Date - 30/05/2021

Experiment No. 6
Client Server using UDP and Socket Programming

Aim:

To write a program to Implement Client-Server communication using Socket Programming and UDP as transport layer protocol.

Theory:

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 other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server. They are the real backbones behind web browsing. In simpler terms there is a server and a client.

Socket programming is started by importing the socket library and making a simple socket.

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

Here we made a socket instance and passed it two parameters. The first parameter is AF_INET and the second one is SOCK_STREAM. AF_INET refers to the address family ipv4. The SOCK_STREAM means connection oriented TCP protocol.

Python provides two levels of access to network services. At a low level, you can access the basic socket support in the underlying operating system, which allows you to implement clients and servers for both connection-oriented and connectionless protocols.

Python also has libraries that provide higher-level access to specific application-level network protocols, such as FTP, HTTP, and so on.

UDP or **User Datagram Protocol** is connectionless protocol which is suitable for applications that require efficient communication that doesn't have to worry about packet loss. In communications using UDP, a client program sends a message packet to a destination server wherein the destination server also runs on UDP.

1. The UDP does not provide guaranteed delivery of message packets. If for some issue in a network if a packet is lost it could be lost forever.
2. Since there is no guarantee of assured delivery of messages, UDP is considered an unreliable protocol.
3. The underlying mechanisms that implement UDP involve no connection based communication. There is no streaming of data between a UDP server or and an UDP Client.
4. An UDP client can send "n" number of distinct packets to an UDP server and it could also receive "n" number of distinct packets as replies from the UDP server.
5. Since UDP is a connectionless protocol the overhead involved in UDP is less compared to a connection based protocol like TCP.

Algorithm:

1. Client

Step 1. Start
Step 2. Import socket module
Step 3. Initialize msgFromClient, server port address, buffersize
Step 4. Create UDP client socket
Step 5. Send server the message via UDP socket
Step 6. Display the server response
Step 7. Stop

2. Server

Step 1. Start
Step 2. Import socket module
Step 3. Initialize localPort, localIP, bufferSize
Step 4. Create a datagram socket
Step 5. Bind it to IP and address
Stelt 6. Print the server running confirmation message
Step 7. Listen for Incoming datagrams from client device
Step 8. If detected display client IP, client message and send server response
Step 9. Stop

Program:

a. server.py

```
import socket
import time

localIP          = "127.0.0.1"
localPort        = 20001
bufferSize       = 1024
msgFromServer    = "Hello UDP Client"
bytesToSend      = str.encode(msgFromServer)


# Create a datagram socket
UDPServerSocket = socket.socket(family=socket.AF_INET,
type=socket.SOCK_DGRAM)

# Bind to address and ip
UDPServerSocket.bind((localIP, localPort))

print("UDP server up and listening")


# Listen for incoming datagrams
while(True):
        bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)
        message = bytesAddressPair[0]
        address = bytesAddressPair[1]
        clientMsg = "Message from Client:{}".format(message)
        clientIP  = "Client IP Address:{}".format(address)
        print(clientMsg)
        print(clientIP)

        # Sending a reply to client
        UDPServerSocket.sendto(bytesToSend, address)
```

b. client.py

```
import socket

msgFromClient       = "Hello UDP Server"
bytesToSend         = str.encode(msgFromClient)
serverAddressPort   = ("127.0.0.1", 20001)
bufferSize          = 1024

# Create a UDP socket at client side
UDPClientSocket = socket.socket(family=socket.AF_INET,
type=socket.SOCK_DGRAM)

# Send to server using created UDP socket
UDPClientSocket.sendto(bytesToSend, serverAddressPort)

msgFromServer = UDPClientSocket.recvfrom(bufferSize)
msg = "Message from Server {}".format(msgFromServer[0])
print(msg)
```

Output:

```
UDP server up and listening
Message from Server b"Hello UDP Client"
Client IP Address : ('127.0.0.1', 48910)
```

Result:

The program to implement Client-Server communication using Socket Programming and UDP as transport layer protocol has been executed successfully.