# **Remote Procedure Call (RPC)**

* **Kathan Adhyaru (121022)**
* **Raj Thakkar (121041)**
* **Urja Patel (121058)**

# **Problem Statement**

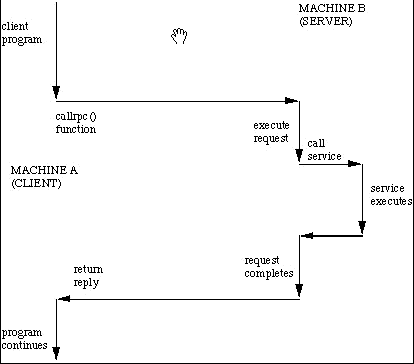
Write a program using RPC mechanism that maintains a log file of the modifications of the predefined files spread across the network.

**Introduction**

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.

That is, the programmer has to write essentially the same code whether the subroutine is local to the executing program, or remote.

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. Above figure shows the flow of activity that takes place during an RPC call between two networked systems.

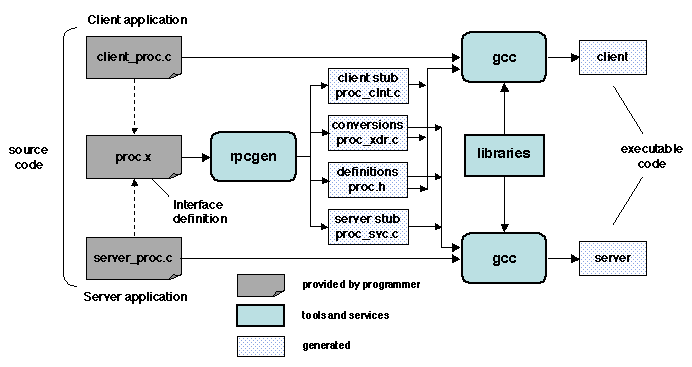


The client makes a procedure call that sends a request to the server and waits. The thread is blocked from processing until either a reply is received, or it times out. When the request arrives, the server calls a dispatch routine that performs the requested service, and sends the reply to the client. After the RPC call is completed, the client program continues.

RPC specifically supports network applications.

**How RPC mechanism works?**

A remote procedure is uniquely identified by the triple: (program number, version number, procedure number) The program number identifies a group of related remote procedures, each of which has a unique procedure number. The program number is a 32-bit number. Sun has reserved the range from 0 to 0x1fffffff. A user can use any number from 0x20000000 to 0x3fffffff.

A program may consist of one or more versions. Each version consists of a collection of procedures which are available to be called remotely. Version numbers enable multiple versions of an RPC protocol to be available simultaneously. Each version contains a number of procedures that can be called remotely. Each procedure has a procedure number.

**Getting Started**

Linux kernel by default does not provide any support for RPC. So we start by installing rpcgen compiler:

sudo apt-get install rpcbind

sudo apt-get install libc6-dev-bin

List of files we have to create:

1. prog.x - This is our IDL file that begins with some type definitions and defines the remote procedure to pass a string. We use rpcgen to compile this file
2. clnt.c - The calling program that calls the remote procedure
3. srvr.c - This is the actual program that acts as the called program ,implements the logic to perform the given task and actually establishes the RPC service

Output files generated by rpcgen:

* a header file prog.h which expresses the interface information and must be included by both the client, mc.c, and the server, mc\_svc\_proc.c
* a c file, prog\_clnt.c , that will be linked with the client. It contains calls to the ONC function **clnt\_call** which performs the calls to the remote procedures and interfaces with the XDR representation standard. This is called the **Client Stub**.
* a c file, pro\_xdr.c, that will be linked with both the client and the server, and contains code for placing/ removing information from the network messages in accordance to the XDR format. This is called the **Server Stub**.

**Compile Steps**

cc -o client clnt.c prog\_clnt.c

cc -o server srvr.c prog\_svc.c

./server

./client localhost

**References and Bibliography**

**1.<http://www.cs.cf.ac.uk/Dave/C/node33.html>**

**2.<http://sardes.inrialpes.fr/~krakowia/MW-Book/Chapters/Intro/intro.html>**

**3.**[**http://en.wikipedia.org/wiki/Remote\_procedure\_call**](http://en.wikipedia.org/wiki/Remote_procedure_call)

[**http://www.codeproject.com/Articles/4837/Introduction-to-RPC-Part**](http://www.codeproject.com/Articles/4837/Introduction-to-RPC-Part)