**Problem Statement**

**Emulating SFTP using TCP socket**

Group No. 4

**Design Document**

1. **INTRODUCTION:**

SFTP (SSH File Transfer Protocol) is a secure file transfer protocol. It runs over the SSH protocol. It supports the full security and authentication functionality of SSH. SFTP has pretty much replaced legacy FTP as a file transfer protocol, and is quickly replacing FTP/S. SFTP (Secure File Transfer Protocol) is the advanced version of FTP (file transfer protocol) which ensures security while transferring files between the organizations/computer. It is also known as SSH (secure shell). It works on port no. 22 and uses the client-server model. A socket programming interface provides the routines required for interposes communication between applications, either on the local system or spread in a distributed, TCP/IP based network environment. Once a peer-to-peer connection is established, a socket descriptor is used to uniquely identify the connection.

**1.1 PURPOSE:**

The purpose of this project is a client-server architecture CLI that allows transfer of files hosted on the server to any number of connected clients, making use of TCP protocol for reliable file transfer.

**1.2 FUNCTIONALITIES OF THE SYSTEM:**

These functionalities have been implemented in the program:

PUT: Transfer a file from client to server

GET: Transfer a file from server to client

MGET (example: MGET .txt): Transfer all files of a given extension from server to client

MPUT (example: MPUT .txt): Transfer all files of a given extension from client to server

File overwrite feature has been implemented along with it i.e. if the file is already present

at the destination, then the client will be asked whether to overwrite the file or not.

Compiling the code:

On Server’s terminal $ gcc server.c -o server On Client’s terminal $ gcc client.c -o client

To run the code, First on the Server’s terminal $. /Server

Then on Client’s terminal run $. /Client

Commands: Enter a choice: GET PUT MGET MPUT QUIT

Closing Client and Server Use command “QUIT server” to quit client program. This will disconnect the client. You can exit the server by simply using Ctrl + C on the server terminal.

**1.3 OPERATING ENVIRONMENT:**

Operating environment for Emulating SFTP using TCP socket:

* Client/server system
* Operating system: Linux
* Platform: Ubuntu/C++

1. **SOFTWARE REQUIREMENTS:**

The application has two primary parts - Client and Server - and the reliable transfer

is ensured using TCP with help of socket () API specifying SOCK\_STREAM.

The Client’s flow:

Checks for validity of arguments - needs server’s Ip and port to fire up

Creates socketfd which controls the flow or the control of the connection

Connects socket ref by socketfd to server\_addr (address and port specified)

Set up data connection in listenfd descriptor and binds the socket to data\_addr port

Then until user quits, listens to commands, parses it using helpers, and executes using

the respective helper functions

The Server’s flow:

Opens a TCP connection and starts listening to incoming connections

Binds the listening file descriptor to the specified port

Creates a fork of its process for every incoming connection, closing listenfd for

that particular fork

Receives client’s data and command

Takes the necessary actions

**3. UNIT TEST:**

Client:

SR1-UT1: Provide valid username and password.

SR2-UT2: Create a valid socket to initiate the communication

SR3-UT3: Client must choose only those options provided by the server.

SR4-UT4: Client must select only the files that are present at the server side to download.

SR5-UT5: If the connect () function fails then the client must receive an error message of connection failure.

SR6-UT6: If client enter commands are not in list won’t be able to download files client to server vice versa

SR7-UT7: If read () or write () function fails then the client won’t be able to send or receive the data

Server:

SR1-UT1: While defining the structure mention a valid element.

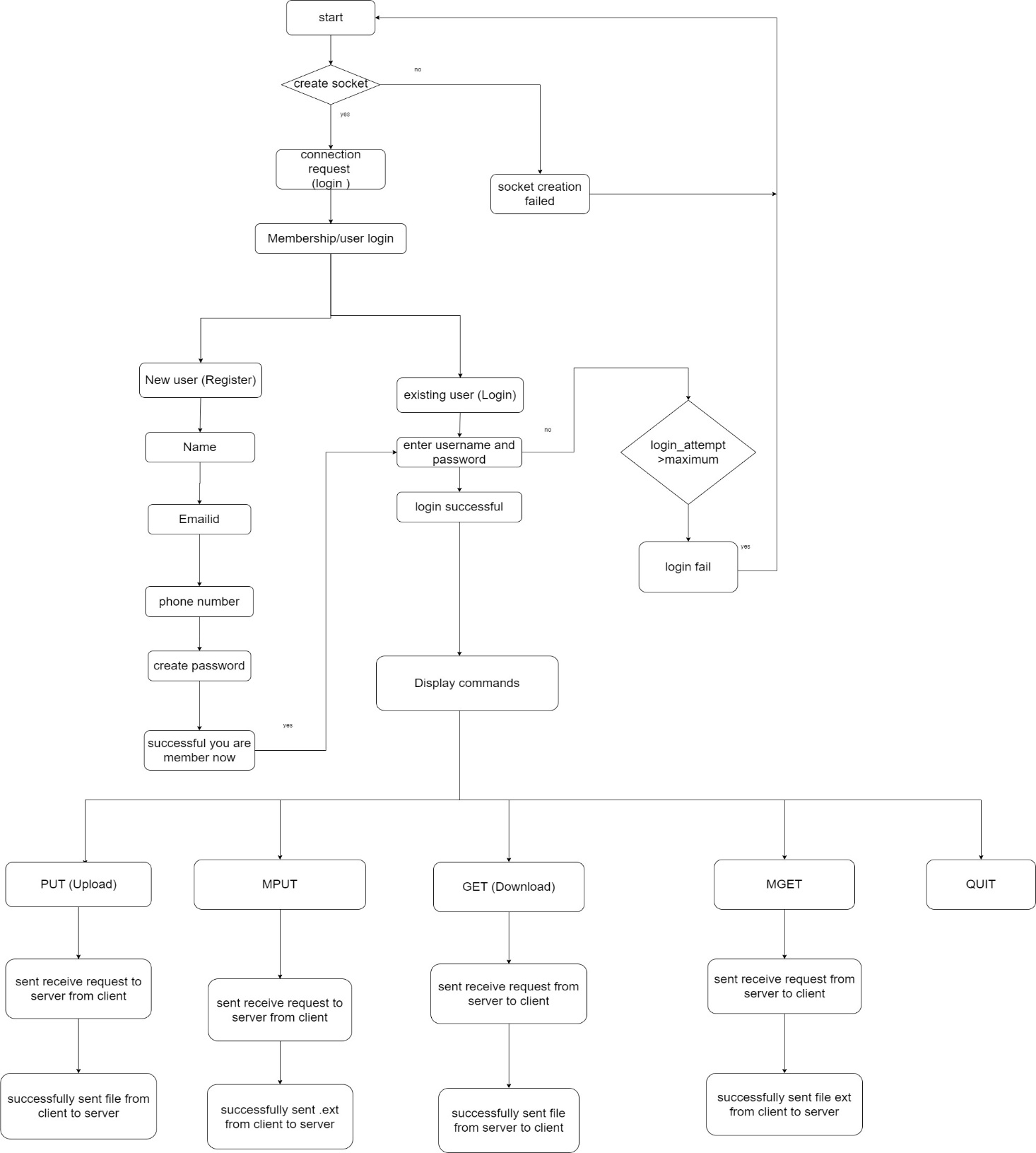
SR2-UT2: The port no. in the client side and server side should be same if no Connection not established

SR3-UT3: The bind () and listen () should bind the client and server if fail must return the Binding error while listen () function must listen on particular port number.

SR5-UT4: Send and receive the request Use the read () and write () file descriptor to

send and receive the data from server.

1. **FLOW DIAGRAM**



1. Start
2. File Transfer Protocol (FTP) using Client-Server socket programming
3. These functionalities have been implemented in the program:
4. PUT <filename> : Transfer a file from client to server
5. GET <filename> : Transfer a file from server to client
6. MGET <extension> (example: MGET .txt): Transfer all files of a given extension from server to client
7. MPUT <extension> (example: MPUT .txt): Transfer all files of a given extension from client to server
8. File overwrite feature has been implemented along with it i.e. if the file is already present at the destination, then the client will be asked whether to overwrite the file or not.
9. Compiling the code:
10. On Server’s terminal $ gcc server -o server
11. On Client’s terminal $ gcc client -o client
12. To run the code, First on the Server’s terminal $ ./server <PORT NUMBER>
13. Then on Client’s terminal run $./client <SERVER IP> <PORT NUMBER>
14. Commands:
15. Enter a choice: download adownload upload aupload quit
16. Closing Client and Server
17. Use command “QUIT server” to quit client program. This will disconnect the client. You can exit the server by simply using Ctrl + C on the server terminal.

**CONCLUSION:**

Using the socket connection with TCP and implementing sftp, ensures a secure file transfer between server and client. Data files of extensions such as .txt, .c etc and image files are transferred through this communication.