Project Report & Design Document

Nidhiben Solanki

Group Members:

Jered Hightower JPH200000

Umaima Memon URM190000

April 27, 2023

Protocol Design/Implementation:

The program represents a client-server interaction using the TCP/IP protocol. This is the breakdown of the user-server interaction in the code for TCPClient.java:

1. The user runs the file passing the ip address and port number as arguments

2.The user is prompted to enter their name via the console using the BufferedReader object.

3. The user is prompted to join the server by entering 'y' or 'n'.

4. If the user enters 'y', a connection is established with the server using the IP address and port number they chpse.

5. If the connection fails, the client waits for 2 seconds and retries the connection up to 3 times.

6. After the connection is established, the user's name is sent to the server using a DataOutputStream object.

7. The user can then send multiple messages to the server until they enter "QUIT".

8. Each message sent by the user is sent to the server using the DataOutputStream object, and the server's response is received using a BufferedReader object. Messages sent by the user should be in the form of a space delimited equation e.g. 1 + 2 + 6

9. The received message is printed to the console. This message should be the answer to the equation if it’s well formed.

10. When the user enters "QUIT", the client closes the connection with the server using the clientSocket.close() method.

Client-Server Interaction for TCPserver.java:

TCPserver.java is an implementation of a server that listens on a specific port for incoming client connections. The client-server interaction is based on a text protocol where the client sends text-based equations to the server, and the server parses and evaluates the equations, and sends the result back to the client as a text message.

The server uses TCP to establish a reliable connection with the client. The server creates a `ServerSocket` object that listens on the port given as an argument for incoming connections. When a client connects, the server accepts the connection by creating a new `Socket` object and starting a new `ClientThread` to handle the communication with the client. The `ClientThread` reads input from the client using a `BufferedReader` object, and writes output back to the client using a `DataOutputStream` object.

The client can send multiple equations to the server, and the server will evaluate each equation and send back the result. If the client sends the string "QUIT", the server will close the connection with the client. Additionally, the server logs the start and end times of each client session to a log file named "log.txt".

Overall, TcpClient.java and TcpServer.java work together to establish a TCP connection between a client and a server.

TcpServer.java is responsible for setting up a server socket that listens for incoming connections from clients. Once a connection request is received, the server socket accepts the connection and creates a new socket dedicated to that specific client connection. The server socket then hands off the client socket to a new thread, which handles all communication with the client.

TcpClient.java is responsible for creating a client socket and connecting to the server socket created by `TcpServer.java`. Once the connection is established, the client can send data to the server and receive data back in response.

By running TcpServer.java on the server side and TcpClient.java on the client side, they can communicate with each other over a TCP connection.

Assumptions:

It is assumed that the server will choose a valid port number.

It is also assumed the client will have a well formed ip address and port number

Message format:

Space delimited equation. Eg. ( 1 + 2 ) \* 3

Supports (),+,-,\*,/,^ operations.

Programming Environment Used:

The programming environment used for this project is VS Code

How To Compile & Execute:

To compile and execute tcpclient.java and tcpserver.java, you would need to:

1. Have the Java Development Kit (JDK) installed on your computer.

2. Open a command prompt or terminal and navigate to the directory where you have saved the files `tcpclient.java` and `tcpserver.java` and the makefile.

3. Compile the files using the make command. This will generate two new files, tcpclient.class and tcpserver.class

4. Start the server by running `java TcpServer “port number”`. This will start the server and it will start listening for incoming client connections.

5. Start the client by running `java TcpClient “ip address” “port number” `. This will start the client and it will connect to the server on that ip address and port number. You should see some output indicating that the client has connected to the server.

6. At this point, you can enter some data into the client console and it will be sent to the server. The server will receive the data and process it, sending it back to the client. You can then enter some more data and see it being sent back and forth between the client and server.

Parameters Needed:

Port # required for TCP-Server

IP and port # required for TCP-Client (asks for name after being ran)

Challenges Faced & Lessons Learned:

One problem we ran into was implementing the equation parser. It took significant trial and error to implement it correctly. We learned how to implement a TCP connection through this process and see how the client interacts with the user based on the program. It was interesting to see how the connection established and then the messages were sent back and forth, which was something we learned in class but got to see implemented first hand.

Output Screenshots: