EXPERIMENT NO: 2

Date of Performance:

Date of Submission:

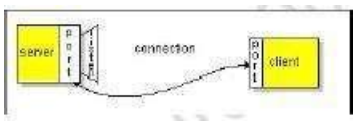
AIM : Client/server using RPC/RMI.

Theory:

The client-server model is one of the most used communication paradigms in networked systems. The server accepts the connection from the client, binds a new socket to the same local port, and sets its remote endpoint to the client's address and port. It needs a new socket so that it can continue to listen to the original socket for connection requests when the attention needs for the connected client.

Creating a server program:

The EchoServer example creates a server socket and waits for a client request. When it receives a client request, the server connects to the client and responds to it.



Following are the steps to create echo server application

- **Create and open a server socket.**

  ServerSocketserverSocket = new ServerSocket(portNumber);

The portNumber argument is the logical address through which the application communicates over the network. It's the port on which the server is running. You must provide the port numberthrough which the server can listen to the client. Don't select port numbers between 0 and 1,023because they're reserved for privileged users (that is, super user or root). Add the server socket inside the try with- resources block.

- **Wait for the client request.**

  Socket clientSocket = serverSocket.accept();

The accept() method waits until a client starts and requests a connection on the host and port ofthis server. When a connection is requested and successfully established, the accept()method returns a new Socket object. It's bound to the same local port, and its remote address and remote port are set to match the client's. The server can communicate with the client over this new object and listen for client connection requests.

- **Open an input stream and an output stream to the client.**
  out = new PrintWriter(clientSocket.getOutputStream(), true);

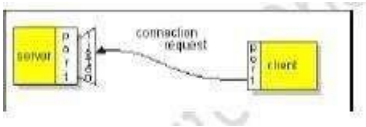
  in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

- **Communicate with the client.**
  Receive data from the client: (inputLine = in.readLine())Send data to the client: out.println(inputLine);

- **Close the streams and then close the socket.**

Creating a client program:

The client knows the host name of the machine on which the server is running. It also knows theport number on which the server is listening. To make a connection request, the client tries to connect with the server on the server's machine and port. Because the client also needs to identify itself to the server, it binds to a local port number that it will use during this connection.The system typically assigns the port number.



Following are the various steps to create client.

- **Create and open a client socket.**

Socket echoSocket = new Socket(hostName, portNumber);

The hostName argument is the machine where you are trying to open a connection, and portNumber is the port on which the server is running. Don't select port numbers between 0 and 1,023 because they're reserved for privileged users (that is, super user or root).

- **Open an input stream and an output stream to the socket.**
  PrintWriter out = new PrintWriter(echoSocket.getOutputStream(), true);BufferedReader in = new BufferedReader(new InputStreamReader(echoSocket.getInputStream()));

- **Read from and write to the stream according to the server's protocol.**

Receive data from the server: (userInput =

stdIn.readLine())Send data to the server:

out.println(userInput);

- **Close the streams and then close the socket.**

CODE:

```
# Server code
from xmlrpc.server import
SimpleXMLRPCServerimport
multiprocessing

def
    add(x,
    y):
    return
    x + y

def start_server():
    server = SimpleXMLRPCServer(("localhost",
    8000))server.register_function(add, "add")
    server.serve_forever()

if _name_ == "_main_":
    # Start server in a separate process
    server_process =
    multiprocessing.Process(target=start_server)
    server_process.start()
```

```
# Client  code
import
xmlrpc.client

def remote_add(x, y):
    proxy =
    xmlrpc.client.ServerProxy("http://localhost:8000/")
    return proxy.add(x, y)

if _name_ ==
    "_main_": result =
    remote_add(4, 5)
    print("Result:",
    result)
```

Output:

```
Result: 9
```

Conclusion: Client/server using RPC/RMI has been implemented.

SIGN AND REMARK

DATE:

R1 (4 Marks)	R2 (4 Marks)	R3 (4 Marks)	R4 (3 Mark)	Total (15 Marks)	Signature