SYSTEM DESIGN DOCUMENT

Remote System Monitoring using RPC

Shikhar Vats

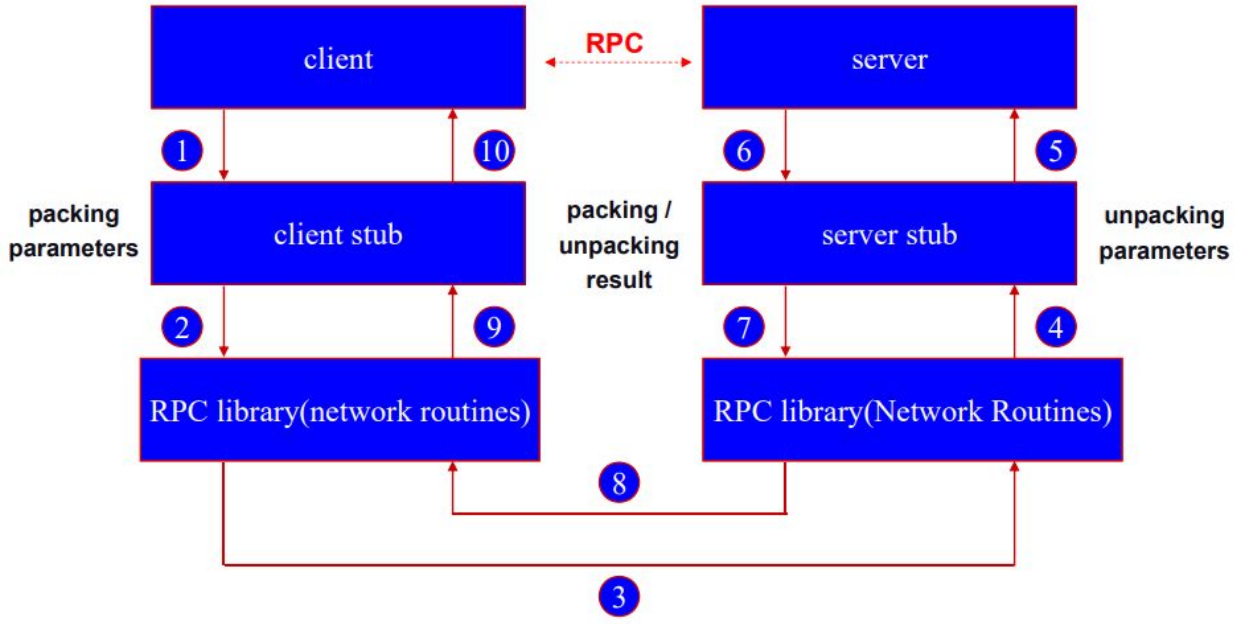
1 INTRODUCTION

1.1 Purpose and Scope

The purpose of this project is to write a program to implement the Remote Procedure Call. We will write a network management application that tracks the user logins, CP usage and other statistics by querying a RPC based network management system.

1.2 Project Executive Summary

1.2.1 System Overview



Packaging/unpackaging of arguments into network messages is called marshaling/unmarshaling.

2 SYSTEM ARCHITECTURE

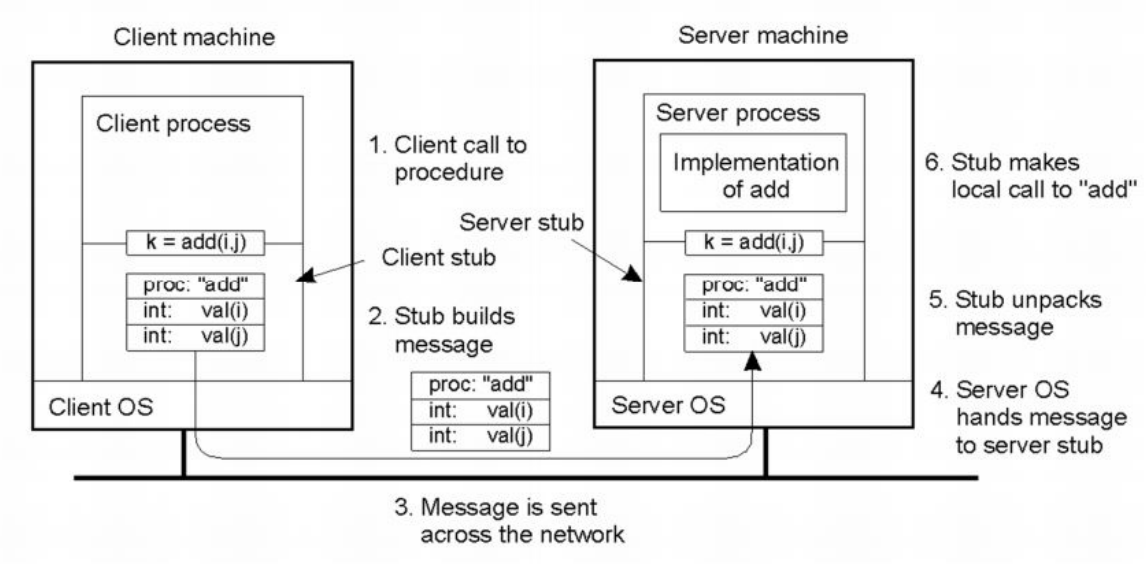
The system architecture has three main components:

1. The client side: client.c
2. The server side: server.c
3. The RPC interface between the client and the server: date.x

2.1 Internal Communications Architecture

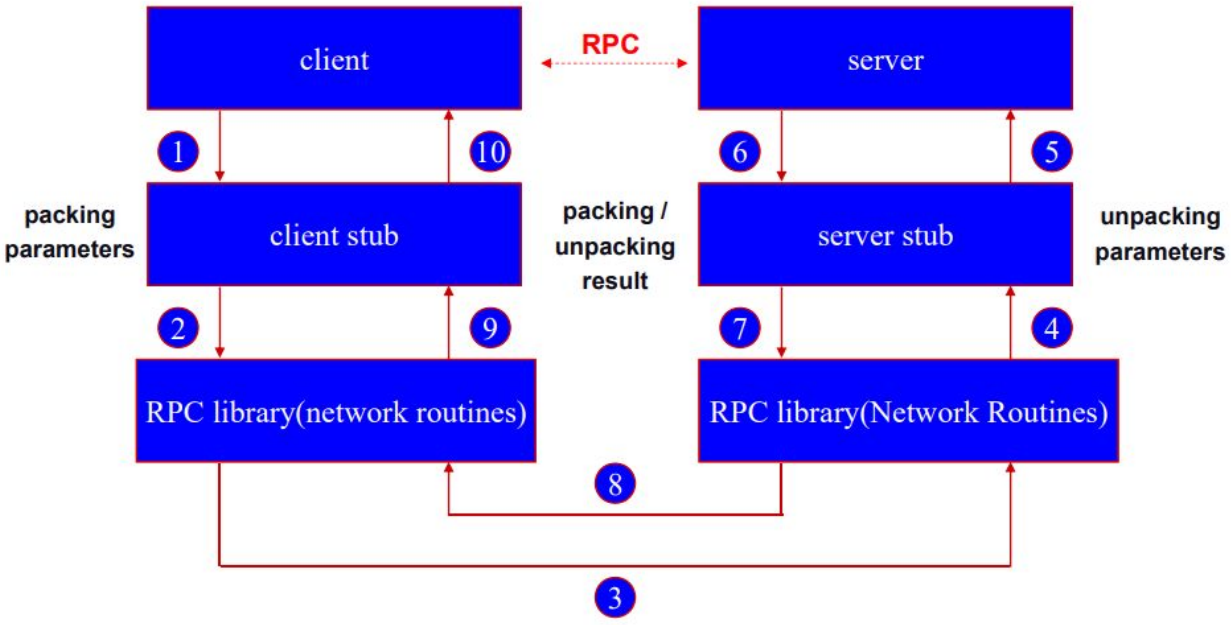
Following procedure is followed while establishing a communication:

1. The server registers the service with portmapper
2. The client contacts portmapper to determine the service port
3. The client contacts server on the specified service p



- **Steps involved in doing remote computation through RPC**

The following steps are involved in a remote procedure call:



Packaging/unpackaging of arguments into network messages is called marshaling/unmarshaling.

1. Client procedure calls client stub in normal way

2. Client stub builds message, calls local OS
3. Client's OS sends message to remote OS
4. Remote OS gives message to server stub
5. Server stub unpacks parameters, calls server
6. Server does work, returns result to the stub

The server performs the following operations on receiving each input :

1. Time: We access the local time on the server system.
2. Date: We access the local time on the server system.
3. Both: We access the local time on the server system.
4. Memory: to retrieve the memory utilization we access the sysinfo struct to get the total ram and the free ram. The difference between the two when divided by the total ram gives the memory usage.
5. Process: Makes a call to getloadavg and returns the first element of the returned array. This function gets the 1, 5 and 15 minute load averages of the system. The values are placed in loadavg. getloadavg will place at most nelem elements into the array but never more than three elements. The return value is the number of elements written to loadavg, or -1 on error.
6. CPU: To get the CPU usage we access the /proc/stat file and monitor the CPU usage values over two different time intervals. The CPU usage can be measured over an interval of time only. We simply sum of all difference between two consecutive reads to get the time elapsed between these reads. The result is equal to multiplying USER_HZ with the number of CPUs on the system and the seconds between the reads. The difference of column 4 (idle) gives us the time spent idle. The sum minus the idle time gives us the total CPU utilization. Divided by the sum we get the percentage of CPU utilization.
7. Exit

7. Server stub packs it in message, calls local OS
8. Server's OS sends message to client's OS
9. Client's OS gives message to client stub
10. Stub unpacks result, returns to client

3 HUMAN-MACHINE INTERFACE

3.1 Inputs

The program asks the user for an input after displaying a menu.
The input can be any number ranging from 1-7.

1. Time : to retrieve the time of the system
2. Date : to retrieve the date of the server system
3. Both : to retrieve both the system date and the time
4. Memory : to retrieve the memory usage
5. Process : to load the number of processes per minute
6. CPU : to retrieve the CPU usage

7. Exit

3.2 Outputs

Based on the input, the program returns the relevant output from the server side.