Practical No.5

AIM:- Implement client-server application for chat server using TCP/IP protocol.

SOFTWARE REQUIRED:-

Operating System: - Ubuntu Python 3

THEORY:-

- TCP protocol :-

TCP is connection oriented – reliable protocol which transfers the data in continues streams. once a connection is established, data can be sent bidirectional. It is a two way communication.

ALGORITHM:

Server-Side Algorithm (TCP Chat Application):

1. **Create two ports**:
    - One for receiving messages from the client (client port).
    - One for sending messages to the client (server port).
2. **Create a TCP socket**, bind it to the client port, and start listening for incoming connections.
3. **Accept a client connection** and establish a communication channel.
4. **Receive client message** and display it.
5. **Get data from the server user** and send a response back to the client.
6. **Repeat steps 4-5** until the client stops sending messages (or a termination keyword like bye is sent).
7. **Close the connection** and the server socket.
8. **Stop the program**.

Client-Side Algorithm (TCP Chat Application):

1. **Create two ports**:
    - One for sending data to the server (server port).
    - One for receiving messages from the server (client port).
2. **Create a TCP socket** and connect it to the server's IP address and server port.
3. **Get data from the client user** to send to the server.
4. **Send the data** to the server and wait for a response.
5. **Receive the server's response** and display it.
6. **Repeat steps 3-5** until the user sends a termination keyword like `bye`.
7. **Close the client socket**.
8. **Stop the program**.

Communication Processs :-

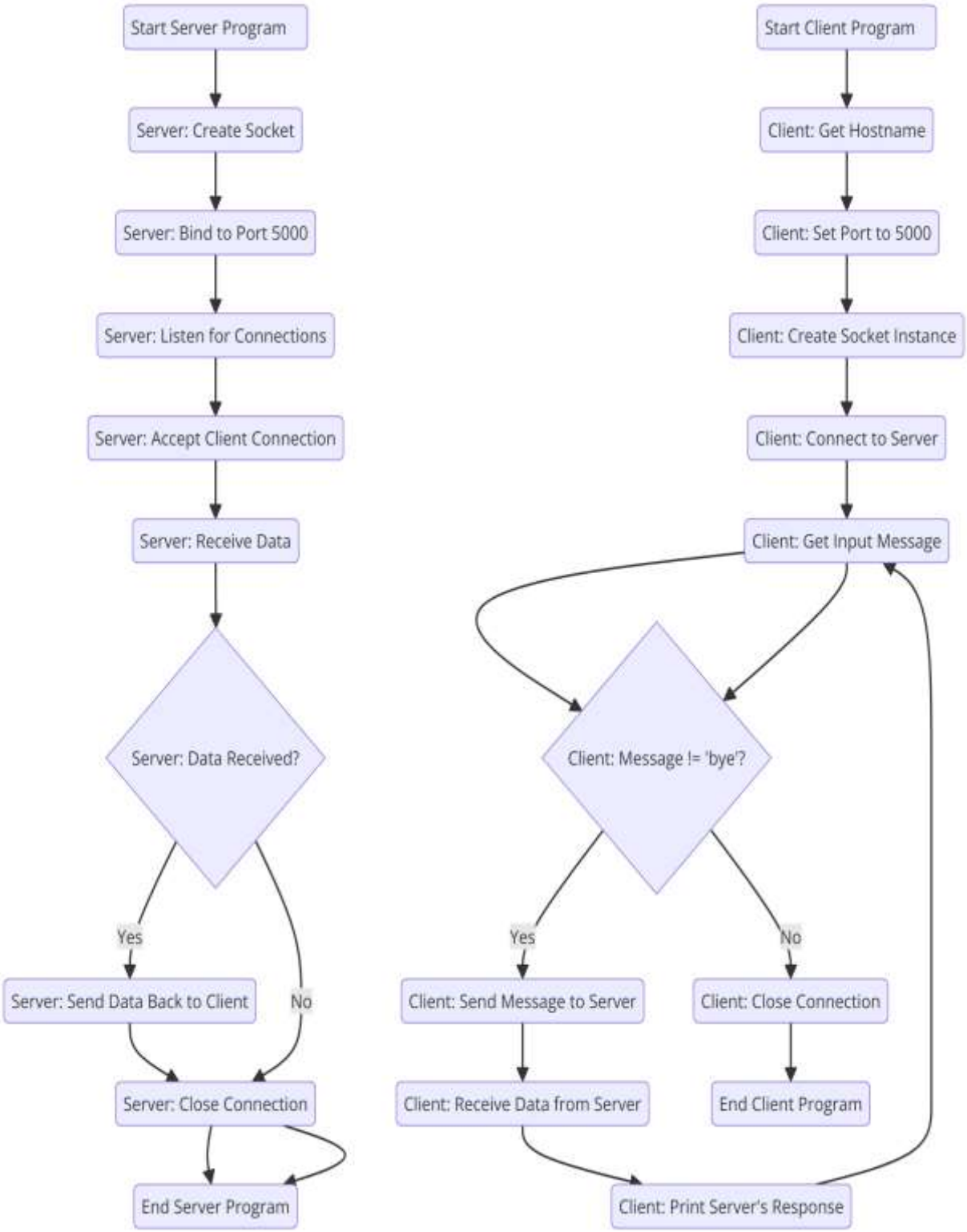


Fig .Communication process of Server-Client Chat

CONCLUSION:-

Thus the program of Client-server application for chat server using TCP/IP protocol was executed and output is verified. Course Teacher

Course Teacher
Ms. Prajakta Sawle

Program : Server

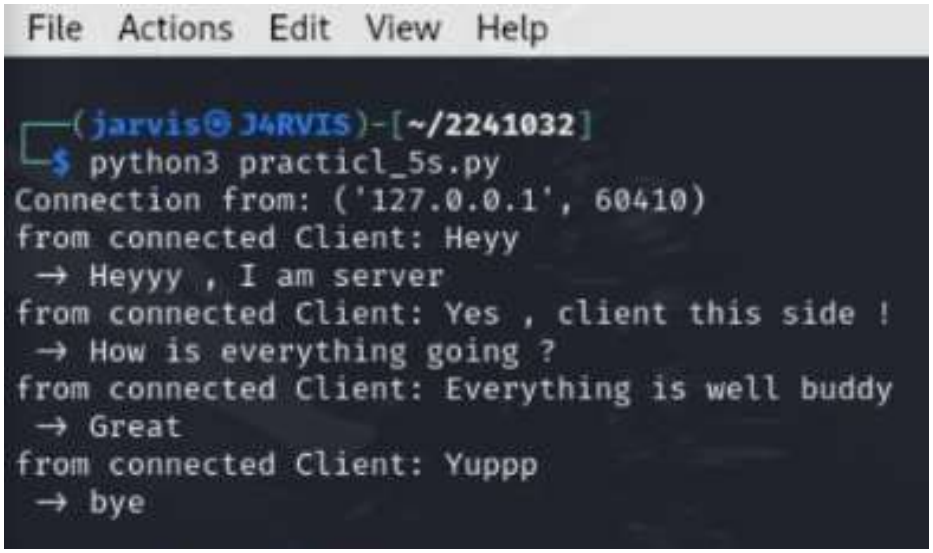
```
import socket

def server_program( ):
        host =socket.gethostname ( )
        port=5000

        server_socket.listen(2)
        conn,address=server_socket.accept( )
        print("Connection from: "+str(address))

        while True:
                data=conn.recv(1024).decode( )
                if not data:
                        break
                print("From connected Client: "+str(data))
                data=input('  - >')
                conn.send(data.encode( ))
        conn.close( )

if__name__= = '__main__' :
        server_program( )
```

Client:

```
import socket

def server_program( ):
        host =socket.gethostname ( )
        port=5000

        client_socket=socket.socket( )
        client_socket.connect((host,port))

        message=input(" ->")

        while message.lower().strip() !='bye':
                client_socket.send(message.encode ( ))
                data=client_socket.recv(1024).decode( )

                print("Received from server: "+data)

                message=input("->")
        client_socket.close ( )

if__name__= = '__main__":
        client_program( )
```

Output:

Server



Client