

LAB 2

Aim: To implement Remote Procedure Call

Lab Outcome:

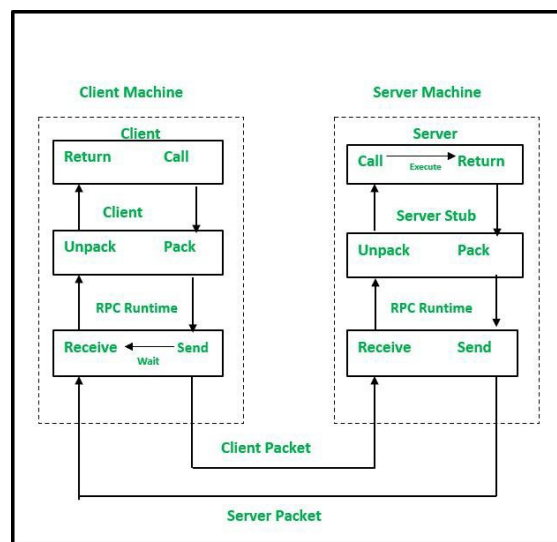
Develop test and debug using Message-Oriented Communication or RPC/RMI based client-server programs

Theory:

RPC is an effective mechanism for building client-server systems that are distributed. RPC enhances the power and ease of programming of the client/server computing concept. It is a protocol that allows one software to seek a service from another program on another computer in a network without having to know about the network. The software that makes the request is called a client, and the program that provides the service is called a server.

There are 5 elements used in the working of RPC:

- Client
- Client Stub
- RPC Runtime
- Server Stub
- Server



- The client, the client stub, and one instance of RPC Runtime are all running on the client machine.

- A client initiates a client stub process by giving parameters as normal. The client stub acquires storage in the address space of the client.
- At this point, the user can access RPC by using a normal Local Procedural Call. The RPC runtime oversees message transmission between client and server via the network. Retransmission, acknowledgment, routing, and encryption are all tasks performed by it.
- On the server-side, values are returned to the server stub, after the completion of server operation, which then packs (which is also known as marshalling) the return values into a message. The transport layer receives a message from the server stub.
- The resulting message is transmitted by the transport layer to the client transport layer, which then sends a message back to the client stub.
- The client stub unpacks (which is also known as unmarshalling) the return arguments in the resulting packet, and the execution process returns to the caller at this point.

Code & Output:

```

server.py
1 import socket
2 import pickle
3
4 # Define the function to be called remotely
5 def add(numbers):
6     return numbers['a'] + numbers['b']
7
8 # Create a socket and bind it to a port
9 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10 server_socket.bind(("localhost", 5000)) # Change "localhost"
11 server_socket.listen(1)
12 print("RPC Server listening on port 8000...")
13
14 while True:
15     # Wait for a connection
16     client_socket, _ = server_socket.accept()
17     print("Connected to client")
18
19     # Receive data from the client
20     data = client_socket.recv(1024)
21     if not data:
22         break
23
client.py
1 import socket
2 import pickle
3
4 # Function to call the RPC server
5 def add_prog(host, x, y):
6
7     # Create a socket and connect to the server
8     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9     client_socket.connect(("localhost", 5000)) # Change "localhost"
10
11     # Prepare data to be sent
12     numbers = {'a': x, 'b': y}
13     data = pickle.dumps(numbers)
14
15     # Send data to the server
16     client_socket.send(data)
17
18     # Receive result from the server
19     result = client_socket.recv(1024)
20
21     # Deserialize received data
22     result = pickle.loads(result)
23
Terminal Output:
PS C:\Users\krisc\Documents\Notes\codes> & C:/Python312/python.exe c:/Users/krisc/
krisc\Documents\Notes\codes/dc2/server.py
RPC Server listening on port 8000...
Connected to client

PS C:\Users\krisc\Documents\Notes\codes> python client.py localhost 5 7
C:\Python312\python.exe: can't open file 'C:\Users\krisc\Documents\Notes\codes\client.py': [Errno 2] No such file or directory
PS C:\Users\krisc\Documents\Notes\codes> cd dc
PS C:\Users\krisc\Documents\Notes\codes\dc> cd 2
PS C:\Users\krisc\Documents\Notes\codes\dc2> python client.py localhost 5 7
Result: 12
PS C:\Users\krisc\Documents\Notes\codes\dc2>

```

Conclusions:

In conclusion, the remote procedure call (RPC) is a powerful technology that enables communication between processes running on different machines in a networked environment. The experiment performed on RPC in C language has demonstrated its ability to enable distributed computing across different machines.

Postlab Questions:

1. In which category of communication, RPC be included?
2. What are stubs? What are the different ways of stub generation?
3. What is binding?
4. Name the transparencies achieved through stubs