Programming Assignment 2: Online chat

Design and develop an online chat system, named csce513fall24Msg, for communications and discussions among students in a class. The csce513fall24Msg is expected to enable the students to chat with the instructor and their classmates for any necessary discussion, e.g., homework problem. Below are the general guideline for the system design.

1. Client-Server Communication using TCP/IP

The first step is to build a server and a client that can communicate with each other through socket programming using TCP/IP. The communication between two clients will go through the server: if client A wishes to initiate a chat with client B, both A and B should connect to the server and the server should forward messages or requests between A and B.

Steps to implement the server:

- Create a socket for communication COMPLETED
- Bind the local port and connection address COMPLETED
- Configure TCP protocol with port number COMPLETED
- Listen for client connection COMPLETED
- Accept connection from client COMPLETED
- Send Acknowledgment COMPLETED
- Receive message from client COMPLETED
- Send message to client COMPLETED

Steps to implement the client:

- Create a socket for communication COMPLETED
- Configure TCP protocol with IP address of server and port number COMPLETED
- Connect with server through socket COMPLETED
- Wait for acknowledgement from server COMPLETED
- Send message to server COMPLETED
- Receive message from server COMPLETED

Using command line is fine, while GUI is highly encouraged. Any language is acceptable. Python and Java are recommended as they offer convenient tools for socket programming. (30 points)

2. Advanced Client

Now you can add the functionality to allow a client to send and receive messages at the same time with less CPU workload. I/O multiplexing can be used in this task. You can use system callback function: a client will be activated if the socket receives data from the server or keyboard input from the user. Hint: try to use select(), poll() and epoll() in your client. (20 points)

3. Multi-Thread Communication Server

We will improve our server in this task to allow multiple students to discuss class topics or homework problems at the same time. You can use any of the following three methods to implement your server:

Use *socketserver* model (Python has provided *socketserver* package)
Thread + socket
I/O multiplexing

Till now, your server should be able to support concurrent connections with multiple clients. (20 points)

4. Client-Client Communication

Now, we are ready to implement the client to client communication in csce513fall24Msg. Your need to implement three core functions:

Client management
Receive message from a sending client
Forward message to a receiving client

Client management is to be implemented in the server, to capture the exception of client absence. For example, if A wishes to chat with B, but B is not in the system, your server should be robust to handle this issue. Clients communicate with each other by passing messages through the server. In your demo, you should show the client window

for sender and receiver, and the server window. Necessary information should be displayed on the respective window to demonstrate that your implementation functions well and satisfies the requirement. (30 points)

5. Bonus functionality

You can get 5 bonus points for the implementation of each extra functionality.

Group chat. Each group member can send and receive messages in the group chat window. These messages are visible to all group members.

File transfer. A client can transfer a file to another client by using csce513fall24Msg.

Offline message. The server can save the message for offline clients (not connecting to the server). Once these client get connected to the server, the stored message can be forwarded to them.

Secure communication. To enhance the security of csce513fall24Msg, please come up with a security model and add encryption & decryption in the transmitted messages in this task. Hint: use the public and private key to generate and transmit session key.

6. Submission requirement

Please submit:

- makefile and readme file to explain how to run your code

- screenshot for your demo for each task

- technical report to explain your implementation

Total points: 100 + 20 (Bonus)