Distributed Systems Project Report	**Team 20**
Mentor: *Tanish Lad*	Ahish Deshpande Pranav Kirsur Sahil Bhatt Vikrant Dewangan

Building a Real Time Collaborative Text Editor

1 Introduction

A collaborative real-time editor is a type of collaborative software or web application which enables real-time collaborative editing, simultaneous editing, or live editing of the same digital document, computer file or cloud-stored data – such as an online spreadsheet, word processing document, database or presentation – at the same time by different users on different computers or mobile devices, with automatic and nearly instantaneous merging of their edits.

With asynchronous collaborative editing systems such as git, each user must typically manually submit (publish, push or commit), update (refresh, pull, download or sync) and (if any edit conflicts occur) merge their edits.

Due to the delayed nature of asynchronous collaborative editing, multiple users can end up editing the same line, word, element, data, row or field resulting in edit conflicts which require manual edit merging or overwriting, requiring the user to choose which edits to use or (depending on the system and setup) automatically overwriting their edits or other people's edits, with or without a warning.

Real-time collaborative editors perform automatic, periodic, often nearly instantaneous synchronization of edits of all online users as they edit the document on their own device. This is designed to avoid or minimize edit conflicts.

Some popular real time collaborative text editors are Google Docs and Notion.

2 Problem Statement

Create a real time collaborative text editor using CRDTs + WebRTC.
Maintain consistency between users and reduce latency.

3 Technologies used

We used the following technologies and concepts to implement our project:

1. Conflict-free Replicated Data Types (CRDTs)
2. WebRTC

3. ReactJS

3.1 CRDTs

A Conflict-free Replicated Data Type (CRDT) is a data structure that simplifies distributed data storage systems and multi-user applications. It helps in many applications, particularly those in the area of collaborative software (such as Google Docs), where conflict resolution is important.

3.2 WebRTC

WebRTC allows us to use Real Time Communication (RTC) in a direct peer-to-peer mode through the use of APIs.

3.3 ReactJS

We use ReactJS to build the website and user interface for our collaborative text editor.

4 Text Editors

A text editor is a space where you can **insert** or **delete** text characters and then save the resulting text to a file. Each character has a value and a numerical index that determines its position in the document. For example, with the text "HAT", the first character has a value "H" and a position of 0, "A" has position 1, and "T" has position 2.

A character can be inserted or deleted from the text simply by referencing a positional index. To insert a "C" at the beginning of the text, the operation is `insert("C", 0)`. This insertion results in the remaining positions being shifted (or incremented) by 1. Now to delete the "H", the operation is `delete(1)`.

The operations performed in a text editor can be seen in Figure 1.

5 Collaborative Text Editing

For a system with multiple text users editing the same document, the following conditions must hold:

1. **Commutativity**: Concurrent insert and delete operations converge to the same result regardless of the order in which they are applied.
2. **Idempotency**: Repeated delete operations produce the same result.

If these conditions do not hold, it can lead to inconsistency. Hence, the challenge while building a real time collaborative text editor is to ensure both consistency and latency at the same time.

For this purpose, there are two main approaches in the literature: Operational Transformation(OT) and Conflict-Free Replicated Data Type (CRDT).

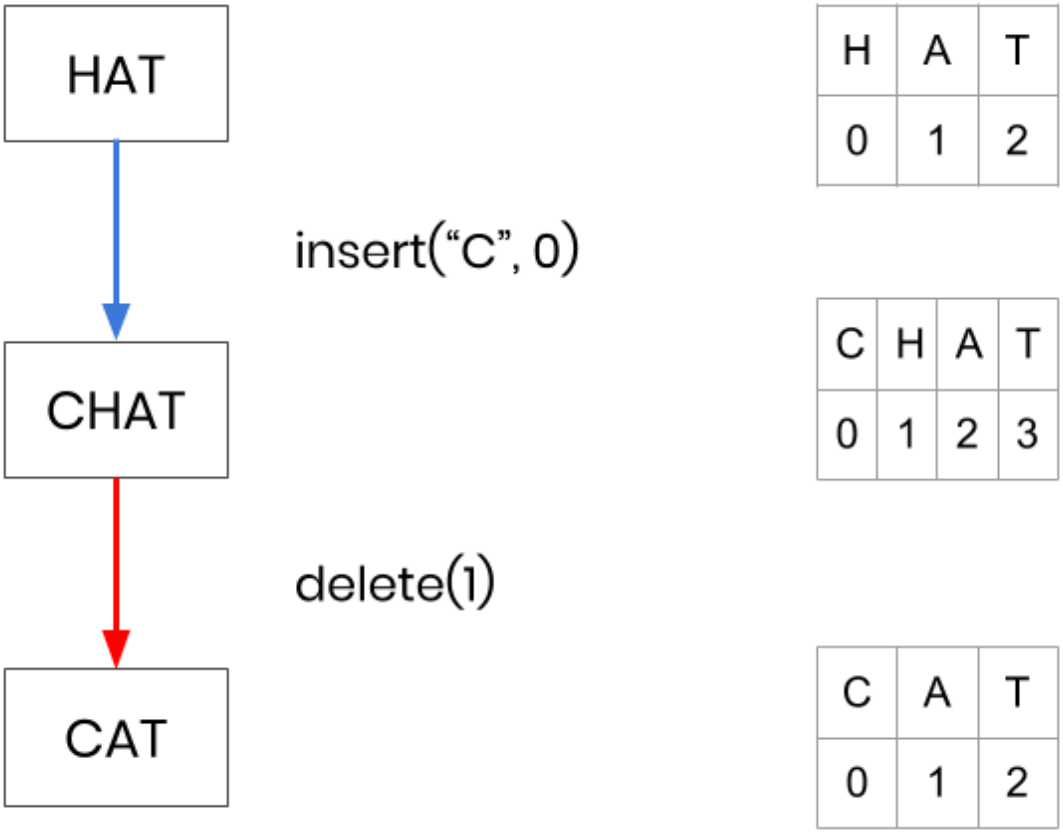

Figure 1: Operations in a text editor

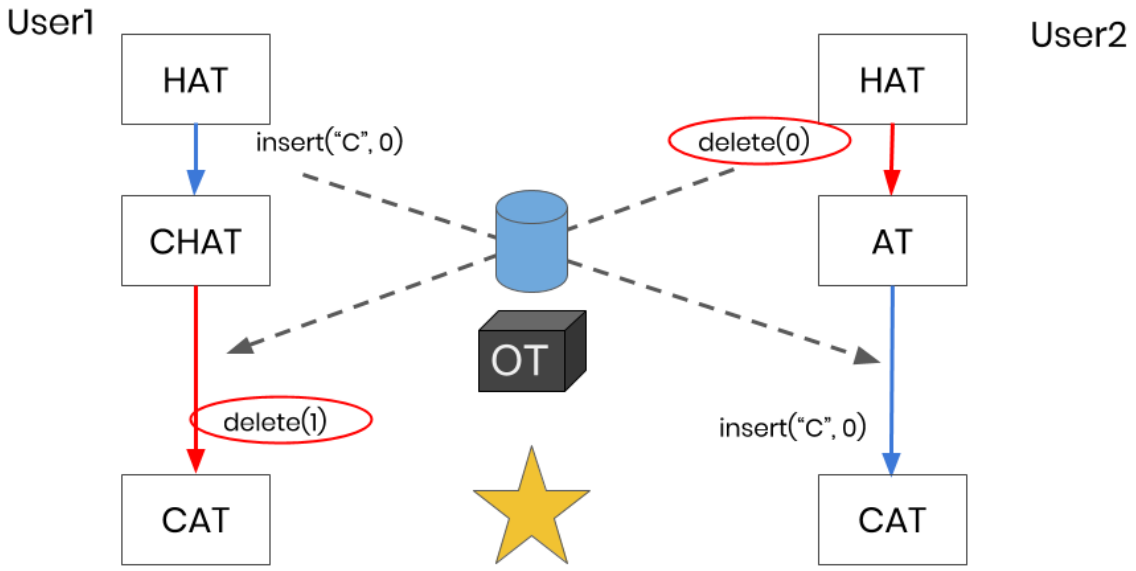

Figure 2: Operational Transformation

6 Operational Transformation

Operational Transformation (OT) is an algorithm that compares concurrent operations and detects if they will cause the documents to not converge. If the answer is yes, the operations are modified (or transformed) before being applied.

As seen in Figure 2, when User1 receives the `delete(0)` operation from User2, OT realizes that since User1 inserted a new character at position 0, User2's operation must be transformed to `delete(1)` before being applied by User1.

When a user tries to delete a character that's already been deleted, OT can easily recognize this and skip the operation.

Thus, OT can ensure commutativity and idempotency of operations for collaborative editing.
However, the implementation is quite complex hence we prefer CRDT for our project.

7 Conflict-Free Replicated Data Type (CRDT)

Conflict-free replicated data type (CRDT) is a data structure which can be replicated across multiple computers in a network, where the replicas can be updated independently and concurrently without coordination between the replicas, and where it is always mathematically possible to resolve inconsistencies that might come up

CRDT is an alternative strategy that was discovered by researchers while trying to strengthen and simplify OT.

OT is implemented without changing the fundamental structure of a basic text editor.

Like a basic editor, OT treats each character as having a value and an absolute position. And to achieve the commutativity and idempotency required by a collaborative text editor, OT relies primarily on an algorithm.

CRDTs take a different approach. Rather than treating the characters as just having a value and absolute position; they change the underlying data structure of the text editor.

Properties are added to each character object that enabled commutativity and idempotency. Using a more complex data structure allows for a much simpler algorithm than OT.

There are many different types of CRDTs with different requirements for different use cases.

For our project, CRDT is used for achieving consistent data between replicas of data without any kind of coordination (e.g. transformation) between the replicas.

We use the CRDT implementation of the library `yjs` which is based on the research paper [1]. The algorithm used is known as YATA.

8 CRDTs for collaborative text editors

The CRDTs used for collaborative text editors have two main requirements:

8.1 Globally Unique Characters

Each character object must be globally unique.
This can achieved by assigning Site ID and Site Counter properties whenever a new character is inserted. Since the Site Counter at each site increments whenever inserting or deleting a character, we ensure the global uniqueness of all characters.

With globally unique characters, when a user sends a message to another user to delete a character, it can indicate precisely which character to delete.

When a user receives a delete operation from another user, it looks for a globally unique character to delete. If it has already deleted that character, then there is nothing more to delete.

By ensuring globally unique character objects, we can achieve idempotency of delete operations.

8.2 Globally Ordered Characters

We need all the characters to be globally ordered and consistent. Thus, when a user inserts a character, it will be placed in the same position on every user's copy of the shared document.

When inserting or deleting a character, the positions of surrounding characters would sometimes need to be shifted accordingly. Since characters can be shifted by a user without the other users' knowledge, we can end up in a situation where insert and delete operations do not commute.

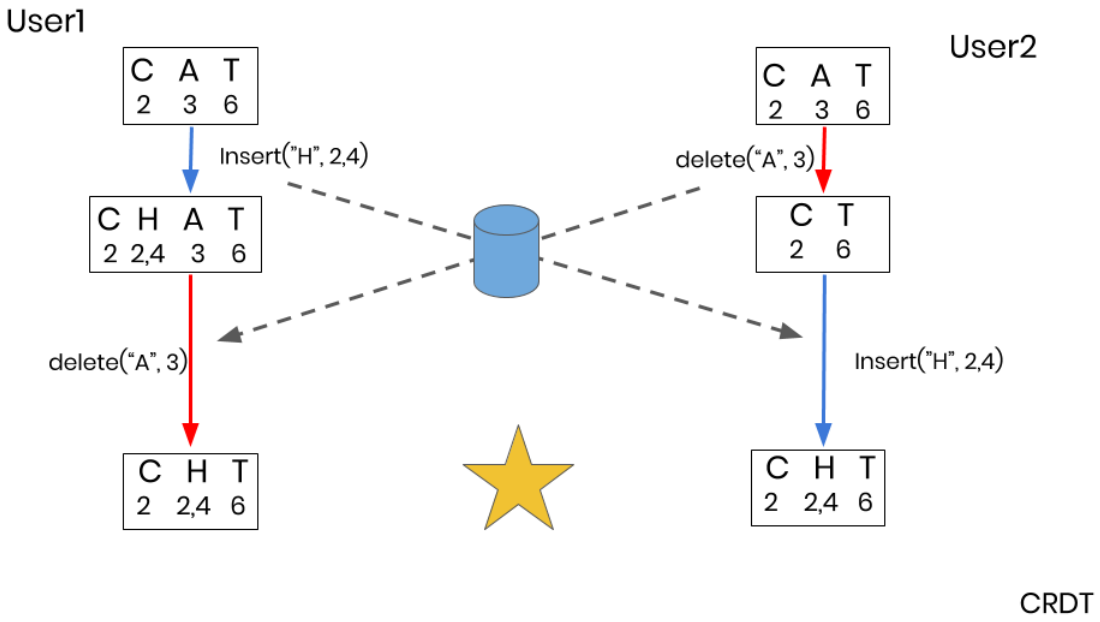

Figure 3: CRDT

We can avoid this problem and ensure commutativity by using fractional indices as opposed to numerical indices.

These ensure commutativity of insert and delete operations, as can be seen in Figure 3.

9 WebRTC

A central relay server for communication would have several disadvantages:

1. High latency between peers
2. Costly to scale
3. Requires trust in the central server

For communication between peers, rather than use a central server, that can slow communication and reduce latency, we use WebRTC to extablish direct connections between peers.

WebRTC (Web Real-Time Communication) is a free and open-source project providing web browsers and mobile applications with real-time communication (RTC) via application programming interfaces (APIs).

It allows audio and video communication to work inside web pages by allowing direct peer-to-peer communication, eliminating the need to install plugins or download native apps.

We use WebRTC for communication between peers to ensure that our collaborative text editor works in real time.

10 Conclusion

Thus, we have built a real time collaborative text editor, using CRDTs to ensure consistency between users, and WebRTC to reduce latency and ensure real time communication.

References

[1] Petru Nicolaescu, Kevin Jahns, Michael Derntl, and Ralf Klamma. Near real-time peer-to-peer shared editing on extensible data types. pages 39–49, 11 2016.