RMI
(REMOTE METHOD INVOCATION)

RMI FOR DISTRIBUTED COMPUTING

RMI ARCHITECTURE

RMI REGISTRY SERVICE

PARAMETER PASSING IN REMOTE METHODS

CREATING RMI APPLICATION

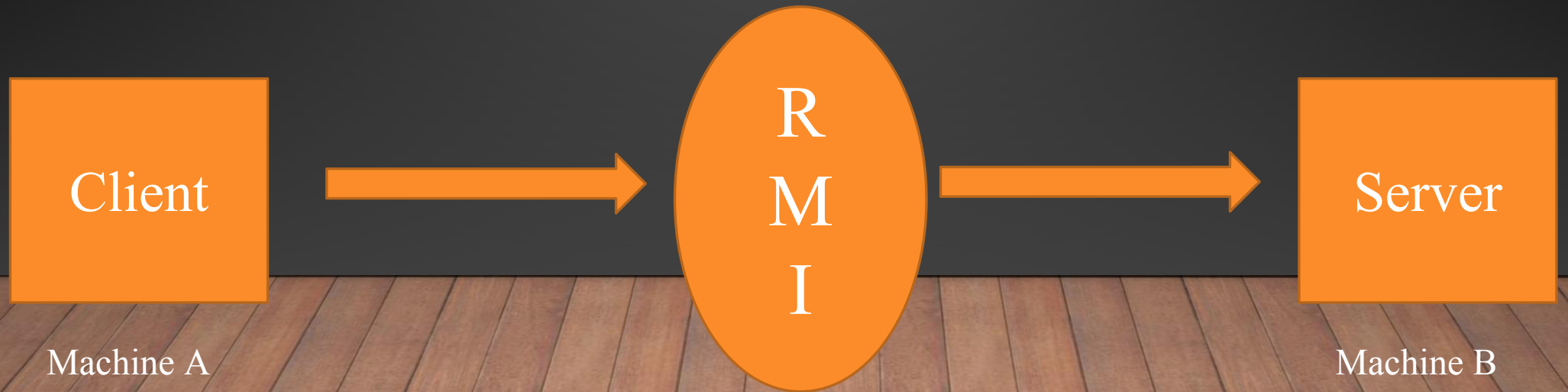
STEPS INVOLVED IN RUNNING THE RMI APPLICATION

WHAT IS RMI?

- The **RMI** (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.
- RMI is used to build distributed applications; it provides remote communication between Java programs. It is provided in the package **java.rmi**.
- RMI is Java's remote procedure call (RPC) mechanism. RMI has several advantages over traditional RPC systems because it is part of Java's object oriented approach. Traditional RPC systems are language-neutral, and therefore are essentially least-common-denominator systems-they cannot provide functionality that is not available on all possible target platforms.RMI is focused on Java, with connectivity to existing systems using native methods. This means RMI can take a natural, direct, and fully-powered approach to provide you with a distributed computing technology that lets you add Java functionality throughout your system in an incremental, yet seamless way.

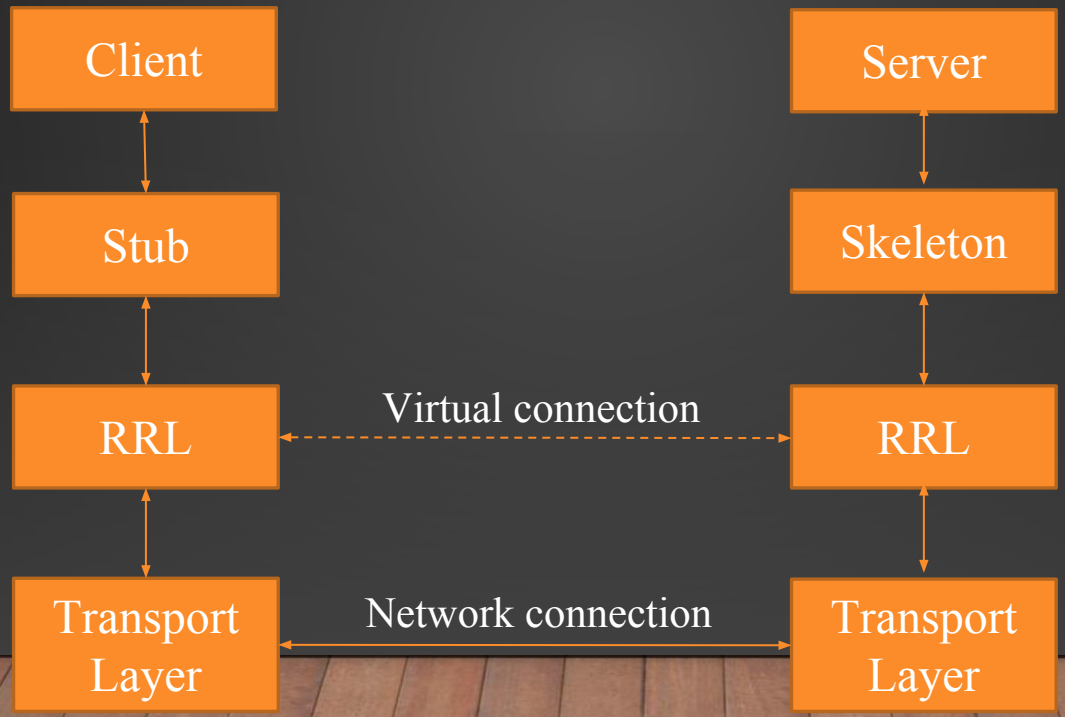
RMI FOR DISTRIBUTED COMPUTING

- Machine A can invoke methods of Machine B
- Two intermediate objects are used in RMI, the object used in client side is “stub” and the object used in server side is “skeleton” .
- If any application performs these tasks, it can be distributed application.
- The application need to locate the remote method
- It need to provide the communication with the remote objects, and
- The application need to load the class definitions for the objects.



RMI ARCHITECTURE

- 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.



RMI ARCHITECTURE

- **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 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.

STUB AND SKELETON

Stub

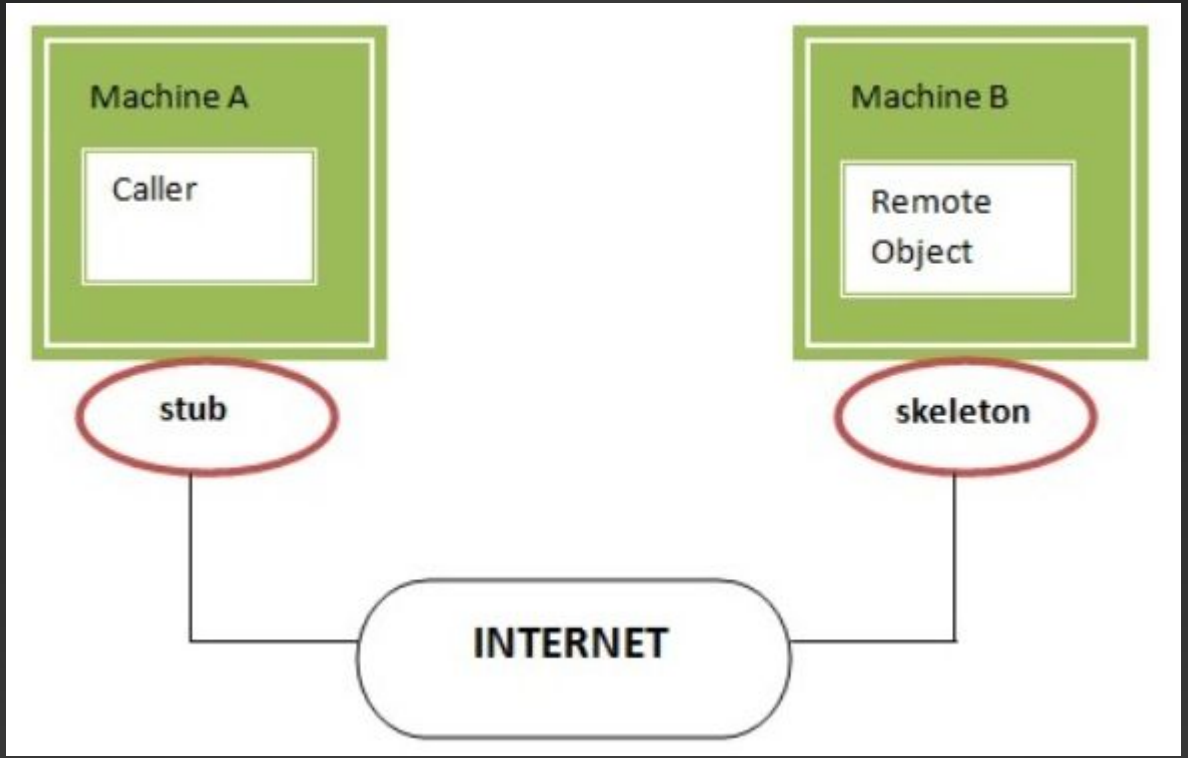
- The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:
- It initiates a connection with remote Virtual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

STUB AND SKELETON

Skeleton

- The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. 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.

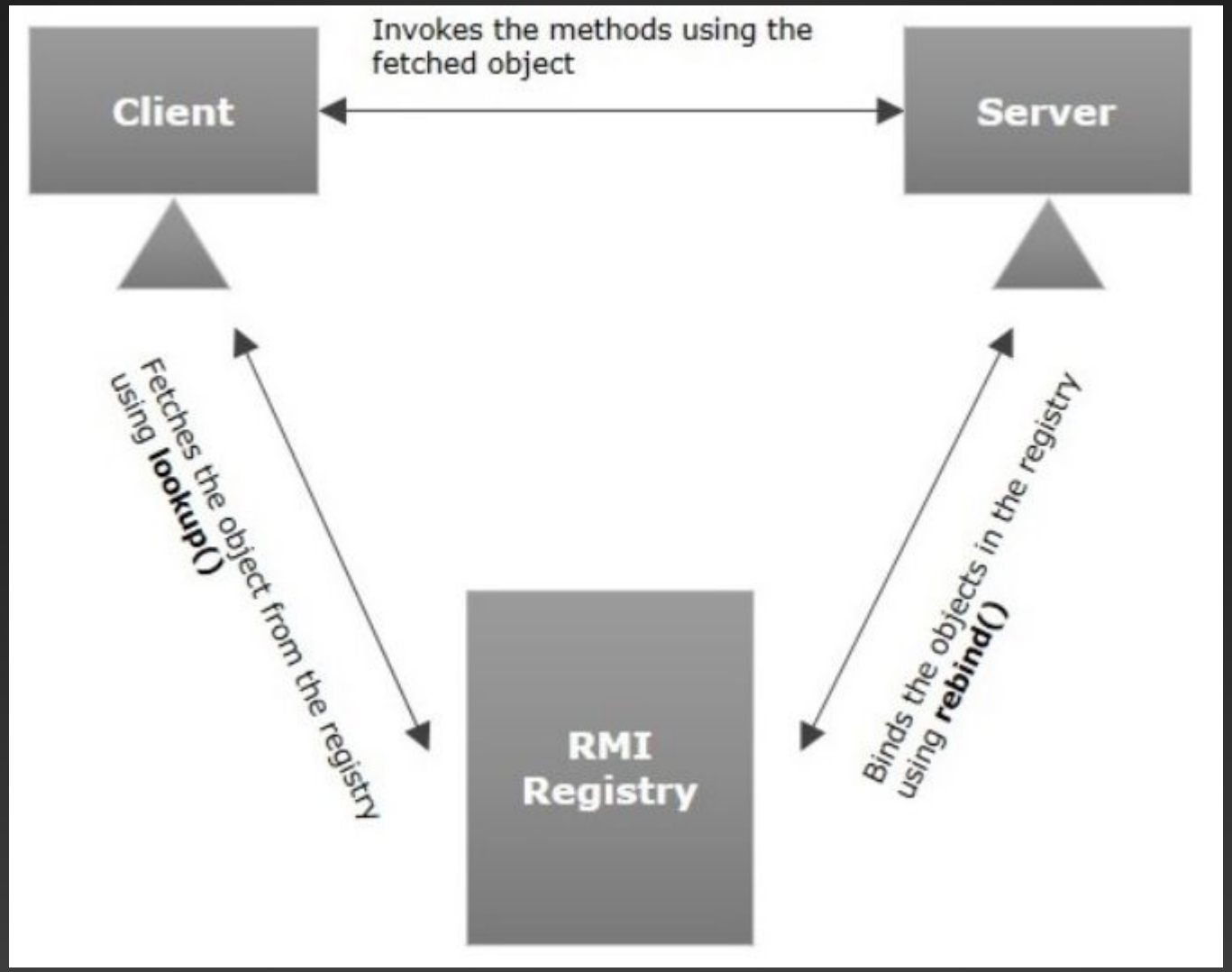
STUB AND SKELETON



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).

RMI REGISTRY



STEPS INVOLVED IN RUNNING THE RMI APPLICATION

1. Create the remote interface (AddI)
2. Implementation of remote interface (AddC)
3. Compile, stub and skeleton (rmic)
4. Start the registry (rmiregistry)
5. Create and start server
6. Create and start client

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.