

END-END communication At transport layer

AIM

to implement chat client server using TCP/UDP
sockets

Algorithm

- * start server: create socket, bind to port
- * receive message: server receive data from client
- * client prints echo message on screen.

Code:

```

import socket
mode = raw_input("Run as server or client?")
if mode == 'S':
    s = socket.socket()
    s.bind(['localhost', 12345])
    s.listen(1)
    print("server waiting for connection...")
    conn, addr = s.accept()
    print("connect", addr)
    data = conn.recv(1024)
    print("received", data)
    conn.send("Echo " + data)

```

client mode = 'c'; server mode = 's'

s = socket.socket()

s.connect('localhost', 12345)

msg = raw_input("Type a message")

s.send(msg)

print(s.recv(1024))

s.close()

Input:

Python TCP.py

Run as (s)erver or (c)lient?

s

server waiting for connection

Connected 12345

Type a message: Hello!

received: hello!

Code:

Implement chat client server using UDP socket

import socket

mode = raw_input("Run as (s)erver or (c)lient")

strip().lower()

if mode == 's':

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

s.bind('localhost', 12345)

Print ("UDP server waiting for message")

while true:

data, addr = s. recvfrom(1024)

Print ("Received", 'data, "from", addr)

s. sendto ("Echo : data, addr)

sleep mode = 'C':

s = socket. socket (socket. AF_INET, socket. SOCK
-D (or RAM))

server = ('localhost', 12345)

msg = new input ("Type a msg:")

s. sendto (msg. server)

data, - = s. recvfrom(1024)

Print ("Received from server", data)

s. close()

Input:

Python. udp. py

Run as (5) server or (1) client?

S

UDP server waiting for message

C

Output:

Type a message! 'Hello!'

Waiting...

received: "Hello!"

8

Result:

Implemented chat client server using TCP/UDP
sockets