Distributed Network Programming

Prativa Nyaupane

Review

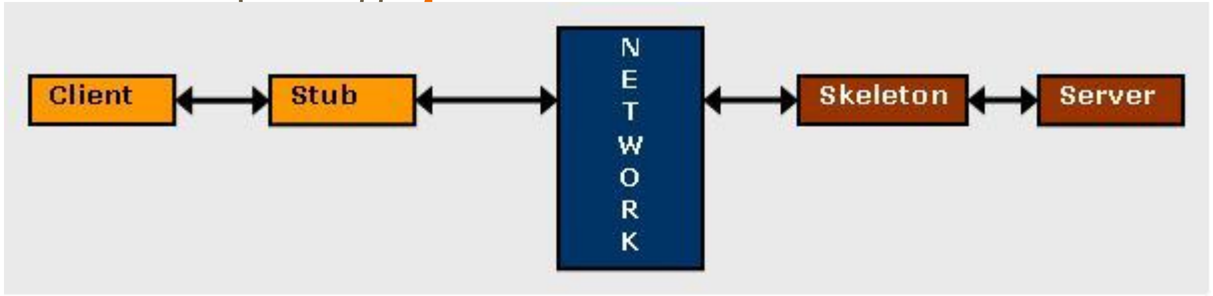
- Email Handling using Java Mail API
- Architecture of RMI

Outline

- TCP
- UDP
- IP Address
- Ports
- Socket Programming using TCP and UDP
- Working with URLs and URL Connection Class
- Email Handling using Java Mail API
- Architecture of RMI
- Creating and Executing RMI applications
- Architecture of CORBA
- RMI vs CORBA
- IDL and Simple CORBA Program

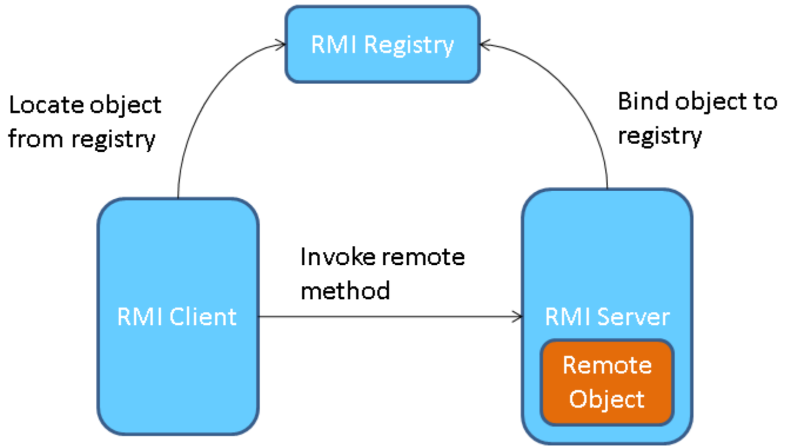
Remote Method Invocation

- **Remote Method Invocation(RMI)** is an API that allows a **Java object running on one virtual machine** to invoke methods on **an object running in another JVM.**
- This API provides a mechanism to build distributed applications in java.
- The RMI provides remote communication between the applications using two objects stub and skeleton.
- It is provided in the package **java.rmi.***



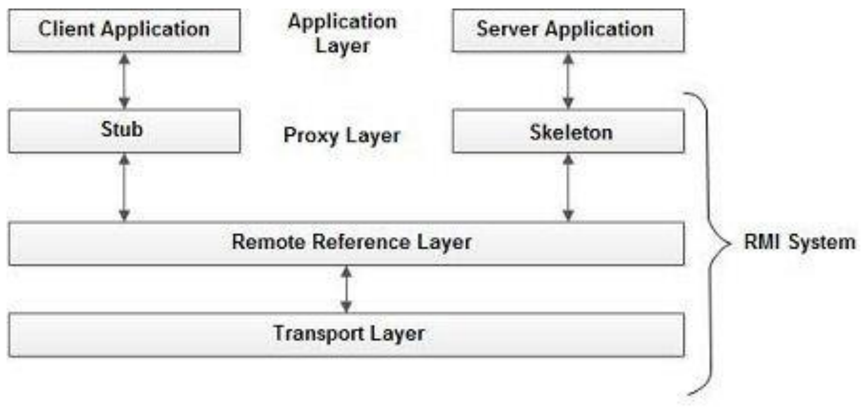
Architecture of RMI

- In an RMI application, we write two programs, a server program (resides on the server) and a client program (resides on the client).
  - Inside the server program, a remote object is created and reference of that object is made available for the client (using the registry).
  - The client program requests the remote objects on the server and tries to invoke its methods.



Architecture of RMI

- Client/Server Application: The java programs that will communicate and pass information back and forth.
- Stub: A stub is a representation (proxy) of the remote object at client. It resides in the client system; it acts as a gateway for the client program.
- Skeleton − This is the object which resides on the server side. stub communicates with this skeleton to pass request to the remote object



Architecture of RMI

Architecture of RMI (contd...)

- RRL(Remote Reference Layer) − It is the layer which manages the references made by the client to the remote object
- Transport Layer − This layer connects the client and the server. It manages the existing connection and also sets up new connections.

Working of an RMI Application

- When the client makes a call to the remote object, it is received by the stub which eventually passes this request to the RRL(Remote Reference Layer).
- When the client-side RRL receives the request, it invokes a method called invoke() of the object remoteRef. It passes the request to the RRL on the server side.
- The RRL on the server side passes the request to the Skeleton (proxy on the server) which finally invokes the required object on the server.
- The result is passed all the way back to the client

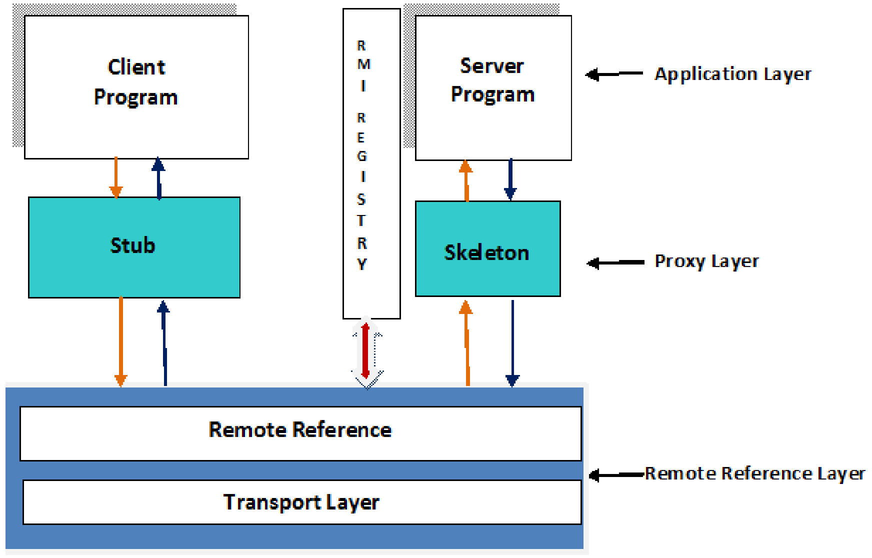


Fig: RMI Architecture

RMI Registry

- RMI registry is a namespace on which all server objects are placed
- Each time the server creates an object, it registers this object with the RMIregistry (using bind() or reBind() methods).
- These are registered using a unique name known as bind name.
- The client fetches the object from the registry using its bind name (using lookup() method).

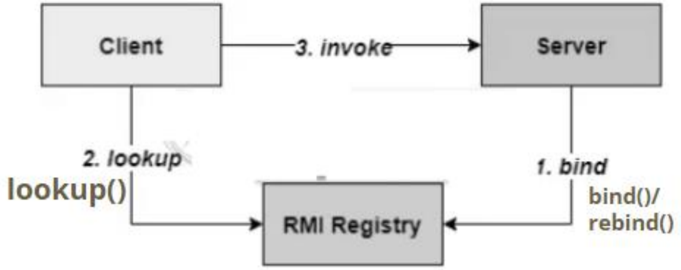
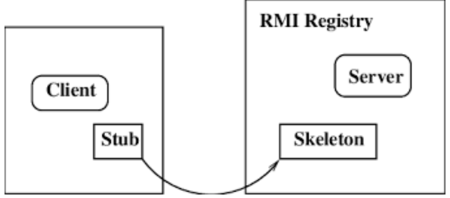


Fig. 1. How Java RMI works

Stub vs Skeleton



Stub	Skeleton
The stub is an object, acts as a gateway for the client side. It resides at the client side and represents the remote object.	The skeleton is an object, acts as a gateway for the server side object.
All the outgoing requests are routed through it.	All the incoming requests are routed through it.
When the caller invokes method on the stub object, it does the following tasks: ○ It initiates a connection with remote Virtual Machine (JVM) ○ It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM), ○ It waits for the result ○ It reads (unmarshals) the return value or exception, and ○ It finally, returns the value to the caller.	When the skeleton receives the incoming request, it does the following tasks: ○ It reads the parameter for the remote method ○ It invokes the method on the actual remote object, and ○ It writes and transmits (marshals) the result to the caller.

Goals of RMI

- To minimize the complexity of the application.
- To preserve type safety.
- Distributed garbage collection.
- Minimize the difference between working with local and remote objects.

Creating and Executing RMI applications

Create an RMI application where a client can remotely invoke a method that send the sum of any two given integers.

This application has four main components:
1. SumInterface - remote interface provided by the server.
2. AddServerImpl.java - implements the remote interface
3. AddServer.java containts the main program from the servermachine
4. AddClient.java implements the client side of the distribute system

Source Code: https://drive.google.com/file/d/1h2LzKvBBqan7-j2rCl5Z9RCE_6k_AbrA/view?usp=drive_link