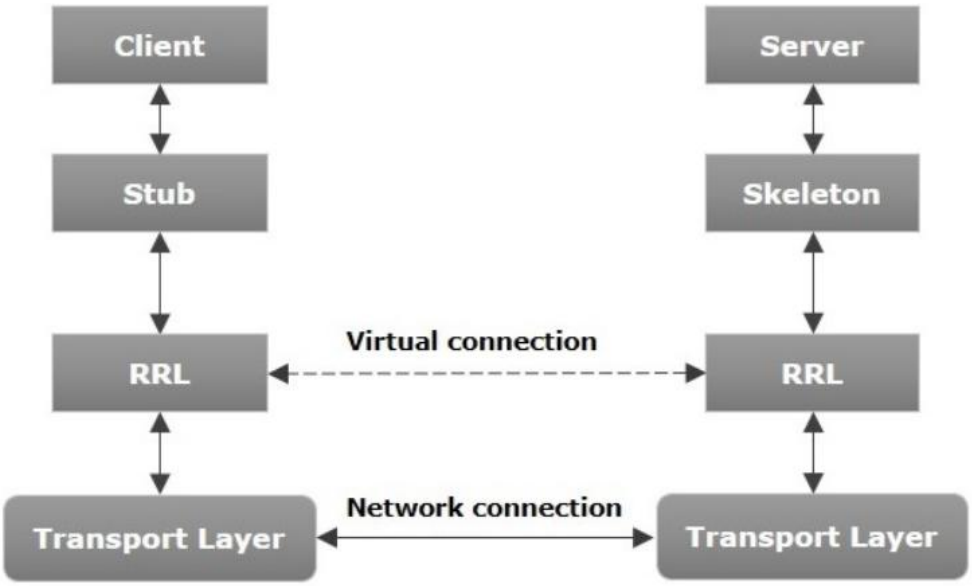
Module 4

Content Beyond Syllabus: Java RMI

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

Architecture of an RMI Application

- 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.



The components of this architecture:

1. **Transport Layer** − This layer connects the client and the server. It manages the existing connection and also sets up new connections.
2. **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.

3. **Skeleton** − This is the object which resides on the server side. **stub** communicates with this skeleton to pass request to the remote object.
4. **RRL(Remote Reference Layer)** − It is the layer which manages the references made by the client to the remote object.

Working of an RMI Application

The following points summarize how an RMI application works −

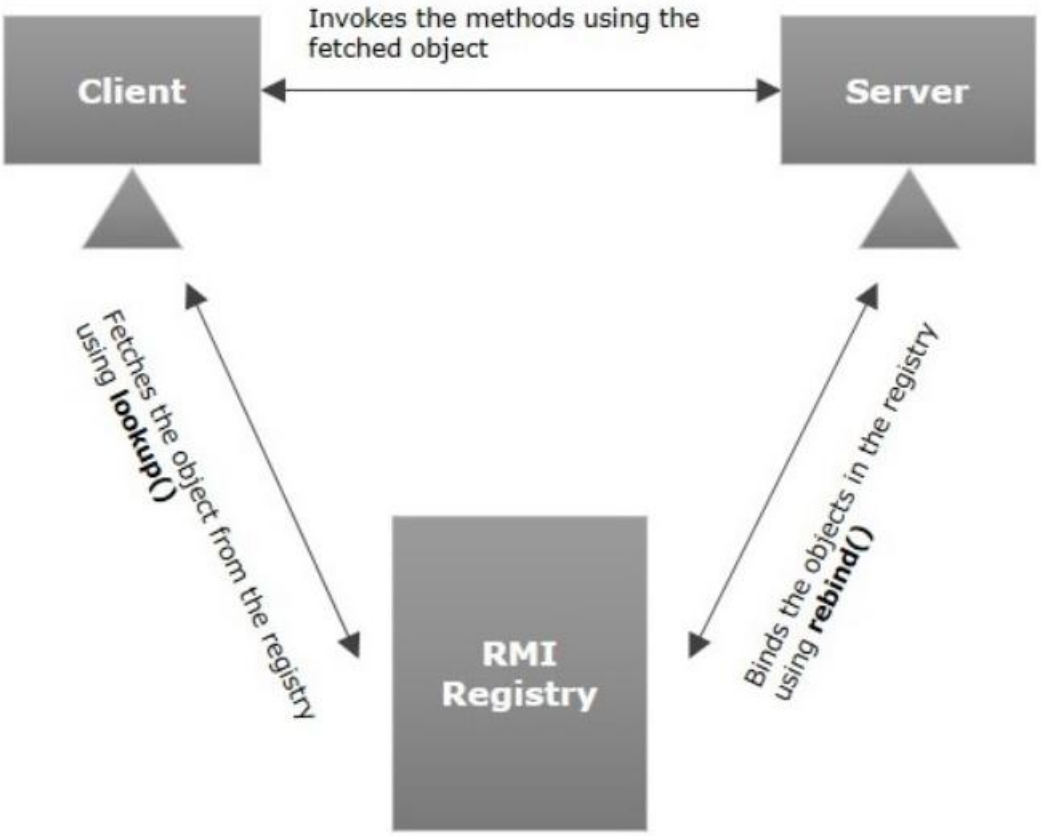
- When the client makes a call to the remote object, it is received by the stub which eventually passes this request to the RRL.
- When the client-side RRL receives the request, it invokes a method called **invoke()** of the object **remoteRef**. It passes the request to the RRL on the server side.
- 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 the 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. These parameters may be of primitive type or objects. In case of primitive type, the parameters are put together and a header is attached to it. In case the parameters are objects, then they are serialized. This process is known as **marshalling**.
- At the server side, the packed parameters are unbundled and then the required method is invoked. This process is known as **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 RMIregistry (using **bind()** or **reBind()** methods). These are registered using a unique name known as **bind name**.
- To invoke a remote object, the client needs a reference of that object. At that time, the client fetches the object from the registry using its bind name (using **lookup()** method).


Client
Invokes the methods using the fetched object
Server
Fetches the object from the registry using lookup()
RMI Registry
Binds the objects in the registry using rebind()