A Report on IDSS630E Course Project

RPC Implementation

Supervised by

DR. ANSHU S ANAND

DEPARTMENT OF INFORMATION TECHNOLOGY

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

ALLAHABAD (PRAYAGRAJ)

Presented By

ANKIT GANGWAR [IEC2017504]

SHIVANSH SRIVASTAVA [IEC2017054]

ADESH SHUKLA [IEC2017088]

ABDUL AHAD SIDDIQUI [IEC2017023]

MAY 10, 2020

Abstract

In this project,we implement a distributed registry service system,in which the clients can use the services provided by the multiple remote servers.The whole system consist of three module:Central Registry and Load Balancer,Server,Client.Central registry stores a list of services along with the server IPs that provide those services.Each new server that wants to provide a new service must first register its service with the registry by specifying name of the service, Arguments required for the service along with their types,return type of the service.Each client when trying to invoke a remote procedure on a distant server must first check with the registry for the server that provides the said service. After receiving the server IP and the service procedure signature from the registry corresponding to the required service, the client sends a port request to the specified server to get the port no. on the server on which the service is exposed and the entire service will be executed on this port. Load balancer which depending upon the number of services currently running on any server, splits the load on different servers providing same service.The server will constantly check if a large no. of requests (greater than a specified threshold) come from the same client in a particular window time, then the requests can be throttled to ensure the security and prevent DOS attacks,so in this way our system is much secure.

CONTENTS

1 | INTRODUCTION

A remote procedure call is an interprocess communication technique that is used for client-server based applications. It is also known as a subroutine call or a function call.

A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumes execution after the server is finished.

The following steps take place during an RPC:

1. A client invokes a **client stub procedure**, passing parameters in the usual way. The client stub resides within the client's own address space.

2. The client stub **marshalls(packs)** the parameters into a message. Marshaling includes converting the representation of the parameters into a standard format, and Copying each parameter into the 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 **(e.g., via a Normal procedure call return)**, 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.

RPC provides **ABSTRACTION** i.e message-passing nature of network communication is hidden from the user. RPC often omits many of the protocol layers to improve performance. Even a small performance improvement is important because a program may invoke RPCs often.RPC enables the usage of the applications in the distributed environment, not only in the local environment. With RPC code re-writing / re-developing effort is minimized. Process-oriented and thread oriented models supported by RPC.

2 | PROBLEM DEFINITION

Remote Procedure Call is an extension of a local procedure call in which the called procedure is part of a different program to the calling procedure, and thus, at run time, the two procedures are in two different process spaces. Perhaps, the main benefit of RPC is that from the programmer's viewpoint, the procedure call works the same regardless of whether it is a local or remote call. Thus, RPC provides location and access transparency and removes the need for the programmer to manually implement the communication aspects. The RPC mechanism can be used for client-server communication. The mechanism ensures basic properties of distributed systems including transparency, security, availability, scalability, and fault tolerance.

So we need to implement the RPC client-server model through which the above-mentioned properties can be achieved for efficient communication between client and server. In our model various services provided by the server can be accessed by the client via the RPC mechanism.

3 | SYSTEM MODEL

A. System Architecture

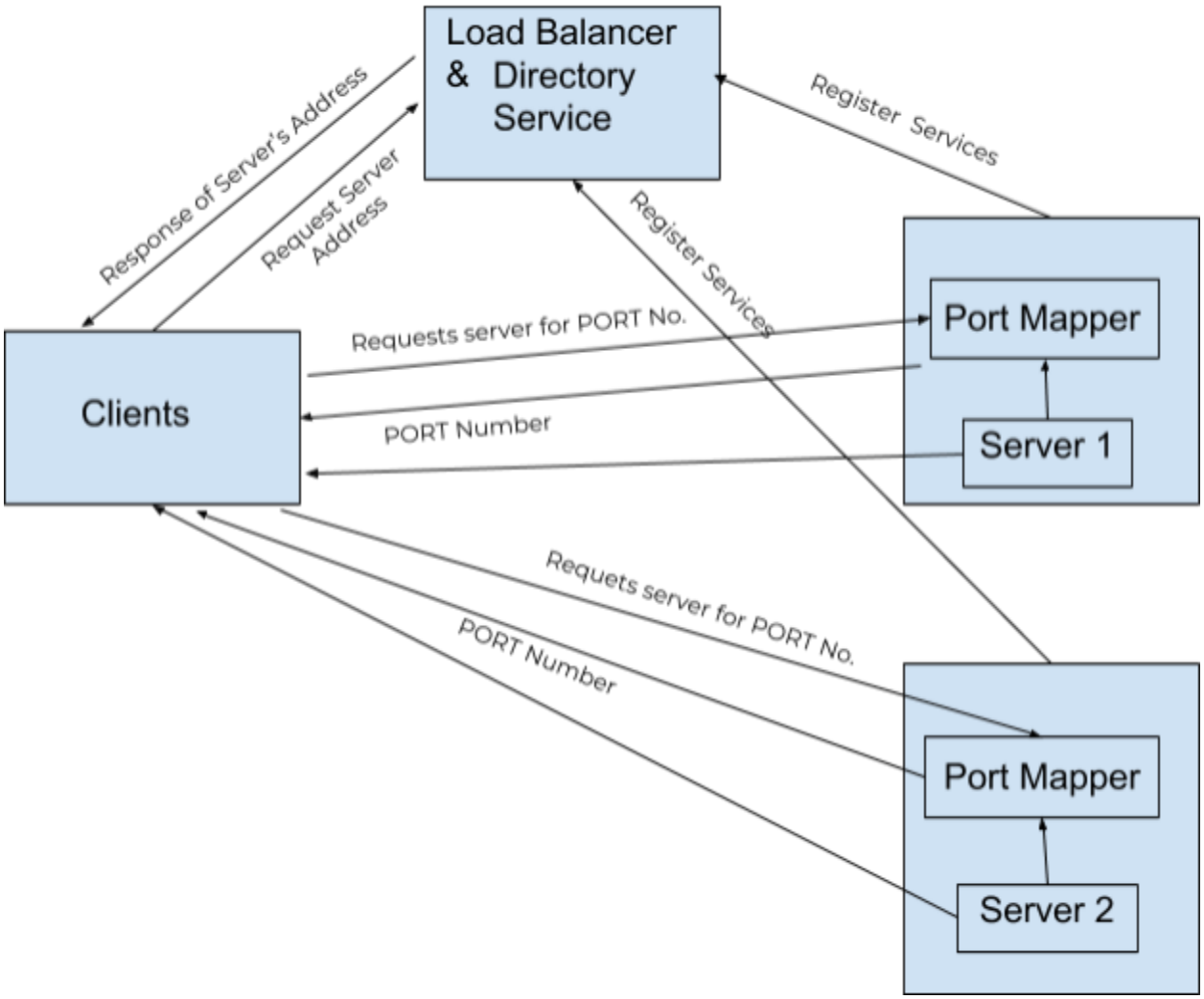
The architecture consists of server each providing some service and client that want to use those services. It also consists of a central registry that stores a list of services along with the server IPs that provide those services.

Each new server that wants to provide a new service must first register its service with the registry by specifying

- Name of the service
- Arguments required for the service along with their types
- The return type of the service

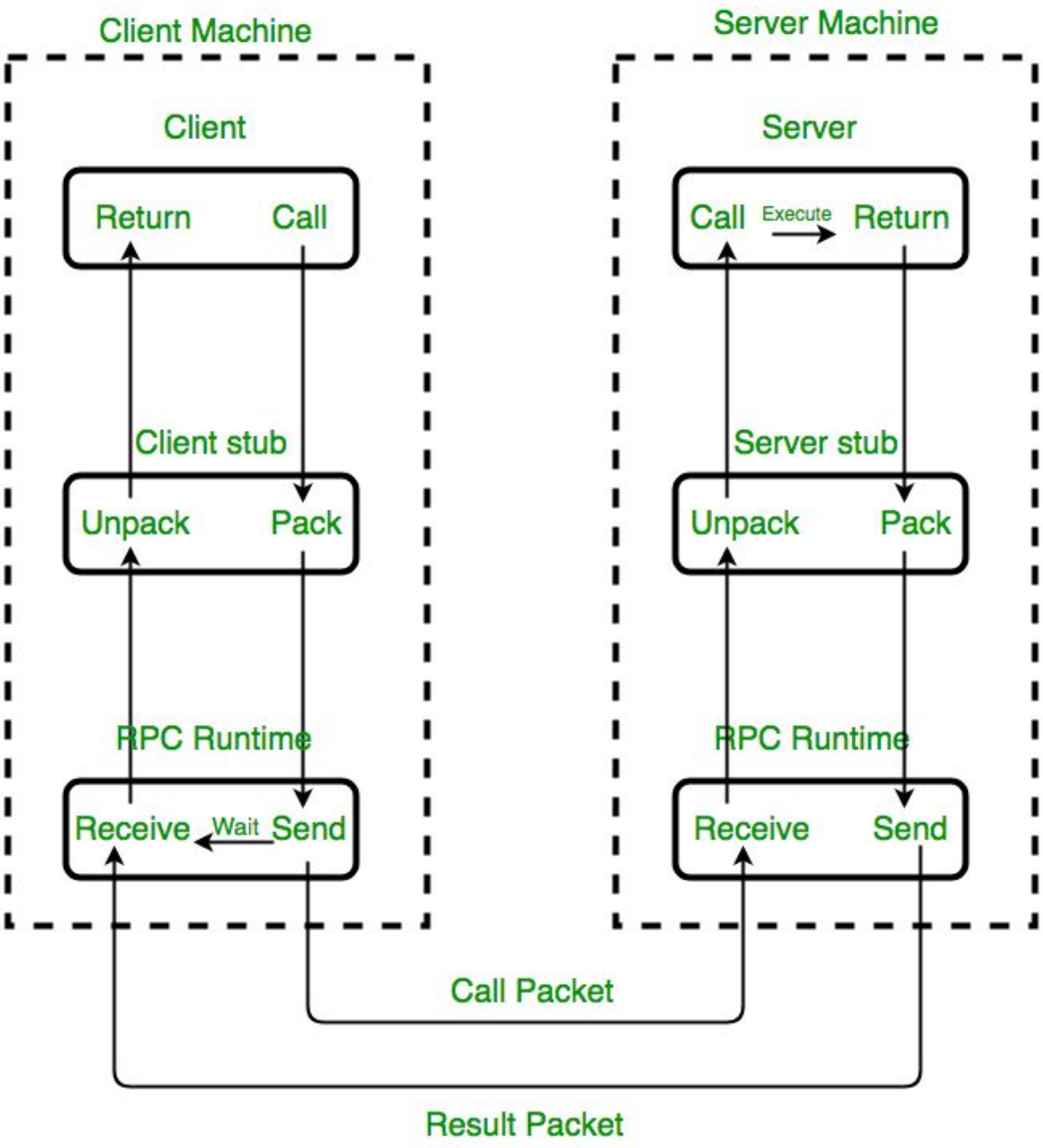
Each client when trying to invoke a remote procedure on a distant server must first check with the registry for the server that provides the said service. After receiving the **server IP** and the **service procedure signature** from the registry corresponding to the required service, the client sends a port request to the specified server to get the **port no.** on the server on which the service is exposed. On getting the port no., the client **marshalls** the parameters according to the remote procedure signature and sends the **request for the invocation** of the remote procedure on the specified server and the specified port. The client gets **blocked** until the server completes the request.

The directory service also contains a **load balancer** which depending upon the number of services currently running on any server, splits the load on different servers providing the same service.



[Fig.1] RPC Architecture

The procurement of server IPs, port number, service procedure signature, and marshaling of parameters is done by the client-side RPC library using the client stub. Also, blocking the client and unmarshalling of returned response from the server is also handled by the client-side RPC library and the client stub.



Implementation of RPC mechanism

The server-side RPC library handles the incoming remote invocation request from the client and determines which procedure needs to be called on the server. It also handles the unmarshalling of sent parameters using the server stub and calls the requested procedure with the sent parameters. After the procedure completes executing, the return value is marshalled again and sent to the client.

B. Technology Used

- The XDR (External Data Representation) will be in **JSON** format. All the procedure signatures will be represented in the above JSON format.
- The registry server will be implemented in **NodeJS** with **MongoDB(Atlas)** as the NoSQL database for storing
    - Service Name
    - Server IP
    - Service Arguments and their types
    - Service Return type
- The clients and the servers along with the exposed service procedures will be implemented in **NodeJS.**
- The client and the server-side RPC libraries will be implemented in **NodeJS** as well.
- Database of Server is also implemented using **MongoDB**

C. Functionality

- The client can call a remote procedure located on a different machine as if it were calling a procedure defined in its own machine.
- The server will constantly check if a large no. of requests (greater than a specified threshold) come from the same client in a particular window time, then the requests can be throttled to ensure the security and prevent DOS attacks.
- The server uses common **HTTP Error Codes** to communicate to the client of basic errors that occurred during the request/reply phase or during transmission of request/response.
- The server also sends the **complete error message** and the **stack trace** to the client in case of an error during the execution of the remote procedure. There will be an error type in the XDR for each procedure for marshalling and unmarshalling such errors.

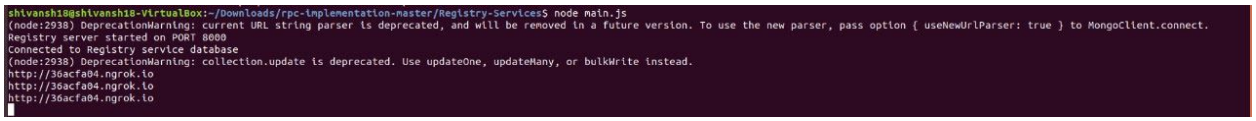
Different modules are:

- Registry and Load Balancer
- Server
- Client

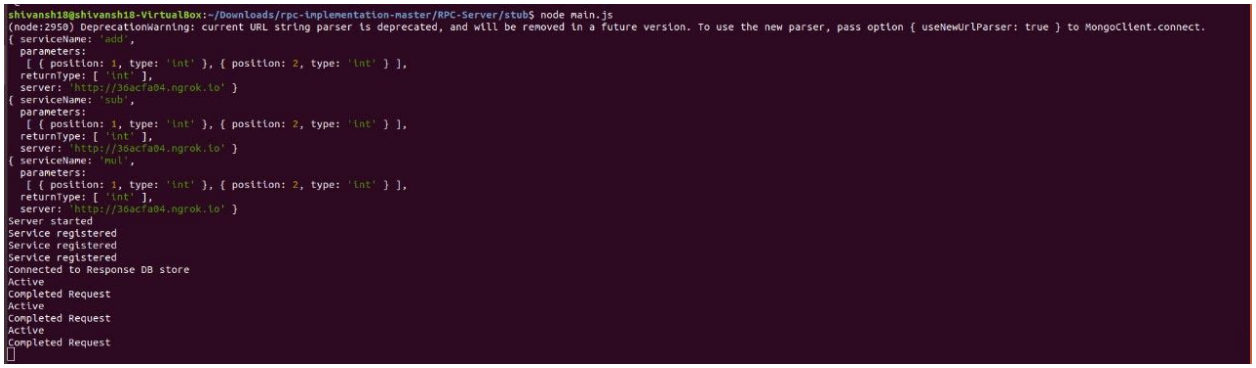
These modules will communicate over the TCP/IP channel.

4 | RESULT

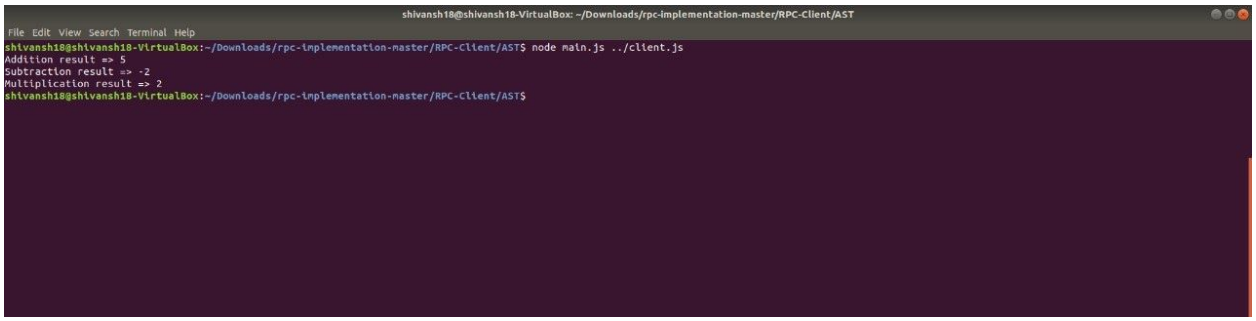
1.Registry Service



2.RPC-Server



3.RPC-Client



5 | CONCLUSION

RPC Provide programmers with familier mechanisms for building distributed systems. RPC Facility is not distributed panacea for all types of applications,it does provide a Valuable communication mechanism that is suitable for building a fairly large number of distributed applications. We were successful in implementing the RPC System (RPC Server and RPC Client).We used basic arithmetic services like addition multiplication and subtraction and client requested for these and we were successful in obtaining the result

6 | FUTURE WORK

- Many error detecting and correcting coding techniques can be implemented on this system so that clients on receiving the result of the service from the server, can detect and correct the error or again request for the same service.
- Similarly server can also detect the error if any from the client side and server can again ask for the corrected argument/parameters required for that service again.

7 | REFERENCES

[1] https://users.cs.cf.ac.uk/Dave.Marshall/C/node33.html

[2] https://www.sciencedirect.com/topics/computer-science/remote-procedure-call

[3] https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world

[4] https://www.geeksforgeeks.org/remote-procedure-call-rpc-in-operating-system/

[5] https://www.tutorialsteacher.com/nodejs/nodejs-tutorials

[6] https://docs.mongodb.com/manual/tutorial/