CN Project Design Documentation

How to run the code:

1. Clone the given github repository
2. Run make on the terminal
3. Run ./server on a machine
4. Change the ip address in the client.cpp to the ip address of the server machines ip address
5. Run ./client
6. Type commands or text in the terminal and see the doc being updated in client_ouput.txt

Features:

Our collaborative documentation allows the below commands:

- Typing any text in the terminal inserts it in the doc at the cursor position
- Typing /exit causes the client to exit the doc
- Typing /backspace {lenght} backspaces from the cursor position till the given length
- Typing /cursor {index} moves the cursor to the given position
- To type “/exit” or any other command in the doc, prepend a /. Eg: “//exit”
- A new client logging in will get the existing doc

Implementation:

We used a basic server client epoll implementation taken from code explained in tutorials and built our implementation on it.

We started by defining an operation class to store and send updates made by a client.
Operations class was defined as

```
class Operation {
   string type;
   int position;
   string content;
   string clientId;
   int localVersion; // will be discussed later
}
```

Now whenever a user enters something in their terminal it sends the operation as a string of the form “OP|type|position|content|clientID|localVersion”.

The server receives such operations, makes necessary changes to the index and broadcasts this packet to all users.

The server creates an instance of the client to the epoll instance when a new client joins. All its information is stored in an unordered_map on the server. Upon connecting, a new client immediately receives the full current state of the document to ensure they start in sync with the server.

The server maintains a linked list of operations to maintain history.

Handling clashes:

The server maintains a global version and every time a client sends an operation packet it includes a local version.
The local version represents the version of the doc the user was looking at when the required change was made

If the local version=server version the operation is applied immediately else if it is less than the server version, i.e, the client is seeing an older version, we go to that point in the linked list and apply to all operations that have occurred since. Once the operation is done we give it a new global version and broadcast the operation to all other clients.

On the client side, on the input thread we check if the text written is a command like ‘/exit’, if not we treat it as text and create a packet and send it to the server. On the receiver thread we listen for incoming packets from the server, deserialize it and make the necessary changes.

Github Link: https://github.com/sjwl07/Collaborative_docs
Video Demo Link: