Socket Programming using Python

Server-Side

- The server accepts connections from the clients to establish a network interface.
- We need to ensure that clients/users are unique. Hence we assign a unique IP address to each client. However many users prefer to be identified by a username. Hence we will assign usernames as well.
- The role of the server is to collect any incoming messages and deliver them to the other client/clients.
- Let’s begin coding the server-side.
- Firstly, create a file named server.py.

Importing Required Libraries

- import socket
- import sys
- import time

  Firstly, we import the socket library because this python library contains the necessary functions to implement sockets.

  The sys library provides the system module which is responsible for providing data related to the system directory, functions, and methods.

  The time module enables us to perform numerous actions about the conversions and descriptions of time.

Creating the Socket and Retrieving the Hostname

- new_socket = socket.socket()
- host_name = socket.gethostname()
- s_ip = socket.gethostbyname(host_name)
- port = 8080

- The socket() function is a constructor of the socket library to create a socket.
- Once the socket is created, we retrieve the hostname/local device name using the gethostname(), which is again a function of the socket library.
- The gethostname() when sent with host_name as parameter retrieves the IP address of the other user and this IP is stored in s_ip.
- The port is assigned as 8080. This port is chosen because this is a default-free port on most machines. Generally, ports like 3000, 5000, etc are used for other applications like express.js. You can also run it on any port such as ‘1234’.

Binding the Host and Port

- new_socket.bind((host_name, port))
- print("Binding successful!”)
- print("This is your IP: ", s_ip)

  Now we will bind the port and host together using the bind function which is invoked on the socket object new_socket. Once the binding is successful, it prints “Binding successful!” on the console.

Listening for Connections

- name = input('Enter name:')
- new_socket.listen(1)

  Here, we use the listen() function which takes in one argument
  namely number_of_connections. This parameter can be any whole number such as 1,2,3,…

Accepting Incoming Connections

- conn, add= new_socket.accept()
- print("Received connection from ", add[0])
- print('Connection Established. Connected From: ',add[0])

The first variable which is conn is connected to the socket and the variable ‘add’ is assigned to the IP address of the client.

Storing Incoming Connection Data

- client = (conn.recv(1024)).decode()
- print(client + ' has connected.')
- conn.send(name.encode())
- The details of the incoming connection are stored in the client_name variable. The client's name can be a maximum of 1024 bytes. It is decoded on the server and prints a message that it has been connected. The server then sends the hostname.

Delivering Packets/Messages

```
while True:
    message = input('Me : ')
    conn.send(message.encode())
    message = conn.recv(1024)
    message = message.decode()
    print(client, ':', message)
```

The user enters the message. This is encoded using encode() and then sent across through the socket. The message is sent using the send() function which is invoked on the connection object created during the invocation of accept() function earlier. It then displays “message has been sent…”.

The incoming message is received using the recv() of the conn object. It can receive up to 1024 bytes of information. The message is decoded on the server-side using decode().

Client-Side

- Here we create a file named client.py

Importing Libraries

- import socket
- import sys
- import time

Creating the Socket and Accepting User Input Hostname

- socket_server = socket.socket()
- server_host = socket.gethostname()
- ip = socket.gethostbyname(server_host)
- sport = 8080

- The socket at the server is created using the socket() method.
- The hostname of the server is retrieved at the client-side and stored as server_host.
- The IP address is stored in ip.
- The server port is stored as 8080 in the sport variable.
- Please note, this port must match with the port mentioned on the server-side code.

Connecting to the Server

- print('This is your IP address: ', ip)
- server_host = input('Enter friend\'s IP address:')
- name = input('Enter Friend\'s name: ')
- socket_server.connect((server_host, sport))

- The details of the server (friend) is entered first. Please note, it is important to enter the exact IP address otherwise the communication will fail.
- The hostname of the server and the port are bound together in a way and connected to the socket.

Receiving Packets/Messages from the Server

- socket_server.send(name.encode())
- server_name = socket_server.recv(1024)
- server_name = server_name.decode()
- print(server_name,' has joined...')
- while True:
- message = (socket_server.recv(1024)).decode()
- print(server_name, ":", message)
- message = input("Me : ")
- socket_server.send(message.encode())

- To receive messages, the socket_server invokes the recv() function to accept 1024 of data. This is stored in the message object and decoded using decode() function. The message is then printed with the hostname of the server and the message received.
- The client can enter any message as input and encode the same and send it to the server using the socket.