Experiment Details

|  |  |
| --- | --- |
| Department Name | Computer Science Engineering |
| Class | B.Tech |
| Semester | 7th |
| Subject Name | Distributed System Lab |
| Experiment No. | Exp 1 |
| Experiment Name | Client-Server With RPC |

Version History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. No. | Version Number | Created By | Approved By | Date |
| 1 | v1.0 | Rushikesh Jirage | Prof. Shivani Kale ma’am | 11/10/2020 |
|  |  |  |  |  |

**AIM:**

To study client-server based program.

**THEORY:**

RPC is a powerful technique for constructing distributed, client-server based applications.

It is based on extending the notion of conventional or local procedure calling, so that the

called procedure need not exist in the same address space as the calling procedure. The two

processes may be on the same system, or they may be on different systems with a network

connecting them. By using RPC, programmers of distributed applications avoid the details of

the interface with the network. The transport independence of RPC isolates the application

from the physical and logical elements of the data communications mechanism and allows

the application to use a variety of transports.

RPC makes the client/server model of computing more powerful and easier to program.

When combined with the ONC RPCGEN protocol compiler clients transparently make

remote calls through a local procedure interface.

An RPC is analogous to a function call. Like a function call, when an RPC is made, the

calling arguments are passed to the remote procedure and the caller waits for a response to

be returned from the remote procedure. Figure shows the flow of activity that takes place

during an RPC call between two networked systems. The client makes a procedure call that

sends a request to the server and waits. The thread is blocked from processing until either a

reply is received, or it times out. When the request arrives, the server calls a dispatch

routine that performs the requested service, and sends the reply to the client. After the

RPC call is completed, the client program continues. RPC specifically supports network

applications.

**PRE TEST:**

1) An RPC (remote procedure call) is initiated by the \_\_\_\_\_\_\_\_\_

a) server

b) **client**

c) client after the sever

d) a third party

2. A remote procedure call is \_\_\_\_\_\_\_

a) **inter-process communication**

b) a single process

c) a single thread

d) a single stream

3. A remote procedure is uniquely identified by \_\_\_\_\_\_\_\_\_

a) program number

b) version number

c) procedure number

d) **all of the mentioned**

4. An RPC application requires \_\_\_\_\_\_\_\_\_

a) specific protocol for client server communication

b) a client program

c) a server program

d) **all of the mentioned**

5. RPC is a \_\_\_\_\_\_\_\_\_

a) **synchronous operation**

b) asynchronous operation

c) time independent operation

d) channel specific operation

**PROCEDURE:**

Steps during RPC:

1. A client invokes a client stub procedure, passing parameters in the usual way.
2. The client stub Marshalls (pack) the parameters into a message.
3. The client stub passes the message to the transport layer, which sends it to the remote server machine.
4. On the server, the transport layer passes the message to a server stub, which Demarshalls (unpack) the parameters and calls the desired server routine using the regular procedure call mechanism.
5. When the server procedure completes, it returns to the server stub, which Marshalls the return values into a message. The server stub then hands the message to the transport layer.
6. The transport layer sends the result message back to the client transport layer, which hands the message back to the client stub.
7. The client stub Demarshalls the return parameters and execution returns to the caller.

**POST TEST:**

1. Remote Procedure Calls are used \_\_\_\_\_\_\_\_\_\_\_\_

a) for communication between two processes remotely different from each other on the same system

b) for communication between two processes on the same system

c) **for communication between two processes on separate systems**

d) none of the mentioned.

2. RPC provides a(an) \_\_\_\_\_ on the client side, a separate one for each remote procedure.

a) **stub**

b) identifier

c) name

d) process identifier

3. What is stub?

a) transmits the message to the server where the server side stub receives the message and invokes procedure on the server side

b) packs the parameters into a form transmittable over the network

c) locates the port on the server

d) **all of the mentioned**

4. What is the full form of RMI?

a) Remote Memory Installation

b) Remote Memory Invocation

c) Remote Method Installation

d) **Remote Method Invocation**

5. The remote method invocation \_\_\_\_\_\_\_\_\_\_

a) allows a process to invoke memory on a remote object

b) **allows a thread to invoke a method on a remote object**

c) allows a thread to invoke memory on a remote object

d) allows a process to invoke a method on a remote object

**REFERENCES:**

1. Distributed Systems: Principles and Paradigms- Tanenbaum, Steen.

2. The Grid Core Technologies”, Maozhen Li, Mark Baker, (Wiley)

3. High performance cluster computing volume 1 Rajkumarbuyya