Date: 24-04-2021

**Lab Assignment No: 09**

**Aim:** To implement the socket programming for client server architecture.

**Lab Outcome Attained:** LO 4:- To implement client-server socket programming

**Theory: -**

# **Socket Programming:-**

* A socket is a mechanism for allowing communication between processes, be it programs running on the same machine or different computers connected on a network.
* More specifically, Internet sockets provide a programming interface to the network protocol stack that is managed by the operating system.
* Using this API, a programmer can quickly initialize a socket and send messages without having to worry about issues such as packet framing or transmission control.
* There are a number of different types of sockets available, but we are only really interested in two specific Internet sockets.
* These are:
  + Stream Sockets (Uses TCP)
  + Datagram Sockets (Uses UDP)
* A stream socket uses the Transmission Control Protocol (TCP) for sending messages. TCP provides an ordered and reliable connection between two hosts.
* This means that for every message sent, TCP guarantees that the message will arrive at the host in the correct order.
* This is achieved at the transport layer so the programmer does not have to worry about this, it is all done for you.
* A datagram socket uses the User Datagram Protocol (UDP) for sending messages.
* UDP is a much simpler protocol as it does not provide any of the delivery guarantees that TCP does.
* Messages, called datagrams, can be sent to another host without requiring any prior communication or a connection having been established.
* As such, using UDP can lead to lost messages or messages being received out of order.
* It is assumed that the application can tolerate an occasional lost message or that the application will handle the issue of retransmission.

# Datagram Sockets:

* A datagram is an independent, self-contained message sent over the network whose arrival, arrival time, and content are not guaranteed.
* The server continuously receives datagram packets over a datagram socket.
* Each datagram packet received by the server indicates a client request for a quotation.
* When the server receives a datagram, it replies by sending a datagram packet.
* The Client sends a single datagram packet to the server indicating that the client would like to receive the service.
* The client then waits for the server to send a datagram packet in response.
* Supports bidirectional flow of data that isn't guaranteed to be sequenced, reliable, or unduplicated.
* That is, a process receiving messages on a datagram socket may find messages duplicated, and possibly in an order other than the one in which they were sent.
* An important characteristic of a datagram socket is that record boundaries in data are preserved.
* Datagram sockets closely model the facilities found in many contemporary packet-switched networks (e.g. Ethernet).

# **Datagram Socket Programming:**

Server.py:

import socket

#  \* The Service provided to the Client is the Month of the Year instead of an integer

UDP\_IP = "127.0.0.1"

UDP\_PORT = 8091

# \* Creating a Socket for UDP

serverSocket = socket.socket(socket.AF\_INET, # Internet

                     socket.SOCK\_DGRAM) # UDP

months = ["January", "Feburary", "March", "April", "May", "June",

          "July", "August", "September", "October", "November", "December"]

serverSocket.bind((UDP\_IP, UDP\_PORT))    # \* Binding the Server to a Socket address of (localhost, 8091)

while True:

    print("Waiting for client")

    # \* Waiting for Client to request for our Service

    # \* Storing the Data that is send & address of the Client

    data, address = serverSocket.recvfrom(1024) # buffer size is 1024 bytes

    intro = """\nHey there I am going to provide you the month, if you Provide me an integer from 1-12\n"""

    # Sending a Intro message to the Client about the Service the Server is providing

    serverSocket.sendto(intro.encode('utf\_8', 'strict'), address)

    print("Connected to ",address)

    print("Client Also sent --> ", data.decode())

    # \* IF the client presses "0" then the connection with the Client will be terminated

    while data != "0":

        # Getting the Data from the Client i.e. integer

        data, address = serverSocket.recvfrom(1024) # buffer size is 1024 bytes

        if data != "0":

            # Sending Back the Month wrt to the Integer received

            serverSocket.sendto(months[int(data) - 1].encode('utf\_8', 'strict'), address)

        print("Client --> ", data.decode())

serverSocket.close() # Closing the Socket Object

Client.py:

import socket

UDP\_IP = "127.0.0.1"

UDP\_PORT = 8091

data = ""

conn = socket.socket(socket.AF\_INET,  # Internet

                     socket.SOCK\_DGRAM)  # UDP

# \* Creating the Connection with the Server with the socket address (localhost, 8091)

conn.connect((UDP\_IP, UDP\_PORT))

conn.sendto( "Hello I just want to use your Service".encode('utf\_8', 'strict'), (UDP\_IP, UDP\_PORT))

recieve, address = conn.recvfrom(1024)

print(recieve.decode()) # Printing the Intro given by the Server

# \* IF the client presses "0" then the connection with the Server will be terminated

while data != "0":

    data = input("Enter an integer ==> ")

    # \* Sending the Data to the Server with encoding

    conn.sendto(data.encode('utf\_8', 'strict'), (UDP\_IP, UDP\_PORT))

    recieve, address = conn.recvfrom(1024)

    # \* Decoding the data received from the Server and printing

    print(recieve.decode())

conn.close()

**Execution:**



Figure Server Side

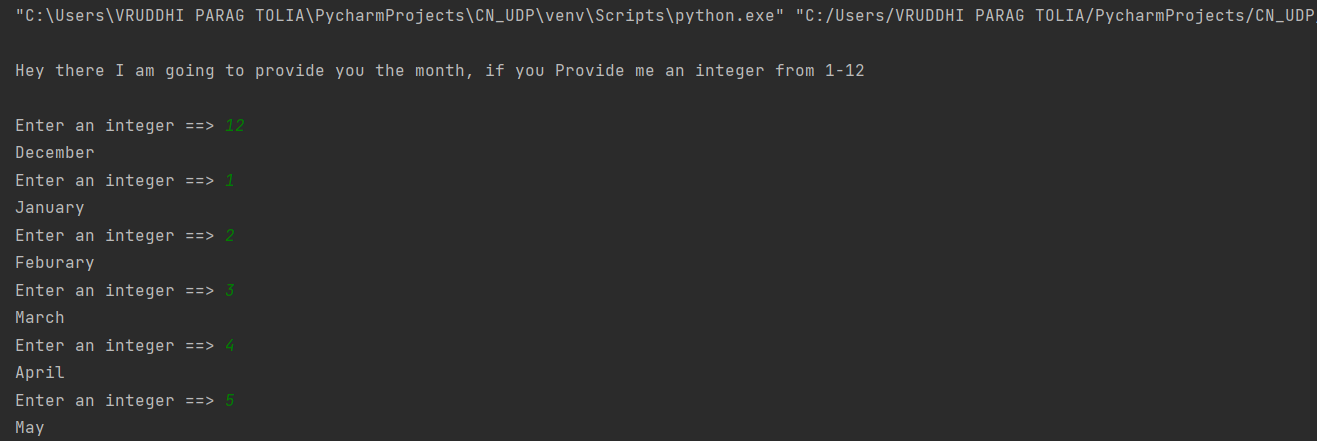


Figure Client Side

**Conclusion**: - Thus, understood to implement client-server datagram socket programming.