Exp.: 12a Echo client server

TCP/UDP client

Date:

Aim:

The implement echo client server using TCP/UDP ~~sent~~ socket.

Algorithm:

Server.py:

→ Create a UDP socket

→ Bind the socket to specific IP address (192.0.0.1) & port (12345)

→ Continuous listen for incoming message

→ display message along with sender address.

→ Repeat infinitely.

client.py:

→ create UDP socket.

→ Set a timeout for socket to avoid waiting.

→ send a predefined message hello to server IP address & 12345.

8 → If no response, recieved in timeout period print timeout message

→ Close socket after sending message

Code:

Server.py

```
import socket;

def start server (host= '127.0.0.1'
                  port=12345):

    with socket.socket (socket.AFINET,
                socket.SOCK_Dgram) as S:

        S.bind ((host, port))
        print( f"UDP server running on
                    {host}: {port}")

        while True:
            data, addr = s.recvfrom (1024)
            print( f"recieved message from
                    {addr}: decode()}")

if __name__ == "__main__":
    start server ()
```

client.py

```
def ping-server (host = "127.0.0.1", port=12345):
    with socket.socket (socket.AF_INET,
                            socket ) as s:


if ~~__name__ == "__main__":~~
        ~~start server ()~~

        s.timeout (5)
        try:
           s.sendto (b'Hello', (host, port))
           print ("Message sent to server")
           except socket.timeout:
              print ("Request timed out")


   if __name__ == "__main__"
        ping-server ()
```

Output:

server.py

Terminal:

```
>python server.py

>>
UDP server running on 127.0.0.1:12345
```

Output: client.py

~~server.py~~ Terminal

```
>python client.py
>> message sent to server.
```

server terminal:

```
Received message from ("127.0.0.1', 5600
Hello.
```

Result:

Thus the program of echo client ~~server~~ using UDP sockets has been implemented & executed successfully.