Denzel Awuah

Request for Comments: 701

COIS 4310

Trent University

October 2019

Version 1.1

Multi Client-Server Chat Application

Summary

This is a chat/messaging application. A server is created on a machine and the server allows up to 5 different clients to establish a network connection with the server. When clients are connected to the server, they are then capable of communicating with each other by sending messages to each other. Clients can choose to send a message which is broadcasted to all other clients connected to the server or send a message to one specific client on the server. The clients also have the option to get a list of all the currently connected clients on the server by entering a specific verb as input. Clients can choose to leave the application at anytime by entering a specific verb that will disconnect them from the server

Application Description

They way that this application is architected that is consists of 4 modules. These modules are client, server, ClientControl, and ServerControl. The main modules that are used to run the application are the client and the server. The client module consists of a socket. This socket contains an IP address and a port number, which is used to connect to the server. In the client module, there exists a packet number to track packets, a version number, a PrintWriter that is

used to send messages on the socket's output stream, and a BufferedReader that is used as an input stream reader to read input from the clients.

The server module is used to first create a Server socket. This server socket contains the port number for the server and which is what clients will connect to once the client program is started. Once the server is started, it waits for clients to connect to it. When a client connects, the server will add that clients socket to an arraylist containing all clients sockets. The same array list, along with the clients socket is then sent to the ClientControl module where it will be used for operations. Depending on the order at which each client has connected to the server, the server will give the client a username called "client1" if they were the first to connect to it, or "client2" if second, and so on. This username is also stored into a separate arraylist full of all clients usernames. The username and the arraylist of usernames are also sent to the ClientControl module. Finally, the server will place the client's ClientControl module into its own thread using ExecutorService.

With the ClientControl module, each ClientControl is running on its own thread and each ClientControl module is capable of reading sockets input and getting output streams. This class is controlled by the server. When a client wants to send a packet, it must go to the server first, it then gets evaluated by the server, then actions are taken by the server based on the assessment of the packet. Each module also contains an arraylist of all client sockets connected to the server and all client usernames. ClientControl will check each input from the client to see if it a valid input before sending. This class is also responsible for checking the Checksum of each packet that is sent by a client. Once the checksum of a packet is checked, if it is valid, then the packet has no errors and can be delivered. The ClientControl also handles client's ability to send

messages to all other clients, or just to specific clients by checking the input from the client and performing the operations based of off it.

The ServerControl module is used to run the client on a different thread then the ClientControl Module. It is also used to send out messages from the server.

Packet Header Description

The Packet head for each socket is containing of the username of the receiving client, the type of request that the client is making (either message, list, or bye). The packet header of each packet also contains the checksum for the specific message that have entered. This checksum is calculated by getting the ascii values of each character in the message and adding all of these values together. The sum of all the ascii values will be the checksum of the packet. The Message verb is the default, and it requires a client to enter the username of another client and the message that they want to send. It must be in the format “username:message”, where username is the username of another client in the server, and message is the data that you want your message to contain. If the client wants to send a message to all clients, they must enter “all” instead of another client’s username. The format of this would be all:message.

Data Portion Description

The data portion of the packet consists of a string of data which can hold a maximum of 256 characters. The purpose of this data portion in the packet is for sending messages to other clients on the server. The data portion is only required when sending messages to other clients. If

a client wants to use the list command to get a list of all the clients connected to the server, then the packet for that command does not require a data portion, only the required verb to perform that operation.

Verbs

Message: message is the default prompt for a client and after each command that a client makes, they are asked again to write a message in the format: “Username:Message”. When a client wants to send a message, they must use this format. The username can be the name of other clients in the server, or “all” to send a message to everyone in the server. The message part in the format is for the data that you want your message to contain. If a client enters an invalid format of input, they will be notified of the invalid input and asked to try again.

List: If a client would like to see a list of all clients usernames that have connected to the server, they can simply type in “list”, and they will be returned with a list of all the clients usernames.

Bye: If a client wants to disconnect from the server, they can simply type in “bye”. This command will close that clients socket and disconnect them from the server and all other clients. Once disconnected, that client is unable to communicate with other clients anymore and their application closes.