CSE 324 – Computer Networks Laboratory

Possible Term Projects – Socket Programming

A. Projects

1. Centralized chat application

The assignment consists of implementing two components: a chat client and a chat server. In this implementation, the server maintains information about the clients that have registered with a specific chat group and pass the messages sent by the clients. The server should use select() or poll() to perform I/O multiplexing among the clients. The operations of client and server are specified as follows:

Client implementation

Clients can communicate with others by registering with a 'group'. All received messages (from other clients) should be immediately sent to standard output. Clients should be started with

- *client [server_host_name] [server_port_number] [client_name]*

After successfully contacting with the server, the client should support the following commands:

- *send [groupID] [message]* : This allows a client to send a message to a group.
- *join [groupID]* : This registers the client with a group, allowing it to receive and send messages from/to the specified group.
- *leave [groupID]* : With 'leave' a client indicates that it wants to stop sending and receiving messages.
- *quit* : This closes the client application.

Server implementation

The server is started with

server [server_port_number]

- When a client joins a group, the server registers the client with the corresponding group.
- A client cannot be member of multiple groups simultaneously and duplicate join requests must be discarded.
- When a client requests to leave a group, the server removes the client from the corresponding group; if a client tries to leave a group in which it is not a member, the request must be discarded.
- When the server receives a send request message for a group, the server delivers the message to all members of the group. The message format should be as follows: "ClientName: msg".

The user interface need not be sophisticated, but should be usable. It can be text-based or can be graphical.

2. Decentralized chat application

The assignment consists of implementing two components: a chat client and a chat server. In this implementation, the server only serves as registry, maintaining contact information for all clients, but not sending any messages on behalf of the clients. That is, clients talk directly to each other. The operations of client and server are specified as follows:

Client implementation

The client will have similar operation as the clients specified in Project 1.

Server implementation

The server is started with

server [server_port_number]

- When a client joins a group, the server registers the client with the corresponding group.
- A client cannot be member of multiple groups simultaneously and duplicate join requests must be discarded.
- At registration, the server informs the new client about all existing clients (contact information) and the client will directly establish a connection with all other clients.
- All messages are then sent directly to all other clients.
- When a client leaves a group, it will terminate the connections to the clients of that group and inform the server with a **'leave' [see the description in Project1]** message, which in turn will inform all other clients.
- You can assume that there are no more than 3 available groups and at most five clients in the system.

The user interface need not be sophisticated, but should be usable. It can be text-based or can be graphical.

Helping links for project 1 and 2 Link . A sample code for chat application is provided here.

3. **File transfer application which can handle one request at a time** (Extra Credit for handling multiple requests at a time with the capability of resemble downloads)

You must write a client and server to support file transfer. The client should send to the server a request to initiate a transfer of a specific file. Upon receiving the request, if the file exists, it should be sent to the client. Be sure your program works with all file types, not just text files.

- When the server starts, it should wait for a client connection (on a port number either obtained as a command line option or input by the user).
- When the client starts, it should connect to the server and send a request for a file. This request may be formatted in any way your group finds convenient.
- The server IP address, server port number, and file name may either be taken by the client program as user input, or as command line options.
- Upon receiving a request, the server should read the requested file if it exists, and send it to the client, which should write the file locally.
- The client and server should each print to the screen each time they send or receive a packet.

4. **Development of an online data base server** (Extra credit for data entry and report)

The client, on the submission of a password, will gain access to the server and would then be able to get the required information.

The server contains a data base of the UIU students. Data base includes:

- **Identification number**
- **Name**
- **Semester**
- **Department**

Server remains in the reception state so that any client can establish a connection with it. Upon establishment of connection between Server and Client, server asks for password form the client to login. After login client can request for information regarding any of the ids available in the database. In case the id is not available, the server notifies the client and then simply terminates the connection.

The program for the client is written such that a client will first try to connect to the IP address of the server that the user wishes to connect. On the submission of the correct password, the client can access the database to get the required information.

B. Report Format

1. **Introduction and Problem Statement:**
   Describe the project details (things to be implemented) in this part.

2. **Flowchart or Diagram:**
   Create a flowchart or diagram to describe the socket operations between server and client programs. Describe every part in details.

3. **Implementation:**
   Describe the server and client codes by listing major functions.

4. **Testing:**
   Describe how you have tested the network to verify that it is fully working. Take screenshots of testing scenarios.

5. **Conclusions and Future Improvement**
   Describe what have you learned and experienced from this project and identify the areas where you can expand or improve the design so that it will be more realistic and efficient in future.