| | |
|---|---|
| Lab report no | 03 |
| Lab report name | Socket programming |
| ID | IT-17037 |

**Objective:-**

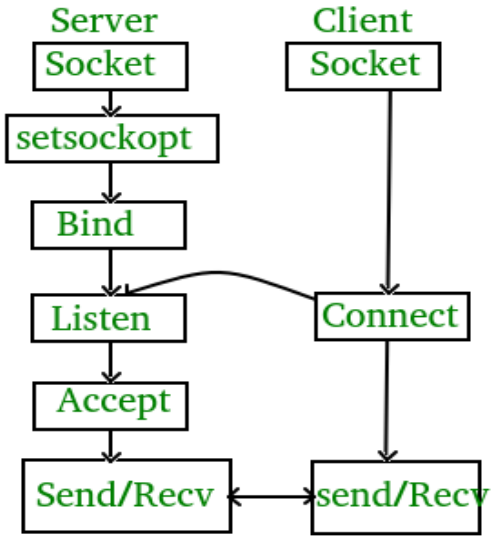
- To know the importance of protocol.
- To get familiar with python socket programming.

**Theory:-**

What is socket programming?

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.

**State diagram for server and client model**



Where is Socket Used?

A Unix Socket is used in a client-server application framework. A server is a process that performs some functions on request from a client. Most of the application-level protocols like FTP, SMTP, and POP3 make use of sockets to establish connection between client and server and then for exchanging data.

**Socket Types**

There are four types of sockets available to the users. The first two are most commonly used and the last two are rarely used.

Processes are presumed to communicate only between sockets of the same type but there is no restriction that prevents communication between sockets of different types.

- Stream Sockets − Delivery in a networked environment is guaranteed. If you send through the stream socket three items "A, B, C", they will arrive in the same order − "A, B, C". These sockets use TCP (Transmission Control Protocol) for data transmission. If delivery is impossible, the sender receives an error indicator. Data records do not have any boundaries.
- Datagram Sockets − Delivery in a networked environment is not guaranteed. They're connectionless because you don't need to have an open connection as in Stream Sockets − you build a packet with the destination information and send it out. They use UDP (User Datagram Protocol).
- Raw Sockets −These provide users access to the underlying communication protocols, which support socket abstractions. These sockets are normally datagram oriented, though their exact characteristics are dependent on the interface provided by the protocol. Raw sockets are not intended for the general user; they have been provided mainly for those interested in developing new communication protocols, or for gaining access to some of the more cryptic facilities of an existing protocol.
- Sequenced Packet Sockets −They are similar to a stream socket, with the exception that record boundaries are preserved. This interface is provided only as a part of the Network Systems (NS) socket abstraction, and is very important in most serious NS applications. Sequenced-packet sockets allow the user to manipulate the Sequence Packet Protocol (SPP) or Internet Datagram Protocol (IDP) headers on a packet or a group of packets, either

by writing a prototype header along with whatever data is to be sent, or by specifying a default header to be used with all outgoing data, and allows the user to receive the headers on incoming packets.

**Implementation of Socket**

Here we are exchanging one hello message between server and client to demonstrate the client/server model.

**Code for server socket:-**

```
import socket
s = socket.socket()
print('socket created')

s.bind(('localhost',9999))

s.listen(3)
print('waiting for connections')

while True:
   c, addr = s.accept()
   name = c.recv(1024).decode()
   print("connected with", addr,name)

   c.send(bytes('welcome ','utf-8'))

   c.close()
```

```
C:\Users\RAFATUL\AppData\Local\Programs\Python\Python38\python.exe C:/Users/RAFATUL/PycharmProjects/Sokets/server.py
socket created
waiting for connections
connected with ('127.0.0.1', 49811) Rafatul
connected with ('127.0.0.1', 49813) Shamim
```

**code for client socket:-**

```
import socket

c = socket.socket()

c.connect(('localhost',9999))

name = input("Enter your name")
c.send(bytes(name,'utf-8'))

print(c.recv(1024).decode())
```

```
C:\Users\RAFATUL\AppData\Local\Programs\Python\Python38\python.exe C:/Users/RAFATUL/PycharmProjects/Sokets/client.py
Enter your nameShamim
welcome

Process finished with exit code 0
```

**Disscussion:-** In this lab, we learn how to new communication with socket by python.