z5281800 Joseph Heflin

COMP3331: Computer Networks and Applications

Assignment: Messaging Application (P2P capable, Python 2.7.16)

This program extends what I have learned during lecture and lab about using TCP and extending that with some creative license to develop an application layer protocol. Two important python modules I chose to use in these programs are threading and socket. Extending the threading module allowed be to write and run a single server program at a single address and port and multiplex and demultiplex those signals so that information is accurately shared and that protocols are followed. The socket module allows me to control the use of the transport layer.

The program server.py acts as the central host for the messaging application, accepting connections from various satellite client connections. Upon accepting a client connection in the welcome socket, I create an instance of the client connection class and pass it the newly allocated socket. ClientConnections are uniquely identified by both their socket (self.sock) and a username if they successfully log in (self.user). ClientConnections extends threading which automatically calls the run() function when start() is called in the body of the program (line 269). With the exception of the private and stopprivate commands, the client sends keyboard inputs directly to the server so the server program in processing most of the application's tasks.

A relatively unique feature in both my server and client programs is a function that serves as a function switchboard. After writing the authorize function in the server program I realized that multiple successive and nested conditional statements would be difficult to write and debug. I use the switchboard functions to select between various functions based on a string argument. The string parameter can also contain data to be passed as an argument to whatever function is selected from the "switchboard".

The program client.py is designed to be run by the message application users. It processes commands and messages sent from the server and displays them for the user. I implement one thread each for the connection to the server and the keyboard input so that the two don't block each other. I also have a threading class for private connections for that a single client can host multiple private conversations. This thread for private conversations is only used to listen for receiving messages. I send private messages using a function outside of the thread that references the correct socket based on the socket value paired to a specified username in the privatelog dictionary.

While my programs work for the purpose of this assignment, there are a few modifications I would consider making to scale the application. Primarily, I would avoid using recursion in the authorize function of server.py. It wouldn't be cost efficient considering the stack space necessarily allocated for each function call when a while loop could perform the same functionality.