



A Web-Service-based Open-Systems Architecture for Achieving Heterogeneity in Synchronous Collaborative Editing Systems

Jon A Preston and Sushil K Prasad

Cooperative Internet Computing 2006

October 25-27, 2006

Hong Kong



Agenda

- Introduction and Motivation
- Concurrency
 - Operational Transformation
 - Locking (multi-grain)
- Architecture
 - Overview
 - Adding the Lock Manager Proxy
 - Events
- Dynamic Locking Algorithm
- Simulation and Results
- Conclusions and Future Work



Introduction and Motivation

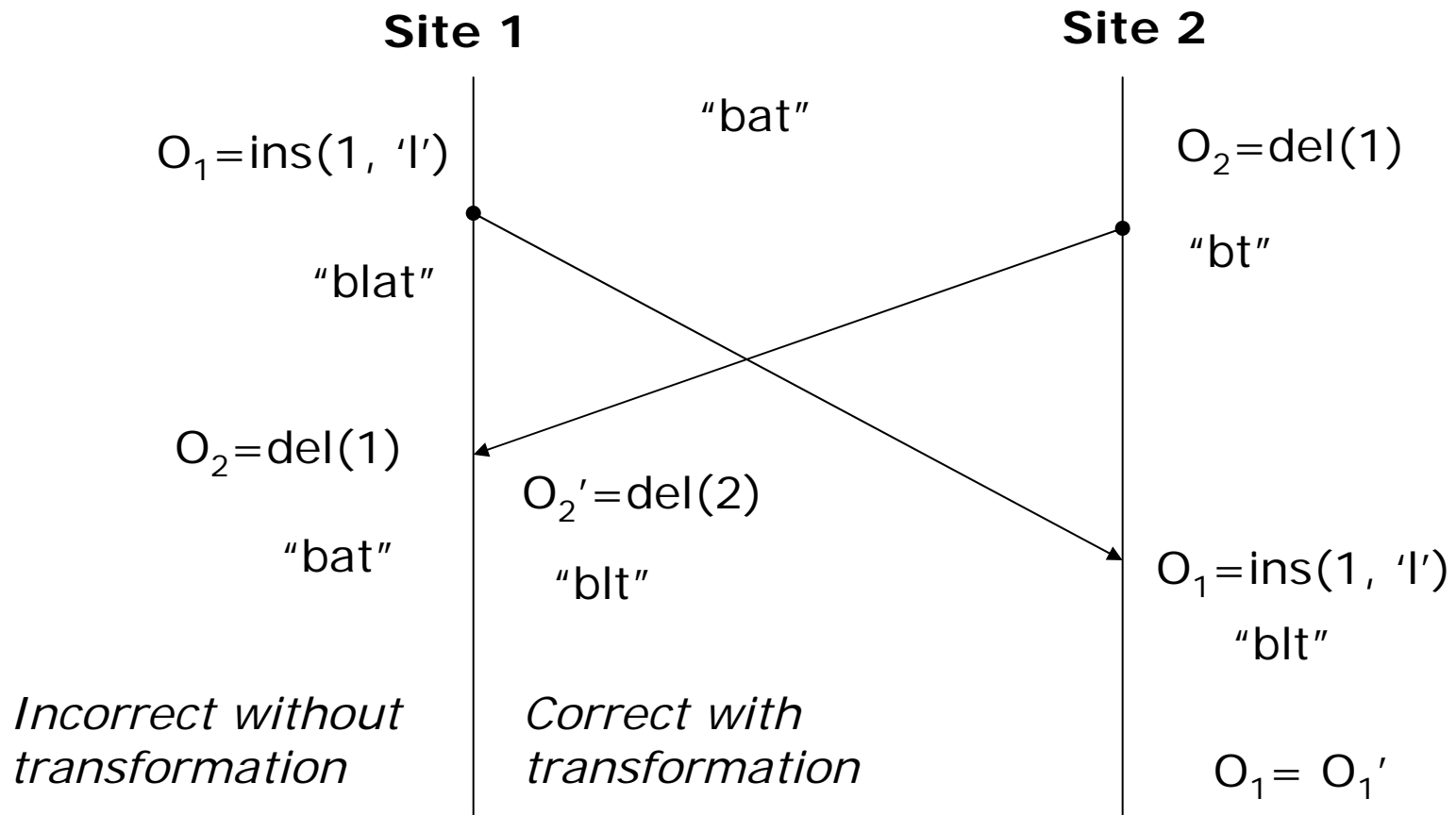
- Combine heterogeneous editing tools and heterogeneous document repositories
- Utilize Web-services to standardize API
- Generalized collaborative editing system
- Increase concurrency while minimizing communication costs



Concurrency

- Operational Transformation
 - Optimistic (no locking)
 - Continuous multicast
 - Transform operations to ensure CCI
 - Convergence, Causality Preservation, and Intention Preservation
- Locking
 - Reduces concurrency (exclusive write)
 - Avoids need to merge/OT
 - Reduces communication overhead
 - Traditionally done at a file level
 - Fine-grain locking offers potential

OT Example

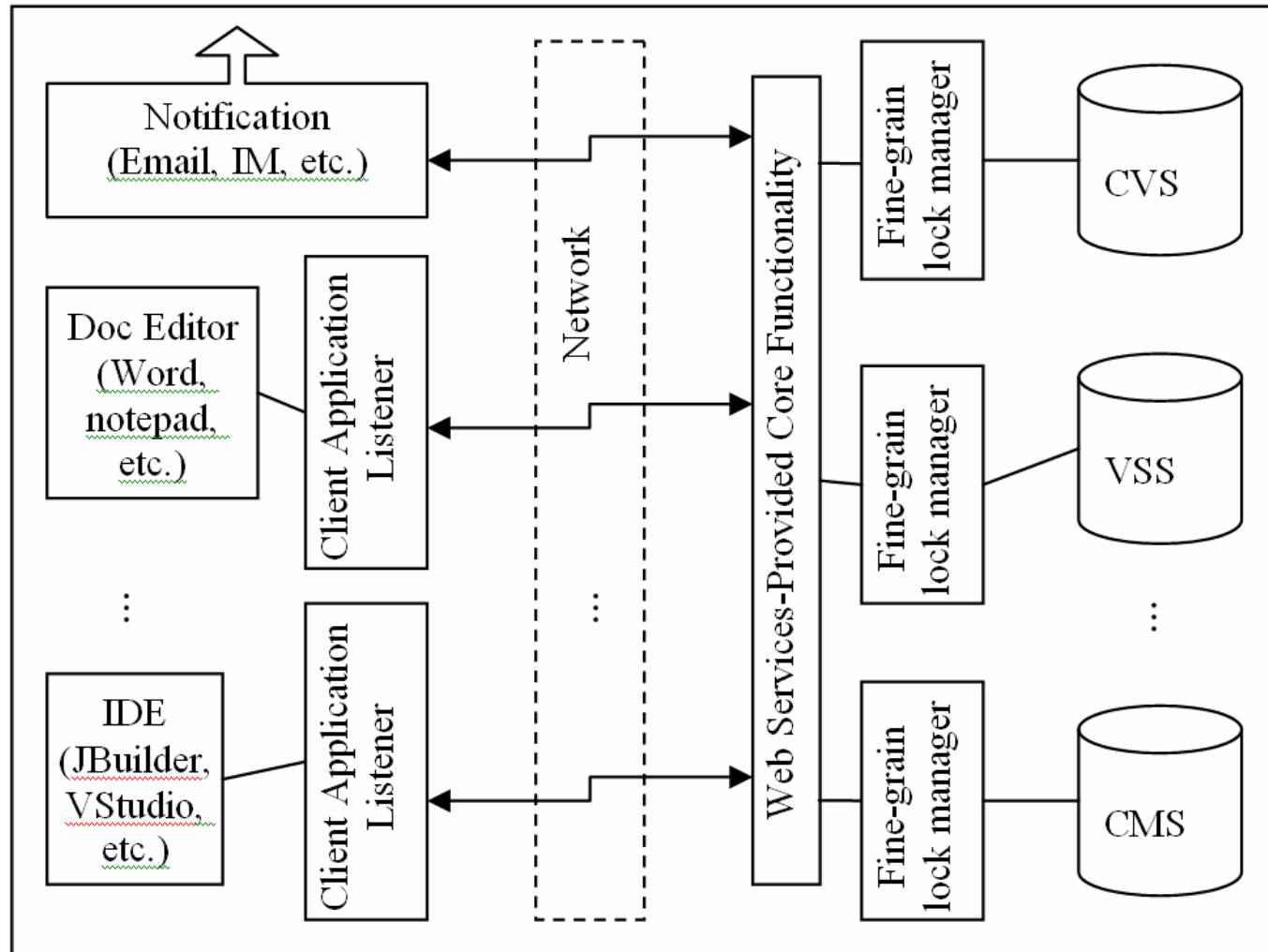




Architecture

- 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.

Achieving Heterogeneity

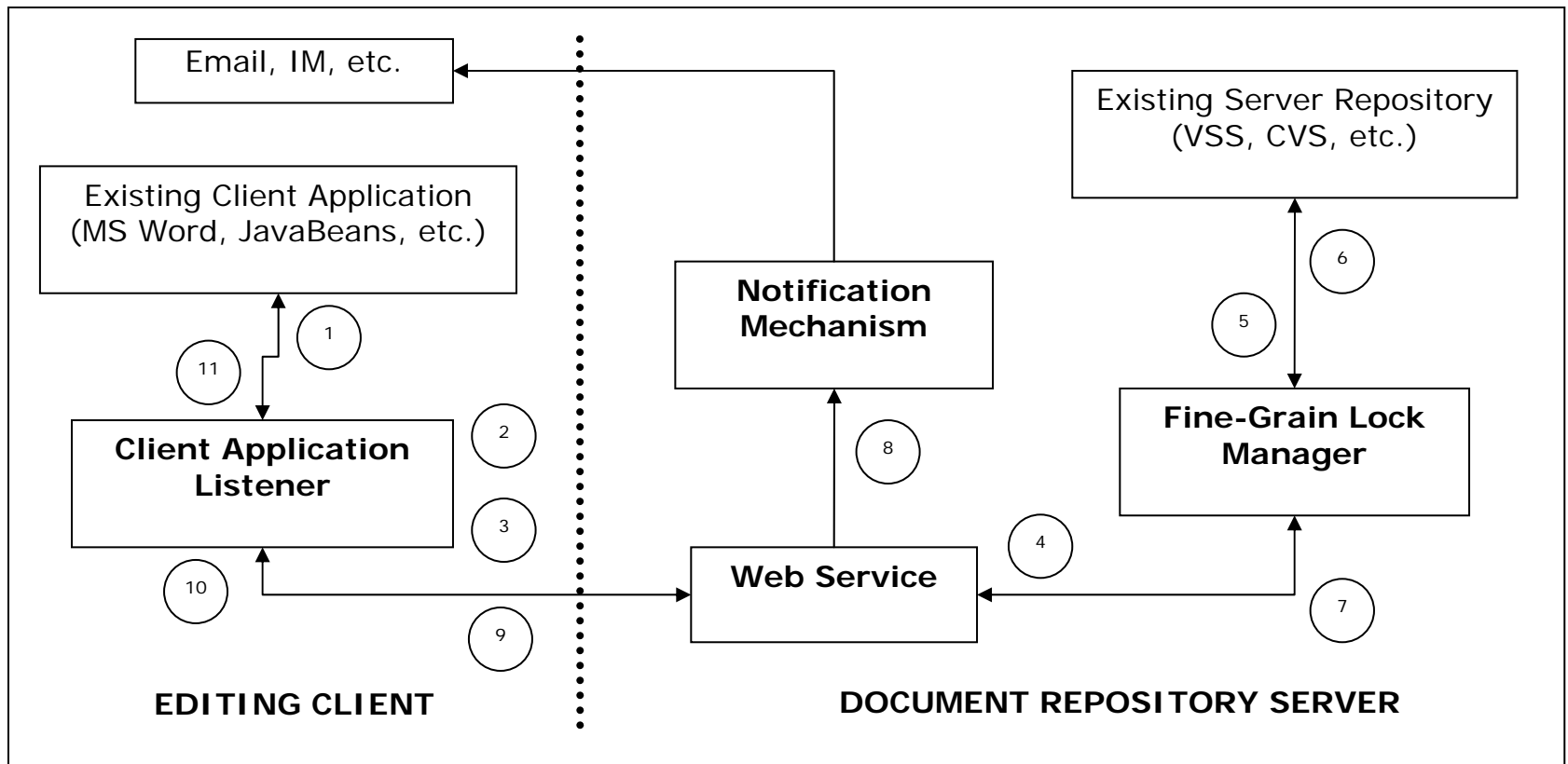




Components

- 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 for distributed repositories





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



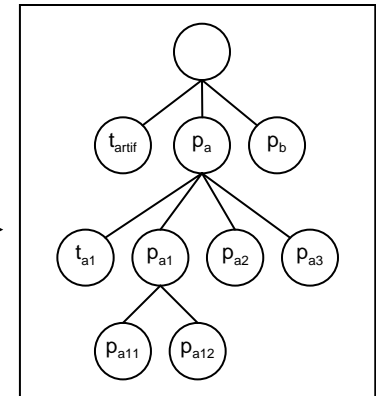
Dynamic Locking Algorithm

- Tree representation of document
- Lock largest sub-tree possible
- Demote lock if request creates contention
- Promote lock if contention is removed
- Deadlock-free
- Can integrate OT to share larger subsections if desired

N-ary Document Tree

- Model the document as an n-ary tree
- Sections and subsections
- Users can “own” a portion of the document without blocking other users from editing other portions of the document

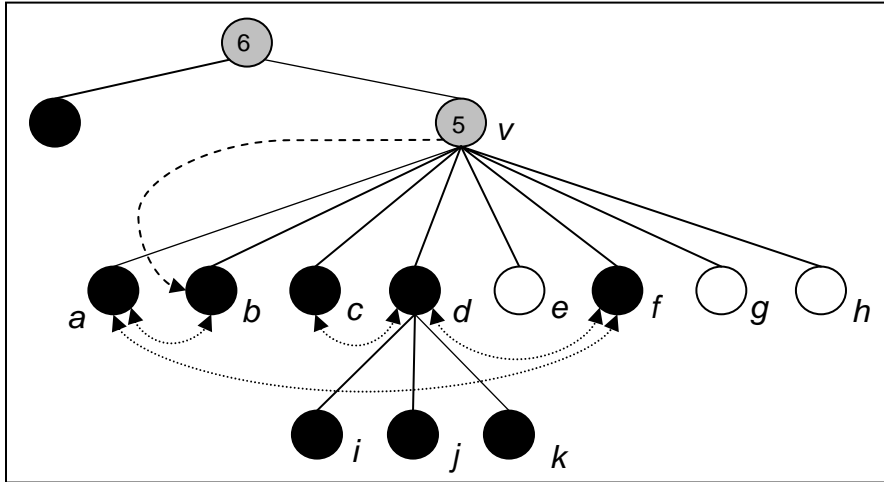
Title (t_{artif})
Paragraph A (p_a)
 Title A1 (t_{a1})
 Paragraph A1 (p_{a1})
 Paragraph A11 (p_{a11})
 Paragraph A12 (p_{a12})
 Paragraph A2 (p_{a2})
 Paragraph A3 (p_{a3})
Paragraph B (p_b)
...





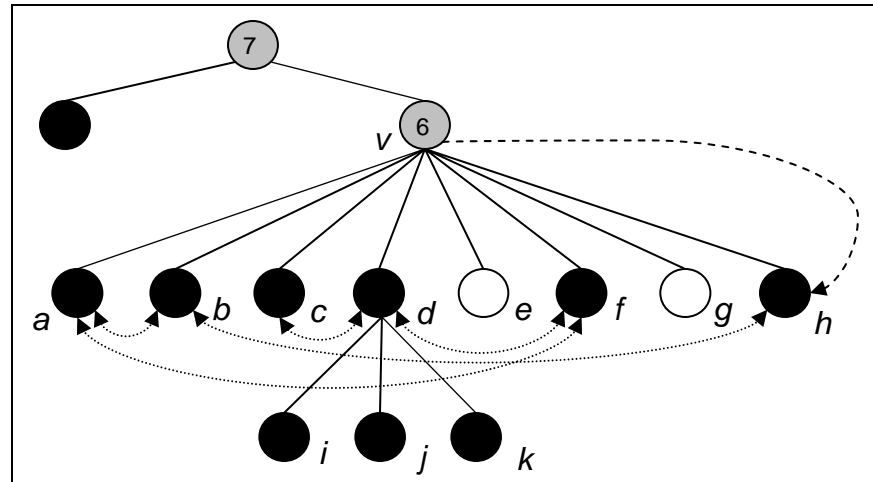
ObtainLock

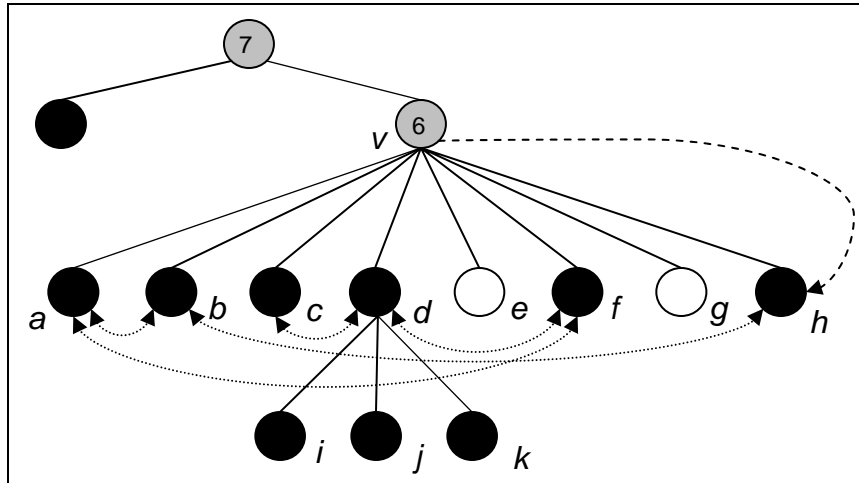
- Traverse top to bottom
- Increase “grey count” as you descend
- Keep descending until you reach a resolution node
 - Obtain an available node
 - Demote another user to resolve conflict
 - Deny request (or adopt OT)



ObtainLock(u_1, h)

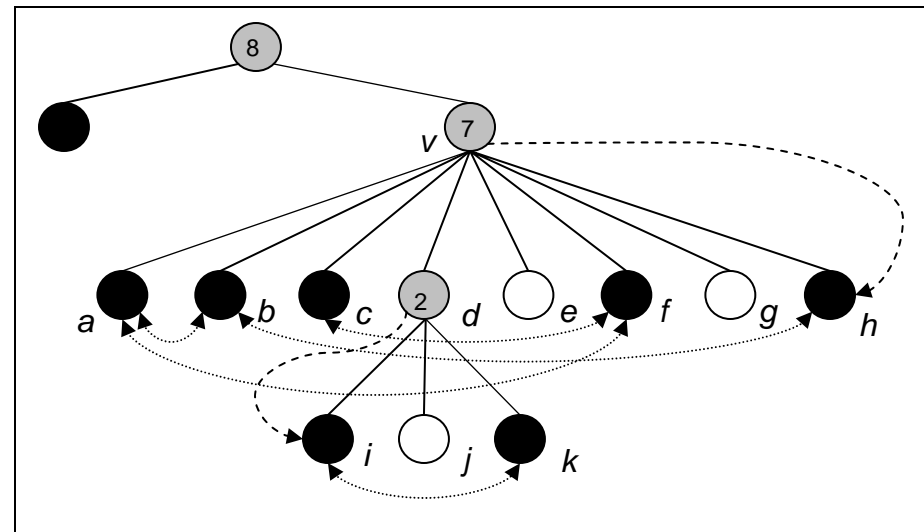
No Demotion





ObtainLock(u_2, k)

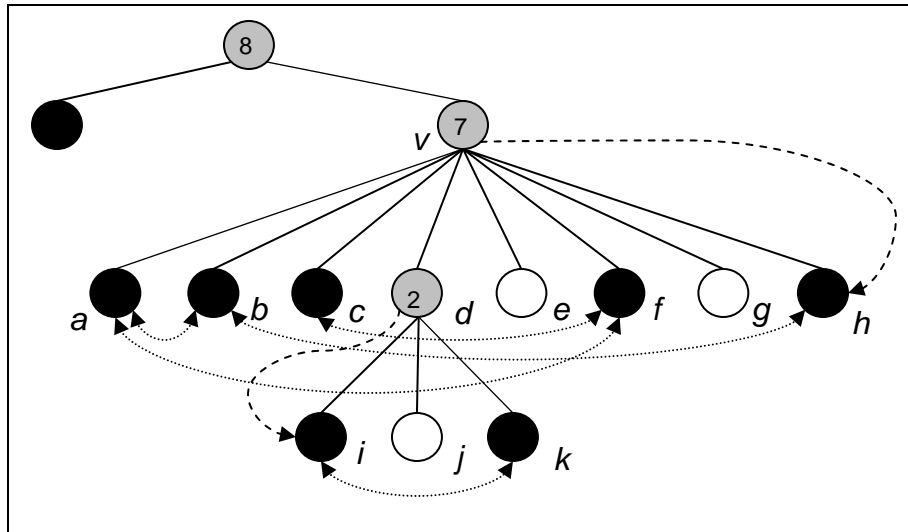
Demotion of u_1 from node d to node i





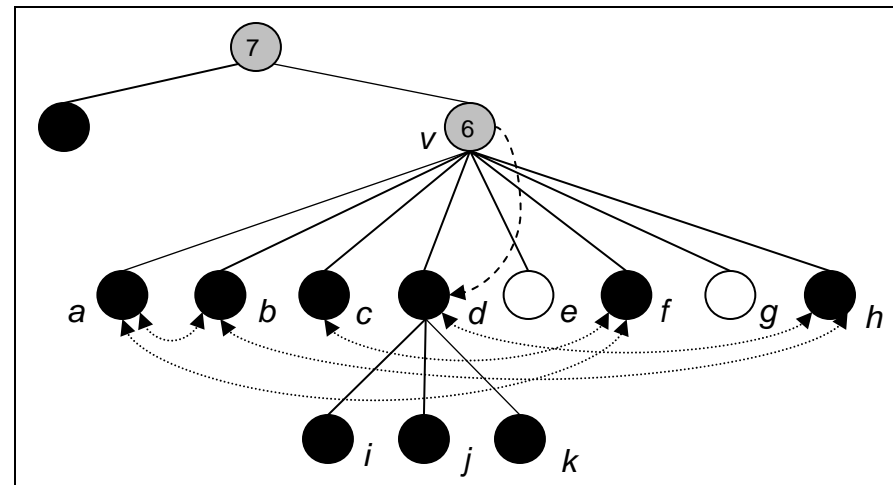
RemoveLock

- Traverse top to bottom
- Decrease “grey count” as you descend
- Keep descending until you reach a resolution node
 - Release owned node
 - Promote another user if conflict is now removed (“grey count” decreased to 1)



RemoveLock(u_1, i)

u_2 lock on node k is promoted to node d

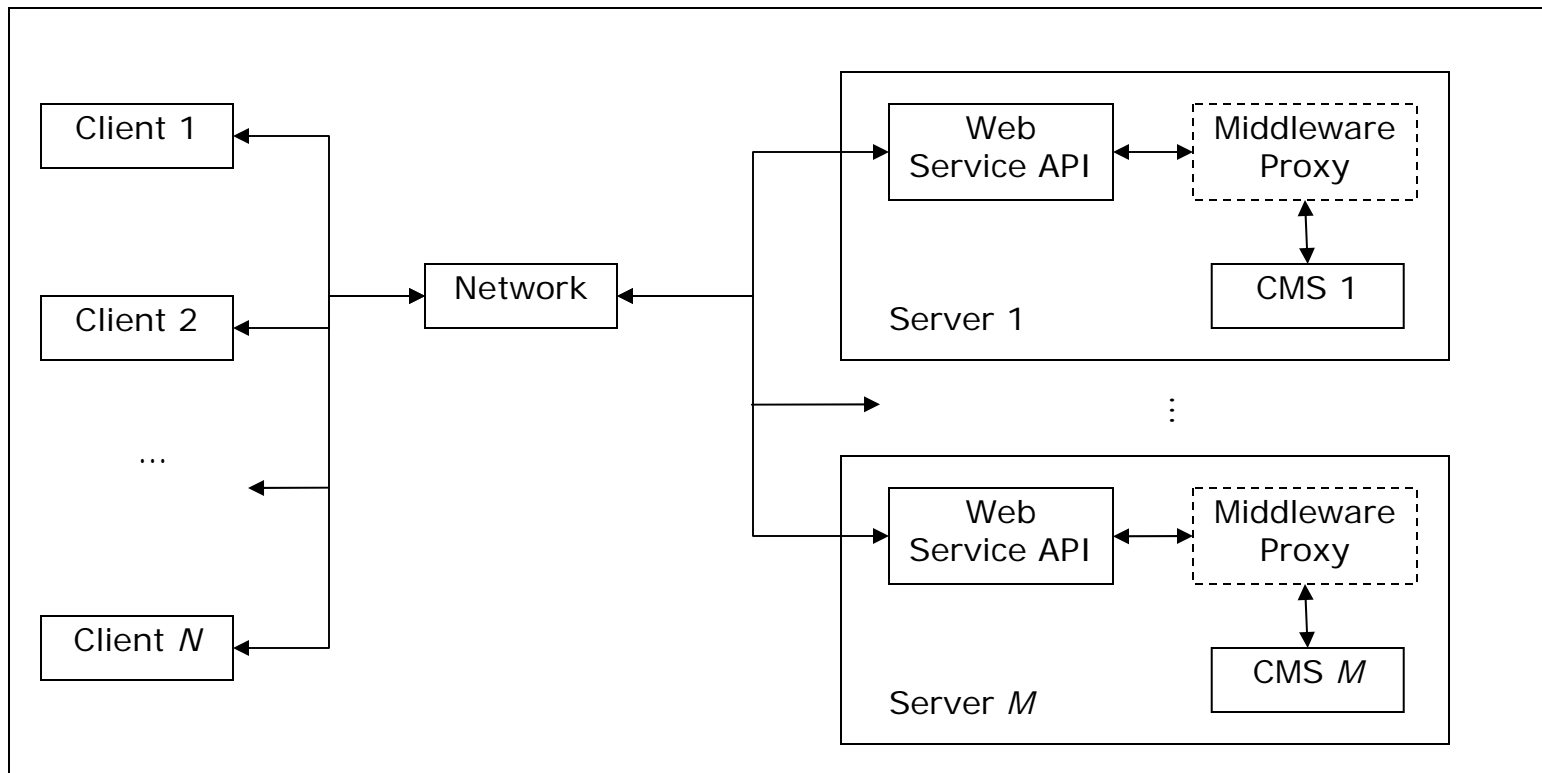




Simulation

- 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 Configurations





Tests

- 3 types of clients
 - Random (selects from all documents)
 - Hybrid
 - Clustered (picks documents localized)
- Various distributions of clients
- 3 or 9 servers (various distribution of documents)
- Various iterations (time duration of simulation)

Simulation Runs

Test	Iterations	Client Distribution (# per type)			Repository Distribution (# Artifacts at each Server)								
		Random	Clustered	Hybrid	S1	S2	S3	S4	S5	S6	S7	S8	S9
1	500	1	1	1	1	2	1						
2	500	3	0	0	1	2	1						
3	500	0	3	0	1	2	1						
4	500	0	0	3	1	2	1						
5	500*	1	1	1	10	20	10						
6	500	10	10	10	30	50	80	30	30	40	40	100	100
7	5000	10	10	10	30	50	80	30	30	40	40	100	100
8	2500	10	10	10	15	25	40	15	15	20	20	50	50
9	5000	1	1	1	1	2	1						

* Test 5 for the fine-grain version was run to 5000 iterations to obtain lock failures



Results

Test	Fail Rate		Improvement
	Without Fine-grain Locking	With Fine-grain Locking	
1	32.75%	7.27%	78%
2	23.33%	11.67%	50%
3	26.92%	6.38%	76%
4	19.64%	7.02%	64%
5	2.00%	0.75%	63%
6	16.39%	5.81%	65%
7	7.91%	2.62%	67%
8	9.08%	2.99%	67%
9	26.55%	7.24%	73%



Conclusion

- Architecture achieves heterogeneous connection
 - Client editing tools
 - Server repositories
- Simulation validates architecture
- Hierarchical locking
 - Maximizes concurrent access
 - Minimizes communication costs



Current/Future Work

- Current Implementation
 - Client Listener (Text Editor and MS Word)
 - Web Service API
 - Lock Manager Proxy (VSS & CVS)
- Load testing under real-world constraints



References

- [1] A. Mehra, J. Grundy, and J. Hosking, "Supporting Collaborative Software Design with a Plug-in, Web Services-based Architecture", *Workshop on Directions in Software Engineering Environments (WoDiSEE) from Proceedings of the 26th International Conference on Software Engineering (ICSE'04)*, IEEE, Edinburgh, Scotland, 2004.
- [2] M. Younas and R. Iqbal, "Developing Collaborative Editing Applications using Web Services", *Proceedings of the 5th International Workshop on Collaborative Editing*, Helsinki, Finland, September 115, 2003.
- [3] V. Bharadwaj and Y. V. R. Reddy, "A Framework to Support Collaboration in Heterogeneous Environments", *SIGGROUP Bulletin*, (24) 3, Dec. 2003, pp. 103-116.
- [4] C. Gutwin and S. Greenberg, "The Importance of Awareness for Team Cognition in Distributed Collaboration", *Team Cognition: Understanding the Factors that Drive Process and Performance*, APA Press, Washington, pp. 177-201.
- [5] C. Sun, X. Jia, Y. Zhang, , Y. Tang, and C. Chen. "Achieving convergence, causality-preservation, and intention-preservation in real-time cooperative editing systems", *ACM Transactions on Computer-Human Interaction*, (5) 1, Mar 1998, pp 63-108.
- [6] C. Ellis and S. Gibbs. "Concurrency Control in Groupware Systems", *ACM SIGMOD Conference on Management of Data*, May 1989, pp. 399-407.
- [7] Cederqvist, P. "Version Management with CVS", Available from info@signum.se, 1993.
- [8] M.C. Chu-Carroll, J. Wright, and D. Shields, "Supporting Aggregation in Fine Grain Software Configuration Management", *SIGSOFT 2002/FSE-10*, Charleston, SC, USA, November 18-22, 2002, pp. 99-108.
- [9] B. Magnusson, U. Asklund, and S. Minör, "Fine-Grained Revision Control for Collaborative Software Development", *Proceedings of ACM SIGSOFT'93 – Symposium on the Foundations of Software Engineering*, Los Angeles, California, December 1993.
- [10] Microsoft Corporation. Visual Source Safe. Available at <http://visualsourcesafe.com>, February 2005.
- [11] B. P. Zeigler and H.S. Sarjoughian. "Introduction to DEVS Modeling & Simulation with JAVA: Developing Component-based Simulation Models", Technical Document, University of Arizona. 2003.
- [12] J. Preston and S. Prasad. "A Deadlock-Free Multi-Granular, Hierarchical Locking Scheme for Real-time Collaborative Editing", *The Seventh International Workshop on Collaborative Editing Systems*, Sanibel Island, Florida, USA, November 2005.