COMPUTER COMMUNICATION NETWORKS

LAB EXPERIMENT 7

NAME: RAHIL SHARMA
PRN: 18070123062
BATCH: 2018-2022
DIVISION: G2; EA 3

Aim: CLIENT SERVER PROGRAM FOR ECHO.

Following are the aim for this Lab.

i. Write a server side program in Python for establishing a socket connection to receive incoming requests from client and echo back whatever is received from client.
ii. Write a client side program in Python for establishing a socket connection with a server. Send a message to server and get echo back.

Client Side: Client-side refers to operations that are performed by the client in a client–server relationship in a computer network. When the server serves data in a commonly used manner, for example according to standard protocols such as HTTP or FTP, users may have their choice of a number of client programs (e.g. most modern web browsers can request and receive data using both HTTP and FTP). In the case of more specialized applications, programmers may write their own server, client, and communications protocol which can only be used with one another.

Programs that run on a user's local computer without ever sending or receiving data over a network are not considered clients, and so the operations of such programs would not be termed client-side operations.

Server Side: **Server-side** refers to operations that are performed by the server in a client–server relationship in a computer network. Client and server programs may be commonly available ones such as free or commercial web servers and web browsers, communicating with each other using standardized protocols. Or, programmers may write their own server, client, and communications protocol which can only be used with one another.

Server-side operations include both those that are carried out in response to client requests, and non-client-oriented operations such as maintenance tasks.

Procedure for Client Side

1. Declare a socket variable.
2. Call wsastartup() to initialize socket library.
3. Call socket() to create a socket
4. Declare a variable for the socket addr structure. Set all the relevant variables in the structure.
5. Call bind() and then listen() since we are writing the program on server side and server listens to client request.
6. Call accept() when a client request comes.
7. Call read() to receive data from client.
8. Call write to send back (echo) whatever data is received from client.
9. Close socket connection and exit.
10. Put proper messages at relevant places in the code to indicate successful execution of any function. Or failure.

Procedure for Server Side

The steps are same as server side except following differences

1. Client will not call bind() and listen() functions.
2. Client will call connect() instead of accept().
3. Client will first call write() to send something to server and then call read(0 to receive from server.
4. If the server is setup to close socket connection on one exho then client has to re-connect after each echo.
5. Write the code based on server program and above instructions.
6. Run the code. You will be able see either successful or failure messages.
7. Take screen shot of the output and paste in the journal

Code and OUTPUT of the program:

CLIENT SIDE:

jupyter CCN_LAB7_18070123062 Last Checkpoint: 11 minutes ago (unsaved changes)

In [*]:

```python
import socket


def server_program():
    host = socket.gethostname()        # to get the host name
    port = 5000                        # initiating port no. above 1024
    print("Creating socket.... ")
    server_socket = socket.socket()    # Create socket
    print("Socket created, now binding with host - {} and port - {}".format(host, port))
    server_socket.bind((host, port))   # Binding host address with port address
    print("Socket Bound")

    server_socket.listen(1)            # Server listening to one client
    print("Listening.... ")
    conn, address = server_socket.accept()  # accepting new connection
    print("Connection from: " + str(address))
    while True:
        # receive data stream. It won;t accept data packet greater than 1024 bytes.
        data = conn.recv(1024).decode()
        if data == 'bye':
            print("From connected user: " + str(data))
            break
        print("From connected user: " + str(data))
        data = "Echo confirmed, thank you for Connecting!"
        conn.send(data.encode())         # Send data to the client

    print("Closing Connection... ")
    conn.close()     # close the connection


if __name__ == "__main__":
    server_program()

```

```
Creating socket....
```

jupyter CCN_LAB7_18070123062 Last Checkpoint: 12 minutes ago (unsaved changes)

```python
def server_program():
    host = socket.gethostname()        # to get the host name
    port = 5000                        # initiating port no. above 1024
    print("Creating socket.... ")
    server_socket = socket.socket()    # Create socket
    print("Socket created, now binding with host - {} and port - {}".format(host, port))
    server_socket.bind((host, port))   # Binding host address with port address
    print("Socket Bound")

    server_socket.listen(1)            # Server listening to one client
    print("Listening.... ")
    conn, address = server_socket.accept()  # accepting new connection
    print("Connection from: " + str(address))
    while True:
        # receive data stream. It won;t accept data packet greater than 1024 bytes.
        data = conn.recv(1024).decode()
        if data == 'bye':
            print("From connected user: " + str(data))
            break
        print("From connected user: " + str(data))
        data = "Echo confirmed, thank you for Connecting!"
        conn.send(data.encode())         # Send data to the client

    print("Closing Connection... ")
    conn.close()     # close the connection


if __name__ == "__main__":
    server_program()

```

```
Creating socket....
Socket created, now binding with host - Rahils-MacBook-Air-3.local and port - 5000
Socket Bound
Listening....
```

SERVER SIDE:

jupyter Untitled1 Last Checkpoint: a minute ago (unsaved changes)

In [1]:

```python
import socket


def client_program():
    host = socket.gethostname()        # As both code are running on same pc
    port = 5000                        # Socket server port number
    print("Creating socket.... ")
    client_socket = socket.socket()    # Instantiate
    print("Socket created, now binding with host - {} and port - {}".format(host, port))
    client_socket.connect((host, port))   # Connect to the server
    print("Socket Bound")

    message = input("Enter echo message to the send (under 1024 bytes): ")

    while message.lower().strip() != 'bye':  # termination check
        print("Sending Message.... ")
        client_socket.send(message.encode())       # send message
        data = client_socket.recv(1024).decode()   # receive response
        print("Received from server: {}".format(data))   # Show in terminal
        message = input("Enter 'bye' to end or another message: ")
        if message == 'bye':
            client_socket.send(message.encode())

    print("Closing Connection... ")
    client_socket.close()     # close the connection


if __name__ == "__main__":
    client_program()
```

```
Creating socket....
Socket created, now binding with host - Rahils-MacBook-Air-3.local and port - 5000
Socket Bound
Enter echo message to the send (under 1024 bytes): Hey Server! It's Rahil Here
```

jupyter Untitled1 Last Checkpoint: a minute ago (unsaved changes)

```python
    message = input("Enter echo message to the send (under 1024 bytes): ")

    while message.lower().strip() != 'bye':  # termination check
        print("Sending Message.... ")
        client_socket.send(message.encode())       # send message
        data = client_socket.recv(1024).decode()   # receive response
        print("Received from server: {}".format(data))   # Show in terminal
        message = input("Enter 'bye' to end or another message: ")
        if message == 'bye':
            client_socket.send(message.encode())

    print("Closing Connection... ")
    client_socket.close()     # close the connection


if __name__ == "__main__":
    client_program()
```

```
Creating socket....
Socket created, now binding with host - Rahils-MacBook-Air-3.local and port - 5000
Socket Bound
Enter echo message to the send (under 1024 bytes): Hey Server! It's Rahil Here
Sending Message....
Received from server: Echo confirmed, thank you for Connecting!
Enter 'bye' to end or another message: Bye
Closing Connection...
```

In [ ]:

<u>CONCLUSION:</u> From this experiment we have learnt how to develop CLIENT and SERVER Side Requests.