ANNA UNIVERSITY

COLLEGE OF ENGINEERING GUINDY

DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY

DSC ASSIGNMENT- 1

TITLE : RPC and RMI Implementation

By,

Akshaya S K - 2023115095

RPC and RMI Implementation Report

Assignment: RPC and RMI Implementation

Name: Akshaya S K

Roll number: 2023115095

1. Introduction

Remote communication is a core concept in distributed systems, where components running on different machines communicate over a network. Two important technologies that enable such communication are **Remote Procedure Call (RPC)** and **Remote Method Invocation (RMI)**. RPC allows a client to invoke procedures on a remote server as if they were local functions, while RMI is Java's object-oriented approach that allows invocation of methods on remote Java objects.

In this experiment, RPC and RMI were implemented using a cloud-hosted server (AWS EC2) to demonstrate real-world distributed system communication.

2. System Architectures

RPC Architecture

- Client sends a remote procedure request to the server
- RPC framework handles serialization and network communication
- Server executes the procedure and returns the result

RMI Architecture

- RMI Registry maintains references to remote objects
- Client looks up the remote object using a name
- Method invocation happens transparently over the network

3. Tools and Technologies Used

- Programming Language: Python (RPC), Java (RMI)
- Frameworks: XML-RPC (Python), Java RMI
- Cloud Platform: AWS EC2
- Operating System: Linux (Server), Windows (Client)
- Networking: TCP/IP

4. RPC and RMI Server Implementation

Description

The server is responsible for hosting remote procedures (RPC) and remote objects (RMI). It listens for incoming client requests and processes them accordingly.

Server Responsibilities

- Accept incoming client connections
- Execute requested procedures or methods
- Send results back to the client
- Handle multiple requests reliably

Server Features

- Cloud-hosted for remote accessibility
- Well-defined interfaces for remote calls
- Console logging for monitoring requests
- Secure port configuration using EC2 security groups

5. RPC and RMI Client Implementation

Description

The client application connects to the remote server and invokes procedures or methods as if they were local. The client abstracts the networking details from the user.

Client Responsibilities

- Locate the remote service
- Send invocation requests
- Receive and display server responses
- Handle communication errors gracefully

Client Features

- Simple command-line execution
- Automatic connection to cloud-hosted server
- Clear display of server responses

6. Error Handling

- Connection timeout handling

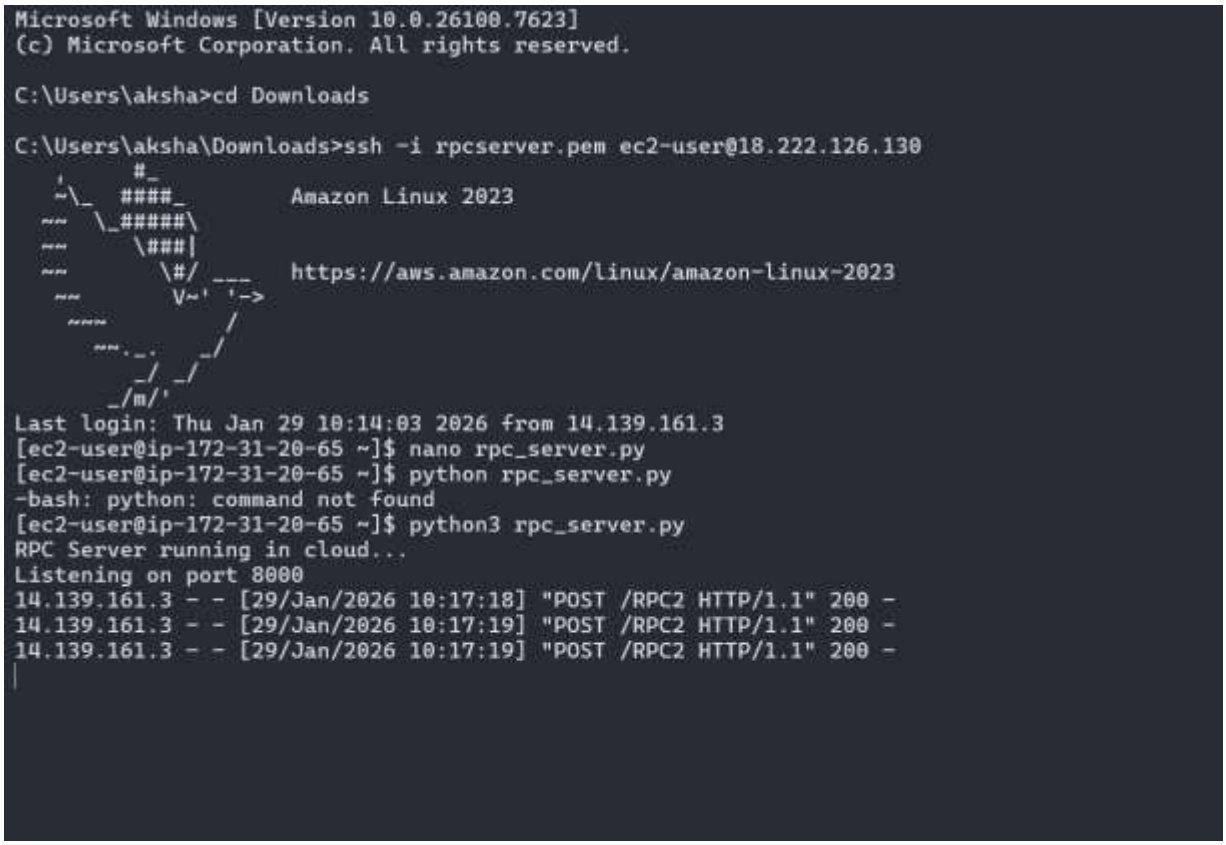
- Invalid host or port detection
- Server unavailability handling
- Exception handling using try-catch (RMI) and error messages (RPC)

7. Cloud Hosting Details

- Cloud Provider: Amazon Web Services (AWS)
- Service Used: EC2
- Ports Opened: 1099 (RMI Registry), Custom service ports
- Security Groups configured to allow client communication
- No cloud credentials exposed in code

8. Output Screenshots

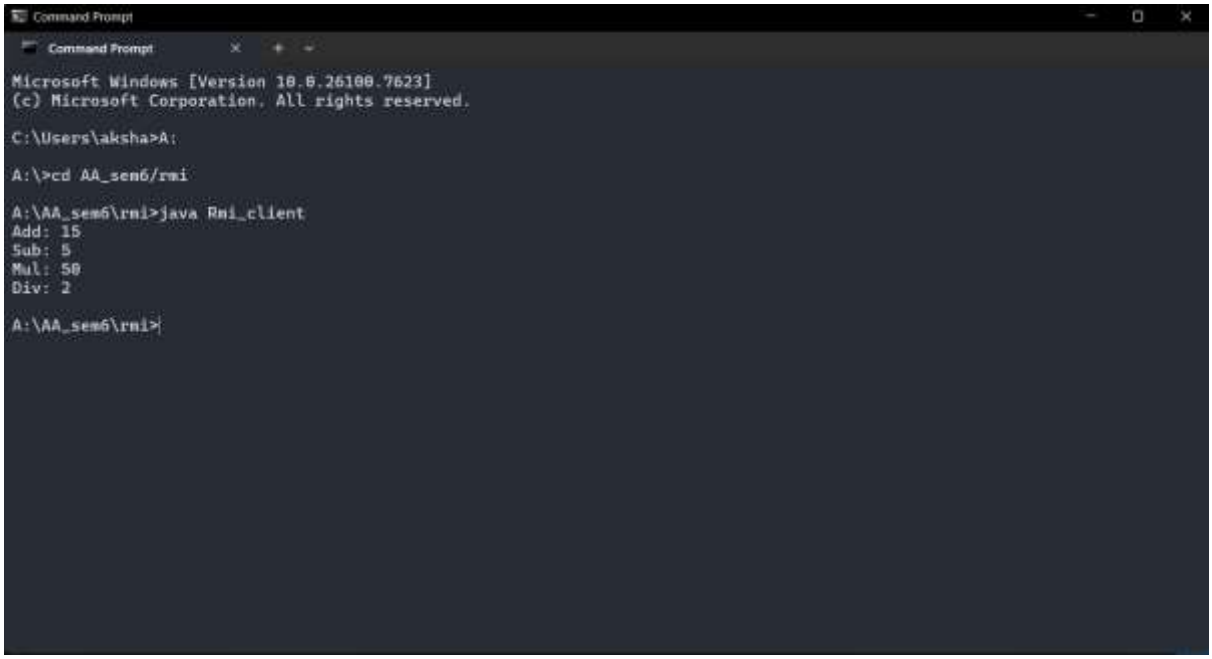
- RPC screenshots



```
Microsoft Windows [Version 10.0.26100.7623]
(c) Microsoft Corporation. All rights reserved.

C:\Users\aksha>A:

A:\>cd sem6/rpc

A:\sem6\rpc>python rpc_client.py
Addition Result: 15
Subtraction Result: 12
Multiplication Result: 42

A:\sem6\rpc>
```

- RMI screenshots

```
C:\Users\aksha\Downloads>ssh -i rpcserver.pem ec2-user@3.145.210.39
       #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Thu Jan 29 15:44:11 2026 from 14.139.161.3
[ec2-user@ip-172-31-20-65 ~]$ ls
rmi  rpc_server.py
[ec2-user@ip-172-31-20-65 ~]$ cd rmi
[ec2-user@ip-172-31-20-65 rmi]$ ls
MyService.class  MyService.java  MyServiceImpl.class  MyServiceImpl.java  Server.class  Server.java
[ec2-user@ip-172-31-20-65 rmi]$ javac Server.java
[ec2-user@ip-172-31-20-65 rmi]$ java Server
RMI Server running on EC2...
```



- AWS EC2 instance creation