# Research and Analysis (Remote Method Invocation: Mechanisms)

*Navjot Singh, Email: Virksaabnavjot@gmail.com, National College of Ireland, Mayor Street, Dublin 1, Ireland*

## Introduction

A Distributed system *(DS)* is a model/collection of independent computers linked together through a network which produce an integrated computing facility using software programs *(middleware)*. Some widely used DS are Word Wide Web *(WWW)*, Email, Cloud services like Google drive and Dropbox, teleconferencing services like Skype. In Distributed communication, services provided by a server can be accessed by multiple clients. The goals in mind while implementing distributed systems are – scalability, reliability, openness, transparency and performance. Inter-process communication is at the core of DS, and there are different ways to achieve that for example - Message Oriented or Stream Oriented communication but for the purpose of this research report will will focus primarily on Remote Method Invocation *(RMI)* which is a Java implementation of Remote Procedure Calls *(RPC)* that allows server and client software to communicate with each other.

Lets, look at high level details about RMI: Remote Method Invocation is an object oriented Application Programming Interface which allows the creation of distributed applications using Java and this distributed environment supports/allows different computers running Java Virtual Machine *(JVM)* to communicate with each other using stub *(on client side)* and skeleton *(on server side)*. Stub and Skeleton are responsible for marshalling and un-marshalling data, the RMI allows an object on client to invoke methods/services on an object *(called servant)* on the server running JVM. RMI does this in a way that the client application thinks its invoking a local Java object's methods.

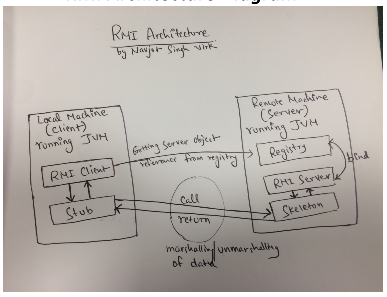
Remote Java Object communication in RMI occurs using the Java Remote Method Protocol *(JRMP).*
Or communication can happen over Internet Inter-Orb Protocol *(IIOP)* as well keeping in mind IIOP stubs are properly connected with ORB (Object Request Broker) before starting operations on IIOP stub whereas this is not required with JRMP. IIOP is an object oriented communication protocol for Common Request Broker Architecture *(CORBA)* it defines how bits are exchanged between CORBA's client and servers. Java implementation of CORBA/IIOP is known as Java IDL *(Interface Definition Language)* and supports mapping for Java. Which is Java IDL, helps to define, implement and access CORBA object using Java. Previously, Java developer's had to choose between RMI and Java IDL.

These days, RMI-IIOP is widely used to bring CORBA capabilities to Java platform with some limitations. RMI over IIOP provides Java developers the freedom to write CORBA applications without learning CORBA IDL.

*So, far we understand that RMI allows communication between remote computers. Now, lets look at RMI Architecture Diagram and RMI mechanisms that allows back and forth communication between server and client.*

Mechanism involves generation of stub on the client-side which takes care of marshalling of parameters *(data)* and passing arguments to Skelton over the network when invoking a remote object method and un-marshal's data when receiving data from the skeleton which is generated on the server-side and is responsible for un-marshalling incoming arguments and marshalling results when returning values from the server object *(servant)* and passing it to client stub over network.

***RMI Architecture Diagram***



Next up let's look at some advantages and drawbacks of Java RMI –

| Advantages | Drawbacks |
|---|---|
| • Easy implementation. | • Can be used only with Java. Strictly Java. |
| • Automatic Marshalling and un-marshalling, puts developers at ease. | • Can be slower especially when compared to CORBA. |
| • Dynamic Interface creation is possible. | • Remote methods are synchronous, which can cause problems when the network is down and the application may freeze. |
| • Remote Objects look like they are local objects. | |

So far we know the basics, advantages and drawbacks of RMI. Next, I will share my views, opinions and experience with RMI.

## Discussion

As we know now, RMI is Java implementation of RPC, and it allows us to bring distributed capabilities to our Java application. Under the Introduction heading, we talked about Java IDL *(CORBA/IIOP)* and RMI-IIOP which can be used to expose Java Objects to CORBA ORBs. But for the purpose of this report, we will focus on the original/basic RMI communication through JRMP used for Java to Java remote calls, and requires the client and server to use Java Objects.

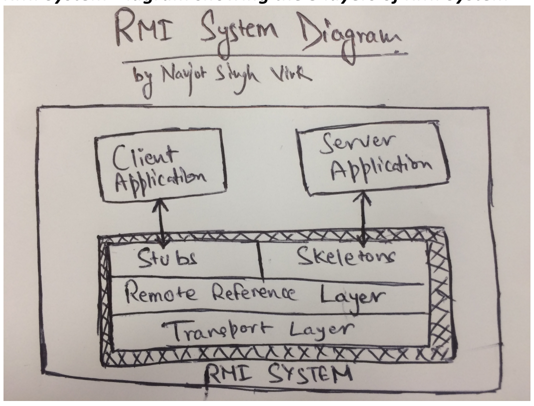
Why use RMI? RMI is a part of the Java core platform and *java.rmi* provides the RMI package. Any object that can be invoked remotely needs to implement Remote Interface *(serves to identify interfaces whose methods can be invoked from remote JVM).*

As a developer with 4 years' experience working with Java, I believe RMI put you at ease as it is easy to implement, uses widely used language Java and its object-oriented nature makes it powerful. RMI is capable of passing full complex objects as arguments *(like a hash map)* and return values unlike in existing RCP technology, the client would have to dissolve the object into primitive data types and transfer and re-assemble the object on the server whereas RMI out of the box allow you to transfer across the wire with no extra code client code putting developers at ease.

But the experience for other people may differ based on different factors like their understanding of Java language, goals and needs because it may or may not be efficient to implement RMI for creating your own distributed system for that you will need to list out your needs and requirements including security and speed and others. RMI uses Java's security principles and when compared to CORBA it may not be the best choice when aiming for speed. Now, we know that RMI may not always suit all your needs and other alternatives like CORBA are available u It allows to access remote objects in a distributed environment as if it's a local object, and pass more than one objects with the request and it automatically takes care of things like marshalling and un-marshalling so the developers don't have to deal working and writing complicated mechanisms to transfer data over the network.

How RMI Works? Here we will discuss how a basic RMI implementation works, we have already seen the architecture diagram of RMI under the Introduction heading, now let's list the high level steps/mechanisms involved in the implementation and working of RMI.
This is how RMI is implemented - Writing interface, Implementing Interface, Writing RMI Server, Writing RMI Client, Starting RMI Registry, Running Server and Client.
Let us try to understand the working with an example –

***RMI System Diagram showing the 3 layers of RMI System***



## Implementation

Brief explanation – How to implement RMI.
The original implementation of RMI depends upon JVM class representation mechanism, meaning client and server must be running JVM in order to communicate and this Java only is possible using JRMP.

- Writing interface
- Implementing Interface
- Writing RMI Server
- Writing RMI Client
- Starting RMI Registry
- Running Server and Client
-

## References

Villanova University, United States, *what is Distributed Systems? Accessed 3rd April 2017*
http://www.csc.villanova.edu/~schragge/CSC8530/Intro.html

Distributed Systems Goals Slides, pp.2-3
https://www.cis.upenn.edu/~lee/07cis505/Lec/lec-ch1-DistSys-v4.pdf

Tanenbaum, A. and Steen, M. (2007). *Distributed systems - Principles and Paradigms*. 2nd ed. pp.115-116.
https://vowi.fsinf.at/images/b/bc/TU_Wien-Verteilte_Systeme_VO_(G%C3%B6schka)_-_Tannenbaum-distributed_systems_principles_and_paradigms_2nd_edition.pdf

What is RMI Basic's?
http://www.javatpoint.com/RMI, https://www.youtube.com/watch?v=YyCUmKojtgk

RMI. How RMI works?
http://infolab.stanford.edu/CHAIMS/Doc/Details/Protocols/rmi/rmi_description.html

What are RMI, IIOP, and RMI-IIOP?
https://www.ibm.com/support/knowledgecenter/SSYKE2_8.0.0/com.ibm.java.hybrid.80.doc/rmi-iiop/overview.html

Getting Started with Java IDL
http://docs.oracle.com/javase/8/docs/technotes/guides/idl/GShome.html?cm_mc_uid=72983530074714919152354&cm_mc_sid_50200000=1491915235

An overview of RMI Applications
https://docs.oracle.com/javase/tutorial/rmi/overview.html

Advantages of RMI
http://www.oracle.com/technetwork/java/javase/tech/index-jsp-138781.html#close

To RMI or Not to RMI?
http://www.devx.com/tips/Tip/25531

Disadvantage of RMI
http://cs.iupui.edu/~aharris/cgi-bin/slides/langs56.html

RMI PATRIK FUHRER http://diuf.unifr.ch/drupal/sites/diuf.unifr.ch.drupal.softeng/files/file/publications/others/RMI.pdf

RMI: Observing the Distributed Pattern
http://www.cs.indiana.edu/~dgerman/tutorials/fie2004.pdf

Introduction to Java RMI by David Reilly (Implementation Steps Described)
http://www.javacoffeebreak.com/articles/javarmi/javarmi.html

Package java.rmi
https://docs.oracle.com/javase/8/docs/api/java/rmi/package-summary.html

The research report also uses moodle resources available on college website (mainly the implementation).
https://moodle.ncirl.ie