PROJECT DESIGN AND PROTOCOL

ericadu - connieh - kahuang

Our document will consist of the **cursor positions** of each of the clients and the **text** of the document.

The cursor positions of the document will be represented by a unique string id and a pointer to an element in a linked list that describes the text (described in greater detail below).

The text of the document will be stored as a LinkedList and each element of this list will be one character of text. The LinkedList will store the following: a forward pointer to the next link (null if none), a backward pointer to the previous link (null if none), a string value representing this character, whether or not this character is a dummy character and whether or not this link has been deleted or not. The first element when the LinkedList is initialized is a dummy character (place holder). Paragraphs are separated by dummy characters. There are three mutating operations (edits) that can be performed on a document's text state: insert, delete, enter.

EDIT ::== INSERT | DELETE | ENTER

INSERT ::== inserts the given character behind the current cursor position (creates a new link and inserts it behind this one by re-assigning pointers). Cursor position isn't changed for any and all other cursors.

DELETE ::== deletes the character behind the given cursor position. (deletes the link and re-assigns the pointers as necessary). Cursor position isn't changed. Cursors that pointed to the deleted character now point to the same link that this cursor points to.

ENTER ::== inserts a dummy character behind the current cursor position. Cursor position isn't changed for any and all other cursors.

We use a linked list to represent the text because it naturally preserves the relative ordering of all the cursors. If we were to use strings to represent the lines of text, then we would have to manually curate the positions of all the cursors (cursors would have to point to a line and an index). This leads to issues down the line with concurrency because edits mutate the string which can lead to edits not editing the document correctly. Since our cursors refer to an object and not coordinates we don't have to worry about how edits shift other cursors.

Mutating the cursor position from the client side consists of two operations:

Left ::== traverse the linked list backwards once. If the backwards pointer is null, don't do anything.

Right ::== traverse the linked list forwards once. If the forwards pointer is null, do nothing.

Every cursor movement GUI-side (client-side) can be translated into a series of lefts or rights which are then sent to the server as Edits.

Protocol

When the first user connects to the server, the server will generate a unique client id that is attached to that particular user. A new document is formed on the server, with a dummy character. The client id is then associated with a pointer and cursor position. The pointer is on the link with a dummy character, and the cursor position is functionally and visually behind the dummy character.

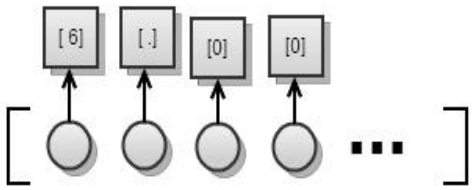
The server will also generate a unique client id for each subsequent user that connects to the server, who will have a pointer and cursor position associated with their id. If there is a "first" link already established, the pointer will be initialized to the "first" link. Otherwise, the pointer will again point to the link with the dummy character. Basically, cursors are instantiated to the first character for new clients.

When a key on the keyboard is pressed by a client, a request for an EDIT is put into a queue. Each key pressed by any given client will be put into this request queue on the client side which is then sent to the server's queue. The server will continue to execute tasks as long as the queue is not empty.

Since we service these edits one at a time, there is no need for locking.

On the server-side we also keep track of how many edits a particular client has in the server-side queue. We increment by one when it enters the queue, and we decrement when we service that edit. When a client's count reaches 0, we send them back the server's copy of the document.

Linked List Representation



Protocol