RMI (Remote Method Invocation)

Architecture of Remote Method Invocation (RMI)

Simple coding way

CLIENT SIDE

```
Add(2,3)
Print(c);
```

SERVER SIDE

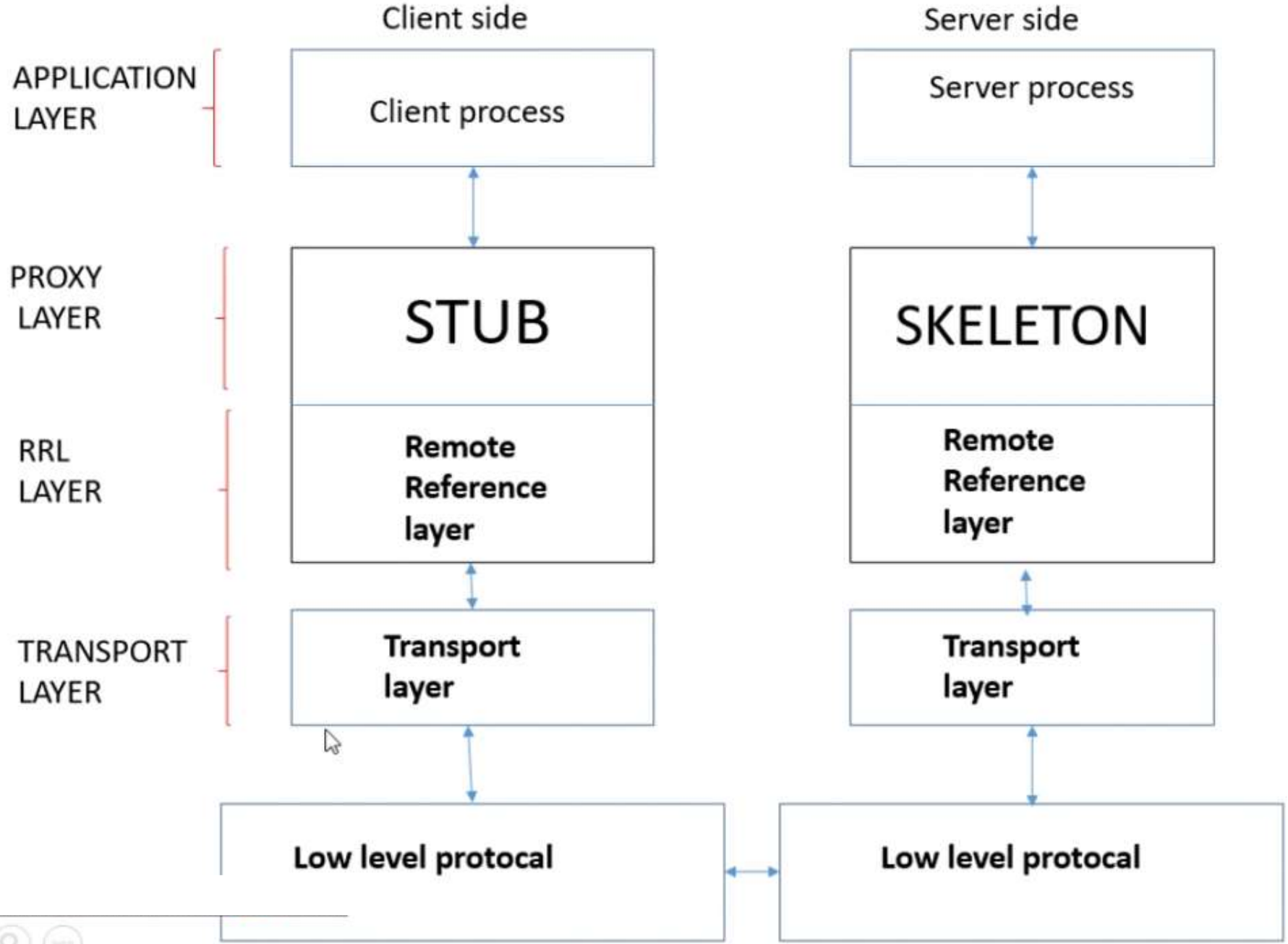
```
int ADD(int a ,int b)
{
int c;
c=a+b;
Return c;
}
```

Defination:

- The remote method invocation (RMI) is a remote object invocation technique used to locate and fetch the objects at the remote side using object references

What is (Java)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.
- The RMI provides remote communication between the applications using two objects *stub* and *skeleton*.
- RMI uses **stub** and **skeleton** object for communication with the remote object.
- A **remote object** is an object whose method can be invoked from another JVM.


Client side
Server side
APPLICATION LAYER
Client process
Server process
PROXY LAYER
STUB
SKELETON
RRL LAYER
Remote Reference layer
Remote Reference layer
TRANSPORT LAYER
Transport layer
Transport layer
Low level protocal
Low level protocal

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:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

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:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.

APPLICATION LAYER

- Responsible for running client and server applications
- Here client application invokes methods defined by server application
- When client invokes a method then request is passed to proxy layer

PROXY LAYER

- Responsible for creating client stub at client side by packing the request msgs sent by client process
- Also responsible for creating skeleton by packing response msgs sent by server
- Once stub and skeleton are created they are passed to RRL

REMOTE REFERENCE LAYER (RRL)

- Checks semantics and remote references used by client process using remote reference protocol
- Finally RRL transmits msgs and data to RMI transport layer

TRANSPORT LAYER

- Responsible for establishing and maintaining stream oriented connection between client and server.
- Also responsible for managing send/receive of request/reply msgs between client and server