Report_f85709jw

Exercise 1 Report

The Protocol

The protocol is very simple, there are 3 keys stored on the server:

1. connected: The number of clients connected
   - This stores the values of '1' or '2'
2. lock: A lock to indicate which client gets to send a message
   - This stores the values of '1' or '2'
3. message: The last message sent
   - This stores the value of the last message sent as a string

All of these fields are expanded on in the next section:

Synchronisation Between Clients

The protocol achieves synchronisation between 2 clients through assigning the roles of 'client 1' and 'client 2' based on which of the clients were first to connect to the server.

At the beginning of the execution of the program with no connected clients, the server key 'connected' has a null value (or the cell doesn't exist). The client changes this to make the value 1, and sets itself as client 1. When the other client begins the program, they'll find that the 'connected' is 1, indicating that they will be client 2. client 2 sets 'connected' to 2, indicating to client 1 that both clients are now connected.

Initially, the 'lock' is set to 1, meaning client 1 gets to message first. Client 2 blocks until the lock changes to 2, at which point client 2 prints the 'message' field. Meanwhile client 1 allows the user to set a message which will change the 'message' field, at which point the client switches the lock to 2. This keeps alternating until the clients are done talking. The lock and the explicit ordering (client 1 goes first) avoids the system from deadlocking.

How to Test

In order to test, run

```
python3 imclient_f85709jw.py
```

for each client.

On the first client run, it should display `Waiting for client 2 to connect..`. After connecting both clients, the first client to connect should be able to type a message. This tests all of the protocol, the first client polls the server until the number of connected clients is 2. The lock is tested as client 2 waits for the lock to change. The message is tested as client 1 gets to set the message field to it's message.

From there each client should be able to alternate in sending messages and the messages received will be printed out to the display. This tests the continuous functionality of the message and lock fields.

One edge case that doesn't work is if both clients connect and then one disconnects/the process is killed, there's no way for that client to rejoin the session. Instead both clients must ensure that they've killed they've quit the program and restarted the program.