

EXP NO: 12A

DATE : 28.9.25

Implementation of echo client server using TCP/UDP sockets

AIM:

To implement an echo client server by using TCP / UDP sockets.

Server - side Algorithm :

Import socket module

Server - socket = socket . socket (socket.AF\_INET, socket.SOCK\_STREAM)

Server - socket . listen (1)

printf ("Server is waiting for connections")

Conn, addr = Server - socket . accept ()

printf ("got connection to %s", addr)

while true:

data = Conn . recv (1024) . decode ()

if not data or data . lower () == "Beige":

printf ("Connection closed")

~~break~~

printf ("Received from Client : %s", data)

Conn . send (data . encode ())

Conn . close ()

Client - side algorithm:

```
import socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
client_socket.connect(('localhost', 12345))
```

while true:

```
    message = input("Enter message: ")
```

```
    client_socket.send(message.encode())
```

```
    if message.lower() == 'bye':
```

```
        break
```

```
    data = client_socket.recv(1024)
```

```
    decode()
```

```
    print(f"Echo from server: {data}")
```

```
client_socket.close()
```

Sample inputs and outputs:

Client side:

Enter message : Hello server.

Echo from server: Hello server.

Enter message : How are you,

Echo from server : How are you?

Enter message : Bye.

Server side : socketserver.py

881: ON 4/3

server is waiting for connection

Connected to ('127.0.0.1', 58949)

Received from client : Hello server

Received from client : How are you?

Connection closed

multicast bind error

: failed to find

((MAESTRA . test02 ) test02 . test02 = server ))

(MAESTRA . S00A . test02

((peer , test.local )) bind . server

of message (0 retur . null

break

("... maintained of pointer to server ") struct

( ) types . null = null , null

("global at maintained "f) struct

and listen

Global . (fd01) var . null = new

NOTE : fd1 = ( ) new . pinu f

RESULT : ( maintained "f) struct

the implementation of echo client

server (using TCP / UDP sockets) is excellent

successfully