Data Communication and Networks Lab

Experiment 8

Name : Ojasa Chitre

TE Comps

Batch : A

UID : 2018130007

CEL 51, DCCN, Monsoon 2020

Lab 7: Socket Programming

Theory :

A *socket* is a communications connection point (endpoint) that you can name and address in a network. Socket programming shows how to use socket APIs to establish communication links between remote and local processes.
The processes that use a socket can reside on the same system or different systems on different networks. Sockets are useful for both stand-alone and network applications. Sockets allow you to exchange information between processes on the same machine or across a network, distribute work to the most efficient machine, and they easily allow access to centralized data. Socket application program interfaces (APIs) are the network standard for TCP/IP. A wide range of operating systems support socket APIs. i5/OS™ sockets support multiple transport and networking protocols. Socket system functions and the socket network functions are thread safe.

Code :

server.py

```
import socket

# creation of socket object
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print("Socket successfully created")

# reserving a port on the computer
port = 12345
```

```
# binding ip address and port
# empty string means server will listen to requests coming from other computer
# s on the network
s.bind(('', port))
print("socket binded to %s" %(port) )

# puting the socket into listening mode
s.listen(5)
print("socket is listening")

while True:

 # Establishing a connection with client.
 c, addr = s.accept()
 print('Got connection from', addr )

 # sending a thank you message to the client.
 c.send(b'Thank you for connecting')

 # Close the connection with the client
 c.close()
```

- First of all we import socket which is necessary.
- Then we made a socket object and reserved a port on our pc.
- After that we binded our server to the specified port. Passing an empty string means that the server can listen to incoming connections from other computers as well. If we would have passed 127.0.0.1 then it would have listened to only those calls made within the local computer.
- After that we put the server into listen mode.5 here means that 5 connections are kept waiting if the server is busy and if a 6th socket trys to connect then the connection is refused.
- At last we make a while loop and start to accept all incoming connections and close those connections after a thank you message to all connected sockets.

client.py

```
import socket

# Creating a socket object
s = socket.socket()

# Defining the port on which we want to connect
port = 12345

# connecting to the server on local computer
s.connect(('127.0.0.1', port))

# receiving data from the server
```

```
print(s.recv(1024) )
# closing the connection
s.close()
```

- First of all we make a socket object.
- Then we connect to localhost on port 12345 (the port on which our server runs) and lastly we receive data from the server and close the connection.
- Now save this file as client.py and run it from the terminal after starting the server script.

Output :

```
PS C:\Users\ojasa\Documents\engineering\TE\DCCN\Lab> python server.py
Socket successfully created
socket binded to 12345
socket is listening
Got connection from ('127.0.0.1', 54927)
```

```
PS C:\Users\ojasa\Documents\engineering\TE\DCCN\Lab> python client.py
b'Thank you for connecting'
PS C:\Users\ojasa\Documents\engineering\TE\DCCN\Lab>
```

Conclusion :

I was able to understand the basics of socket programming. I have been able to establish a communication between a client and server.

Reference :

- https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_71/rzab6/rzab6soxoverview.htm
- https://www.geeksforgeeks.org/socket-programming-python/#:~:text=Socket%20programming%20is%20a%20way,reaches%20out%20to%20the%20server.