Alan Bird, Jordan Otsuji, Joe Rodman, Kyle Charlton, Kai Zheng

Overview

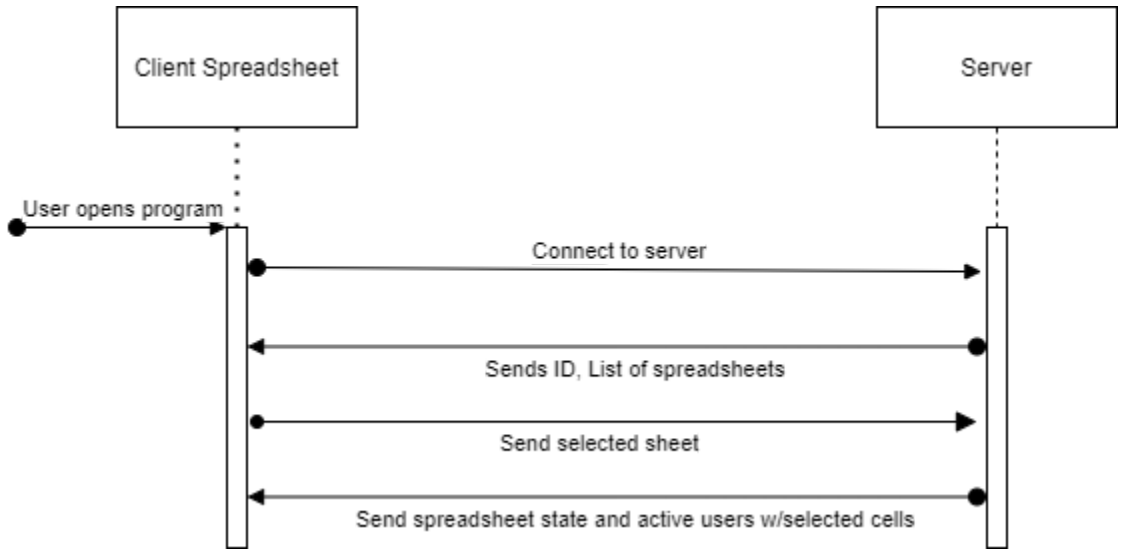
The client's main job is to receive raw data from the server as strings, then parse, evaluate, and display the current state of the spreadsheet through a GUI. Every time the client receives data, it will immediately process and display it. The types of data that the client will accept include raw string data tied to specific cells on the spreadsheet, and cells that are currently occupied by other users. For cell data, clients will evaluate the type of data (string, integer, formula) and evaluate its interaction with other cells–if a circular dependency, mathematical error (divide by 0), invalid formula, or other problem is detected, the associated cells will display a specified error message. The client will still send the server the cell edit that causes an error, and all other clients will receive the cell edit that causes the error. The client will not save any local data beyond the current session, and will rely on the initial handshake with the server to load a document. The client will never request a save, it will rely on the server to save changes automatically.

The server will handle communication between connected clients, and the short-term/long-term storage of the spreadsheet data. Once a client connects, the server will send the client all of the non-empty cells in the spreadsheet, the string data associated with them, currently connected users, and the cells they're currently occupying. The server will listen for commands from the client including connections, edits, cell selection, and other commands specified below. As clients send edits to the server, the server will immediately execute the edit on its internal spreadsheet, saving the edit to the server's history for both the spreadsheet as a whole and for the individual cell. The server will not immediately communicate changes to the clients, but instead, will send a json message to each client after a certain (very short) interval, containing all of the changes made within that interval. The server won't do any technical evaluation, completely ignoring mathematical errors, circular dependencies, and other errors, leaving them to be dealt and displayed by the client. All edits will be handled on a first-come-first-serve basis, and messages received at the same time will be processed in order of which processing thread arbitrarily reaches the lock first. The server will save the state of the spreadsheet to long term storage on a chosen time interval (an implementation details).

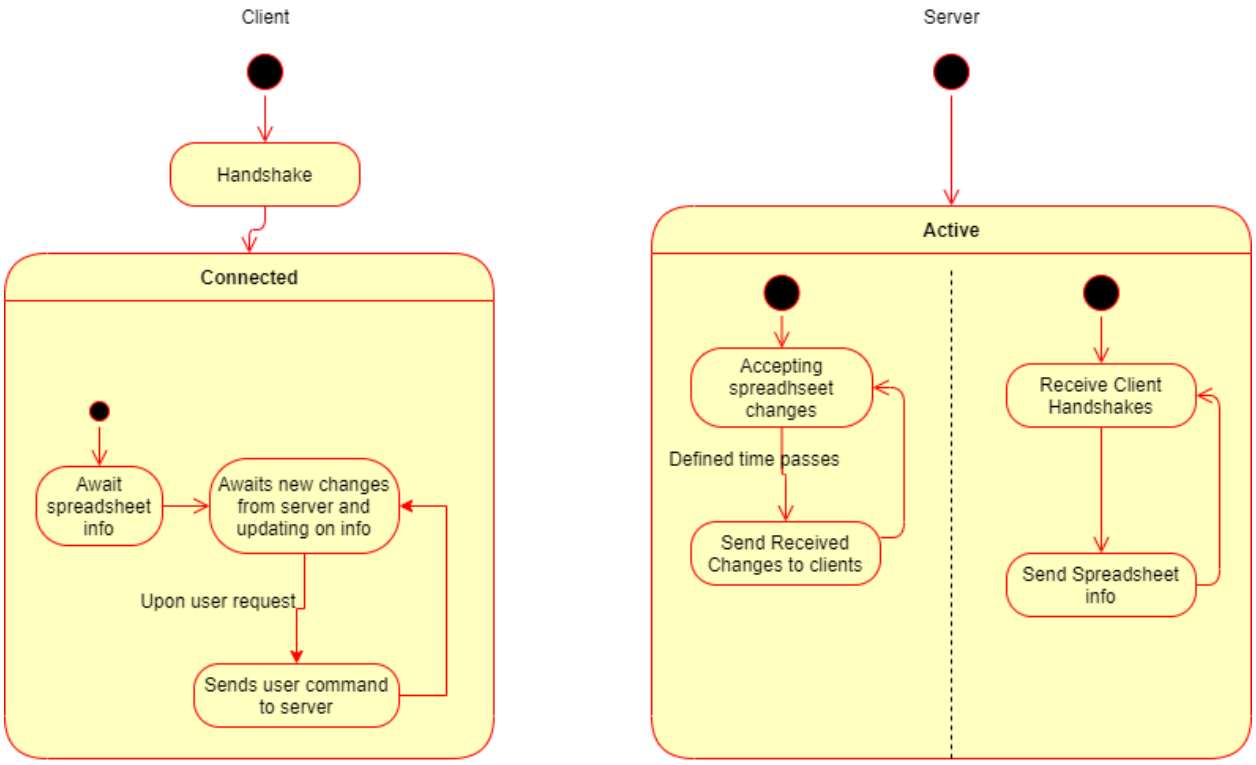
Clients and the server will communicate via TCP sockets.

Connection Details

Every message is identified by a command, followed by the data that should go with that command. All of these messages will be sent and parsed as JSON objects. The client will never timeout. The client has an option to disconnect from the server. The server will run on port 41968. With the server running, the handshake is as follows:



Now that the user is connected, it can make edits to the spreadsheet. The general state of the client and server is shown in the diagram below.



Once activated, the server will continuously accept client connections and go through the handshake protocol defined above. It will asynchronously accept spreadsheet changes from and send update messages to previously connected clients.

Example messages:

Messages are formatted in JSON:

{ Command: “connect”, Name: “John” }

All cell IDs are sent uppercase:

A4 not a4

The client will send the server the following messages, and the server will respond accordingly

Client message	Server action
Command: “cellSelection” CellID: “A4”	Sends cell selection message to other clients and updates other clients with the following message: Command: “cellSelection”, CellID: “A4”, ClientID: “John1”
Command: “rename” Name: “new spreadsheet name”	Renames sheet on server and updates other clients using that sheet with the same message
Command: “new” Name: “spreadsheet name”	Creates a new empty spreadsheet on the server
Command: “undo”	Undoes the spreadsheet to the last change. Sends an update command to all clients using that spreadsheet Command: “update”, CellID: [“A3”, “A4 * A3”]
Command: “revert” CellID: “A4”	Reverts the change of the provided cell in the server spreadsheet. Sends update command to each client using that spreadsheet Command: “update”, Cell: [“A3”, “A4 * A3”]
Command: “edit” CellID: “A3” Contents: “A4 * A3”	Edits the server spreadsheet with the following data, and sends an update command to all clients using that spreadsheet Command: “update”

	Cell: [“A3”, “A4 * A3”]
Command: “close”	Saves the current state of the spreadsheet? Closes the connection to the socket. Sends message to all clients who were using that spreadsheet Command: “clientClosed” ClientID: “John”
Command: “getFiles”	Server responds with a list of files that are available to access as follows: Command: “getFiles”, Files: [“fileName1”, “fileName2”, “fileName3”, ...]
Command: “open”, File: “fileName”	Updates server and all relevant clients for what file the user is now viewing. Updates client with data of opened spreadsheet as follows: Response to Connecting User: Command: “update”, Cell: [“A3”, “A4 * A3”], Cell: [“B7”, “12”] … Response to Users Already Connected to Spreadsheet: Command: “newUser” User: [userID, “Name”]

If a server detects that a client has disconnected, it will send a client disconnection message to the remaining connected clients.

Command: “clientClosed”
ClientID: “John”

Error Handling:

If a client tries to send the server the message, and fails, the local change that was made is reverted. If the server fails to send a message to a client, it disconnects that client and forces it to go through the reconnection handshake again.