**Experiment 07**

**Aim:** Socket programming using TCP or UDP.

**TCP Socket Connection :**

TCP server:

**(server.py)**

import socket

def server\_program():

host = socket.gethostname()

port = 5000

server\_socket = socket.socket()

server\_socket.bind((host, port))

server\_socket.listen(2)

conn, address = server\_socket.accept()

print("Connection from: " + str(address))

while True:

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

if not data:

break

print("from connected user: " + str(data))

data = input(' -> ')

conn.send(data.encode())

conn.close()

if \_\_name\_\_ == '\_\_main\_\_':

server\_program()

**(client.py)**

import socket

def client\_program():

host = socket.gethostname()

port = 5000

client\_socket = socket.socket()

client\_socket.connect((host, port))

message = input(" -> ")

while message.lower().strip() != 'bye':

client\_socket.send(message.encode())

data = client\_socket.recv(1024).decode()

print('Received from server: ' + data)

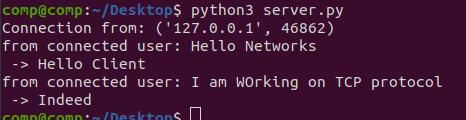
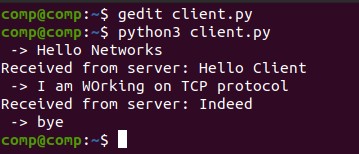
message = input(" -> ")

client\_socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

client\_program()

**Output:**

****

**UDP Socket Connection :**

UDP server :

**(server.py)**

import socket

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

udp\_host = socket.gethostname()

udp\_port = 5000

print ("UDP target IP:", udp\_host)

print ("UDP target Port:", udp\_port)

sock.bind((udp\_host,udp\_port))

while True:

data,addr = sock.recvfrom(1024)

print ("Received Messages:",data," from",addr)

if not data:

Break

data = input(" -> ")

sock.sendto(data.encode(),addr)

sock.close()

UDP client :

**(client.py)**

import socket

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

udp\_host = socket.gethostname()

udp\_port = 5000

print ("UDP target IP:", udp\_host)

print ("UDP target Port:", udp\_port)

message = input(" -> ")

while message.lower().strip() != 'bye':

sock.sendto(message.encode(),(udp\_host,udp\_port))

data,addr = sock.recvfrom(1024)

print ("Received Messages:",data," from",addr)

message = input(" -> ")

sock.close()

**Output:**

