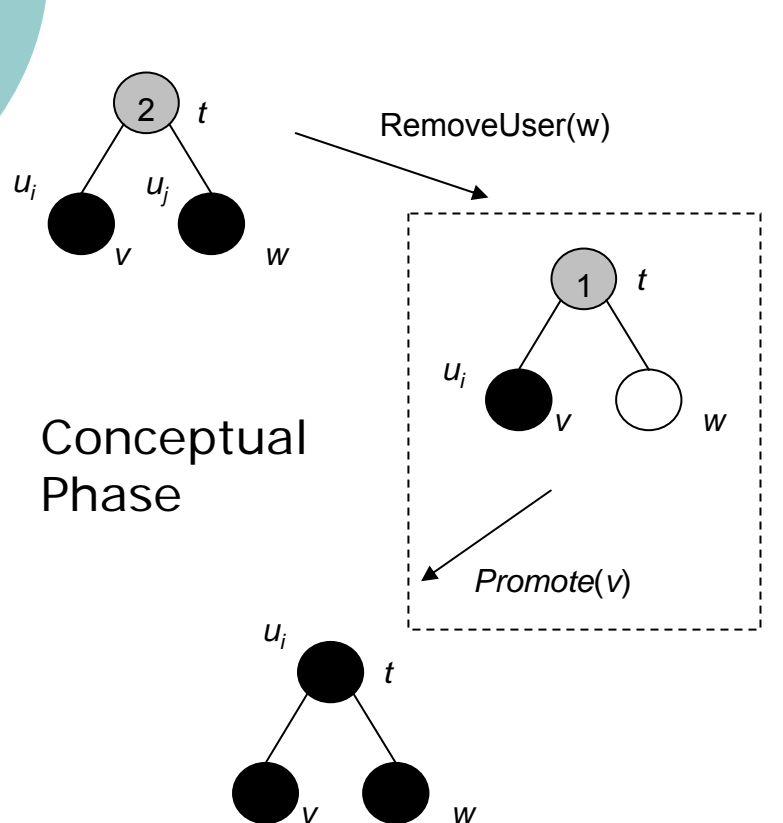
- User u_i releases lock on node w
- All nodes in the path from t to v must be grey
 - If white, lock higher
 - If black, release higher
- Decrease grey-color in path
- Promote if possible when grey-color goes from 2 to 1



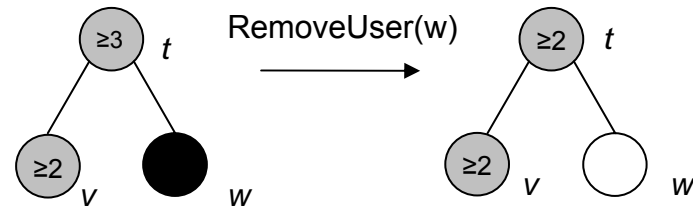


Remove User Cases

○ With promotion



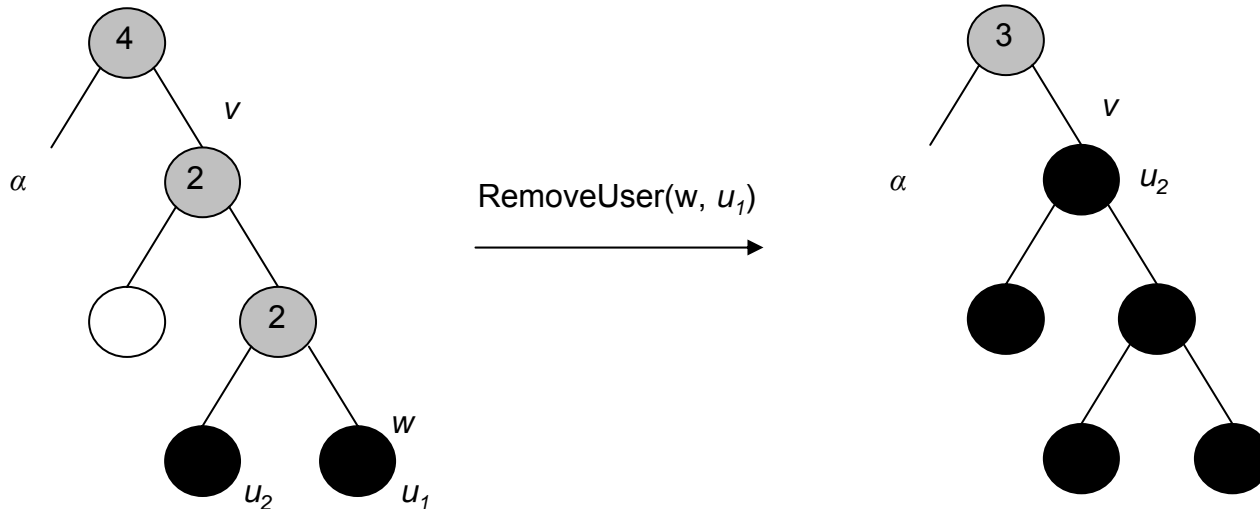
○ Without promotion





Remove User Promotion

When grey-color reduces from 2 to 1,
promote





RemoveUser

```
RemoveUser(w, ui)
  if w.owner = ui
    then RecurseRemove(root, w, ui)
         RecurseRemove(n, w, ui)
  if n.color = black and n.owner = ui
    then ReleaseOwner(n)
  else if n.color = grey // keep looking down
    then n.greyColor = n.greyColor - 1
         if n.greyColor = 1 and w.sibling.color = black // sibling promotion priority
           then SetOwner(n, w.sibling.owner, w.sibling)
         else if n.greyColor = 1 and w.color = black // w must be all that remains
           then SetOwner(n, w.owner, w)
         else if n.greyColor = 0 and // paint n white as w and w.sibling are white
           then ReleaseOwner(n)
         else RecurseRemove(NextInPath(n, w), w, ui)
```

```
ReleaseOwner(w)
  w.color = white
  w.owner = NIL
  w.originalRequest = NIL
```





RemoveUser Special Cases

- If an insertion failed, then we invoke a RemoveUser to reduce the artificially-inflated grey-count along the path from root to desired node
- It is possible that legitimate removals occurred between the time of the failed insertion and the concomitant removal (to compensate for the failed insertion)
- Consequently, the grey-count of a node can fall to 1
 - Promote sibling or the node associated with the failed insert
- Additionally, the grey-count of a node could fall to 0
 - Promote nothing (everything is now removed below this node)
 - Paint this node white





Analysis

- RemoveUser makes a single pass of the tree from top to bottom
 - $O(h)$ where h is equal to the height of the tree.
- InsertUser can fail and require an “undo” of the artificially-inflated grey-colors (via a call to RemoveUser)
 - Thus at most two passes of the tree from top to bottom
 - Also $O(h)$ where h is equal to the height of the tree
- Promotion occurs in $O(1)$ because we know the sibling is to be promoted (or in the special cases, the failed insert node is promoted or nothing is promoted)





Current Research Summary

- Theoretical foundations for maximizing concurrency while minimizing communication costs
- **Heterogeneous architecture for CES**
- Prototype system to validate architecture





Heterogeneous Architecture

- Proxy-based configuration management
- Multi-granular locking simulation used to validate architecture





Architecture

- Connects to existing, heterogeneous client applications
 - MS Word, OpenOffice, JavaBeans, MS Studio, etc.
- Connects to existing, heterogeneous server document repositories
 - VSS, CVS, RCS, etc.
- Synchronous and asynchronous change notification
 - Email, IM, etc.





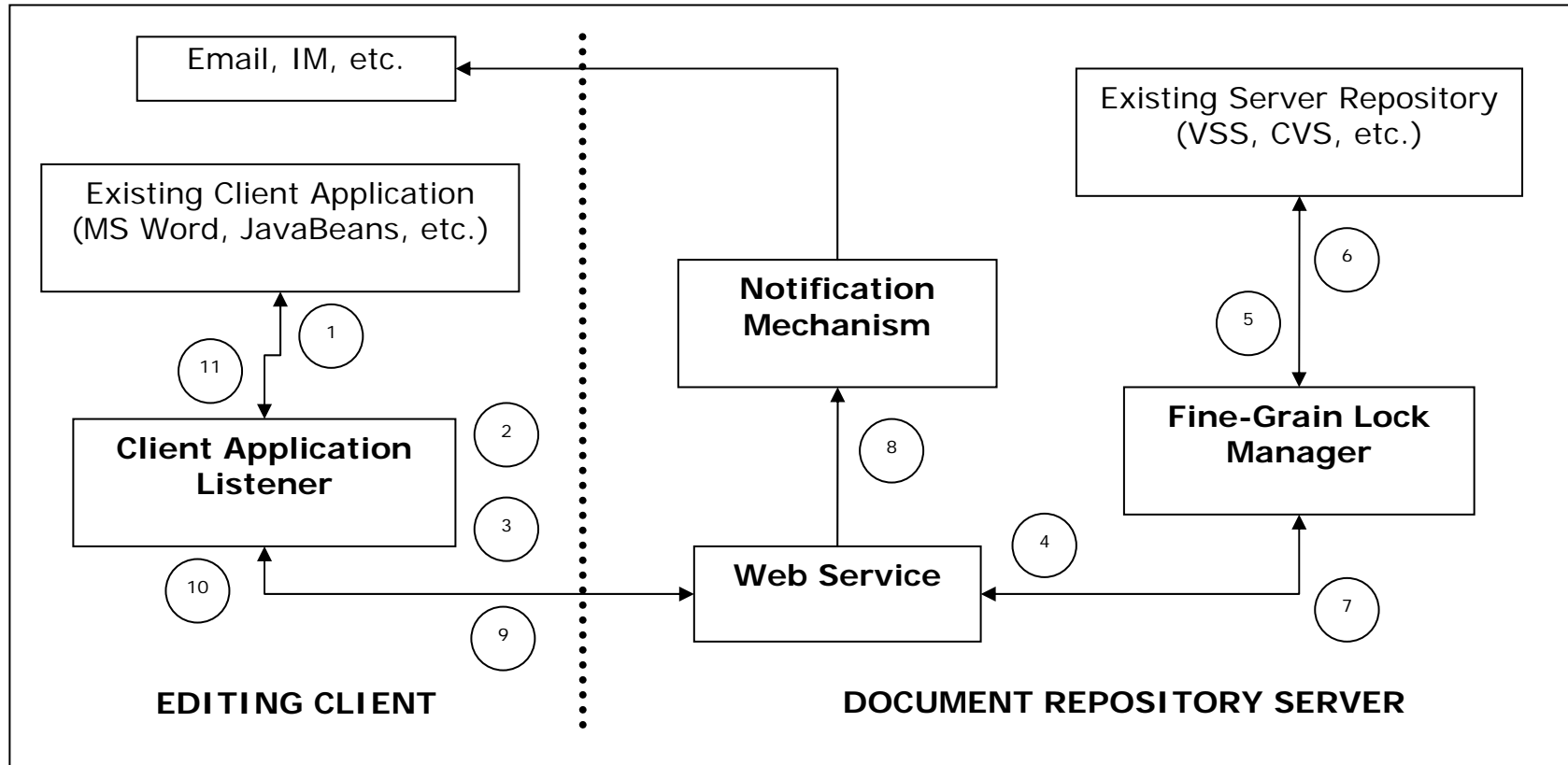
Components

- Client Application Listener
 - Hooks to existing editing software
 - Caches changes
 - Sends changes to Web Service
- Web Service
 - Publishes API – check-in, check-out, subscribe
 - Manages subscriptions and connected users
- Fine-Grain Lock Manager
 - Connects to existing document repository (CMS)
 - Intercepts check-in and check-out messages
 - Acts as check-in, check-out proxy
 - Adds fine-grain locking
- Notification Mechanism
 - Communicates to Email, IM, etc. to notify users of changes based upon their subscription preferences





Open-system architecture





Events

1. State update message (user edit of artifact) is sent to the Client Application Listener
2. The Client Listener caches the change
3. When the cache must be flushed changes are sent to the Web Service on the server
4. The Web Service sends updates to the Fine-Grain Lock Manager
5. Fine-Grain Lock Manager updates its data store and sends the check-out or check-in message to the existing Server Repository.
6. The Server Repository confirms update of the artifact to the Fine-Grain Lock Manager
7. The Fine-Grain Lock Manager notifies the Web Service component
8. For each client subscribed for notification concerning this document being changed, the Web Service component sends a message to the Notification Mechanism
9. The Web Service component selectively broadcasts change notifications to each client interested in the change
10. The Client Application Listener will receive the update notification and cache it if the user is not currently viewing the updated section
11. Consistency is maintained as the user views the content of the shared document (cache flushed to client application).





Fine-Grain Lock Manager Proxy

- Situated between the Web Service component and the existing CMS
- Intercepts check-in and check-out requests
- Maintains state of who has which section of each document
- Checks-in and checks-out via proxy
- CMS is unchanged
- Adds ability to do fine-grain locking





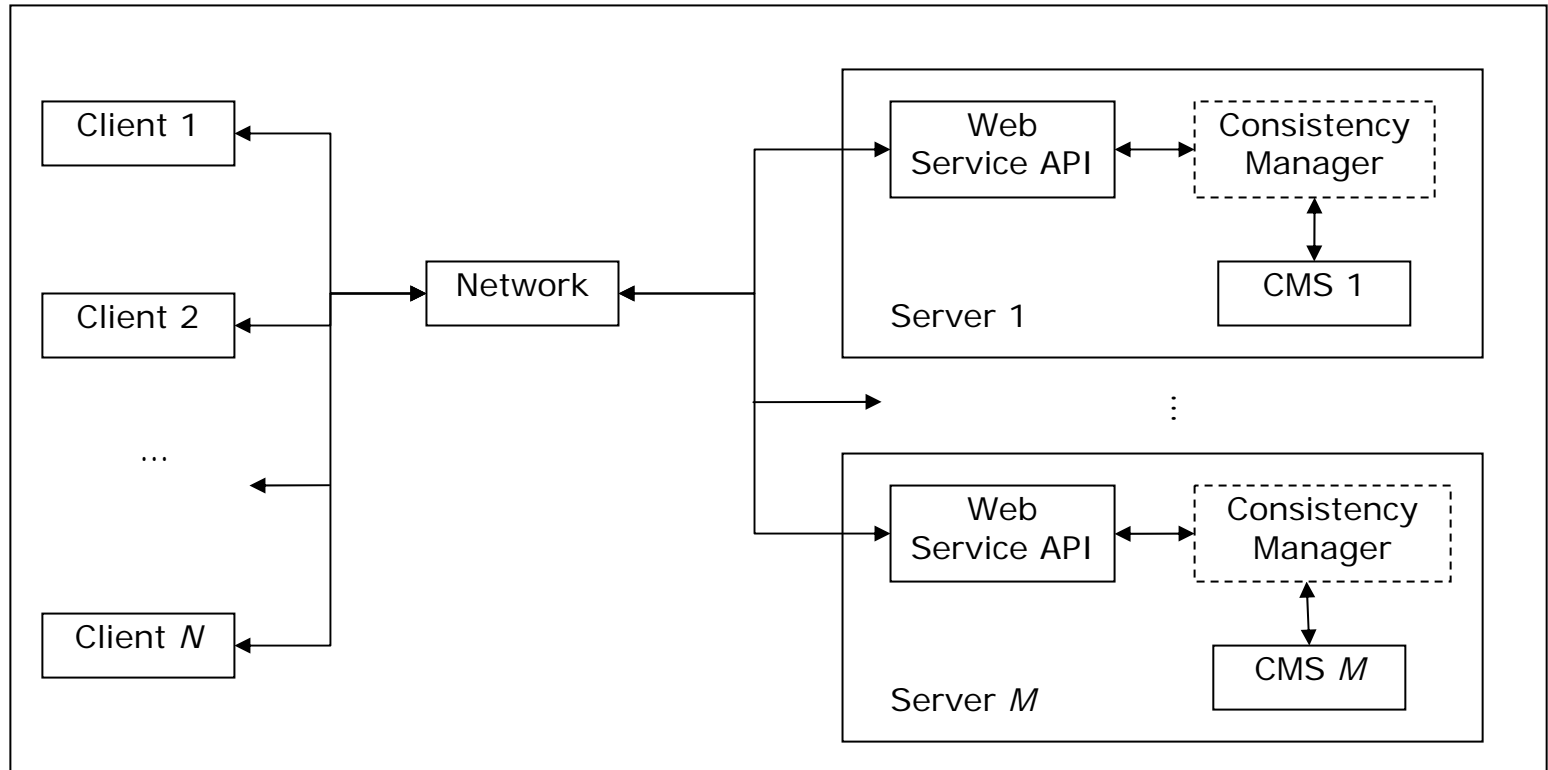
Simulation

- Validates our architecture
- Utilizes DEVS Java (discrete event simulation engine)
- 2 configurations
 - Traditional (no lock manager on server)
 - Lock Manager Proxy (with fine-grain locking)





Simulation Configuration





Simulation and Results

- 9 test runs with various configurations of clients and servers
- Improvement of fine-grain locking over non-granular ranged from 50% to 78%
- As expected, decreasing the unit locked decreases the collision rate among clients





Current Research Summary

- Theoretical foundations for maximizing concurrency while minimizing communication costs
- Heterogeneous architecture for CES
- **Prototype system to validate architecture**





Prototype Development

- Peer-to-Peer Communication Analysis
- Hooking into Existing Editors
- Implement Algorithms and analyze performance





Peer-to-Peer Prototype

- Simple text editor with “chat”
- Allowed for locking on a per-line basis of the document
- Peer-to-peer utilized to avoid central server bottleneck for communication
- Allowed new users to join the collaboration late and receive most up-to-date version of document
- Broadcast change on a per-edit basis (no caching)
- Communication costs analyzed





Peer-to-Peer Screenshot

The screenshot shows a window titled "Synch Edit 1.0 ID = -815206130". The window is divided into two main sections. The left section is a code editor with a yellow background, displaying the following C++ code:

```
File
Number of Participants: 2
C:\Documents and Settings\Jon A Preston\Desktop\doc1.txt
#include <stdio.h>

void main (int argc, char** argv)
{
    int i;

    for(i=0; i < 100; i++)
    {
        cout << "The variable 'i' is = " << i << endl;
        i += 2;
    }
}
```

The right section is a chat log with a "Send" button. The chat log contains the following messages:

<Jon Preston> Ok - we're both in now
<David Gilmour> Great - thanks for starting the code up
<Jon Preston> Now - what do we need to edit next?
<David Gilmour> Let's run it to see output first

Below the chat log, there is a "Send" button. The chat log also shows the following output:

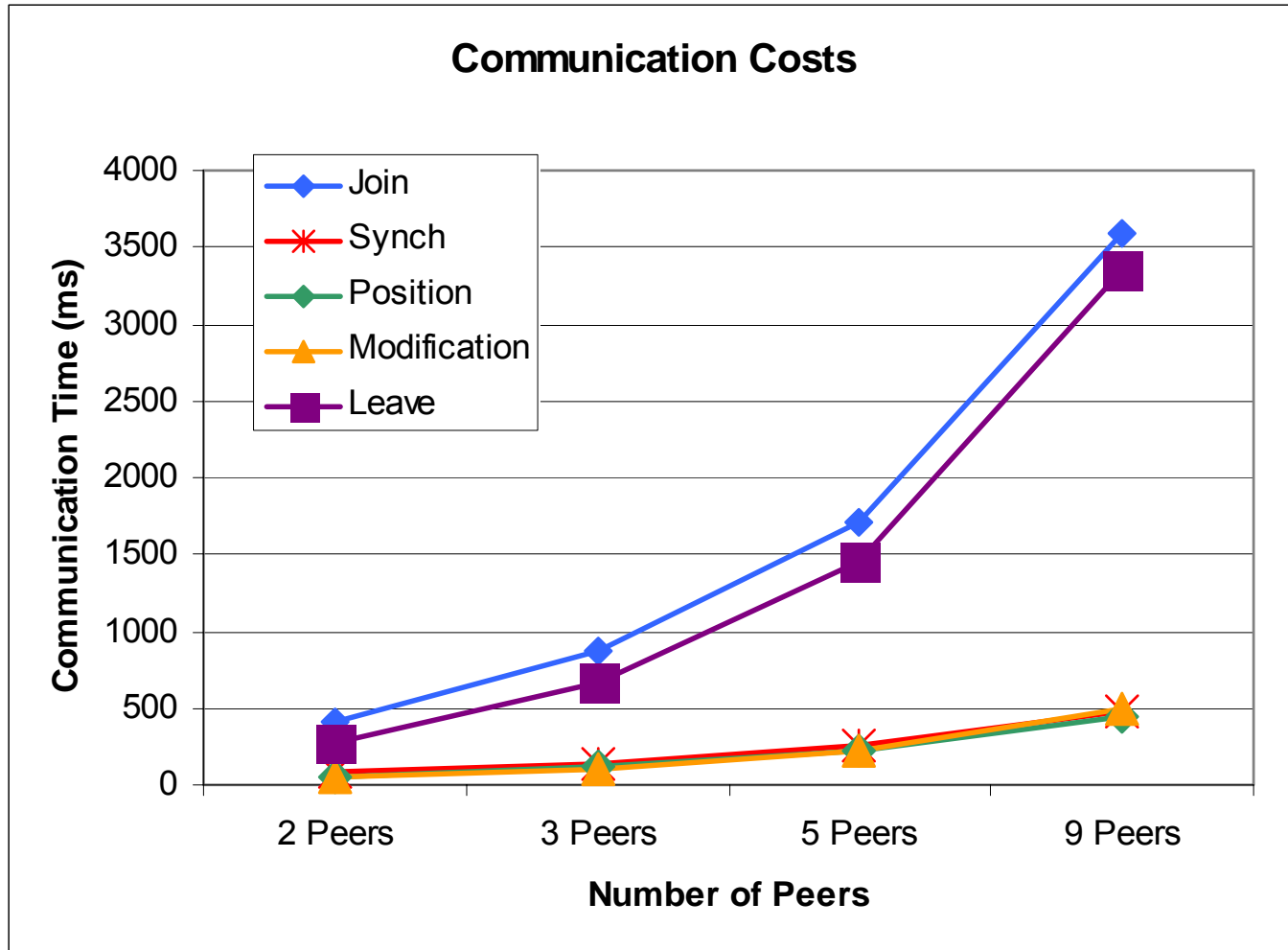
<David Gilmour> 155 0
<Jon Preston> on line -1
<Jon Preston> on line 10
<David Gilmour> on line 11
<David Gilmour> on line 10
<David Gilmour> on line 9
<David Gilmour> on line 10

At the bottom of the window, there is a status bar with the following text: "lower", "shift off", "Current Line: 11", and "OWNED".





Peer-to-Peer Editor Results





Hooking Into Existing Editors

- Each client and server application/repository must have a custom hook
 - Listen for events in existing application
 - Map to meta-event to support heterogeneity
- Preliminary research indicates adding capability to existing systems easily achieved
 - Eclipse and Visual Studio 2005 support plug-ins
 - MS Office Macros/VBA and OpenOffice (open source)
 - Handle OS-level events (WinSpy, Spy++)





Agenda

- Introduction
- Motivation
- Problem Statement
- Contributions
- Related Work
- Research Goals
- Current Research Summary
- **Conclusion/Future Work**
- Proposal Related Publications





Conclusions/Future Work

- Computing is increasing supportive of collaboration among distributed users, and document editing systems are increasingly looking for solutions to allowing multiple authors to create documents synchronously and asynchronously.
- To date, much work has been done in this field, but there exists an opportunity to improve concurrent access while still maintaining reasonable communication costs – resulting in reasonable real-time responsiveness for users.
- Concomitantly, we propose to investigate and develop solutions to this open problem.
 - First, we will establish the **theoretic foundational algorithms and data structures** to maximize concurrent access to shared documents while minimizing communication cost. Initial work in developing a deadlock-free dynamic multi-granular algorithm and tree structure has been presented.
 - Second, we will articulate an **architecture that supports heterogeneous client editing tools, heterogeneous server document repositories, and the ability for subscription and notification mechanisms** upon document change. Our preliminary architecture supports all three of these, but this architecture will be refined to better accommodate reuse among the client and server components.
 - Third, a **prototype testbed that validates the architecture will be developed**, and communication and responsiveness data will be measured to demonstrate the viability of our approach.
- Finally, we have presented our intended scope and methodology to realize our research goals.





Agenda

- Introduction
- Motivation
- Problem Statement
- Contributions
- Related Work
- Research Goals
- Current Research Summary
- Conclusion/Future Work
- **Proposal Related Publications**





Proposal Related Publications

- Jon A Preston. *Exploring Communication Overheads and Locking Policies in a Peer-to-Peer Synchronous Collaborative Editing System*. ACMSE 2005, Kennesaw State University, GA, USA. Poster presentation.
- Jon A Preston and Sushil K Prasad. *A Web-Service-based Open-Systems Architecture for Collaborative Editing Systems*. Proceedings of CollabTech 2006, Ibaraki, Japan, 2006, pending acceptance.
- Jon A Preston and Sushil K Prasad. *A Deadlock-Free Multi-Granular, Hierarchical Locking Scheme for Real-time Collaborative Editing*. Proceedings of the 7th International Workshop on Collaborative Editing Systems. Sanibel Island, FL, 2005.





Target Publication Venues

- CollaborateCom 2006 (Nov 17-20, 2006)
 - Submission deadline is June 27, 2006
 - *Improved version of Architecture*
- CSCW 2006 (Nov 4-8, 2006)
 - Demo submission deadline is July 7, 2006
 - *Prototype developed*
- DEVS'07 (Spring 2007)
 - Submission deadline will probably be in November 2006
 - *Simulation of architecture and performance results of prototype*





References

1. Atul Prakash "Group Editors", Computer Supported Cooperative Work, Edited by Beaudouin-Lafon, 1999 John Wiley & Sons Ltd, pgs 103-133.
2. Saul Greenberg and Mark Roseman, "Groupware Toolkits for Synchronous Work", Computer Supported Cooperative Work, Edited by Beaudouin-Lafon, 1999 John Wiley & Sons Ltd, pgs 135-168.
3. Prasun Dewan, "Architectures for Collaborative Applications", Computer Supported Cooperative Work, Edited by Beaudouin-Lafon, 1999 John Wiley & Sons Ltd, pgs 169-193.
4. Paul Dourish, "Software Infrastructures", Computer Supported Cooperative Work, Edited by Beaudouin-Lafon, 1999 John Wiley & Sons Ltd, pgs 195-219.
5. G. Henri Ter Hofte, Working Apart Together: Foundations for Component Groupware, Telematica Instituut, The Netherlands, ISBN 90-75176-14-7, 1998.
6. Peter Manhart and DaimlerChrysler AG. "A System Architecture for the Extension of Structured Information Spaces by Coordinated CSCW Services", Proceedings of GROUP 1999, Phoenix Arizona USA, pgs 346-355.
7. Kevin L. Mills, "Introduction to the Electronic Symposium on Computer-Supported Cooperative Work", ACM Computing Surveys, Vol. 31, No. 2, June 1999, pgs
8. Arregui, D., Pacull, F., Willamowski, J.: "Yaka: Document notification and delivery across heterogeneous document repositories". In: Proc. of CRIWG'01, Darmstadt, Germany (2001)
9. Anita Sarma and André van der Hoek, "A Conflict Detected Earlier is a Conflict Resolved Easier", Proceedings of the 4th Workshop on Open Source Software Engineering, Edinburgh, United Kingdom, May 2004.
10. Anita Sarma , Zahra Noroozi and André van der Hoek , Palantir: Raising Awareness among Configuration Management Workspaces . In Proceedings of Twenty-Fifth International Conference on Software Engineering, pp 444-454, May 2003, Portland, Oregon.
11. Anita Sarma, "A Survey of Collaborative Tools in Software Development", UCI, ISR Technical Report, UCI-ISR-05-3, March 2005.
12. André van der Hoek , David Redmiles , Paul Dourish , Anita Sarma , Roberto Silva Filho , and Cleidson de Souza, "Continuous Coordination: A New Paradigm for Collaborative Software Engineering Tools", In Proceedings of the Workshop on Directions in Software Engineering Environments, pp 29-36, Edinburgh, United Kingdom, May 2004.
13. Dewayne E. Perry and Harvey P. Siy and Lawrence G. Votta, "Parallel Changes in Large Scale Software Development: An Observational Case Study", International Conference on Software Engineering 1998, pgs 251-260.





References (cont)

14. Goopeel Chung and Prasun Dewan, "Towards Dynamic Collaboration Architectures", Proceedings of the 2004 ACM conference on Computer supported cooperative work, November 6-10, 2004, Chicago, Illinois, USA. pgs 1-10.
15. Perry, D.E., H.P. Siy, and L.G. Votta, Parallel Changes in Large-Scale Software Development: An Observational Case Study. ACM Transactions on Software Engineering and Methodology, 2001. 10(3): p. 308-337.
16. Mens, T., A State-of-the-Art Survey on Software Merging. IEEE Transactions on Software Engineering, 2002. 28(5): p. 449-462.
17. Ronald van der Lingen and André van der Hoek, "Dissecting Configuration Management Policies", Proc. of the International Conference on Software Engineering Workshops: Software Configuration Management 2001.
18. Steven Xia, David Sun, Chengzheng Sun, David Chen and Haifeng Shen, "Leveraging Single-user Applications for Multi-user Collaboration: the CoWord Approach", Proceedings of the 2004 ACM conference on Computer supported cooperative work, Chicago, Illinois, USA, 2004. pgs 162-171.
19. J. Estublier. Defining and Supporting Concurrent Engineering Policies in SCM. Proceedings of the Tenth International Workshop on Software Configuration Management, 2001.
20. Shengzheng Sun and Clarence Ellis, "Operational Transformation in Real-Time Group Editor: Issues, Algorithms, and Achievements", Proceedings of 1998 ACM Conference on Computer Supported Cooperative Work, Seattle USA, Nov 14-18, pgs 59-68.
21. C. Sun, X. Jia, Y. Zhang, Y. Yang, D. Chen: "Achieving convergence, causality-preservation, and intention-preservation in real-time cooperative editing systems," ACM Transactions on Computer-Human Interaction, Vol.5, No.1, March, 1998, pp.63-108.
22. C. Sun, X. Jia, Y. Zhang, Y. Yang: "` REDUCE: a prototypical cooperative editing system," Proceedings of the 7th International Conference on Human-Computer Interaction , pp.89-92, San Francisco, USA, Aug. 24-30, 1997.
23. C. Sun, Y. Zhang, Y. Yang, D. Chen: "` Distributed concurrency control in real-time cooperative editing systems," Proc. of the 1996 Asian Computing Science Conference, , Lecture Notes in Computer Science, #1179, Springer-Verlag, Singapore, pp.84-95, Dec. 1996.
24. C. Sun, Y. Yang, Y. Zhang, and D. Chen: "` A consistency model and supporting schemes in real-time cooperative editing systems," Proc. of the 19th Australian Computer Science Conference, Melbourne, pp.582-591, Jan. 1996.
25. C. Sun and R. Sasic: "` Optional Locking Integrated with Operational Transformation in Distributed Real-Time Group Editors," In Proceedings of the 18th ACM Symposium on Principles of Distributed Computing. pp.43-52, Atlanta, GA, USA, May 4-6, 1999.
26. Muhammad Younas and Rahat Iqbal, "Developing Collaborative Editing Applications using Web Services", The Fifth International Workshop on Collaborative Editing, ECSCW 2003, Helsinki, Finland, September 15, 2003





References (cont)

27. Knister, M. and Prakash, A., DistEdit: A distributed toolkit for supporting multiple group editors. In Proceedings of the Third Conference on Computer-Supported Cooperative Work, pages 343–355, Los Angeles, California, October 1990.
28. A. van der Hoek and et.al. Continuous coordination: A new paradigm for collaborative software engineering tools. In Proceedings of Workshop on WoDISEE, Scotland, 2004.
29. M. Locasto et al. CLAY: Synchronous Collaborative Interactive Environment. The Journal of Computing in Small Colleges, vol. 17, issue 6, pp. 278-281, May 2002.
30. Korel, B. et al. Version Management in a Distributed Network Environment. In Proceedings of the 3rd International Workshop on Software Configuration Management, pp. 161-166, May 1991.
31. Magnusson, Boris, Asklund, Ulf. Collaborative Editing – distribution and replication of shared versioned objects. European Conference on Object Oriented Programming 1995, in Workshop on Mobility and Replication, Aarhus, August 1995.
32. Magnusson, Boris, and Guerraoui, Rachid. Support for Collaborative Object-Oriented Development. International Symposium on Parallel and Distributed Computing Systems (PDCS'96), Dijon, France, September 1996.
33. de Souza, Cleidson R. B., Redmiles, David, and Dourish, Paul, "Breaking the code", moving between private and public work in collaborative software development, Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work, November 09-12, 2003, Sanibel Island, Florida, USA.
34. Velazquez M. *A Survey of Distributed Mutual Exclusion Algorithms*. Colorado State University Department of Computer Science Technical Report CS-93-116, September 1993.
35. Walter, J. et al. A K-Mutual Exclusion Algorithm for Wireless Ad Hoc Networks. Principles of Mobile Computing '01. Newport, Rhode Island USA. 2001.
36. Bulgannawar, S. and Vaidya, N. A Distributed K-mutual Exclusion Algorithm. International Conference on Distributed Computing Systems, pp. 153-160, 1995.
37. Shari Lawrench Pfleeger. *Software Engineering: Theory and Practice*. Prentice Hall, New Jersey, NJ, 1998, pgs 1-34.
38. Roger S. Pressman. *Software Engineering, A Practitioner's Approach: Sixth Edition*. McGraw Hill, Boston, MA, 2005, pgs 596-613 and 739-765.
39. Jon A Preston, Exploring Communication Overheads and Locking Policies in a Peer-to-Peer Synchronous Collaborative Editing System. ACMSE 2005, Kennesaw State University, GA, USA. Poster presentation.





References (cont)

40. Jon A Preston and Sushil K Prasad. A Web-Service-based Open-Systems Architecture for Collaborative Editing Systems. Proceedings of CollabTech 2006, Ibaraki, Japan, 2006, pending acceptance.
41. Jon A Preston and Sushil K Prasad. A Deadlock-Free Multi-Granular, Hierarchical Locking Scheme for Real-time Collaborative Editing. Proceedings of the 7th International Workshop on Collaborative Editing Systems. Sanibel Island, FL, 2005.
42. Greenberg, S. and Marwood, D. Real time groupware as a distributed system: Concurrency control and its effect on the interface. In *Proceedings of the ACM conference on Computer-Supported Cooperative Systems*, November, 1994, 207-217.
43. Gu, N., Yang, J., and Zhang Q. Consistency Maintenance Based on the Mark & Retrace Technique in Groupware Systems. Proceedings of GROUP 2005, November 6-9, 2005, Sanibel Island, Florida, USA. pgs 264-273.
44. Wang, X., Bu, J., and Chen C. A New Consistency Model in Collaborative Editing Systems. Proceedings of the 4th International Workshop on Collaborative Editing. New Orleans, Louisiana, USA, 2002.
45. Li D. and Li R. *Transparent Sharing and Interoperation of Heterogeneous Single-User Applications*. In Proceedings of CSCW'02, New Orleans LA, pp. 246-255, November 2002.
46. Tigris.org. Subversion.
47. Borland.com/jbuilder. Borland JBuilder.
48. Microsoft.com/windowsserver2003/technologies/sharepoint. Microsoft SharePoint Services
49. Groove.net. Groove Networks.
50. Chu-Carroll, Mark C., Wright, James, and Shields, David. Supporting Aggregation in Fine Grain Software Configuration Management. SIGSOFT 2002/FSE-10, pp. 99-108. November 18-22, 2002, Charleston, SC, USA.
51. B. Magnusson. Fine-Grained Version Control in COOP/Orm. European Conference on Computer Supported Cooperative Work 1995, Workshop on Version Control in CSCW Applications, Stockholm, Sept. 1995.
52. 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.
53. Cheng L. et al. *Building Collaboration into IDEs*. ACM Queue. December/January 2003-2004. 40-50.





References (cont)

54. James D. Herbsleb , Audris Mockus , Thomas A. Finholt , Rebecca E. Grinter, An empirical study of global software development: distance and speed, Proceedings of the 23rd International Conference on Software Engineering, p.81-90, May 12-19, 2001, Toronto, Ontario, Canada
55. Donald Norman. Collaborative Computing: Collaboration First, Computing Second. Communications of the ACM. Vol 34, No. 12. December 1991. pgs. 88-90.
56. Katherine M. Everitt, Scott R. Klemmer, Robert Lee, James A. Landay. Two Worlds Apart: Bridging the Gap Between Physical and Virtual Media for Distributed Design Collaboration. Proceedings of CHI 2003, April 5-10, 2003, Ft. Lauderdale, Florida, USA. pgs 553-560.
57. Borghoff U. and Teege G. *Application of Collaborative Editing to Software-Engineering Projects*. ACM SIGSOFT, 18(3), pp. 56-64, July 1993.
58. J. D. Herbsleb and R. E. Grinter. Architectures, coordination, and distance: Conway's law and beyond. IEEE Software, pages 63-70, 1999.
59. Horstmann T. and Bentley R. *Distributed Authoring on the Web with the BSCW Shared Workspace System*. StandardView, vol. 5, no. 1, pp. 9-16, March 1997.
60. Grudin J. *CSCW Introduction*. Communications of the ACM, vol. 34, no. 12, pp. 30-34, December 1991.
61. Kock M. *The Collaborative Multi-User Editor Project IRIS*, Technical Report TUM-I9524, University of Munich, Aug. 1995.
62. Aguido Horatio Davis, Chengzheng Sun, Junwei Lu. Generalizing Operational Transformation to the Standard General Markup Language. Proceedings of CSCW 2002, New Orleans, Louisiana, USA. November 16-20. pgs. 58-67.
63. Shen H. and Sun C. *Flexible Notification for Collaborative Systems*. In Proceedings of CSCW'02, New Orleans Louisiana, pp. 77-86, November 2002.
64. Cheng L. et al. *Jazz: A Collaborative Application Development Environment*. In Proceedings of OOPSLA'03, Anaheim CA, 102-103, 2003.
65. Korel B. et al. *Version Management in Distributed Network Environment*. In Proceedings of the 3rd International Workshop on Software Configuration Management, pp. 161-166, May 1991.
66. Glance N. et al, *Collaborative Document Monitoring*. In Proceedings of GROUP'01, Boulder CO, pp. 171-178, September 2001.





References (cont)

67. Vidot N. et al. *Copies convergence in a distributed real-time collaborative environment*. In Proceedings of CSCW'00, Philadelphia PA, pp. 171-180, December 2000.
68. Hao M. C., Karp A. H, and Garfinkel D. *Collaborative Computing: A Multi-Client Multi-Server Environment*. In Proceedings of COOCS'95, Milpitas CA, pp. 206-213, August 1995.
69. Conradi R. and Westfechtel B. *Version Models for Software Configuration Management*. ACM Computing Surveys, vol. 30, no. 2, pp. 232-282, June 1998.
70. Y-J. Lin and S.P. Reiss. Configuration management with logical structures. In Proceedings of the 18th international conference on Software engineering, pages 298-307, Berlin, Germany, 1996. IEEE Computer Society.
71. Chu-Carroll, Mark C., and Sprenkle, Sara. Coven: brewing better collaboration through software configuration management, Proceedings of the 8th ACM SIGSOFT international symposium on Foundations of software engineering: twenty-first century applications, p.88-97, November 06-10, 2000, San Diego, California, United States.
72. Uri Dekel and Steven Ross. Eclipse as a Platform for Research on Interruption Management in Software Development. OOPSLA'04 Eclipse Technology eXchange (ETX) Workshop, Oct. 24-28, 2004, Vancouver, British Columbia, Canada. Copyright 2004 ACM. pgs. 12-16.
73. Teege G. and Borghoff U. W. *Combining Asynchronous and Synchronous Collaborative Systems*. In Proceedings of the 5th International conference on Human-Computer Interaction, Amsterdam Netherlands, pp. 516-521, 1993.
74. Nickson R. C. *A Taxonomy of Collaborative Applications*. <http://hsb.baylor.edu/ramsower/ais.ac.97/papers/nickers.htm>.
75. Roth J. and Unger C. *An extensible classification model for distribution architectures of synchronous groupware*. 4th International Conference on Cooperative Systems. 2000.
76. O'Reilly, C., P. Morrow, and D. Bustard. Improving Conflict Detection in Optimistic Concurrency Control Models. In Proceedings of the Eleventh International Workshop on Software Configuration Management. 2003. Portland, Oregon. pgs 191-205.
77. Ciaran O'Reilly: A Weakly Constrained Approach to Software Change Coordination. ICSE 2004: 66-68
78. van der Hoek A., Heimbigner D., and Wolf A. L. *A Generic, Peer-to-Peer Repository for Distributed Configuration Management*. Proceedings of the 18th international conference on Software Engineering, pp. 308-317, May 1996.
79. Sarma A., Noroozi Z., and van der Hoek A. *Palantir: Raising Awareness among Configuration Management Workspaces*. Proceedings of the 25th international conference on Software engineering, Portland OR, pp. 444-454, May 2003.





References (cont)

80. Begole J., Rosson M. B., and Shaffer C. A. Flexible Collaboration Transparency: Supporting Worker Independence in Replicated Application-Sharing Systems. *ACM Transactions on Computer-Human Interactions*, vol. 6, no. 2, pp. 95-132, June 1999.
81. Li D. and Patrao J. Demonstrational Customization of a Shared Whiteboard to Support User-Defined Semantic Relationships among Objects. In *Proceedings of GROUP'01*, Boulder CO, pp. 97-106, October 2001.
82. C. A. Ellis and S. J. Gibbs. Concurrency control in groupware systems. In *Proceedings of the ACM Conference on the Management of Data 1989*, pages 399-407, Portland Oregon, May 1989. ACM.
83. Roth, J. and Unger C. *Developing synchronous collaborative applications with TeamComponents*. 4th International Conference on Cooperative Systems. 2000.
84. Sunderam V. et al. *CCF: Collaborative Computing Frameworks*. SC'98: High Performance Networking and Computing Conference (Orlando, Florida USA). IEEE. 1998.
85. T. Mens. A state-of-the-art survey on software merging. *IEEE Transactions on Software Engineering*, 28(5):449-462, 2002.
86. David Sun, Steven Xia, Chengzheng Sun, and David Chen: "Operational transformation for collaborative word processing," *Proceedings of ACM 2004 Conference on Computer Supported Cooperative Work*, Nov 6-10, Chicago, IL USA.
87. Geyer W., Vogel J., Cheng L., and Muller M. *Supporting Activity-centric Collaboration through Peer-to-Peer Shared Objects*. In *Proceedings of GROUP'03*, Sanibel Island FL, pp. 115-124, November 2003.
88. Begole J., Rosson M. B., and Shaffer C. A. Supporting Worker Independence in Collaboration Transparency. In *Proceedings of UIST'98*, San Francisco CA, pp. 133-142, 1998.
89. Grinter, R. E. Recomposition: Putting It All Back Together Again. In *Proceedings of CSCW'98*, Seattle, Washington, USA, 1998. pgs 393-402.
90. L. Osterweil. Software processes are software too. In *Proceedings of the 9th International Conference on Software Engineering*, pages 2-13, Monterey, CA, 1987.
91. R.E. Grinter. Using a configuration management tool to coordinate software development. In *Conference on Organizational Computing Systems*, pages 168-177, 1995.
92. D.L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053-1058, 1972.





References (cont)

93. J. Grudin. Cscw: History and focus. *IEEE Computer*, 27(5):19–27, 1994.
94. Haifeng Shen and Chun Ti Cheong. CoStarOffice: Towards a Flexible Platform-independent Collaborative Office System. 6th International Workshop on Collaborative Editing Systems. Chicago, IL, USA, November 6, 2004.
95. Steven Xia, David Sun, Chengzheng Sun, David Chen, Yanxin Shi. Supporting Interactive Presentations with CoPowerPoint. 6th International Workshop on Collaborative Editing Systems. Chicago, IL, USA, November 6, 2004.
96. Mehra, A. et al. Supporting Collaborative Software Design with a Plug-in, Web Services-based Architecture. Workshop on Directions in Software Engineering Environments. ICSE 2004. IEEE. Edinburgh, Scotland, UK. May 23-28.
97. Tam, J., and Greenberg, S. (In Press - Accepted May 2005) A Framework for Asynchronous Change Awareness in Collaborative Documents and Workspaces. *International Journal of Human Computer Studies*, Elsevier
98. Chawathe Y., McCanne S., and Brewer E. *RMX: Reliable Multicast in Heterogeneous Networks*. In Proc. IEEE INFOCOM, March 2000.
99. Fu S., Tzeng N., and Li Z. *Empirical Evaluation of Distributed Mutual Exclusion Algorithms*. International Parallel Processing Symposium '97. 1997.
100. Drury J. *Developing Heuristics for Synchronous Collaborative Systems*. In Proceedings of CHI'2001, pp. 447-448, March/April 2001.
101. Walpole J. et al. *A Unifying Model for Consistent Distributed Software Development Environments*. In Proceedings of the third ACM SIGSOFT/SIGPLAN software engineering symposium on Practical software development environments, pp. 183-190, January 1989.
102. Wu D. and Sarma R. *Dynamic Segmentation and Incremental Editing of Boundary Representations in a Collaborative Design Environment*. Proceedings of the sixth ACM symposium on Solid Modeling and Applications, Ann Arbor Michigan, pp. 289-300, May 2001.
103. Buszko, D., Lee W., and Helal A. *Decentralized Ad-Hoc Groupware API and Framework for Mobile Collaboration*. In Proceedings of ACM 2001 International Conference on Supporting Group Work, Boulder, Colorado, pages 5-14, 2001.
104. Harrison W. H., Ossher H., and Sweeney P. F. *Coordinating Concurrent Development*. In Proceedings of CSCW'90, 157-168, October 1990.
105. C. Sun and D. Chen: "Consistency Maintenance in Real-Time Collaborative Graphics Editing Systems," *ACM Transactions on Computer-Human Interaction*, vol 9, no 1, March 2002. pgs 1-41.
106. P.H. Feiler. Configuration management models in commercial environments. TechnicalReport SEI-91-TR-07, Software Engineering Institute, Carnegie Mellon University, 1991.

