



Simulation-based Architectural Design and Implementation of a Real-time Collaborative Editing System

2007 DEVS Integrative Modeling &
Simulation Symposium (DEVS'07)

Jon A Preston
Xiaolin Hu
Sushil Prasad



Agenda

- Goals
- Background
- Simulation Design Process
- Models
- Simulation Configurations
- Results
- Conclusions



Goals

- Design and develop a Real Time Collaborative Editing System (RTCES) using simulation design
 - Multiple simultaneous users
 - Shared document
 - Distributed collaboration
- Reduce communication costs
 - Existing solutions are costly

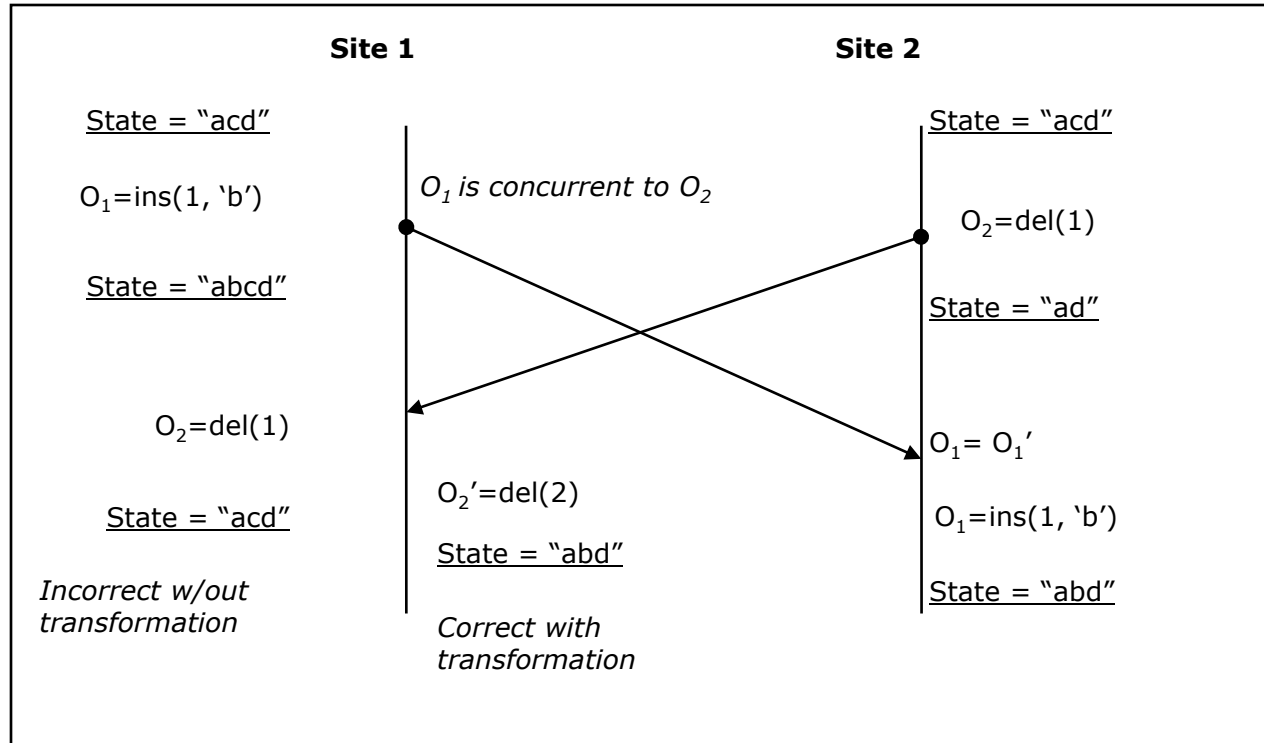
Background

- RTCES are a subset of Computer Supported Collaborative Work (CSCW)
- Musts provide responsiveness in the editor
 - Thus replication of document is employed
- Operational Transformation (OT) use to ensure CCI
 - CCI defined (next)

Background: CCI Defined

1. **Convergence:** when all sites have executed the same set of operations, the copies of the shared document at all sites are identical.
2. **Causality-preservation:** for any pair of operations O_a and O_b , if $O_a \rightarrow O_b$, then O_a is executed before O_b at all sites.
3. **Intention-preservation:** for any operation O ,
 - (a) both the local and remote execution effects of O equal to the intention of O , and
 - (b) if there exists an operation O_x such that $O_x \parallel O$, then the execution effect of O_x does not interfere with the execution effect of O , and vice versa.

OT Example



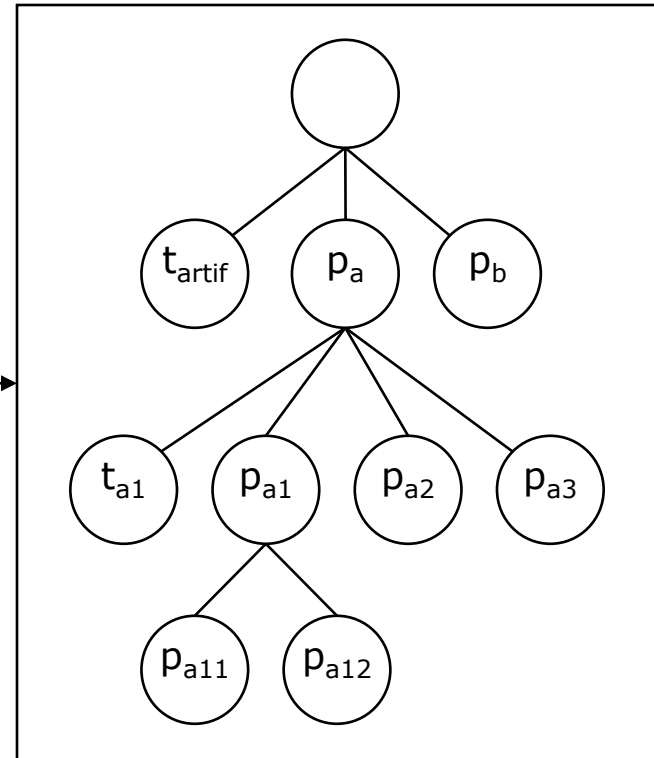


Our Previous Work

- Dynamic, hierarchical locking
 - Manage a tree representing the shared document
 - Locks are automatically moved up and down the tree to allow multiple users access
- Previous simulation demonstrated dynamic locking can reduce communication costs significantly
 - But had write request fails (locks demoted to leaf nodes, but then rejected thereafter)
 - Caused some users to be blocked from editing

Previous Work: Document Tree

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)
...



Message Types

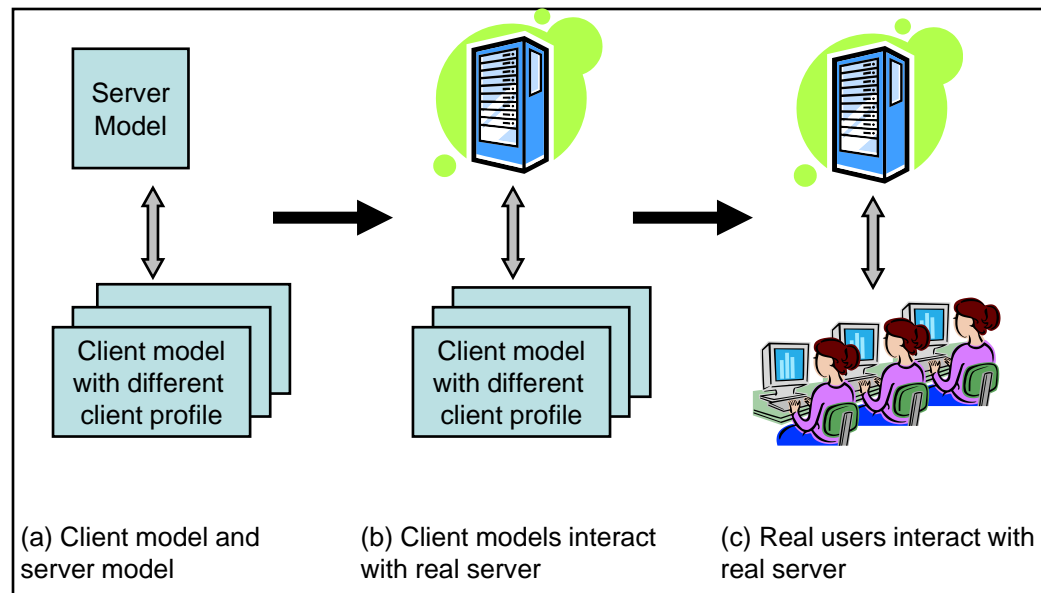
- *Document Check-out* (**CO**) – the client would like to check out and become a reader of a document.
- *Document Check-in* (**CI**) – the client is no longer interested in the document and releases it.
- *Lock Request* (**LK**) – the client wants to write to a section of the document
- *Unlock* (**ULK**) – the client has left the section and no longer needs the ability to write to it

- *Promotion* (**P**) – informs the user that he now owns more of the document that he previously owned.
- *Demotion* (**D**) – informs the user that he now owns less of the document that he previously owned.

- *OT Added* (**OTA**) – signals a user within a section that another user has been added to the section and future changes must be sent to this new user
- *OT Deleted* (**OTD**) – signals a user within a section that a user has left the section and no longer needs to have changes sent to him
- *OT Join* (**OTJ**) – tells the user requesting a lock that he has been granted write access to a section that is already using OT; this message contains a list of the existing users within the section so that the new user can send future changes to these users
- *OT Modify* (**OTM**) – this message tells a client that the section has been modified and a local OT must be performed based upon the operation being communicated.

Simulation Design Process

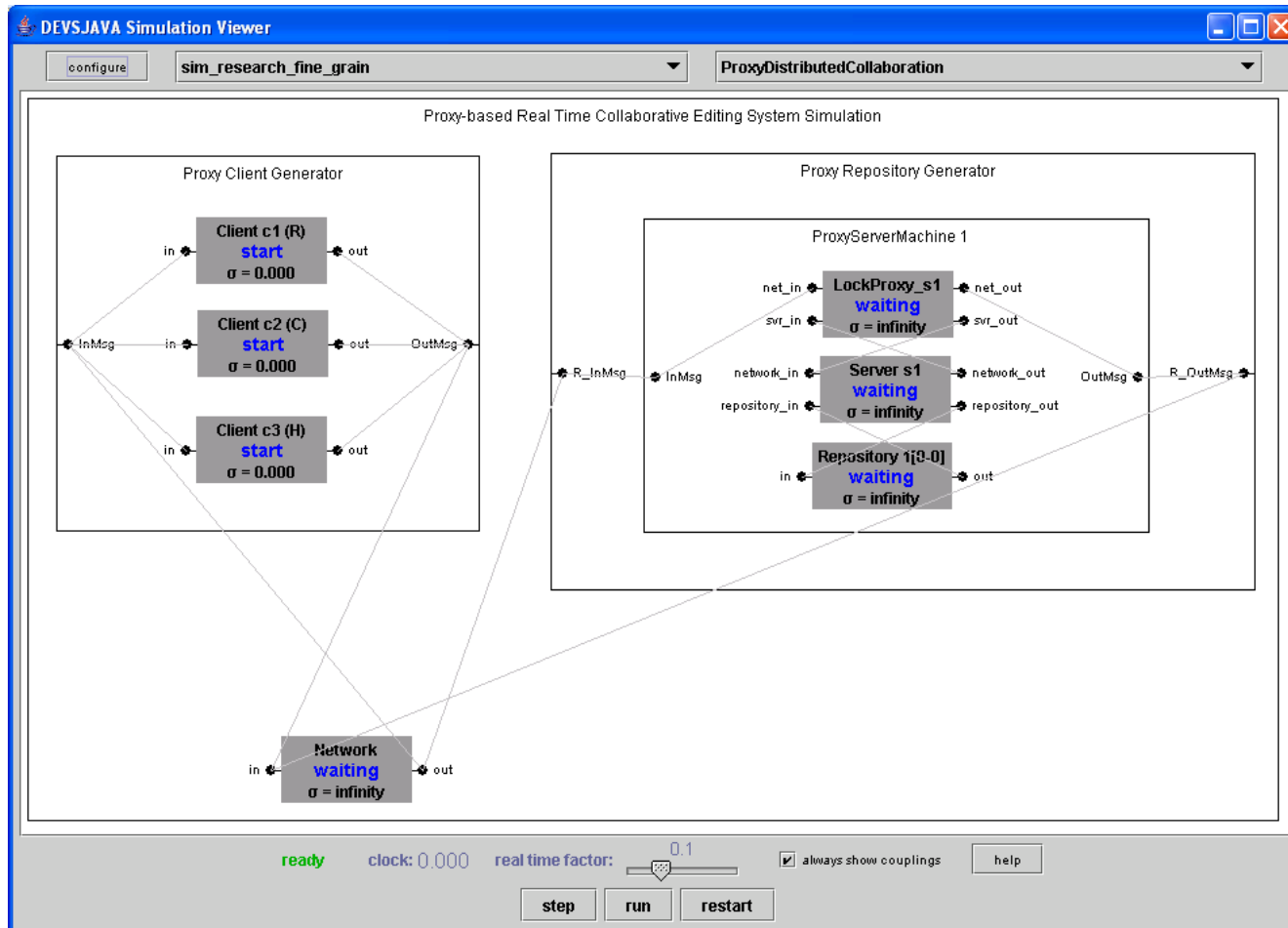
- Simulate both clients and server
- Simulate clients and use real server
- Use real clients and real server



Models

- Client
 - Behaviors (states – reading or writing)
 - Internal state transition function changes
 - Based upon simulated user actions
 - External state transition function changes
 - Based upon messages from clients and server
- Server
 - Repository
 - Lock proxy (integrates with legacy repositories) and implements our algorithms
 - Only external state transition function implemented
- Network
 - Simple (all messages arrive in constant time)
 - Could make more complex (not focus)
 - Used to monitor and count message types

Models in DevsJava



This configuration shows one server and three clients.



Simulation Configurations

- 6 different document structures
 - Varied depth and number of leaves
- 2 different number of clients
 - Used 3 and 9 clients
- 4 different client behavior patterns
 - Random (R)
 - Clustered (C)
 - Hybrid (H)
 - Uniform distribution of R, C, and H

Document Structures

Document Structure	Number of Leaves	Maximum Depth	Average Depth
1	4	3	2.75
2	8	4	2.875
3	16	4	2.875
4	48	3	3
5	96	3	3
6	192	3	3

Document structures 5 and 6 represent 4 and 8 page conference papers (respectively) if paragraphs are used as the leaf nodes in the document tree.

Simulation Configurations

Simulation Configuration	Number of Clients	Document Structure
1	3	1
2	9	1
3	3	2
4	9	2
5	3	3
6	9	3
7	3	4
8	9	4
9	3	5
10	9	5
11	3	6
12	9	6

Results

- Since all messages are broadcast to all users other than the originating user in a pure OT system, we define the number of messages generated in a pure OT system as

$$M_{PureOT} = (n - 1)W$$

where n is the number of users and W is the number of write requests (the number of times users modified the document).

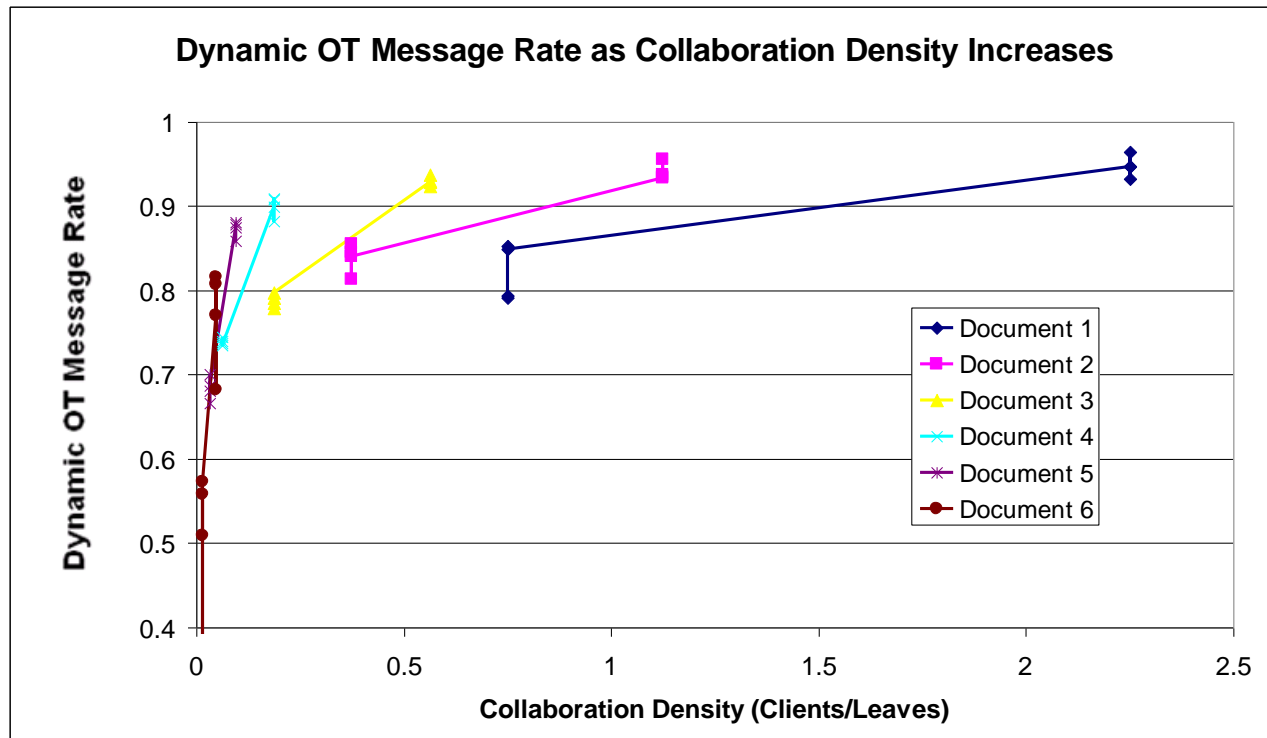
- The relative message overhead, Mo , of our dynamic lock OT system is defined as

$$Mo = \frac{LK + ULK + P + D + OTA + OTD + OTJ + OTM}{M_{PureOT}}$$

- We ignore LK and ULK since these are constant in both pure OT and our system.
- Thus a relative message overhead of 1 reflects the dynamic lock with OT system incurs the same number of communication cost as a pure OT system. Mo above 1 reflects our system incurs more communication than a pure OT system. Mo below 1 reflects our system incurs less communication than a pure OT system. Thus a lower value is a reduction in communication costs.

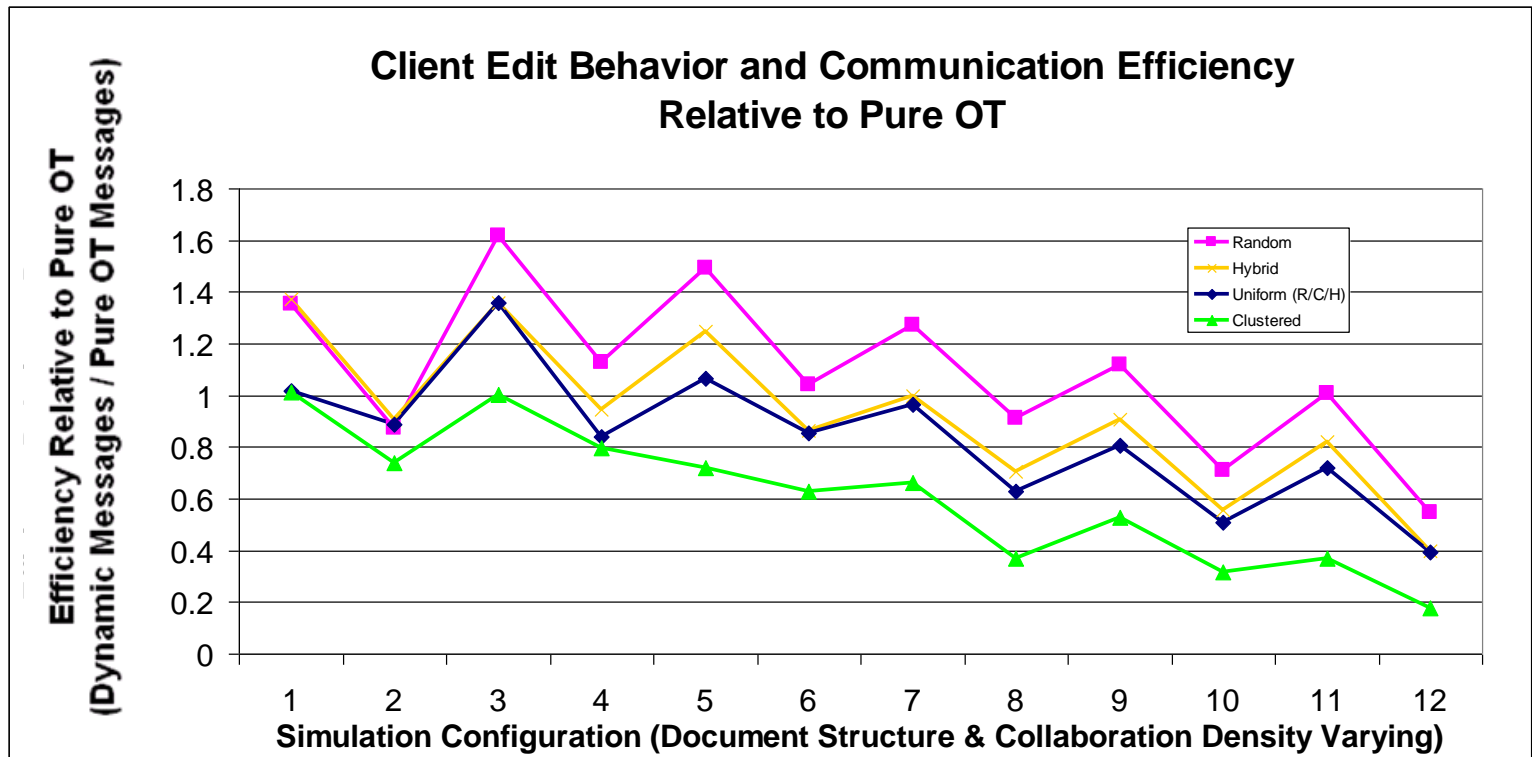
Results: Dynamic Message Overhead

Dynamic OT Message Rate = % messages dealing with OT relative to all messages

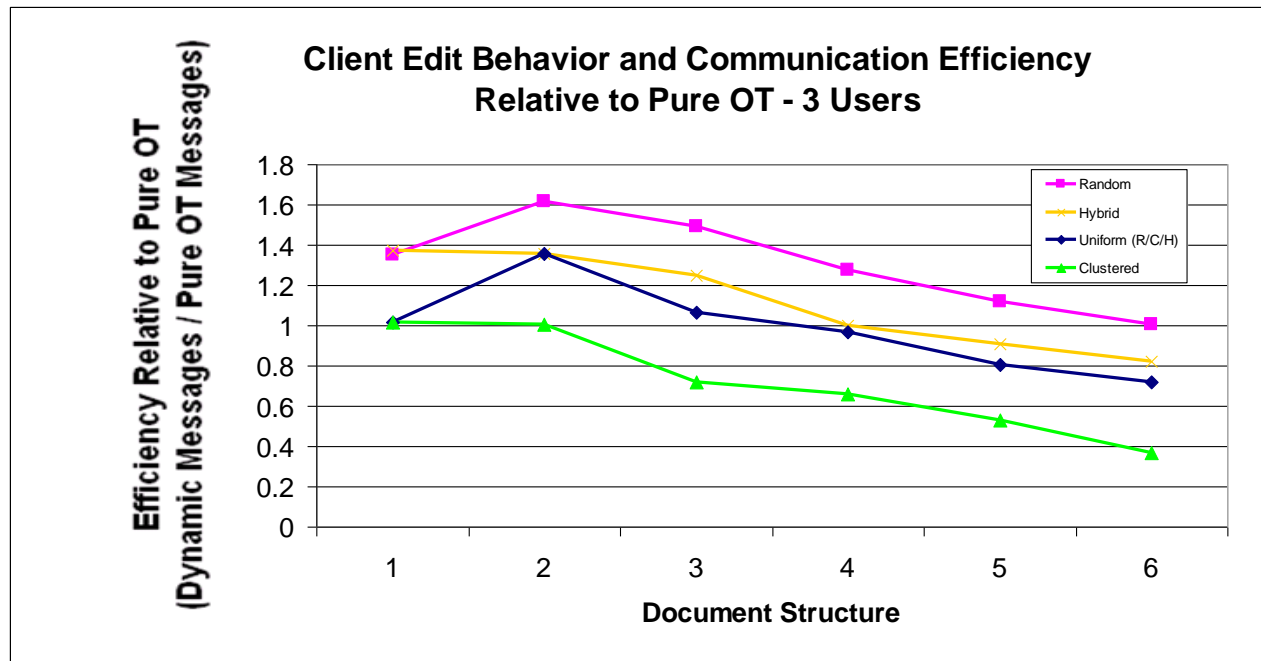


Results: Communication Efficiency

- Odd simulations = 3 users
- Even simulations = 9 users

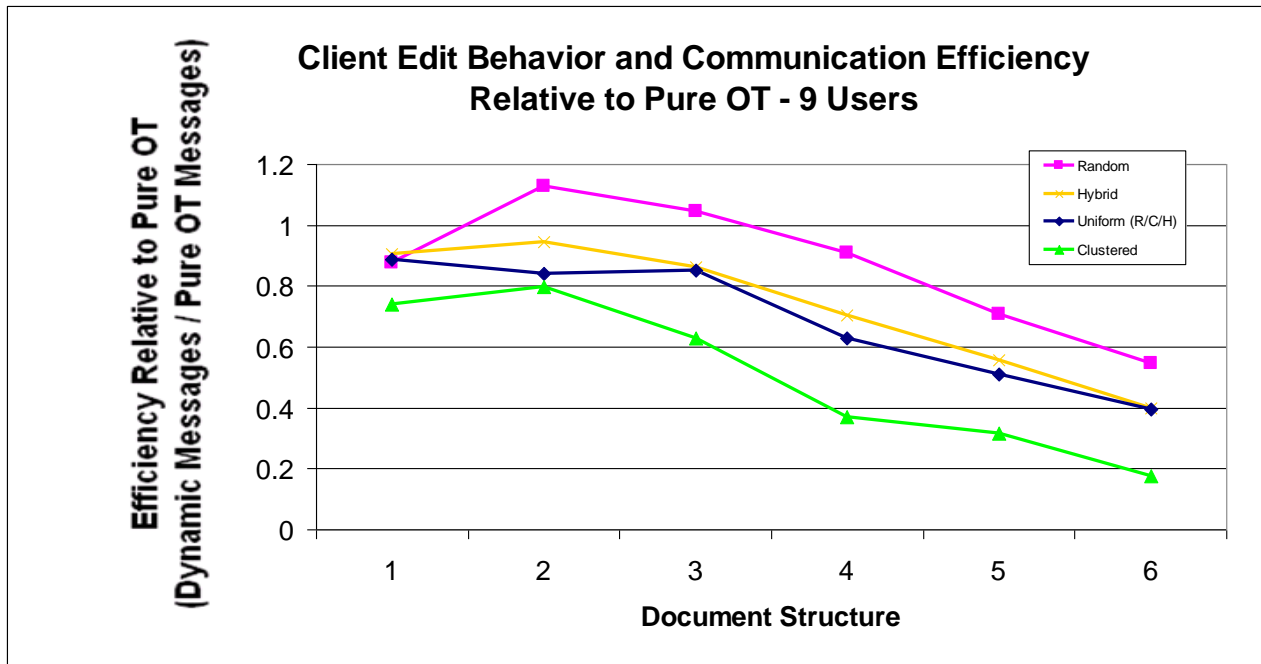


Results: 3 Users



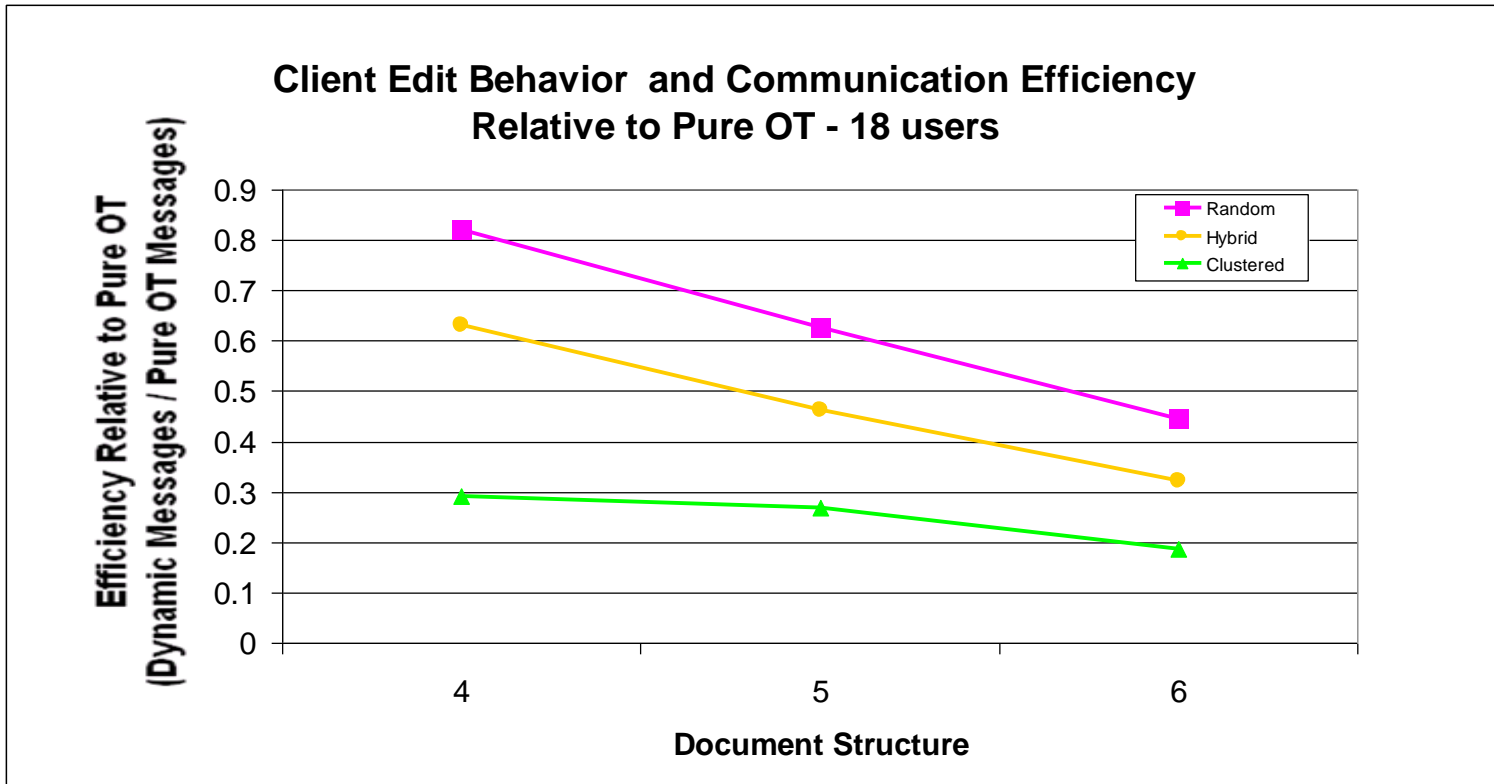
Document Structure	Number of Leaves	Maximum Depth	Average Depth
1	4	3	2.75
2	8	4	2.875
3	16	4	2.875
4	48	3	3
5	96	3	3
6	192	3	3

Results: 9 Users



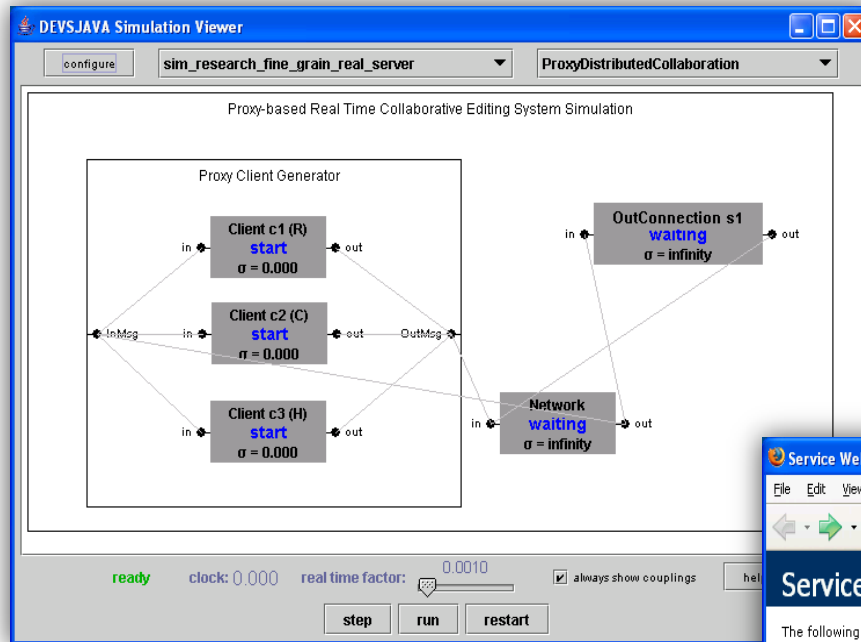
Document Structure	Number of Leaves	Maximum Depth	Average Depth
1	4	3	2.75
2	8	4	2.875
3	16	4	2.875
4	48	3	3
5	96	3	3
6	192	3	3

Results: 18 Users



Document Structure	Number of Leaves
4	48
5	96
6	192

Real Server



The screenshot shows a Mozilla Firefox browser window titled "Service Web Service - Mozilla Firefox". The address bar shows the URL "http://localhost:2077/Lock_Proxy/Service.asmx". The page content is titled "Service" and lists supported operations:

The following operations are supported. For a formal definition, please review the [Service Description](#).

- [CheckIn](#)
- [CheckOut](#)
- [GetTreeState](#)
- [ListFiles](#)
- [ObtainLock](#)
- [ReleaseLock](#)
- [Validate](#)

The browser's status bar at the bottom shows "Done" and weather information: "Now: Cloudy, 68° F", "Tue: 70° F", and "Wed: 70° F".

Real Client Editor

The screenshot shows the 'Client Text Editor' application window. The main editing area contains a document with the following structure:

- Providing a Dynamic RTCES Client
- Section 1
 - In this demo we discuss how an application can connect with a server to download a document from We also show how locks can be maintained dynamically as users move about the document.
- Section 2
 - A demo of the system would be here
 - Section 2.1
 - Architecture of the system
 - Section 2.2
 - Components of the system
 - Section 2.2.1
 - Client components
 - Section 2.2.2
 - Server components
 - Section 2.2.3
 - Communication/notification components
 - Section 2.3
 - Analysis of atchitecture
- Section 3
 - Simulation design.
 - Simulation results
- Section 4
 - Conclusions
- Section 5
 - References
- Section 6
 - Appendix 1
 - Appendix 2

The right-hand pane displays a tree structure of the document, with nodes corresponding to the sections and sub-sections listed in the main editor. The tree structure is as follows:

- 1
 - 2
 - 2.1
 - 2.2
 - 2.3
 - 3
 - 3.1
 - 3.2
 - 3.3
 - 3.3.1
 - 3.3.2
 - 3.4
 - 3.4.1
 - 3.4.2
 - 3.4.3
 - 3.4.3.1
 - 3.4.3.2
 - 3.4.4
 - 3.4.4.1
 - 3.4.4.2
 - 3.4.5
 - 3.4.5.1
 - 3.4.5.2
 - 3.5
 - 3.5.1
 - 3.5.2
 - 4
 - 4.1
 - 4.2
 - 4.3
 - 5
 - 5.1
 - 5.2
 - 6
 - 6.1
 - 6.2
 - 6.3
 - 6.4
 - 6.5
 - 6.6
 - 6.7
 - 6.8
 - 6.9
 - 6.10
 - 6.11
 - 6.12
 - 6.13
 - 6.14
 - 6.15
 - 6.16
 - 6.17
 - 6.18
 - 6.19
 - 6.20
 - 6.21
 - 6.22
 - 6.23
 - 6.24
 - 6.25
 - 6.26
 - 6.27
 - 6.28
 - 6.29
 - 6.30
 - 6.31
 - 6.32
 - 6.33
 - 6.34
 - 6.35
 - 6.36
 - 6.37
 - 6.38
 - 6.39
 - 6.40
 - 6.41
 - 6.42
 - 6.43
 - 6.44
 - 6.45
 - 6.46
 - 6.47
 - 6.48
 - 6.49
 - 6.50
 - 6.51
 - 6.52
 - 6.53
 - 6.54
 - 6.55
 - 6.56
 - 6.57
 - 6.58
 - 6.59
 - 6.60
 - 6.61
 - 6.62
 - 6.63
 - 6.64
 - 6.65
 - 6.66
 - 6.67
 - 6.68
 - 6.69
 - 6.70
 - 6.71
 - 6.72
 - 6.73

The bottom status panel contains the following information:

- Position Information
 - Cursor Index: 364
 - Active Line: 12
 - Active Node: TreeNode: 3.4.1
- Currently Editing: document_1.txt
- Login: jon ID 1
- Success: 3.4.1
- Lock obtained successfully

Editing Space

Location State

Tree Structure

Messages from Server

Conclusions

- Simulation design and implementation process useful
 - Allows rapid testing/validation of approach
- Communication costs can be reduced using dynamic locking
 - Larger documents
 - Clustered editing
 - Larger number of clients/collaborators