Experiment 12 a): End-to-End Communication at Transport Layer

AIM:

To implement **an Echo Client-Server** and **a Chat Program** using **TCP/UDP socket programming** for end-to-end communication at the Transport Layer.

PROGRAM CODE

```
# ---------------------------------------------------------

# Title: Echo Client-Server and Chat Program using TCP Sockets

# ---------------------------------------------------------

# Name:

# Roll No:

# Date:

# ---------------------------------------------------------


import socket

import threading


# ------------------------

# Server Code

# ------------------------

def handle_client(client_socket, client_address):

    print(f"[+] New connection from {client_address}")

    while True:

        try:

            msg = client_socket.recv(1024).decode()

            if not msg:

                break

            print(f"[Client {client_address}] {msg}")

            # Echo the message back to client

            client_socket.sendall(f"Server received: {msg}".encode())

        except ConnectionResetError:

            break
```

```
    print(f"[-] Connection closed {client_address}")
    client_socket.close()

def start_server(host="127.0.0.1", port=5000):
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen(5)
    print(f"[SERVER] Listening on {host}:{port}...")

    while True:
        client_socket, client_address = server_socket.accept()
        client_thread = threading.Thread(
            target=handle_client, args=(client_socket, client_address)
        )
        client_thread.start()



# ------------------------
# Client Code
# ------------------------
def start_client(server_host="127.0.0.1", server_port=5000):
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server_host, server_port))
    print(f"[CLIENT] Connected to server {server_host}:{server_port}")

    try:
        while True:
            msg = input("Enter message (or 'quit' to exit): ")
            if msg.lower() == "quit":
                break
            client_socket.sendall(msg.encode())
```

```
            response = client_socket.recv(1024).decode()
            print(f"[SERVER RESPONSE] {response}")
    finally:
        client_socket.close()
        print("[CLIENT] Disconnected")


# ------------------------
# Run as server or client
# ------------------------
if __name__ == "__main__":
    import sys

    if len(sys.argv) > 1 and sys.argv[1] == "server":
        start_server()
    else:
        start_client()
```

SAMPLE INPUT AND OUTPUT

Step 1: Run the Server

$ python chat_program.py server

Server Output:

[SERVER] Listening on 127.0.0.1:5000...

[+] New connection from ('127.0.0.1', 60628)

[Client ('127.0.0.1', 60628)] Hello Server!

[Client ('127.0.0.1', 60628)] How are you?

[-] Connection closed ('127.0.0.1', 60628)

Step 2: Run the Client

$ python chat_program.py

Client Interaction:

[CLIENT] Connected to server 127.0.0.1:5000

Enter message (or 'quit' to exit): Hello Server!

[SERVER RESPONSE] Server received: Hello Server!

Enter message (or 'quit' to exit): How are you?

[SERVER RESPONSE] Server received: How are you?

Enter message (or 'quit' to exit): quit

[CLIENT] Disconnected

RESULT:

The **Echo Client-Server** and **Chat Program** were successfully implemented using **TCP sockets**. The client could send messages to the server, and the server echoed the same messages back, confirming reliable **end-to-end communication** at the **Transport Layer**.