جامعة الملك فهد للبترول والمعادن
King Fahd University of Petroleum & Minerals

COE 523, Term 232

Distributed Computing

Assignment# 1
Due date: Saturday March 2, 2024
Instructor: Dr. Ayaz ul Hassan Khan

Objective:

In this assignment, you will learn:
- Implementing client/server communication through python sockets
- Programming multi-threaded server to handle multiple clients' requests.
- Implementing process/thread synchronization in python

Problem Statement:

You are required to produce two programs, **chatserver** and **chatclient**. There can be more than one instances of **chatclient** running at a time and the purpose of **chatserver** is to provide a link between all the available clients so that they can talk to each other.

Requirements:

chatserver:
On startup, server will create a ***socket*** to receive client connections and messages. One client can send a message to another client via the server. All messaging is done through the server. It means that whenever a client has to send a message to another client; it will send the message with the target ***clientid*** to the server to be forwarded to another client. Clients can also send messages to the server to get some information in response. The server if it receives a message for a client, who is not online anymore, respond to the source client with a message that the target client is not online. The server will keep a list of online clients with it and clients can ask about this list by sending a message to the server. The server is also responsible for sending the latest client list to all the clients whenever there is a change in it.

chatclient:
On startup, each client will connect to the server and sends its id provided by the user. The server will register this client. Each client is required to send an alive message to the server after regular intervals so that the server is updated about its presence. The server will tell the client about the length of this interval in response to the connect message. If the server doesn't receive an alive message from some client, then it de-lists it and sends the latest list to all online clients. Whenever a client wants to send a message, it sends a message to the server with the target ***clientid***. Clients can also request the latest client list by sending a message to server.
message:

A message sent by a process (client/server) to another. It is a null terminated string with the maximum length of 255 bytes (256th byte is null). First 8 bytes are the name of destination, next 8 bytes are the name of the source and the remaining bytes up-to the null byte are the actual message. It means that the actual message length can't be more than 239 bytes. It also implies that a client name can't be more than 8 bytes, and if some client has name less than 8 bytes then you have to perform padding to keep it of 8 Bytes. Moreover, as the server sends the list of online clients through a message and the message length is limited, therefore, either you have to limit the number of concurrent clients or use some other way to send the list in more than one iteration.

Standard/Control Message Formats:

The following standard/control messages should be implemented:

Connect
Syntax: Connect clientid
Purpose: automatically sent by a client to the server when the client comes online
@Quit
Syntax: @Quit
Purpose: user types it at the client prompt to end his session
Quit
Syntax: Quit clientid
Purpose: automatically sent by a client to the server when a user requests for session end
@List
Syntax: @List
Purpose: user types it at the client prompt to view the current list of online clients
List
Syntax: List
Purpose: automatically sent by a client to the server when a user requests the list of online clients
Alive
Syntax: Alive clientid
Purpose: automatically sent by client to server after regular intervals that it is still alive
General Message to some other client
Syntax: (otherclientid) message-statement
Purpose: typed by the user at the client prompt when he wants to send a message to an online client

Note: In all messages to the server, the destination address is "-SERVER-". You can implement server in a way that it can broadcast a single message to all clients by typing destination as "ALL" or something else, but it is not required. You need to implement a multithreaded server such that each client is connected with the server in a separate thread. The online client list at the server is shared among all the threads and you may be required to use proper thread synchronization method to avoid data inconsistencies.
Required Deliverables:

Two source code files and a report should include the usage of each source code and test cases (covering all the defined requirements) with brief explanation and screen shots of sample executions.

Note: You should provide proper comments in source code to get full marks.

Grading Scheme:

Server (50)
Receiving Message from client (includes tokenization of message): 5
Forward/Reply Message to client (includes creation of message in specified format): 5
Creating Client List: 5
Sending Client List: 5
Adding new Client to List: 5
Deleting Client from List: 5
Retiring Client: 5
Informing about change in List: 5
Reply Message if message for offline Client: 5
Resetting Client life on receiving alive message: 5

Client (50)
Exchange Messages with another clients via server: 10
Receiving client list from server (includes tokenization of message): 10
Sending Message to server (includes creation of message in specified format): 10
Displaying Client List: 5
Sending alive message: 5
Sending connect message: 5
Sending quit message: 5