Java RMI

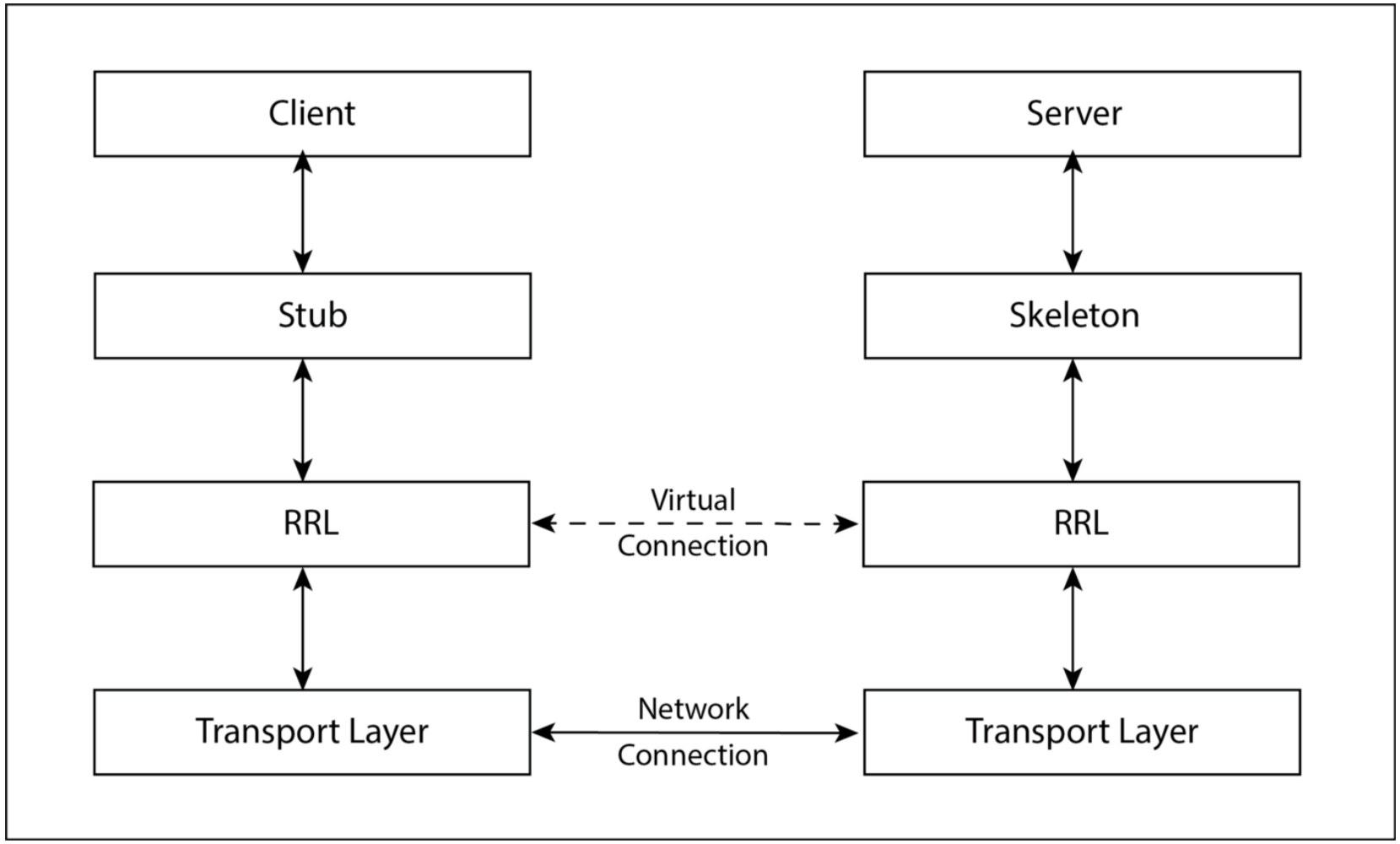
Introduction

- RMI stands for **Remote Method Invocation**.
- It is a mechanism that allows an object residing in one system (JVM) to access or invoke an object running on another JVM.
- RMI is used to build Distributed applications.
- It provides remote communication between Java programs. Java RMI is provided in the package *java.rmi*

Architecture of an RMI Application

- Java RMI consists with two main programs known as a server program (resides on the server) and client program (resides on the client)
- Inside the server, a remote object is created, and reference of the 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 method.

Architecture of an RMI Application



Architecture of an RMI Application

- Transport Layer – This layer connects the client and the server. It manages the existing connection and also sets up new connections.
- Stub – A stub is a representation (proxy) of the remote object at the 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.
- RRL/ Remote Reference Layer – It is the layer which manages the references made by the client to the remote object.

Implementation of an RMI Application

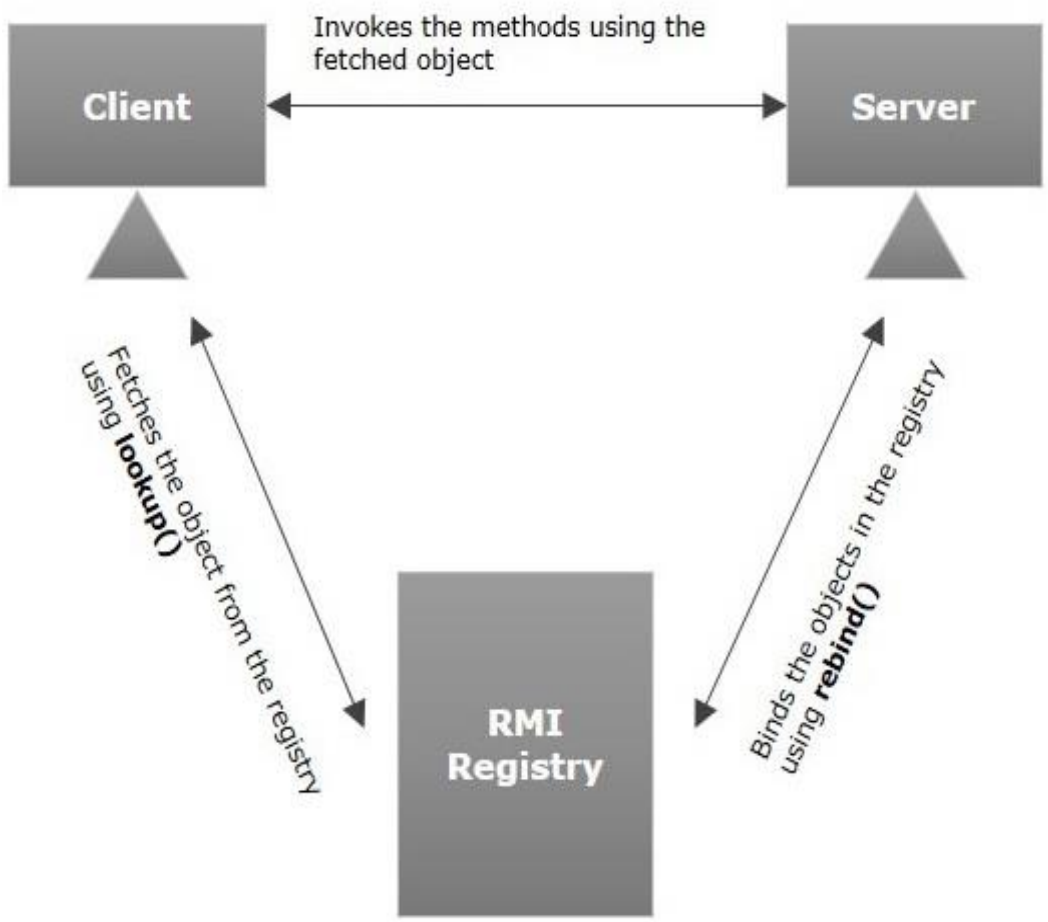
- When the client makes a call to the remote object, it is received by the stub which eventually passes this request to the RRL
- When the client side RRL receives the request, it invokes a method called **invoke( )** of the object **remoteRef**. It passes the request to the RRL on the server side.
- 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 client.

Marshalling and Unmarshalling

- Whenever a client invokes a method that accepts parameters on a remote object, the parameters are bundled into a message before being sent over the network including header information.
- This process is known as marshalling.
- At the server side, the packed parameters are unbundled and then the required method is invoked. This is unmarshalling.

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 RMI registry (using **bind( )** or **reBind( )** methods)
- To invoke a remote object, the client needs a reference of the object.
- At the time, the client fetches the object from the RMI registry using its bind name (using **lookup( )** method)



Implementing JAVA RMI Application

- Define the remote interface
- Develop the implementation class (remote object)
- Develop the server application
- Develop the client application
- Compile and execute the application