University of Dhaka
Computer Science and Engineering
4th Year 2nd Semester B.Sc.: 2024
CSE4269: Parallel and Distributed Systems
Published : 2025/09/22

Assignment Code: A2
Assignment Title: Design a Matrix Operator using Remote Procedure Call (RPC)
.

- ## Objectives
  The objective of this assignment is to design a Matrix Operator which runs on a remote machine. The client will need to accomplish the matrix operations: (i) matrix addition, (ii) matrix multiplication, (iii) matrix inverse and (iv) matrix transpose by calling those procedures which reside on the remote machine/server. For this purpose you have to implement the whole functionality using the RPC mechanism.

- ## What is RPC?
  Remote Procedure Call (RPC) is a protocol that one program can use to request a service from a program located in another computer in a network without having to understand network details. RPC makes coding easy in distributed systems' applications since the programmer writes essentially the same code whether the subroutine is local to the executing program, or remote. That is, the programs are coded as if the procedure call were a normal (local) procedure call, without explicitly coding the details for the remote interaction.

- ## How RPC works?
  Figure 1 presents the RPC working principle.

- ## Message passing in RPC
  - The client calls the client stub. The call is a local procedure call, with parameters pushed on to the stack in the normal way.
  - The client stub packs the parameters into a message and makes a system call to send the message. Packing the parameters is called marshalling.
  - The client's local operating system sends the message from the client machine to the server machine.
  - The local operating system on the server machine passes the incoming packets to the server stub.
  - The server stub unpacks the parameters from the message. Unpacking the parameters is called unmarshalling.
  - Finally, the server stub calls the server procedure. The reply traces the same steps in the reverse direction.

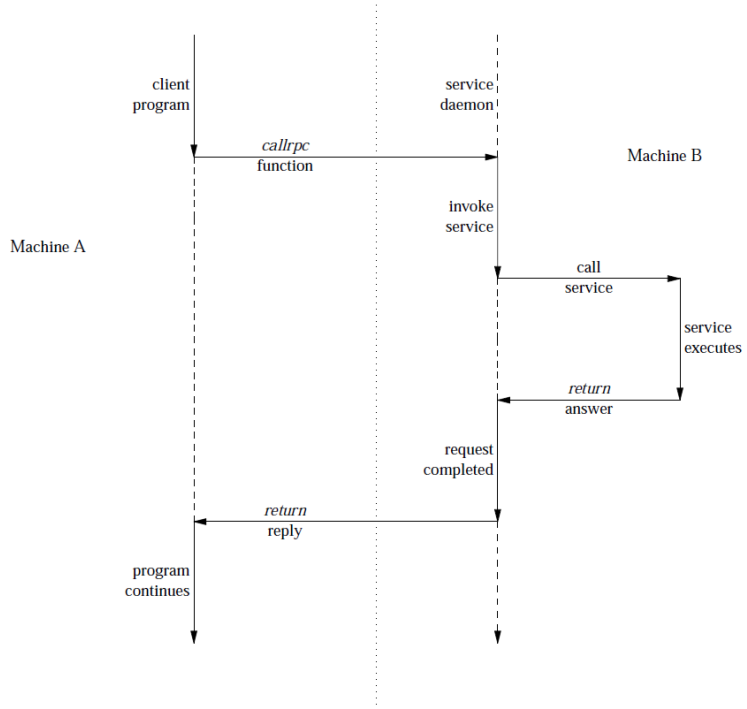

Figure 1: Network Communication with the Remote Procedure Call

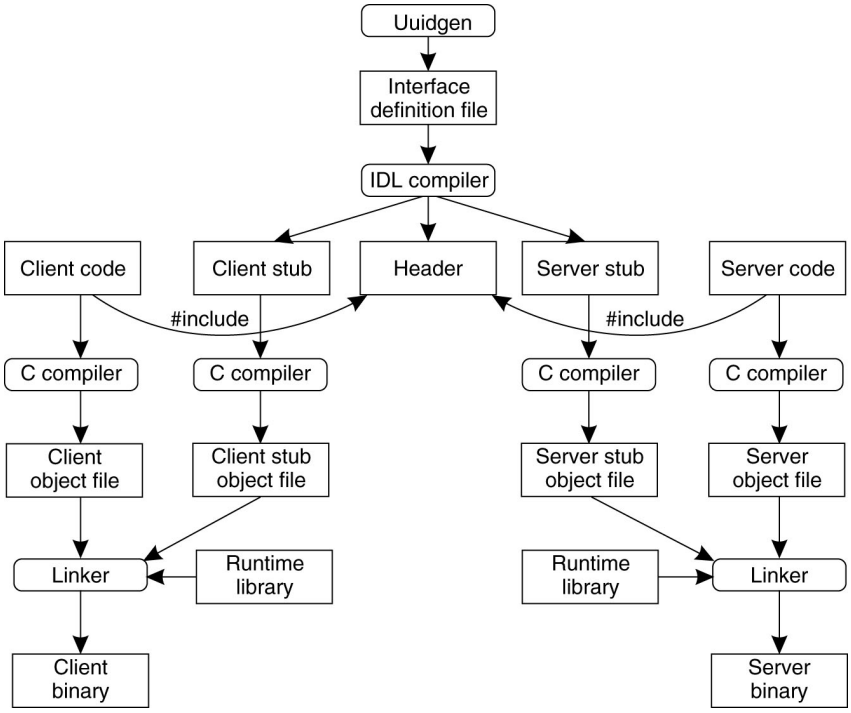

Figure 2: The steps in writing a client and a server in DCE RPC.

- **Writing a Client and a Server**
  - Three files output by the IDL compiler:
    - A header file (e.g., interface.h, in C terms).
    - The client stub.
    - The server stub.
    - Registration of a server makes it possible for a client to locate the server and bind to it.
    - Server location is done in two steps:
      1. Locate the server's machine.
      2. Locate the server on that machine.
- **RPC Application Development** Steps involved in developing a RPC application developing:
  - Specify the protocol for client server communication
  - Develop the client program
  - Develop the server program
- **Step1 : Specify the protocol for client server communication**
  IDL (Interface Description Language)

  Writing stub code by hand is painful, tedious, error prone, difficult to maintain and difficult to reverse engineer the wire protocol from the implemented code. A better approach is specify the data objects, messages and services and automatically generate the client and server code. For this purpose IDL is used. An interface description language (IDL) to let various platforms call the RPC. The IDL files can then be used to generate code to interface between the client and servers. So, in IDL file, we define the program structure like below. Save this IDL file with .x extension. Say the name of the file is matrixOp.x
- **Step 2: Compile the matrixOp.x file with rpcgen**
  rpcgen

  The rpcgen tool generates remote program interface modules. It compiles source code written in the RPC Language. RPC Language is similar in syntax and structure to C. rpcgen produces one or more C language source modules, which are then compiled by a C compiler. The default output of rpcgen is:
  - A header file of definitions common to the server and the client
  - A set of XDR routines that translate each data type defined in the header file
  - A stub program for the server
  - A stub program for the client
- When compiled using rpcgen compiler, this IDL file generates the followings:
  rpcgen -a -C matrixOp.x
  1. Client stub : matrixOp_cln.c
  2. Server stub : matrixOp_svc.c
  3. Sample client program : matrixOp_client.c
  4. Sample server program : matrixOp_server.c
  5. Header file : matrixOp.h
  6. XDR routines used by both the client and the server : matrixOp_xdr.c

7. Makefile

- rpcinfo
- yum install rpcbind

- **Step 3**: Edit the client and server programs.
- **Step 4**: Compile all the files.
- **Step 5**: Run the server and the client.
- **Language and Platform**: You have to use C to implement this assignment . Your assignment will be tested on Linux Platform. Make sure you develop your code under Linux.
- **Submission** :
  A single package containing all necessary files, codes and instructions for running the program. You can submit through the google classroom. The compressed filename must be of the format: [Roll No.]_[Assignment Code].zip.
- **Evaluation Procedure**

1. IDL : 30%
2. Client
   (a) The client should connect with the server (remote machine) and allow the user to communicate properly by calling the procedure. : 10%
   (b) Once a client completes all the matrix functionalists, the client should terminate properly. : 10%
   (c) There MUST be a minimum of two clients that can run in parallel. : 10%
3. Proper Comments: 5%
4. Test Case: 10%
5. Coding Style: 10%: Writing the code in a professional manner
6. Matrix Operations: 10%
7. Additional Features: 5%

Thank You