Morgan Trembley

CSE 160

Project 4 Description

For this project, I created a new interface called Chat. There are start client and server functions similar to the transport protocol from project 3 that create a connection between the specified server port and a client. When that is complete, A client can send a hello message. After transport, the server reads this message, splits it into its components: instruction, username, port, then saves the username, socket number and port number in a list of users. Then the client is free to send messages to other clients. To handle these messages, there are 2 new functions. There is a setSocket function only for hello messages and a handleMsg function that handles and transmits a msg, whisper, or listUsrRply. When handleMsg reads a new message from the rcvdbuff of a socket, it breaks the message up using strtok and decides what to do with the message. If the instruction is msg, a new message is built using the name of the sender ": " then the message. This is done using a helper function called concat. Then, it iterates through the list of users and sends the message to everyone. If the instruction is whisper, everything is the same except when the message is sent. It is only sent if the username in the list is equal to the username provided in the client's whisper message. If the instruction is listusrs, a new string is created starting with "listUserRply: " then I iterate through the list of users and add each name to the string in the list. Then that is sent to the user that requested it.

There are a few decisions I made that might not have been the best choice. First, I decided to only send messages through the python script. This is useful because you can control which node, ports, messages you are sending from easily, but you are very limited on message size. The packet has a maximum size of 28 characters including the instruction and username, if there is one. It might have been more useful to ask for an input in the interface so the size of the message could be much larger but I wasn't sure how I'd go about changing the node I was sending from and I thought it'd be more difficult to demo. The other decision I made was to separate the chat user list and its information from the socket list in TCP. This required more thought to get working than it should have and would probably be confusing at first to someone else trying to understand what my code is doing.

I had one problem that basically made my program useless that I could not solve by the due date. The server does not send reply messages. It picks the correct destination and fills the correct buffer to send back to the client but the send does not occur. I think this is an issue with how the acknowledgements work in my implementation but I could not fix it. Even though the reply from the server doesn't reach the client, the client is set up right now to receive and print out the message from the server in a debug message. That could be easily changed to just output to the client node but the client shouldn't have to do anything to the message to interpret it correctly.