Design collaborative text editor

Planning

1. Requirements
    a. Functional
    b. Non-functional
2. Component Architecture
3. Data Model / Data APIs
4. Data Entities
5. Data Store
6. Performance and Optimization
7. Accessibility
8. Security

Functional Requirement

1. Low latency
2. Real time update
3. Having smooth experience.
4. Offline support
5. Allow multiples user to edit the document at the same time
6. Conflicts Resolution

Non-Functional Requirement

1. It should support variety of devices
2. Responsive nesss
3. Web vitals.
4. Design Consistency

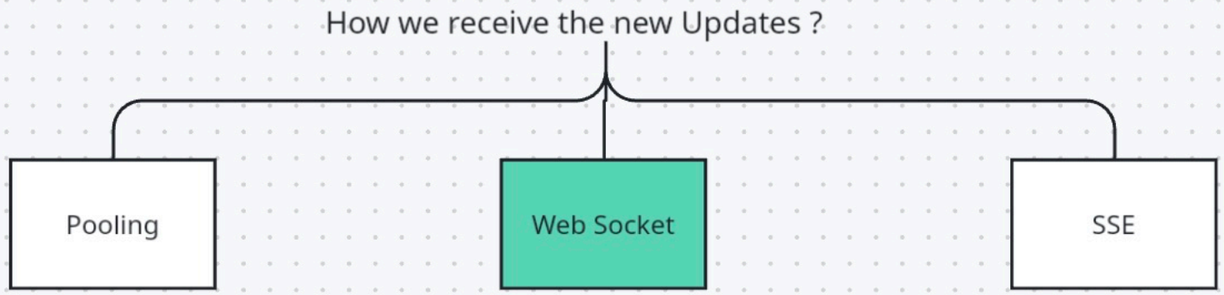
Design a collaborative editor that allows multiple users to work on a single document or file simultaneously across a network

eg: Google docs, Microsoft word.


```
Type UpdateDocument = {
    DocumentID: String
    userId: String
    Operation: {
        type: ChangeType
        position: Integer
        text: String (insertedText)
        length: Integer
        timeStamp: String
        version: String
    }
}

enum ChangeType = {
    INSERT,
    DELETE,
    UPDATE
}
```

```
{
 "documentId": "12345",
 "userId": "user_678",
 "operation": {
  "type": "insert", // or "delete", "update"
  "position": 105, // character index where the change happened
  "text": "Hello", // inserted text (if type = insert)
  "length": 5, // number of characters deleted (if type = delete)
  "timestamp": "2025-04-26T14:10:00Z",
  "version": 25 // optional: to track document version
 }
}
```



Calls to the sever at
specific interval, to get the update

Unidirectional connection

we can close the connection from
client side

We have two way connection
If any update present the server will
send to us

we can close the connect from client
side

Unidirectional connection
We can't send data from client
only from sever we can receive

We can't close the connection from
client side

We have to write the logic in
sever , how we are going to handle conflicts
and keep every user state common

Approach - 1

Can we go ahead with the lock-based approach,
What does it means, Since 15 users are using the same docs, we will allow them in
The Round Robin fashion, this is also known as **Pessimistic concurrency control**
This is not a good approach, for collaborative editor

Droping this appraoch

Lead AY

Appraoch - 2 **(We need lock free architecuture)**

Optimistic concurrency control

Sync Strategies
1. Event Parsing (OT - Operational transformation)
2. Differential synchronization.

Let's talk about OT

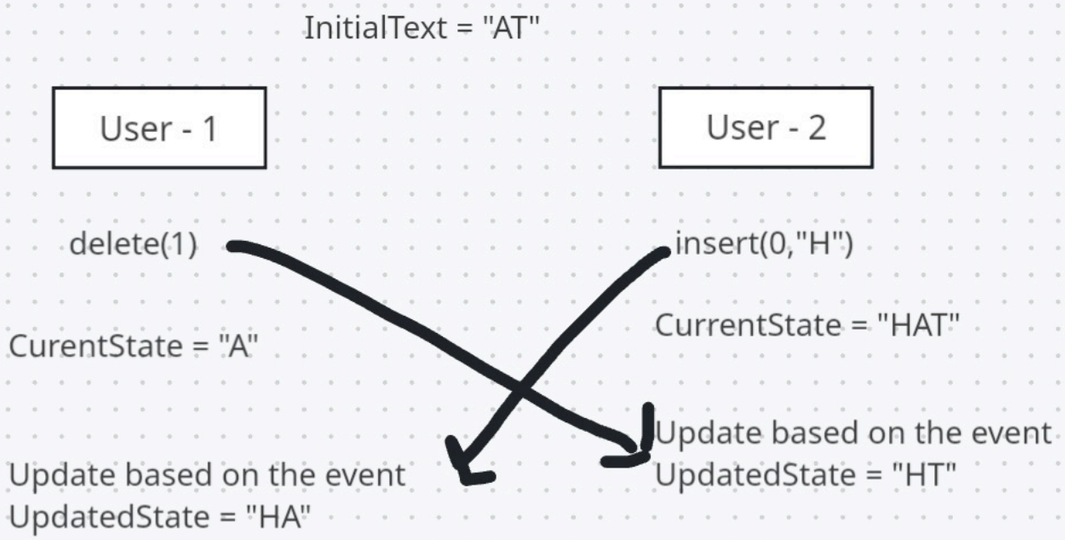
Operational Transformation (OT) is a **conflict resolution algorithm** used in real-time collaborative applications (like Google Docs) to allow **multiple users to edit the same document at the same time**.

In OT every changes, is done by user is considered as an event, which we are keep sending to server and receiving from the server

eg: insert(), delete(), update() .....etc.

there can be so many other operations like, changing the color, changing the background color , changing the font size , changing the font family etc

But this is not correct, right because the final state should be "HA" for both the cases, So we can't simply apply event, what we received we need to figureout something



Final State Should be "HA" for both the case

So, we need to do some transformation of the operation which we are getting, that's why this Algo is also known as Operation transformation.

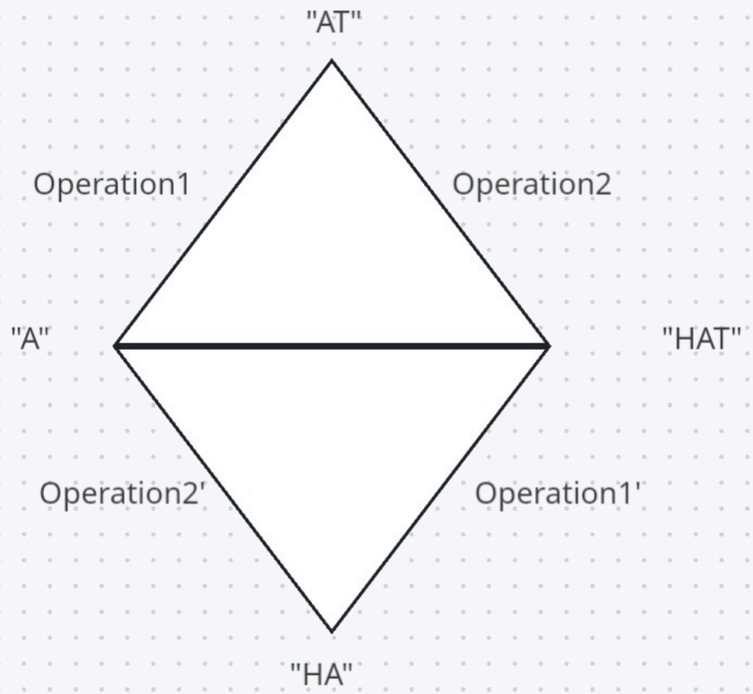
Formula of OT

tForm(Operatin1, Operation2) => {Operation1' , Operation2'}

and then we have to apply the Opeartion1' with the User2 and Operation2' with the User1

Opeartion1 = Delete(1)
Operatoin2 = Insert(0, "H")

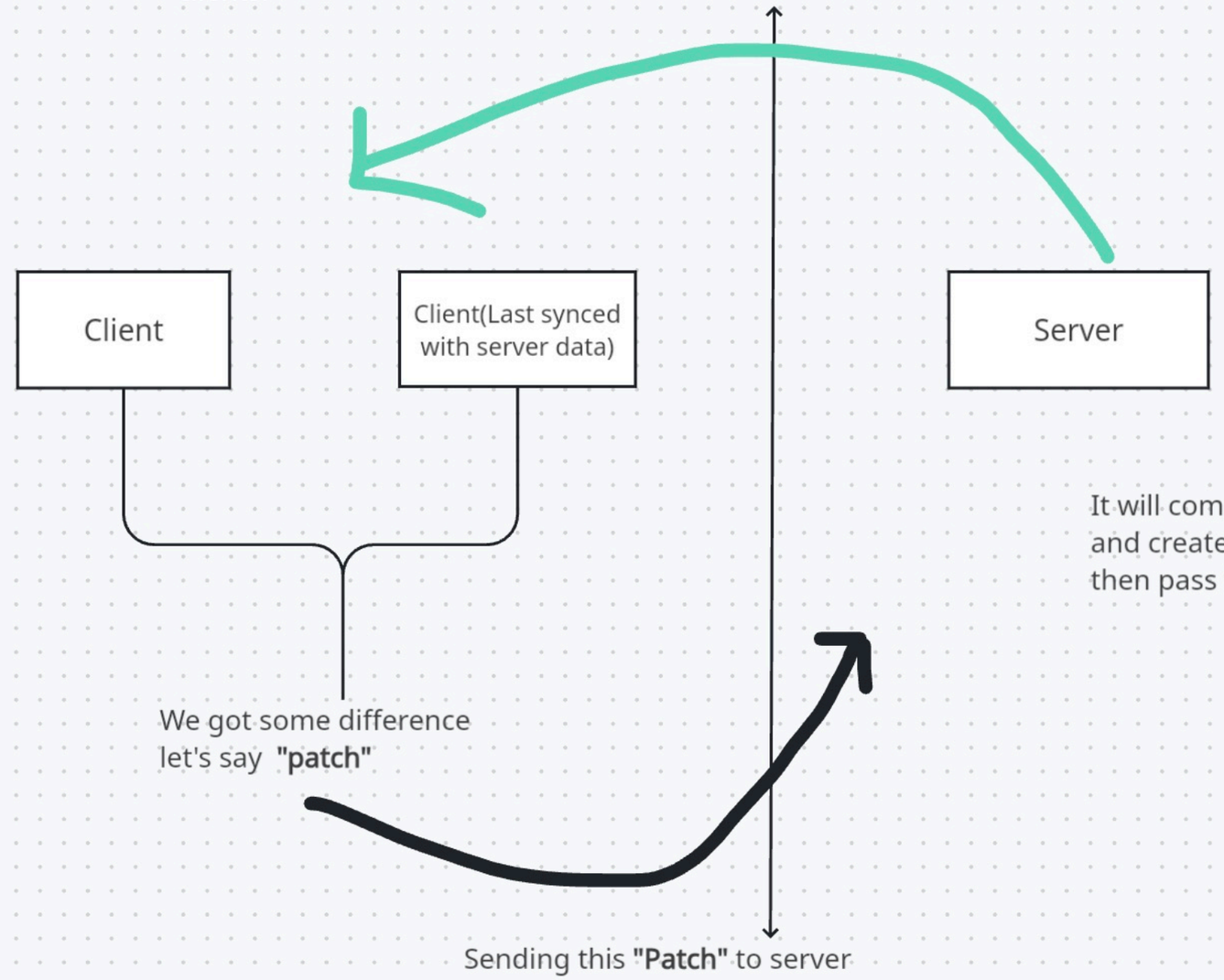
Opeartion1' = Delete(2)
Operatoin2' = Insert(0, "H")



Let's talk about Differential synchronization

Differential Synchronization (aka **diff sync**) is a **real-time synchronization technique** where:

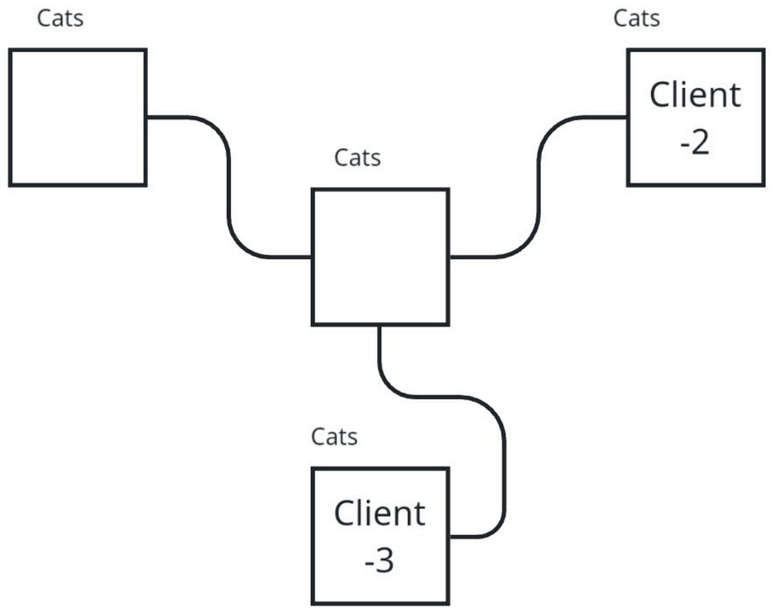
- **Clients and server** maintain **copies** of the document.
- **Periodically**, the **differences** ("diffs") between a client's version and its *shadow copy* are calculated.
- Only the *diff* (not the full document) is sent to the server.
- Server **merges** these diffs smartly, updates the master copy, and sends diffs back to other clients.



Technique to handle the collabrative text editor

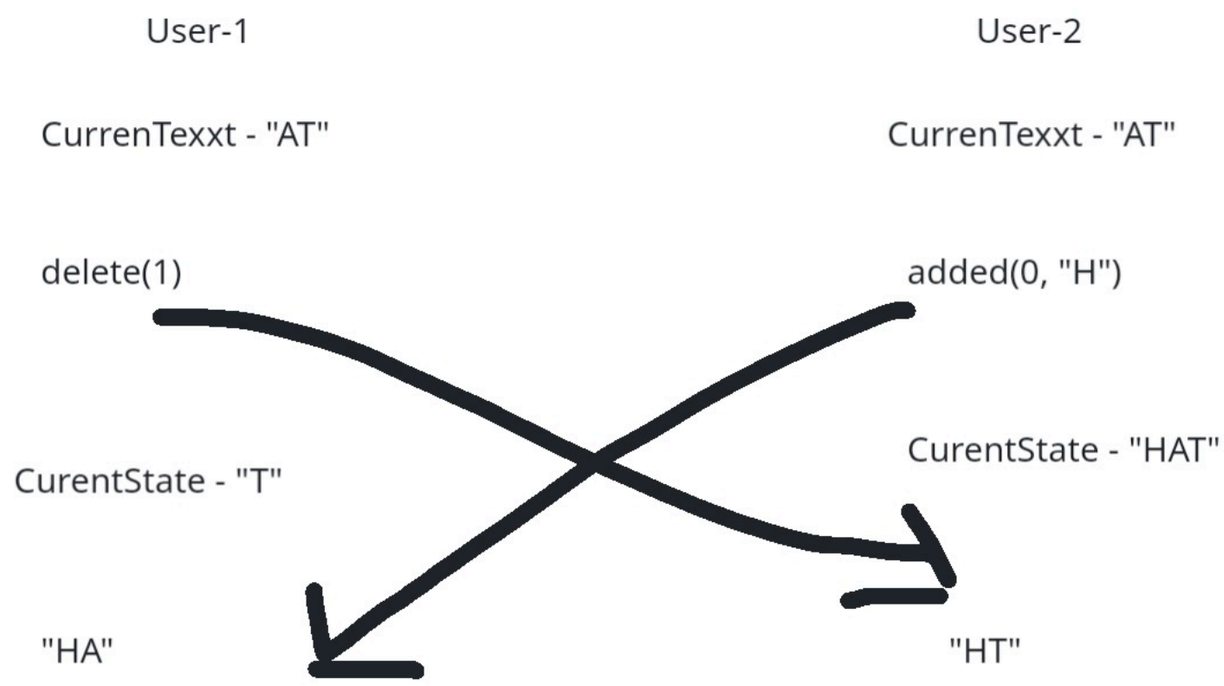
1. Locking Strategy (Pissimestic conncurency control)
2. Event Passing (Operational transformation)
3. Three-way merges

Locking Strategy



Event Passing (Operation Transformation)

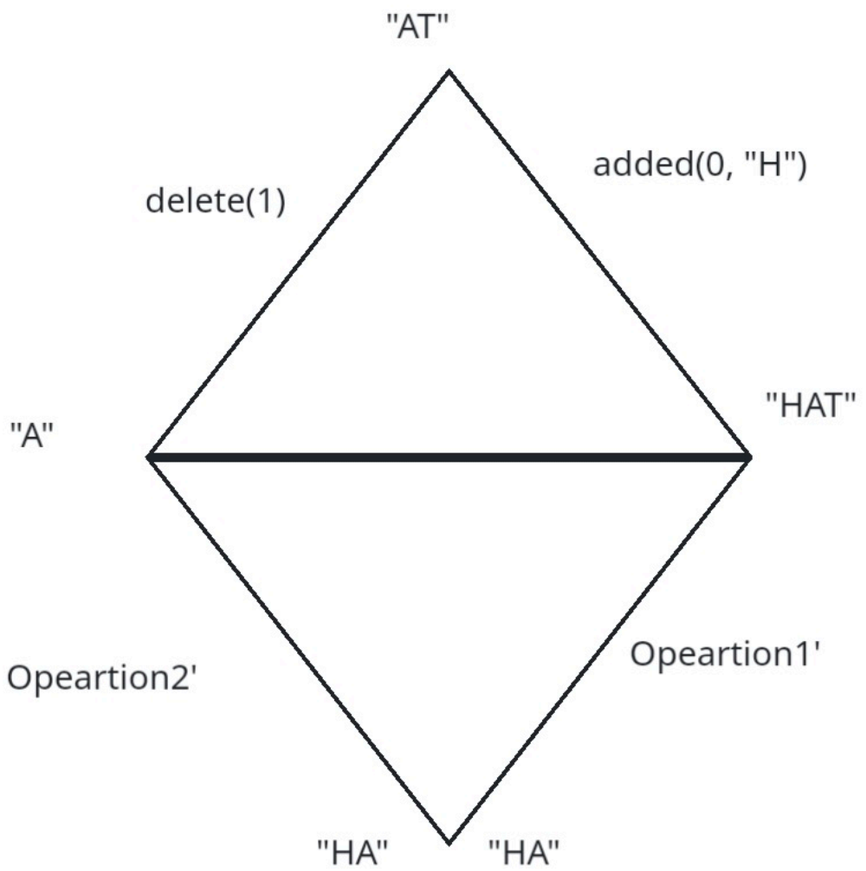
Event Passing (Operation Transformation)



eTransfor(Operation1, Operation2) = {Opeartion1', Operation2'}

Opeaton1 = delete(1)
Opeartion2=added(0,"H")

Operation1' = delete(2)
Operation2' = added(0, 'H")



Three Way Merge