Ex No : 12

DATE : 15-10-25

END - END COMMUNICATION AT TRANSPORT LAYER.

A) Implement echo client Server using TCP / UDP Sockets

AIM :

TO implement echo client Server using TCP / UDP Sockets

PROGRAM :

```
import Socket
import time
import sys
# UDP. ping [Echo] Server
def - udp- ping - server (host = "127.0.0.1", port = 1200)
Server_ socket = socket. Socket ( socket. AF _ INET, Socket -
    Sock _ DGRAM)
Server_ socket. Bind ((host, port))
Print (f" [SERVER] Listening on {host} : {port}")
while True :
    msg. client_ address = Server_ socket. receive from (1024)
    Print (f" [SERVER] received ' {msg. decode ()}'. from
    {client_ address}")
    Server - socket. send to (msg, client_ address)
#UDP. ping ( Echo) client
def _ udp _ ping _ client ( Server _ host = "127. 0. 0. 1", Serv from
                                    = 1200, count = 3);
    client _ socket = socket. Socket ( socket. AF=INET,
    socket. SOCK _ DGRAM)
for i in range (1, count + 1):
    msg = " ping {i} . {time. time()}"
    start = time. time()
    Client - Socket . Send to (msg. encode (), ( Server host,
Server _ port))
```

```
try:
    data = client_socket.recv(1024)
    end = time.time()
    rtt = (end - start) * 1000
    print(f"Reply from {server_host}: {server_port}")
    {data.decode()} | RTT = {rtt:.2f} ms")
except socket.timeout:
    print(f" Request {P} timed out")
client_socket.close()
# Run as server or client.
if __name__ == "__main__":
    if len(sys.argv) > 1 and sys.argv[1] == "server":
        udp_ping_server()
    else:
        udp_ping_client()
```

Input:

```
python udp_ping_program.py server.
python udp_ping_program.py
```

output:

Server Side output:

```
[SERVER] listening on 127.0.0.1:12000
[SERVER] Received 'ping' 1738842178.937154 from
('127.0.0.1', 54321)
[SERVER] Received 'ping' 1738842178.987654 from
('127.0.0.1', 54321)
```

Client Side Output:

Reply from 127.0.0.1:2000 | ping | 1738842178.987654 |

RTT = 1.23ms.

Reply from 127.0.0.1:2000 \ping 3 1738842178.987654)

RTT = 1.10 ms.

RESULT:

Therefore Implementation of echo client server using TCP | UDP Sockets is executed.

22/x/25

B) Implement chat client server using TCP / UDP Socket.

AIM:

To implement chat client server using TCP / UDP sockets.

PROGRAM: (TCP chat Server - client)

```
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}")
    client - socket. Send all ("f" Server received: {msg}".
encode ())
        except connection Reset Error:
            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 our server or client
if __name__ == "__main__":
    import sys
    if len(sys.argv) > 1 and sys.argv[1] == "server":
        start_server()
    else:
        start_client()
```

Input:

python tcp_chat_program.py server

python tcp_chat_ program.py

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

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

Enter message (or 'quit to exit): quit

output:

Server side output:

[SERVER] listening on 127.0.0.1:5000 . . .

[+] New connection from ('127.0.0.1', 54321)

[client ('127.0.0.1', 54321)]: Hello server

[client ('127.0.0.1', 54321)]: How are you?

[-] connection closed ('127.0.0.1', 54321)

client side output:

[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:

Therefore the implementation of chat client server using TCP/UDP sockets is executed.