

# END-END communication At transport layer

## AIM

to implement chat client server using TCP/UDP sockets

## Algorithm

- \* start server: create socket, bind to host
- \* receive message: server receive data from client
- \* client print echo message on screen.

## Code:-

```
import socket

mode = input ("Run as (server or client?)")

stop().lower()

if mode == 's'
    s = socket.socket()
    s.bind(['local host', 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))
    E
```

elif mode == 'c':

s = socket.socket()

s.connect(('localhost', 12345))

msg = raw\_input("Type a message")

s.send(msg)

Print(s.recv(1024))

s.close()

Input:

Python T/P.py

Run as (s) server or (c) client?

s

server waiting for connection

Connected 12345

c

Type a message: Hello!

received: Hello!

Code:

# Implement chat client server using UDP socket

import socket

mode = raw\_input("Run as (s) server or (c) client")

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.send to ("Echo: data, addr)

elif mode == 'c':

S = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server = ('localhost', 12345)

msg = raw\_input ("type a msg: ")

S.send to (msg, server)

data, \_ = S.recvfrom(1024)

Print ("Received from server", data)

S.close()

Input:

Python . udp.py

Run as (s) server or (c) client?

s

UDP server waiting for message

c

Output:

Type a message: 'Hello!'

waiting...

received: "Hello!"

Result:

Implemented chat client server using TCP/UDP sockets