Real-Time Collaborative Editing of Documents

by

Your Name

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy

Department of Computer Science
University of Toronto

© Copyright 2026 by Your Name

Real-Time Collaborative Editing of Documents

Your Name
Doctor of Philosophy

Department of Computer Science
University of Toronto
2026

Abstract

This thesis compares two approaches to synchronising a document that several people edit at the same time, operational transformation and conflict-free replicated data types, and evaluates both on a prototype of a web editor. The abstract should summarise the problem, the method, the results and the conclusions of the work, and be readable on its own.

To my family.

Acknowledgements

I would like to thank my supervisor for the guidance throughout this work, and my colleagues at the institute for the many discussions that shaped it.

Contents

Introduction **1**

1 Collaborative Document Editing **2**
1.1 System Requirements 3

2 Approaches to Synchronization **4**
2.1 Operational Transformation 4
2.1.1 Transformation Function 4
2.2 Conflict-Free Replicated Data Types 4

3 Design and Implementation **6**
3.1 Storing the History 6

4 Evaluation **7**
4.1 Discussion of Results 8

Conclusion **9**

A Supplementary Material **10**

References **11**

List of Tables

4.1 Mean latency and size of stored data by number of users 7

List of Figures

1.1 Typical course of collaborative document writing 2

4.1 Number of edits by day of the week and hour in the observed sessions 8

Introduction

Today, text documents are only rarely written by a single author. Students write term papers in pairs, developers maintain technical documentation together, and research teams finish papers at the last minute from several places at once. For such writing to work, the editor must ensure that all participants see the same text and that no edit is lost.

The aim of this thesis is to compare the two main approaches to document synchronization in real-time collaborative editing, operational transformation and conflict-free replicated data types, and to evaluate them on a prototype web editor. The first chapter defines the requirements for such a system, the second describes both approaches, the third the design of the prototype, and the fourth the measurement results.

Chapter 1

Collaborative Document Editing

By collaborative editing we mean a situation in which several users edit the same document and their changes propagate to one another without manual merging. Depending on how quickly changes propagate, we distinguish three basic modes of work:

- **asynchronous editing**, in which authors exchange entire versions of the document and merge them only afterwards;
- **editing with locks**, in which only one author at a time may edit a given part of the document;
- **real-time editing**, in which every change is immediately reflected for the other users.

This thesis deals exclusively with the last mode. It is the most convenient for users, but it places the greatest demands on system design, because changes originate concurrently in many places and arrive in different orders.

Real-time editing, moreover, changes not only the technique but also the course of the entire collaboration. The typical process of collaborative writing, as described by the interviewed users, is shown in Figure 1.1.

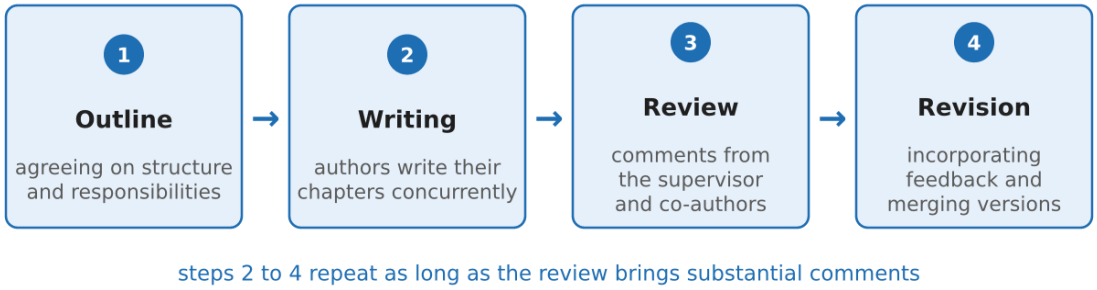

Figure 1.1: Typical course of collaborative document writing

1.1 System Requirements

From interviews with users of existing editors and from the literature, we derived four requirements that a system for collaborative editing must meet:

1. convergence, i.e. once all changes have been delivered, all copies of the document have the same content;
2. intention preservation, i.e. the result of an edit corresponds to what the author saw at the moment of editing;
3. responsiveness, i.e. a local edit takes effect immediately without waiting for the server;
4. robustness, i.e. a short-term loss of connection does not lead to the loss of text in progress.

The first two requirements concern correctness, the remaining two usability. In practice, it is precisely the combination of immediate responsiveness and guaranteed convergence that proves to be the most difficult part of the whole design.

Chapter 2

Approaches to Synchronization

Systems for real-time collaborative editing first apply each change locally and only then distribute it to the other participants. They differ mainly in how they handle the situation in which two changes arise concurrently and each participant receives them in a different order.

2.1 Operational Transformation

Operational transformation builds on the work of Ellis and Gibbs [1], who were the first to describe how to adjust a concurrent operation so that it can be applied even after another operation of which its author was unaware. Each edit is recorded as an insert or delete operation at a particular position in the text.

If, for example, one user inserts a word at the beginning of a paragraph and another simultaneously deletes a word at its end, the position of the deletion must be shifted by the length of the inserted word. This shift is computed by the transformation function.

2.1.1 Transformation Function

The transformation function takes a pair of concurrent operations and returns their adjusted versions. For all copies of the document to converge, the function must satisfy the so-called transformation properties, which are difficult to prove for general operations. Most deployed systems therefore rely on a central server that determines a single order of operations.

2.2 Conflict-Free Replicated Data Types

The second approach is represented by conflict-free replicated data types, collectively referred to by the abbreviation CRDT [2]. Instead of adjusting operations after the fact, they choose a representation of the document in which concurrent operations commute and their order has no effect on the result.

> A replicated data type is conflict-free if any two replicas that have received the same set of updates have the same state, regardless of the order in which the updates were delivered.

Each character of the document is given a unique identifier that remains the same for the entire lifetime of the character. The insertion position is then determined not by a number but by a reference to the identifier of the neighboring character, so concurrent edits elsewhere in the text cannot shift it.

Chapter 3

Design and Implementation

The editor prototype consists of a web client, a synchronization server, and a database in which the server continuously stores the state of the documents. Both the client and the server are written so that the synchronization layer can be replaced and the two described approaches can be compared on the same user interface.

The client maintains a local replica of the document and converts each user edit into an operation on the replicated data type. Operations are sent to the server in batches so that the network is not flooded during fast typing. A simplified processing of an incoming batch on the server is shown in the following listing:

```
def process_batch(document, batch):
    for operation in batch.operations:
        if operation.id in document.applied:
            continue
        document.apply(operation)
        document.applied.add(operation.id)
    broadcast_to_others(document.id, batch)
```

Because the operations commute, the server does not need to order or transform the batches. It suffices to ensure that no operation is applied twice and that every batch reaches all connected clients.

3.1 Storing the History

The server stores not only the resulting text but the entire log of operations. From the log, any earlier version of the document can be restored and the replica state of a client that has been disconnected for a longer time can be rebuilt. This approach corresponds to an architecture based on an immutable event log, as described by Kleppmann [3]. To keep the log from growing without bound, the server periodically compacts it into snapshots.

Chapter 4

Evaluation

We compared the prototype with a reference implementation of operational transformation on a set of recordings of real editing sessions. For each session we measured the latency, i.e. the time between an edit by its author and its display to the other participants, and computed the mean latency of the session according to (4.1):

$$\bar{t} = \frac{1}{n} \sum_{i=1}^{n} \left(t_i^{\text{delivery}} - t_i^{\text{edit}} \right) \tag{4.1}$$

where n is the number of operations in the session. The measured values for different numbers of concurrently working users are summarized in Table 4.1.

Table 4.1: Mean latency and size of stored data by number of users

Users	**OT latency (ms)**	**CRDT latency (ms)**	**Data (kB)**
2	38	35	12
5	52	41	19
10	87	48	31
20	164	63	58

As the number of users grows, the latency of operational transformation increases much faster, because the server must transform each operation against all concurrent operations. The price of the lower latency of CRDT is a larger volume of stored data, since each character carries its own identifier. The distribution of edits over time on which the load test is based is shown in Figure 4.1.

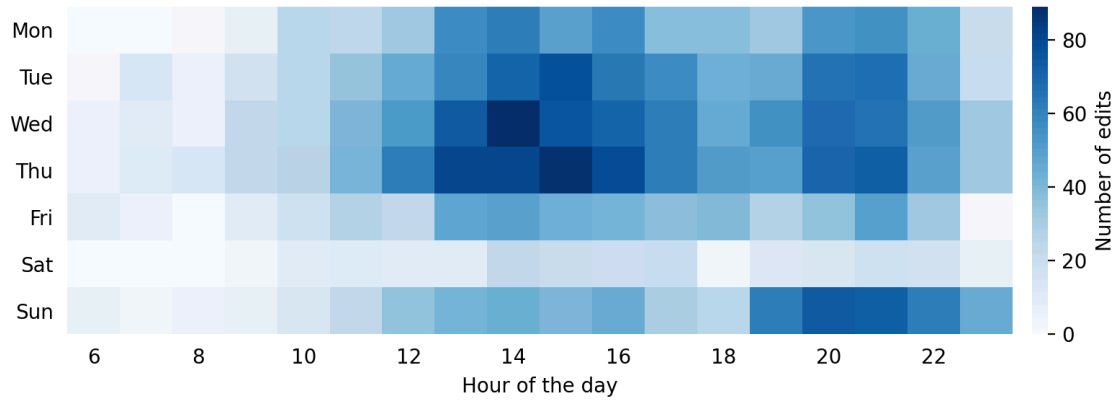

Figure 4.1: Number of edits by day of the week and hour in the observed sessions

4.1 Discussion of Results

The results confirm that for smaller teams both approaches are equivalent and the simplicity of implementation is the more decisive factor. With a larger number of concurrent users, CRDT has a clear advantage both in latency and in robustness to connection outages. Supplementary interviews with users, conducted according to the principles of qualitative research [4], showed that users do not perceive a difference in latency below one hundred milliseconds, whereas they consider the loss of text in progress after a connection outage to be a serious problem.

Conclusion

This thesis compared operational transformation and conflict-free replicated data types as two approaches to document synchronization in real-time collaborative editing. Using a prototype web editor, we showed that CRDT allow a simpler server without central ordering of operations and offer lower latency with a larger number of users.

The main disadvantage of CRDT remains the larger volume of stored data. The next step will therefore be to examine how the data size is affected by the periodic compaction of the log into snapshots and how the prototype behaves in long-term operation with tens of thousands of documents.

Appendix A

Supplementary Material

An appendix holds material that supports the text but is not needed to follow it, such as the questionnaire of a survey, measurement data or a longer listing.

References

[1] C. A. Ellis and S. J. Gibbs, "Concurrency control in groupware systems," in *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, New York: ACM, 1989, pp. 399–407. doi: 10.1145/67544.66963.

[2] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, "Conflict-free replicated data types," in *Stabilization, Safety, and Security of Distributed Systems*, Berlin: Springer, 2011, pp. 386–400. doi: 10.1007/978-3-642-24550-3_29.

[3] M. Kleppmann, *Designing data-intensive applications*. Sebastopol: O'Reilly Media, 2017.

[4] J. Hendl, *Kvalitativní výzkum: základní metody a aplikace*. Praha: Portál, 2005.