EXP NO: 9 End-End communication at Transport Layer
10/10/25

AIM: To implement an Echo client-server & a chat Program using TCP/UDP socket Programming for end-end at the TP layer.

CODE: server code:

```
import socket
import threading
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}")
            client_socket.sendall(f"server received {msg}".encode())
        except ConnectionResetError:
            break
    print(f"[-] Connection closed {client_address}")
    client_socket.close()
```

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 & Output:

```
$ python chat_program.py server

O/P:
[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)

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 & chat program were successfully implemented using TCP sockets. The client could send messages to the server, & the server echoed the same messages back, confirming reliable end-end at TP layer.

9/10

(6)

AIM: To implement a UDP- based Echo (Ping) client - server program using socket program that measures RTT for each packet.

CODE:

```
import socket, time, sys

def udp_ping_server(host = "127.0.0.1", port=
                                    12000):
    s= socket.socket(socket.AF_INET,
        socket.SOCK_DGRAM)
    s.bind((host, port))
    while TRUE:
        msg, addr = s.recvfrom(1024)
    print(f"Received ")
    s.sendto(msg, addr).

def udp_ping_client(server = "127.0.0.1", port =
    12000, count=5):
    c= socket.socket(socket.AF_INET, socket.
                SOCK_DGRAM)
    c.settimeout(1)
    for i in range(1, count+1):
        msg = f"Ping {i}
        c.sendto(msg.encode(), (server, port))
        try:
            data, _ = c.recvfrom(1024)
            print(f"Reply: {data.decode()}
        except socket.timeout:
```

```
        print(f"Request {i} timed out")
    c.close()

if __name__ == "__main__":
    udp_ping_server()
    if len(sys.argv) > 1 and sys.argv[1] == "server":
        else udp_ping_client()
```

Sample Output:

Server:

```
[SERVER] Listening on 127.0.0.1:12000
Received 'Ping 1 ...' from ('127.0.0.1', 60642)
```

Client:

```
Reply: Ping 1 ... | RTT = 0.48 ms
Reply: Ping 2 ... | RTT = 0.50 ms.
```

RESULT:

The UDP Echo (Ping) client-server was successfully implemented.
It demonstrated connectionless, unreliable communication and measured RTT for each message.

End –End Communication at Transport Layer

a) Implement echo clientserver using TCP/UDP sockets.
b) Implement a chat program using socket programming.

TCP:

```
import socket
import threading

# ---------- SERVER ----------
def start_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('127.0.0.1', 5000))
    server.listen(1)
    print("[SERVER] Listening on 127.0.0.1:5000")

    conn, addr = server.accept()
    print(f"[SERVER] Connected by {addr}")

    while True:
        data = conn.recv(1024).decode()
        if not data or data.lower() == 'quit':
            print("[SERVER] Connection closed by client.")
            break
        print(f"[SERVER] Received: {data}")
        conn.send(f"Server received: {data}".encode())

    conn.close()
    server.close()
```

```
# ---------- CLIENT ----------
def start_client():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(('127.0.0.1', 5000))
    print("[CLIENT] Connected to server")

    while True:
        msg = input("Enter message (or 'quit' to exit): ")
        client.send(msg.encode())
        if msg.lower() == 'quit':
            break
        response = client.recv(1024).decode()
        print(f"[SERVER RESPONSE] {response}")

    client.close()

# ---------- MAIN ----------
if __name__ == "__main__":
    # Run server in a background thread
    server_thread = threading.Thread(target=start_server,
daemon=True)
    server_thread.start()

    # Run client in main thread
    start_client()
```

```
========== RESTART: C:/Users/tkala/Downloads/tcp.py ==========
[SERVER] Listening on 127.0.0.1:5000[CLIENT] Connected to server

[SERVER] Connected by ('127.0.0.1', 53458)Enter message (or 'quit' to exit):
hello
[SERVER] Received: hello
[SERVER RESPONSE] Server received: hello
Enter message (or 'quit' to exit):
========== RESTART: C:/Users/tkala/Downloads/tcp.py ==========
[SERVER] Listening on 127.0.0.1:5000[CLIENT] Connected to server

[SERVER] Connected by ('127.0.0.1', 60328)Enter message (or 'quit' to exit):
HI
[SERVER] Received: HI
[SERVER RESPONSE] Server received: HI
Enter message (or 'quit' to exit): quit
[SERVER] Connection closed by client.
```

```
import socket
import threading

# ---------- SERVER ----------
def udp_server():
    s = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
    s.bind(('127.0.0.1', 12000))
    print("[SERVER] Running on 127.0.0.1:12000")
    while True:
        data, addr = s.recvfrom(1024)
        msg = data.decode()
        if msg.lower() == 'quit':
```

```
            print(f"[SERVER] Client {addr} ended the
connection.")
            break
        print(f"[SERVER] Received from {addr}: {msg}")
        s.sendto(data, addr)  # Echo back
    s.close()
    print("[SERVER] Stopped.")


# ---------- CLIENT ----------
def udp_client():
    c = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
    server = ('127.0.0.1', 12000)
    while True:
        msg = input("Enter message (or 'quit' to stop): ")
        c.sendto(msg.encode(), server)
        if msg.lower() == 'quit':
            print("[CLIENT] Closing connection...")
            break
        data, _ = c.recvfrom(1024)
        print("Echo from Server:", data.decode())
    c.close()
    print("[CLIENT] Stopped.")
```

```
# ---------- MAIN ----------
if __name__ == "__main__":
    threading.Thread(target=udp_server,
daemon=True).start()
    udp_client()
```

```
Enter message (or 'quit' to stop): [SERVER] Running on 127.0.0.1:12000hi

[SERVER] Received from ('127.0.0.1', 49670): hi
Echo from Server: hi
Enter message (or 'quit' to stop): hello
[SERVER] Received from ('127.0.0.1', 49670): hello
Echo from Server: hello
Enter message (or 'quit' to stop): how are you
[SERVER] Received from ('127.0.0.1', 49670): how are you
Echo from Server: how are you
Enter message (or 'quit' to stop): quit
[CLIENT] Closing connection...[SERVER] Client ('127.0.0.1', 49670) ended the connection.

[CLIENT] Stopped.[SERVER] Stopped.
>>>
```

```
Command Prompt
Microsoft Windows [Version 10.0.26200.7019]
(c) Microsoft Corporation. All rights reserved.

C:\Users\tkala>cd Downloads

C:\Users\tkala\Downloads>python chatudp.py
Run as (s)erver or (c)lient? c
[CLIENT] Chat started with server
Client: hi
Server: hello
Client: where are u?
Server: college..
Client: ok
Server: quit

C:\Users\tkala\Downloads>
```

```
Command Prompt
Microsoft Windows [Version 10.0.26200.7019]
(c) Microsoft Corporation. All rights reserved.

C:\Users\tkala>cd Downloads

C:\Users\tkala\Downloads>python chatudp.py
Run as (s)erver or (c)lient? s
[SERVER] Waiting for messages...
Client: hi
Server: hello
Client: where are u?
Server: college..
Client: ok
Server: quit

C:\Users\tkala\Downloads>
```