**GROUP 7**

**Problem Statement:** Emulating custom TCP (acknowledge based) protocol using UDP

1. **Introduction:**

**Transmission Control Protocol:**

TCP (Transmission Control Protocol) is an important network protocol that lets two hosts connect and exchange data streams. TCP guarantees the delivery of data and packets in the same order as they were sent.

TCP's role is to ensure the packets are reliably delivered, error free. TCP has concurrency control, which means the initial requests start small, increasing in size to the levels of bandwidth the computers, servers, and network can support.

**User Datagram Protocol:**

UDP (User Datagram Protocol) is a long-standing protocol used together with IP for sending data when transmission speed and efficiency matter more than security and reliability. UDP uses a simple connectionless communication model with a minimum protocol mechanism.

UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network; there is no guarantee of delivery, ordering, or duplicate protection. If error-correction facilities are needed at the network interface level, an application may use the Transmission Control Protocol (TCP).

**Difference between TCP and UDP:**

TCP is a connection-oriented protocol, whereas UDP is a connectionless protocol. A key difference between TCP and UDP is speed, as TCP is comparatively slower than UDP. Overall, UDP is a much faster, simpler, and efficient protocol. However, retransmission of lost data packets is only possible with TCP.

* 1. **PURPOSE**

The purpose of this project is to get a solid grasp on the fundamentals of the UDP & TCP. Writing such an application in C gives a basic understanding on the drawbacks of UDP and TCP and creates an enhanced UDP whose drawback is resolved by using some feature of TCP.

* 1. **FUNCTIONALITIES OF THE SYSTEM:**

The user will first do registration with providing username and password and get an auto-generated ID which will be used to login. After successful login, the user will be able to access the system. Then, the user will input port number in a given range between 49152 to 65535 and can transfer the data after getting acknowledged from client side and also the successful completion of handshake which ensure that both server and client are ready for data transfer and for logout client must send goodbye message to server to end communication or data transfer.

* 1. **OPERATING ENVIRONMENT:**

Operating environment for implementing TCP over UDP are:

* Operating system: Linux
* Platform: Ubuntu
* Language: C programming Language
* editor: text editor

**Software requirements:**

**Client:**

SR1 : Login credentials:

Register if a new user and login using the auto generated id received when user will register

SR2: Input port number

Client will have to give port number in range of 49152 to 65535

SR3 : Request to server to establish connection and perform 3 way handshake

Client will send a synchronization message to the server to establish connection between them and initiate a 3 way handshake which will be completed when the server will get the acknowledgement message which he requested.

SR4: Data send to server

client will send data or message to server with an acknowledgement message and message transfer will end when client will send logout and it will get out of the system.

**Server:**

SR5 : Server Login credentials:

Server will have to login using server login ID and password

SR6 : Input port number

Client will have to give port number in range of 49152 to 65535

SR7: Establish 3 way handshake

Server will receive the synchronization message to initiate handshake after that server will wait for getting acknowledged by client to complete handshake.

SR8: Receive data from client

Server will receive data from client after getting acknowledgement message from client and will come out from the system after receiving logout message from client.

**Functional Requirements:**

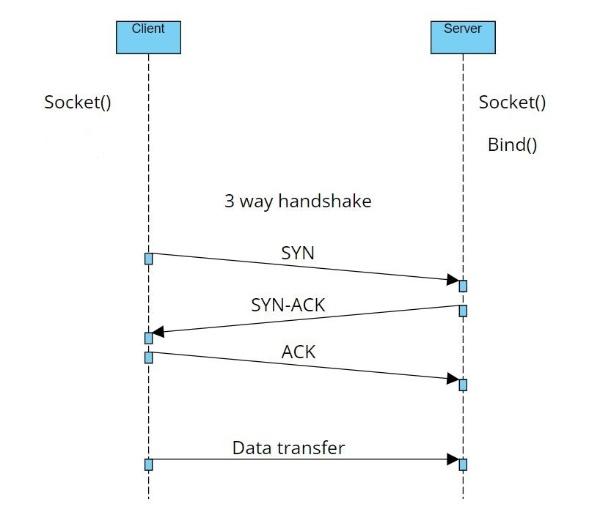
1. User validation
2. Port validation
3. Server socket
4. Client socket
5. Socket binding
6. Handshake function
7. Sending data
8. Receiving data
9. logout/exit

**Software design:**

**High level design:**



**Flow diagram:**

****

1. Start

This is the start block which indicates the start of the program.  
which will accept the client and server credentials (like the username and password). On validation of these credentials the system will allow the client and server to further communicate else if the credentials are wrong it will display an error message indicating “Invalid Credentials”.

1. Server Login

This is the module used for the server login where the server has to enter the credentials (username and password).

1. Client Login

This is the module used for the client login where the client, if new, has to register with an username and password and will get an auto generated ID and then login by entering the same ID. If already registered the client can login with the ID.

1. Login Credentials

In this module the credentials entered by the server/client are then validated by the system. If the server/client enters valid credentials then it will move to the further step else the system will prompt the server/client with an error message.

1. Enter your port number

Client/server enters port number in range of 49152 to 65535

1. Client request for connection

client will request server to establish connection by sending synchronization message to server.

1. server side handshake

Once the server gets the synchronization message it will ask for acknowledgement, and once server gets acknowledgement message from client, connection successfully established by performing 3 way handshake.

1. Client send data to server

client sends data to server after the successful establishment of 3 way handshake and also sends acknowledgement message for it.

1. Server Receives data

Server will receive data after getting acknowledged by the client .

1. Client Exit or logout from data transfer

client will send a logout message to get out from data transfer .

1. Server Exit or logout from data transfer

Server will receive a logout message and will get out from data transfer .

1. Display error message
2. End

Both server and client programs will end after selecting the exit option.

**Test plan:**

1. Unit test cases-

**Client:**

SR1-UT1: Client must provide valid user ID

SR2-UT2: Create a valid socket for connection

SR3-UT3: If the sendto() call fails to send a synchronization message, then it will throw an error message. If the recvfrom() call fails to receive an acknowledgement message, then it will throw an error message.

SR4-UT4: if the handshake is failed the data won’t get sent to the server and it will throw an error message.

**Server:**

SR5-UT1: Server must provide valid server name and password.

SR6-UT2: Create a valid socket for connection

SR7-UT3: If the recvfrom() call fails to receive an synchronization message, then it will throw an error message. If the sendto() call fails to send an acknowledgement message, then it will throw an error message. if acknowledgement messages fail to receive, data will not be displayed.

SR8-UT4: if the handshake is failed the data received from client and it will throw an error message

**Bug Report:**

BR1

**Debugging Report:**

DR1

**Summary:**

This project will help implement a TCP protocol using UDP transmission. TCP 3-way handshake or the send/receive methods should be implemented. In this regard, we will be implementing TCP over UDP Socket.