ASSIGNMENT # 2

Implementation of RPC Principles

Abstract

This assignment is based on principles of Remote Procedure Call(RPC). In this we are simulating client-server architecture using RPC, with the goal of calling remote procedure as if we are calling local procedure. We are performing distributed programming like conventional programming, to achieve transparency.

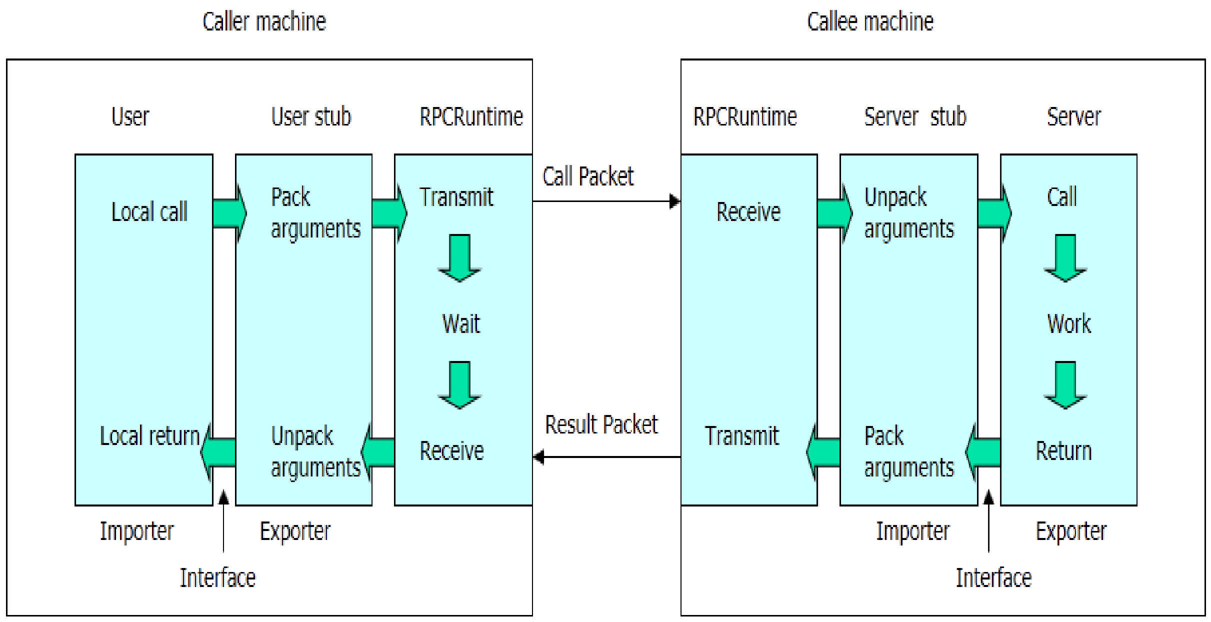
Pawar, Rohit Sidram
rspawar@iu.edu

RPC Principle

Fairly simple idea, in which we are extending normal procedure call in distributed environment. RPC is a kind of request–response protocol in which client sends a request message to a remote *server* to execute a specified procedure with supplied parameters and then remote server sends a response to the client. During this, client gets blocked and wait for server to respond back and client resumes it execution when response comes back.

Workflow of RPC events:

1. The client calls the client stub and push the parameters to stack as it is a local procedure call.
2. The client stub packs the parameters and RPC runtime at client side transmit these call packets over network. Packing the parameters is called marshalling.
3. When RPC runtime at server machine receives call packet it sends that packet to server stub.
4. The server stub unpacks the parameters from the packets. Unpacking the parameters is called unmarshalling.
5. Finally, the server stub calls the server procedure as it is normal procedure call. Then server execute the procedure and the result is returned to the client in reverse direction by following the same steps and then the client resumes it execution.



"["Reference taken from: - dpnm.postech.ac.kr/cs600/tanenbaum/**RPC**-Intro.ppt*"]"*

Components of RPC:

1) RPC Runtime
   - This component is responsible for transferring messages.

2) Client Stub
   - After getting request from client it packs the arguments and wait for response from server.
   - After getting the response from server, the client stub unpacks the data and send it to the client.

3) Server Stub
   - It receives the client request.
   - Upon receiving request, it unpacks the arguments and sends to server for execution.
   - After execution, server sends the data back to server stub, it packs the data and sends it back to the client.

Design Issues for RPC:

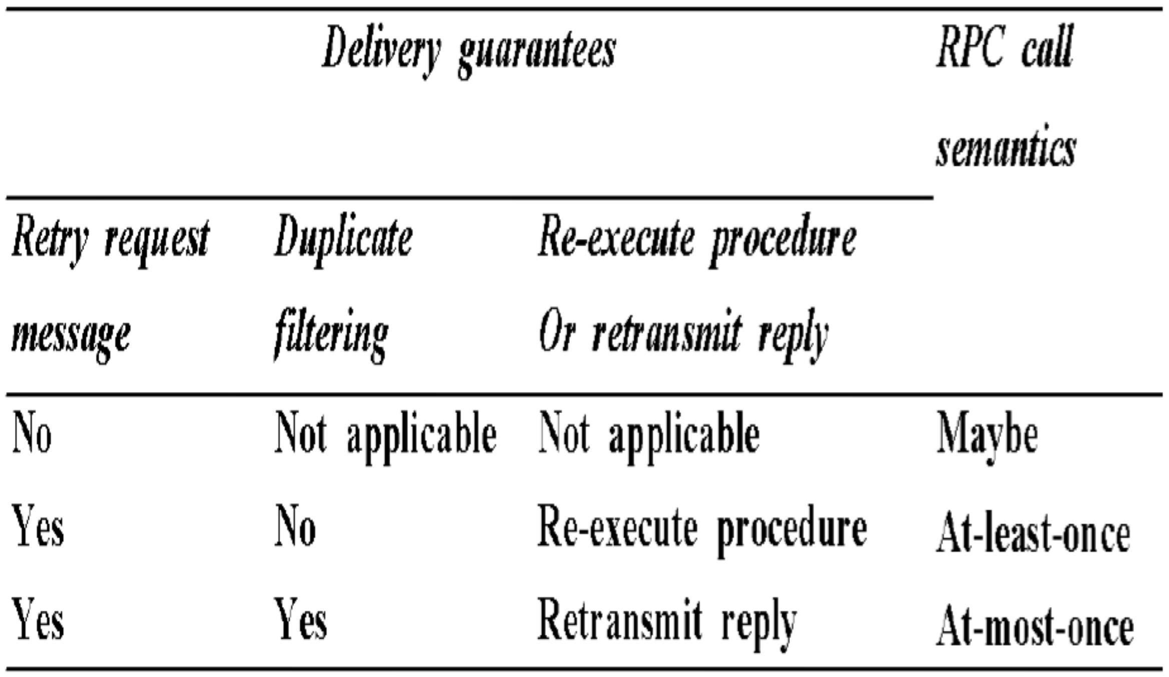
1) Parameter Passing to Procedure
   - Parameters are passed as pass by value only. Since, pass by reference doesn't make any sense in other's memory space.

2) Transport layer services for RPC
   - In RPC, communication can be done using TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).
   - If we opt for TCP service, then we will get connection-oriented reliable service. But message delivery will be slow, and server will be having additional overhead of maintaining client connection.
   - If we opt for UDP service, then packets will be delivered faster but no guarantee of message delivery.
   - So, it's up to programmer, to decide transport layer service based on applications requirement.

3) RPC interface compiler
   - Interface compiler which is helpful to generate client and server stub automatically.

4) Different methods used to implement request-reply protocol
   - Retry request message:
     Used in situation where no response is received from server. Timeouts can be used to retransmit request after certain interval or to detect server crash after several attempts.
   - Duplicate filtering:
     To avoid re-execution of same tasks
   - Retransmission of results:
     Decides whether to keep the copy of the results in case the sending of results gets failed.

5) RPC semantics



Delivery guarantees			RPC call semantics
Retry request message	Duplicate filtering	Re-execute procedure Or retransmit reply	
No	Not applicable	Not applicable	Maybe
Yes	No	Re-execute procedure	At-least-once
Yes	Yes	Retransmit reply	At-most-once

"["Reference taken from: - Distributed System Concepts and Design by Colouris"]"

6) RPC Exchange protocol

- Request protocol: Used in scenario where client don't need any acknowledgement from server. It just requests server to execute certain task and proceeds its execution.
- Request Reply (RR) protocol: Used in scenario where client explicitly ask server to send reply after execution till that time client is blocked.
- Request Reply Acknowledge Reply (RRA) Protocol: Used in scenario where server also wants to confirm whether client has received a response. Client sends ACK to server when it receives server's response.

Pros of RPC

- RPC supports both process-oriented as well as thread-oriented models.
- Distributed systems development is simple because of RPC semantics.
- Procedure which contains business logic can be kept separate from application.
- Fairly simple as it extends concepts of local procedure calls.

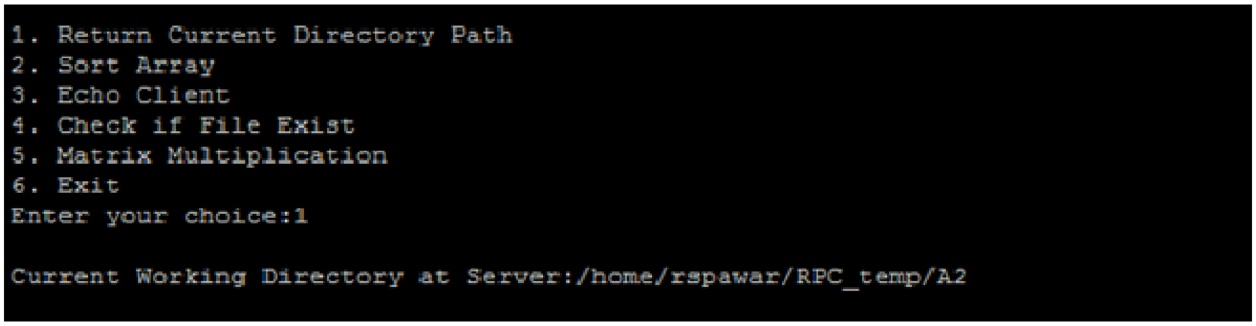
Cons of RPC

- Context switching cost is more.
- RPC may not be suitable for many distributed applications.
- RPC's performance is slow when we transfer large volume of data

Operations Performed

1) Path: Returns path of the current working directory to the client upon request.

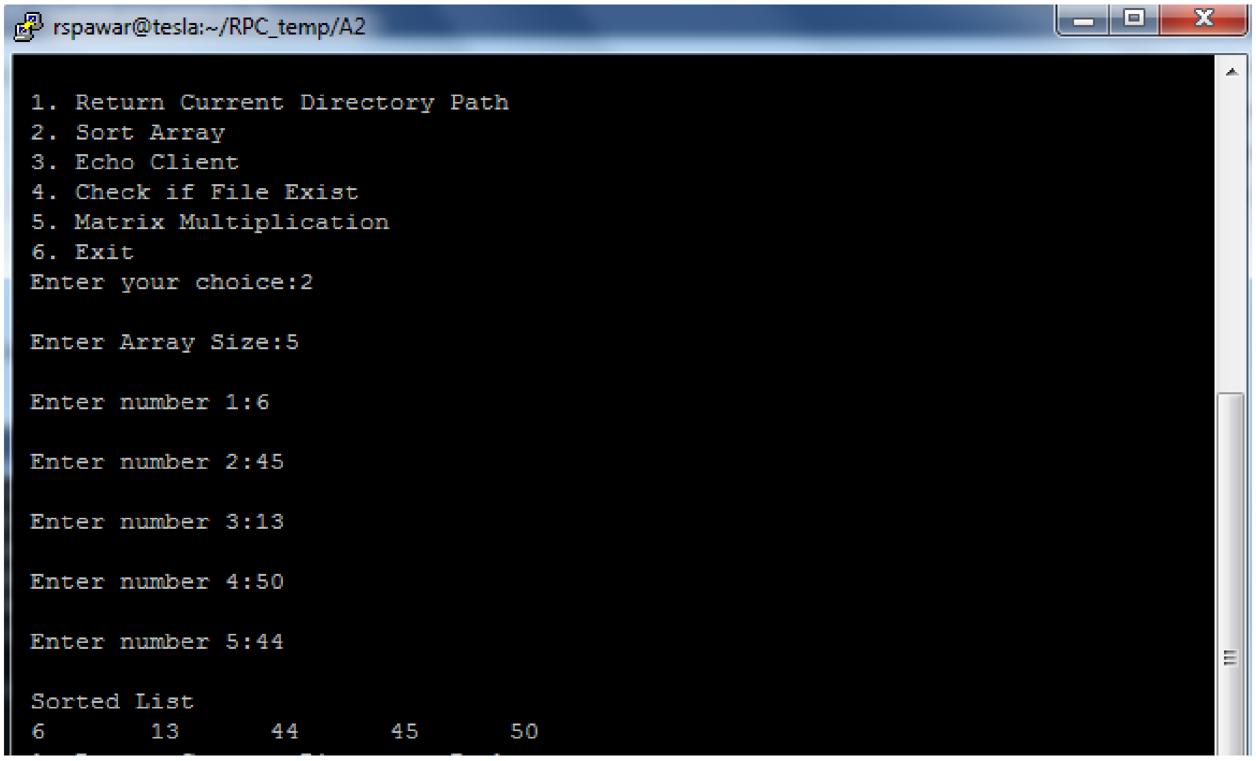
Execution Result



```
1. Return Current Directory Path
2. Sort Array
3. Echo Client
4. Check if File Exist
5. Matrix Multiplication
6. Exit
Enter your choice:1

Current Working Directory at Server:/home/rspawar/RPC_temp/A2
```

2) Sort: Client request to sort given input array and server sorts the array and returned sorted array to client.

Execution Result

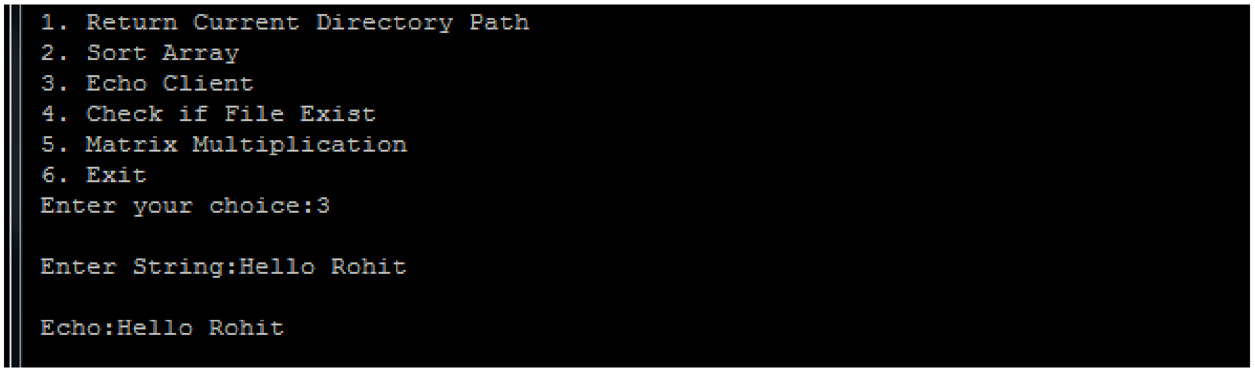


rspawar@tesla:~/RPC_temp/A2

```
1. Return Current Directory Path
2. Sort Array
3. Echo Client
4. Check if File Exist
5. Matrix Multiplication
6. Exit
Enter your choice:2

Enter Array Size:5

Enter number 1:6

Enter number 2:45

Enter number 3:13

Enter number 4:50

Enter number 5:44

Sorted List
6       13      44      45      50
```

3) Echo: Client sends some message to server and server echoes back the message to the client.

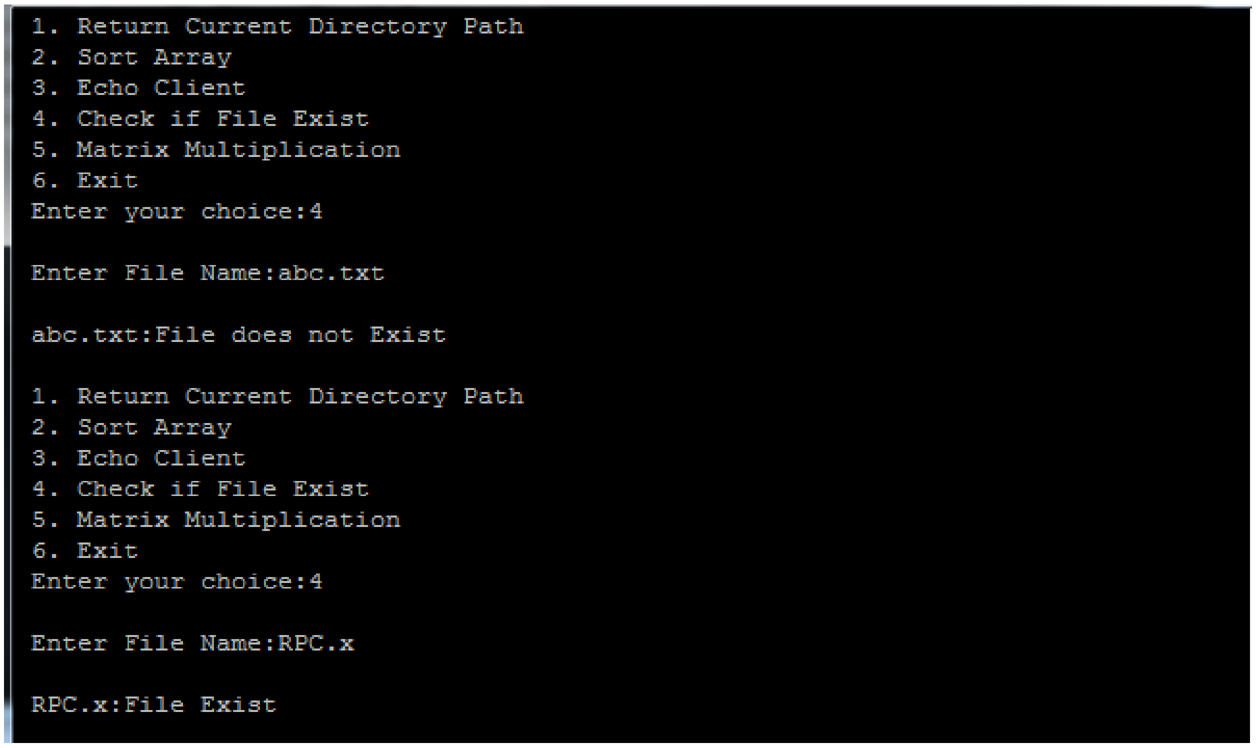
Execution Result



```
1. Return Current Directory Path
2. Sort Array
3. Echo Client
4. Check if File Exist
5. Matrix Multiplication
6. Exit
Enter your choice:3

Enter String:Hello Rohit

Echo:Hello Rohit
```

4) Check file exist: Client request server to check whether file exist or not at server. Server responds back the status of the file existence to the client. Server returns status as *File exist* if file is present and *File does not exist* if that file is not available at server side.

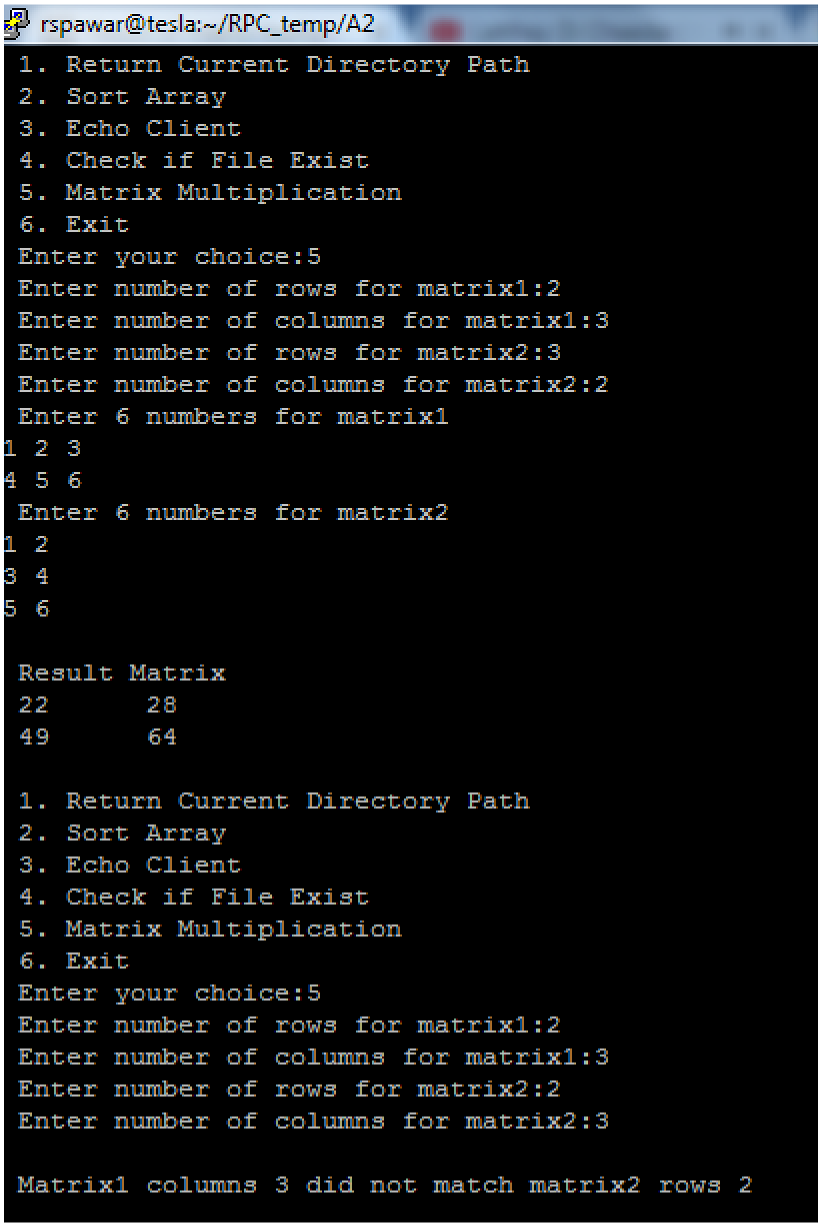
Execution Result



```
1. Return Current Directory Path
2. Sort Array
3. Echo Client
4. Check if File Exist
5. Matrix Multiplication
6. Exit
Enter your choice:4

Enter File Name:abc.txt

abc.txt:File does not Exist

1. Return Current Directory Path
2. Sort Array
3. Echo Client
4. Check if File Exist
5. Matrix Multiplication
6. Exit
Enter your choice:4

Enter File Name:RPC.x

RPC.x:File Exist
```

5) Matrix Multiplication: Clients request matrix multiplication by passing two input matrices and in response server sends multiplication of two input matrix. I have added validation at client side to check whether matrix multiplication is possible or not based on given input matrix range.

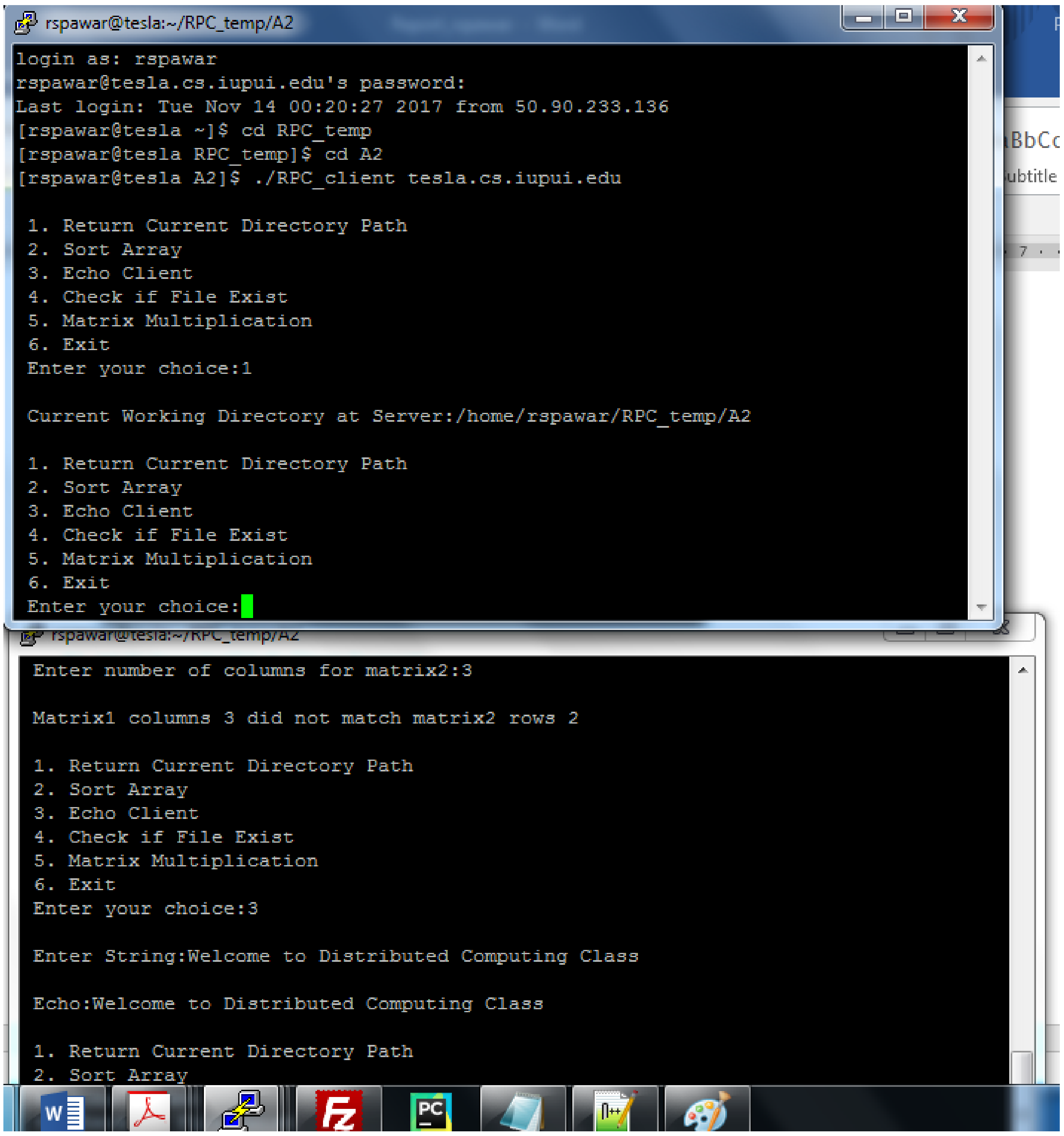
Execution Result



```
rspawar@tesla:~/RPC_temp/A2
 1. Return Current Directory Path
 2. Sort Array
 3. Echo Client
 4. Check if File Exist
 5. Matrix Multiplication
 6. Exit
 Enter your choice:5
 Enter number of rows for matrix1:2
 Enter number of columns for matrix1:3
 Enter number of rows for matrix2:3
 Enter number of columns for matrix2:2
 Enter 6 numbers for matrix1
1 2 3
4 5 6
 Enter 6 numbers for matrix2
1 2
3 4
5 6

 Result Matrix
 22      28
 49      64

 1. Return Current Directory Path
 2. Sort Array
 3. Echo Client
 4. Check if File Exist
 5. Matrix Multiplication
 6. Exit
 Enter your choice:5
 Enter number of rows for matrix1:2
 Enter number of columns for matrix1:3
 Enter number of rows for matrix2:2
 Enter number of columns for matrix2:3

 Matrix1 columns 3 did not match matrix2 rows 2
```

6) Multi-threaded/Multi-Client Support:
Two clients requesting server to perform two different operation and server can handle multiple clients request.

Execution Result



```
login as: rspawar
rspawar@tesla.cs.iupui.edu's password:
Last login: Tue Nov 14 00:20:27 2017 from 50.90.233.136
[rspawar@tesla ~]$ cd RPC_temp
[rspawar@tesla RPC_temp]$ cd A2
[rspawar@tesla A2]$ ./RPC_client tesla.cs.iupui.edu

 1. Return Current Directory Path
 2. Sort Array
 3. Echo Client
 4. Check if File Exist
 5. Matrix Multiplication
 6. Exit
 Enter your choice:1

 Current Working Directory at Server:/home/rspawar/RPC_temp/A2

 1. Return Current Directory Path
 2. Sort Array
 3. Echo Client
 4. Check if File Exist
 5. Matrix Multiplication
 6. Exit
 Enter your choice:
```

```
 Enter number of columns for matrix2:3

 Matrix1 columns 3 did not match matrix2 rows 2

 1. Return Current Directory Path
 2. Sort Array
 3. Echo Client
 4. Check if File Exist
 5. Matrix Multiplication
 6. Exit
 Enter your choice:3

 Enter String:Welcome to Distributed Computing Class

 Echo:Welcome to Distributed Computing Class

 1. Return Current Directory Path
 2. Sort Array
```

Conclusion

In this assignment, we have successfully implemented and used the principles of RPC (remote procedure call) for the implementation of client-server architecture. Compiled and executed the code to perform five different functionalities using RPC. Used *rpcgen* for the automatic creation of client stub and server stub together with XDR file. In this assignment, we got to know how RPC is the extension of local procedure call as well as how request -response protocol like RPC can be used for distributed applications.