# TOPIC: Socket Programming

**Objective:** Implement Socket Programming program using:

a. TCP Client Server programming.

b. UDP Client Server Programming.

**Problem Statement:**

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.

**Algorithm**

**TCP Server –**

1. using create(), Create TCP socket.
2. using bind(), Bind the socket to server address.
3. using listen(), put the server socket in a passive mode, where it waits for the client to approach the server to make a connection
4. using accept(), At this point, connection is established between client and server, and they are ready to transfer data.
5. Go back to Step 3.

**TCP Client –**

1. Create TCP socket.
2. connect newly created client socket to server.

**SERVER TCP:**

import socket

if \_name\_ == "\_main\_":

ip = "localhost"

port = 9999

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.bind((ip, port))

server.listen(5)

while True:

client, addr = server.accept()

print(f"Connection Established - {addr[0]}:{addr[1]}")

string = client.recv(1024).decode("utf-8")

string = string.upper()

client.send(bytes(string,"utf-8"))

  client.close()

**CLIENT TCP:**

import socket

if \_name\_ == "\_main\_":

ip = "localhost"

port = 9999

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.connect((ip, port))

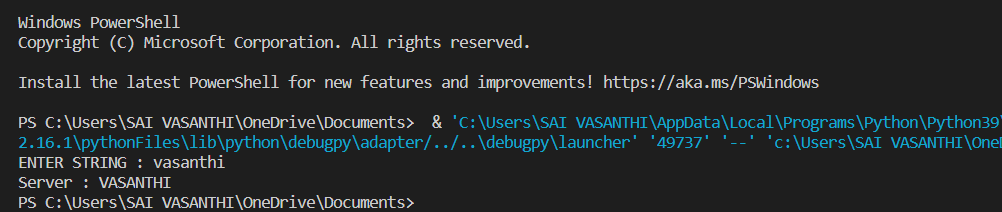
string = input("ENTER STRING : ")

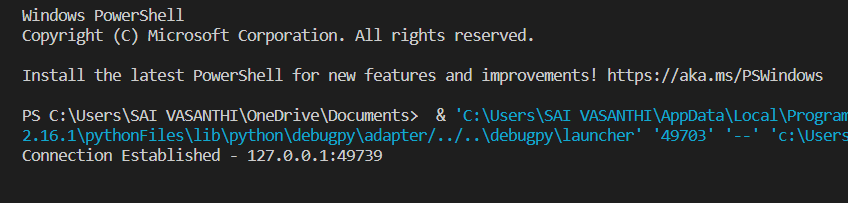
server.send(bytes(string, "utf-8"))

buffer = server.recv(1024).decode("utf-8")

print(f"Server : {buffer}")

**OUTPUT:**

****

****

An acknowledgment segment is present.

**Algorithm**

**UDP Server :**

1. Create a UDP socket.
2. Bind the socket to the server address.
3. Wait until the datagram packet arrives from the client.
4. Process the datagram packet and send a reply to the client.
5. Go back to Step 3.

**UDP Client :**

1. Create a UDP socket.
2. Send a message to the server.
3. Wait until response from the server is received.
4. Process reply and go back to step 2, if necessary.
5. Close socket descriptor and exit.

**UDP Server :**

import socket

if \_name\_ == "\_main\_":

host = "localhost"

port = 4455

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

server.bind((host, port))

while True:

data, addr = server.recvfrom(1024)

data = data.decode("utf-8")

print(data)

if data == "EXIT!!!":

print("Client disconnected.")

break

print(f"Client: {data}")

data = data.upper()

data = data.encode("utf-8")

server.sendto(data, addr)

**UDP Client :**

import socket

if \_name\_ == "\_main\_":

host = "localhost"

port = 4455

addr = (host, port)

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

while True:

data = input("Enter a word: ")

if data == "!EXIT":

data = data.encode("utf-8")

client.sendto(data, addr)

print("Disconneted from the server.")

break

data = data.encode("utf-8")

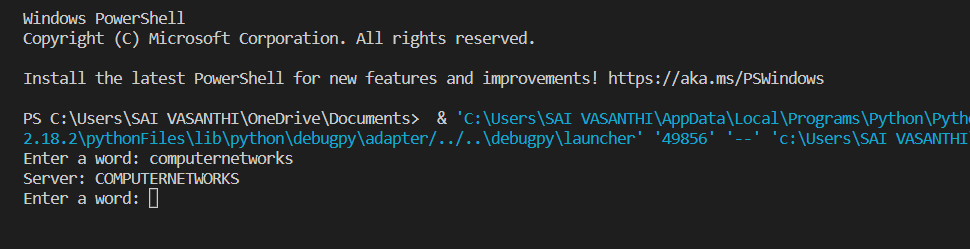
client.sendto(data, addr)

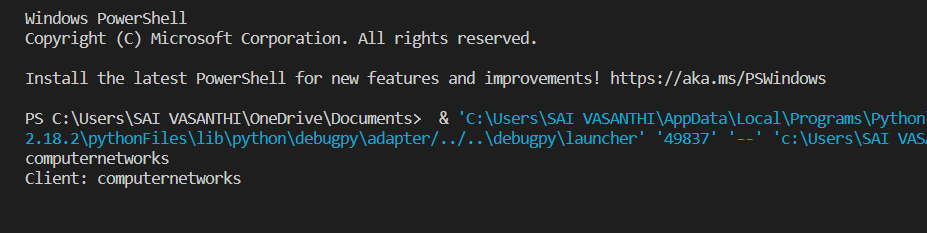
data, addr = client.recvfrom(1024)

data = data.decode("utf-8")

print(f"Server: {data}")

**OUTPUT:**





No acknowledgment segment.

**Problems Faced:**

In this experiment, I didn’t face much problems. Because, already we are familiar with this socket programming so it became easy to do.

**Conclusion:**

TCP is a connection-oriented, reliable, and stream-based protocol. It provides acknowledgment of the received packets. In order to establish a logical connection between the two ends, TCP exchanges some connection establishment packets. One socket is used by the TCP server to make connections, and the other one is used to carry data.

UDP is an unreliable, connectionless, and datagram service. There is no logical connection between two ends exchanging messages in this connection and does not provide acknowledgment. Both the client and the server only need one socket during UDP communication.