Computing Laboratory II (CS69202)

Spring Semester 2024

Assignment : Client-Server Chat Application with FAQ Chatbot (Part - 2)

Assignment date: March 27, 2024

Important Instructions:

1. **Using Linux**
    a. This assignment is based on Network programming and will be beneficial for the students to do this assignment in a Linux OS.
    b. If you don't have linux, use virtualbox (https://www.virtualbox.org/wiki/Downloads) + linux image, both free. We suggest any Debian based Linux Distribution (specially Ubuntu, Pop OS and Linux Mint), but feel free to use any linux version. If you decide to use cygwin or wsl, it's up to you, just remember that most of your batch mates (and the instructors) will use Linux. Hence, we might not be able to provide support for other systems. Install the latest version (22.04 LTS for Ubuntu or Pop OS) to avoid any problems.
    c. To install Linux in Virtual Box refer to this tutorial : https://youtu.be/nvdnQX9UkMY?si=MG46UuES1krp1L-6

2. **Programming language**
    a. This assignment is using the C and Python programming language. You can also use cpp (without STL) if you want, as the important calls remain mostly the same.
3. **Error handling**
    a. A proper error handling (e.g., when the syscall or the library calls fail) is expected in this Assignment.
4. **Input/output**
    a. The inputs should be taken from the Terminal and results displayed in the Terminal, **if explicitly not mentioned**.
    b. The IP address and port number will be provided by Command Line Interface (if required).
5. **Documentation**
    a. Include a README file explaining how to compile and run the server and client, any dependencies, and any additional features or improvements you made.

6. **Submission**:
    a. Please upload a single zip file (<Roll_no>_FAQ_D1.zip) with the tasks and any additional files. Please also include the README
7. **Tutorial**
    a. https://nikhilroxtomar.medium.com/tcp-client-server-implementation-in-c-idiot-developer-52509a6c1f59

Objectives:

1. Develop a peer-to-peer (P2P) chat functionality using a client-server architecture using TCP sockets.
2. Incorporate a FAQ chatbot feature that can be turned on or off.
3. Handle multiple client connections and manage communication between clients and the server including chat histories for individual clients.

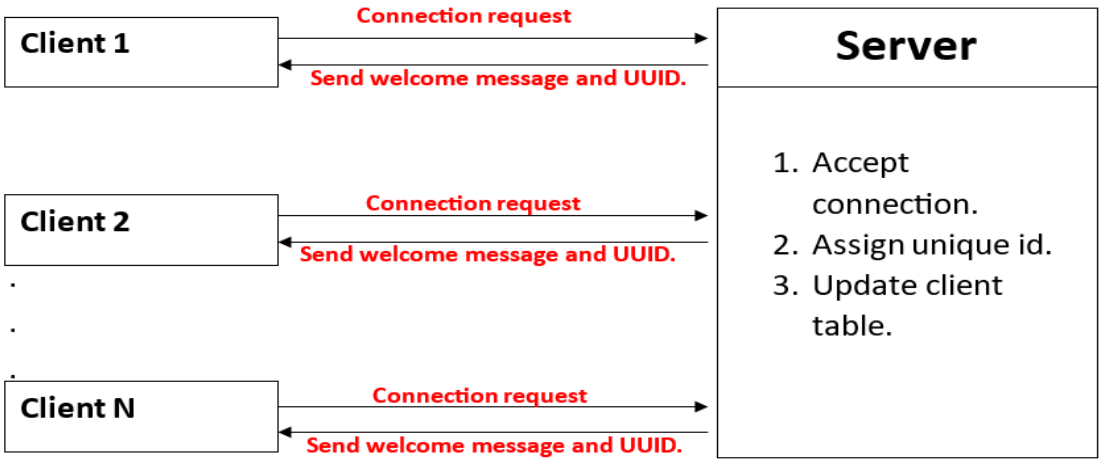

Fig 1: Connection Establishment between Server and Client

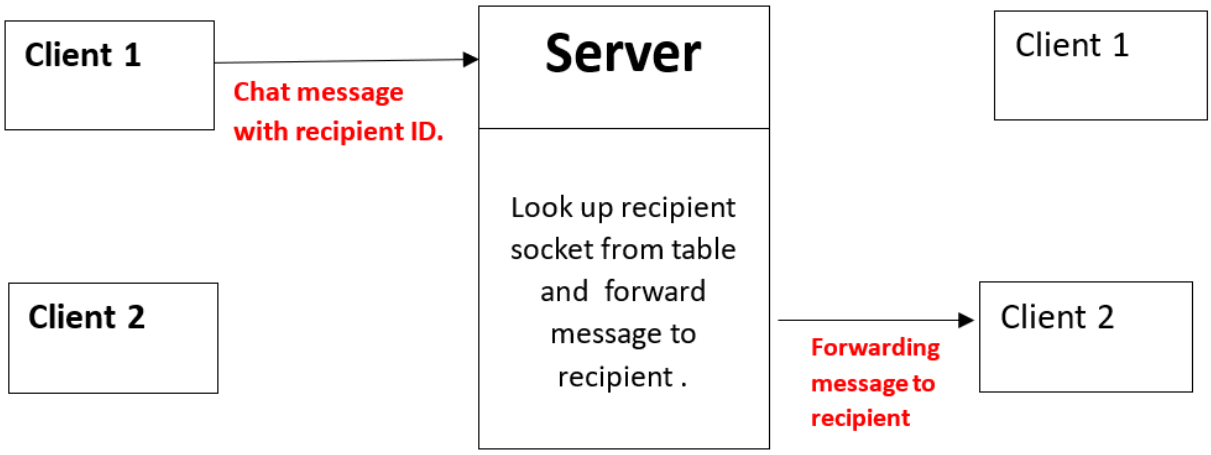

Fig 2: Peer to Peer Chat using server

Task 1: Peer-to-Peer Chat

Implement a basic peer-to-peer chat functionality between clients (see tutorial above). The server should handle multiple client connections (up to 10 concurrent connections) and facilitate smooth communication between clients.

1. Connection Establishment
    - Clients should send a connection request to the server.
    - The server should assign unique identifiers (ID) to clients (use random uuid4, for example using this) and send a welcome message along with their ID upon successful connection.
    - The server should maintain a table of connected clients details like socket ID, unique ID
2. Chatting
    - Implement the following features for the clients:-
        i. "/active" - retrieve a list of active clients
        ii. "/send <dest_id> <message>" - send messages to other clients using their unique IDs
        iii. "/logout" - client requests to exit the application. In such cases also, the server must send a message "Bye!! Have a nice day" as an acknowledgement to the client.
    - The server should handle message sharing between clients.
    - Appropriate error handling should be implemented (e.g., notifying the sender if the recipient goes offline).

Task 2: FAQ Chatbot

Implement a FAQ chatbot feature that can be turned on or off by clients. This FAQ will be given by the server.

1. FAQ Handling

- Implement a set of frequently asked questions (FAQs) and their corresponding answers.
- When the chatbot is active, if a client's message matches any FAQ, the server should respond with the corresponding answer.
- If no matching FAQ is found, the server should respond with a default message **"System Malfunction, I couldn't understand your query."**
- The FAQ matching can be implemented using simple string matching (exact and case sensitive) .

2. Chatbot Activation/Deactivation
    - Clients should be able to toggle the chatbot feature using commands
        - "/chatbot login" - to avail the chatbot feature
        - "/chatbot logout" - to disable the feature
    - The server should maintain the chatbot status for each client.
    - When the clients avail the chatbot feature, the conversation must continue using two prompts "**stupidbot>**" (for messages from chatbot) and "**user>**" (for messages from the user).
    - Also, you need to display an appropriate message on login (**Hi, I am stupid bot, I am able to answer a limited set of your questions**) and logout (**Bye! Have a nice day and do not complain about me**) features. One such example is given for reference:-
        - /chatbot login
        - stupidbot> Hi, I am stupid bot, I am able to answer a limited set of your questions
        - user> How are you?
        - stupidbot> Bad
        - user> When will the green line metro of Kolkata be completed?
        - stupidbot> System Malfunction, I couldn't understand your query.
        - user> /chatbot logout
        - stupidbot> Bye! Have a nice day and do not complain about me

Task 3:Chat History

- Maintain a chat history for each client, storing their previous messages and conversations in a log file on the server.
- Implement the following features for the client :
    - "/history <recipient_id>" : the server should retrieve the conversation history between the requesting client and the specified recipient.
    - "/history_delete <recipient_id>" : delete chats of specified recipient from requesting client chat history.
    - "/delete_all" : delete complete chat history of requesting client.

Whenever a client sends a message to another client, the server should store the message in the chat history for both clients.

Task 4: FAQ Chatbot (using GPT-2) [Python Implementation]

Implement a FAQ chatbot feature that can be turned on or off by clients. This FAQ will be given by the server.

3. FAQ Handling
    - When the chatbot is active, if a client's message matches any FAQ, the server should respond with the corresponding answer using a pre-trained GPT-2 (https://huggingface.co/openai-community/gpt2) Large Language Model .
    - **There is no need to train the GPT-2 LLM using any custom data. Utilize the already pre-trained model to get the response.**

4. Chatbot Activation/Deactivation
    - Clients should be able to toggle the chatbot feature using commands
        - "/chatbot_v2 login" - to avail the chatbot feature
        - "/chatbot_v2 logout" - to disable the feature
    - The server should maintain the chatbot status for each client.
    - When the clients avail the chatbot feature, the conversation must continue using two prompts "**gpt2bot>**" (for messages from chatbot) and "**user>**" (for messages from the user).
    - Also, you need to display an appropriate message on login (**Hi, I am updated bot, I am able to answer any question be it correct or incorrect**) and logout (**Bye! Have a nice day and hope you do not have any complaints about me**) features. One such example is given for reference:-
        - /chatbot_v2 login
        - gpt2bot> Hi, I am updated bot, I am able to answer any question be it correct or incorrect
        - user> How are you?
        - gpt2bot> Good!! Who are you?
        - user> When will the green line metro of Kolkata be completed?
        - gpt2bot>What is Kolkata? Do you mean CCU?
        - user> /chatbot_v2 logout
        - gpt2bot> Bye! Have a nice day and hope you do not have any complaints about me

Things to remember:

Server Side:

- Use select() for the implementation.
- Consider using send() and recv() system calls.
- If client B wants to send a message to client A, B won't be able to send to A's socket directly. Instead use a message details table at the server side. Client B will send a message to server and then server will pass on the message to client A.
- Server will be implemented in C and the GPT-2 (inference only) model in Python. Fig. 3 shows a server side implementation using exec system call.

Client Side:

- Reading from the standard input and writing to the server (send() system call) and reading from the server (recv() system call) will be handled by different processes.

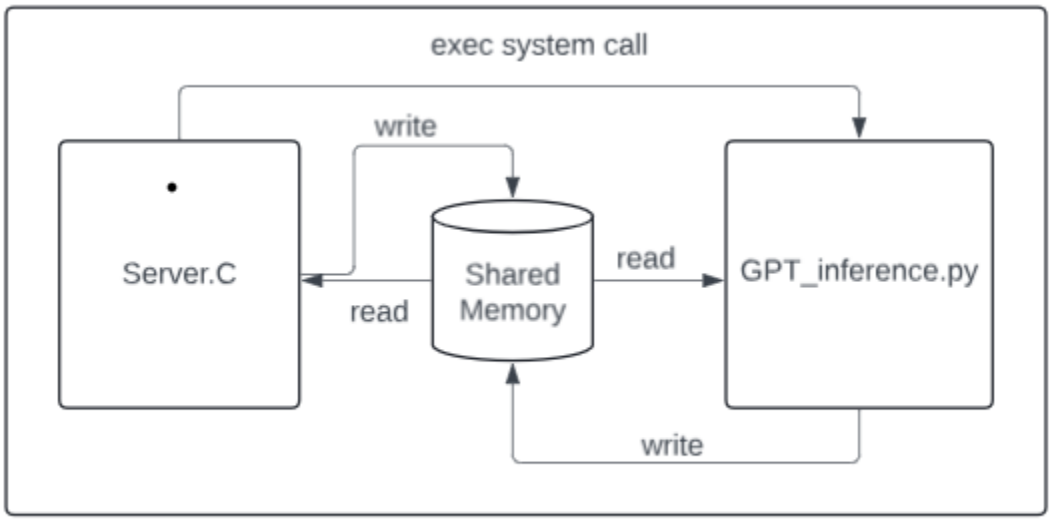

Fig 3 :Server Side Implementation