Chat Application using TCP and UDP

# 1. Introduction

In this assignment, we implemented a simple chat application using Python's socket programming. The objective was to understand the difference between TCP (Transmission Control Protocol) and UDP (User Datagram Protocol), their working principles, and how they can be used to establish communication between two devices.

# 2. Objectives

The objectives of this project are:

• To implement a chat application using TCP protocol (reliable, connection-oriented).

• To implement a chat application using UDP protocol (faster, connectionless).

• To understand the differences between TCP and UDP.

• To test and verify communication between client and server.

# 3. Theory

## 3.1 Transmission Control Protocol (TCP)

TCP is a reliable, connection-oriented protocol. It ensures error-free delivery of data, guarantees order, and performs retransmission in case of packet loss. TCP is widely used in applications like web browsing, email, and messaging apps.

## 3.2 User Datagram Protocol (UDP)

UDP is a connectionless protocol that provides fast transmission but does not guarantee delivery or order. It is suitable for applications where speed is critical, such as video streaming, gaming, and VoIP.

## 3.3 Comparison between TCP and UDP

TCP provides reliability but is slower, whereas UDP is faster but may lose data. The choice between them depends on the application's requirements.

# 4. Implementation Steps

The following steps were followed to implement the chat application:

1. Create a server program using sockets.

2. Create a client program using sockets.

3. For TCP: Establish connection using connect() and accept().

4. For UDP: Send and receive messages without explicit connection.

5. Exchange messages between server and client.

6. Verify communication using print statements.

# 5. Explanation of Code

## 5.1 TCP Server

- socket.socket(AF\_INET, SOCK\_STREAM): Creates a TCP socket.  
- bind(): Binds the server to an IP and port.  
- listen(): Waits for client connections.  
- accept(): Accepts connection from a client.  
- recv(): Receives message from client.  
- send(): Sends message to client.

## 5.2 TCP Client

- connect(): Connects to the server.  
- send(): Sends message to server.  
- recv(): Receives reply from server.

## 5.3 UDP Server & Client

- socket.socket(AF\_INET, SOCK\_DGRAM): Creates a UDP socket.  
- bind(): Binds the server to an IP and port.  
- sendto(): Sends message to specified IP and port.  
- recvfrom(): Receives message from specified address.

# 6. Connection Diagram

The chat application connection can be represented as follows:

[ Client ] <--TCP/UDP--> [ Server ]  
  
The client initiates communication, while the server listens on a specific port. Messages are exchanged in both directions once the connection is established.

# 7. Verification of Successful Connection

To check if the connection is successful:  
• In TCP: If the client connects successfully, the server will display the client's IP address.  
• In UDP: The server will display the incoming message from the client.  
• If the client and server exchange messages without errors, the connection is working correctly.

# 8. Conclusion

This assignment helped in understanding the practical differences between TCP and UDP. TCP ensures reliable delivery of messages, while UDP provides faster but less reliable communication. The chat application demonstrated both protocols effectively.