Institute of Computer Science
Department of Mathematics and Computer Science

Freie Universität Berlin

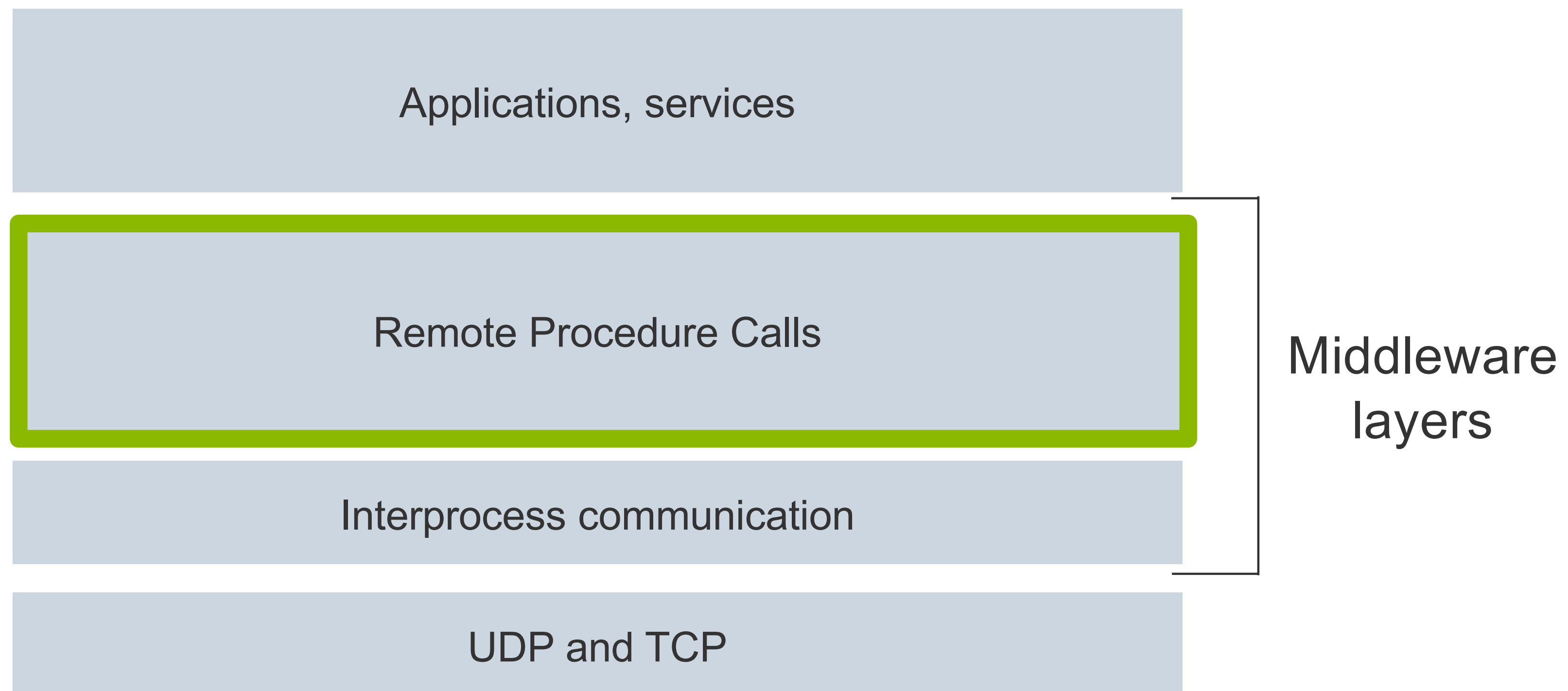
Algorithms and Programming IV

Remote Method Invocation

Summer Term 2023 | 26.06.2023

Claudia Müller-Birn, Barry Linnert

Middleware Layers



Our Topics Today

- Recap
- Remote Object invocation
- Components of RMI
- Implementation of RMI
- Java RMI

Remote Call Variantes

Remote procedure call (RPC)

- A procedure is called (typically part of a module) for procedural languages (e.g., Modula, C)
- Also for heterogeneous infrastructures, e.g. Distributed Computing Environment (DCE)

Remote object invocation (ROI)

- For object-oriented languages (e.g., Java, C++, C#), where an operation/method of an object is called.
- In Java it is called Remote Method Invocation (RMI) and in C# ".NET Remoting"
- Also for heterogeneous infrastructures, e.g. CORBA.

Commonalities of ROI and RPC

- Support of programming languages with interfaces.

- Both are typically constructed on top of the request-reply protocol, offering several call semantics (exactly once, at most once, and at least once).

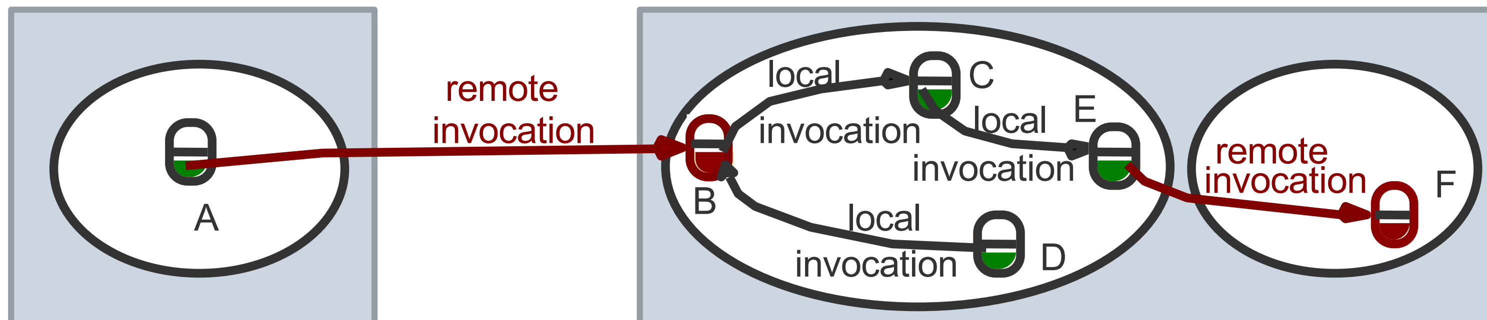
- Offer a similar level of transparency, which means local and remote calls employ the same syntax, but remote interfaces expose the distributed nature, for example, by supporting remote exceptions.

Freie Universität Berlin

Remote method invocation

REMOTE OBJECT INVOCATION

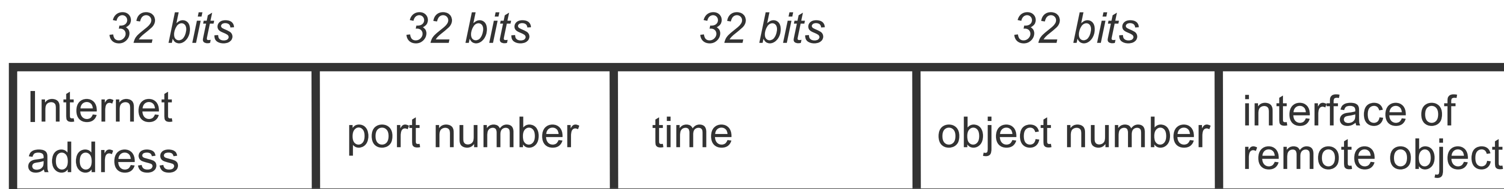
Remote and Local Method Invocation



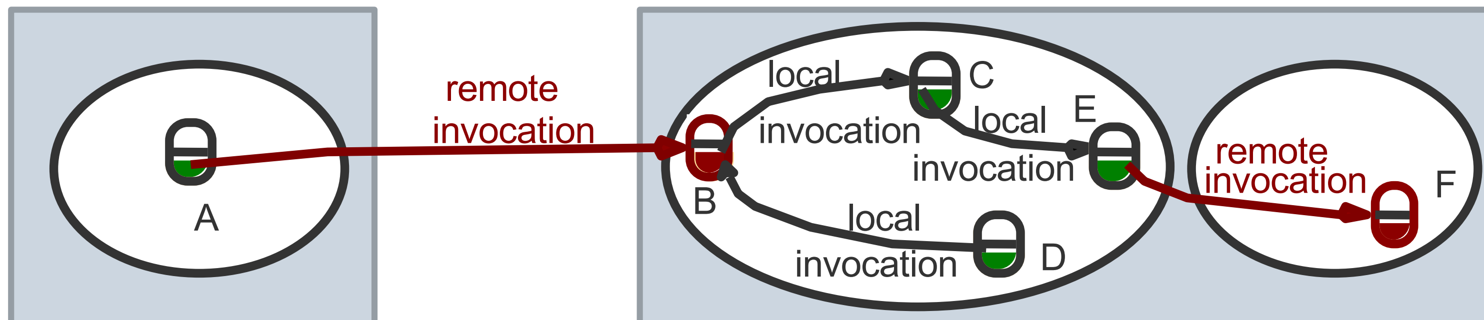
Remote object reference: Other objects can invoke the methods of a remote object if they have access to its remote object reference.

Remote Object Reference

The remote object reference is an identifier for a remote object that is valid throughout the distributed system. It is passed in the invocation message to specify which object is to be invoked.



Remote and Local Method Invocation

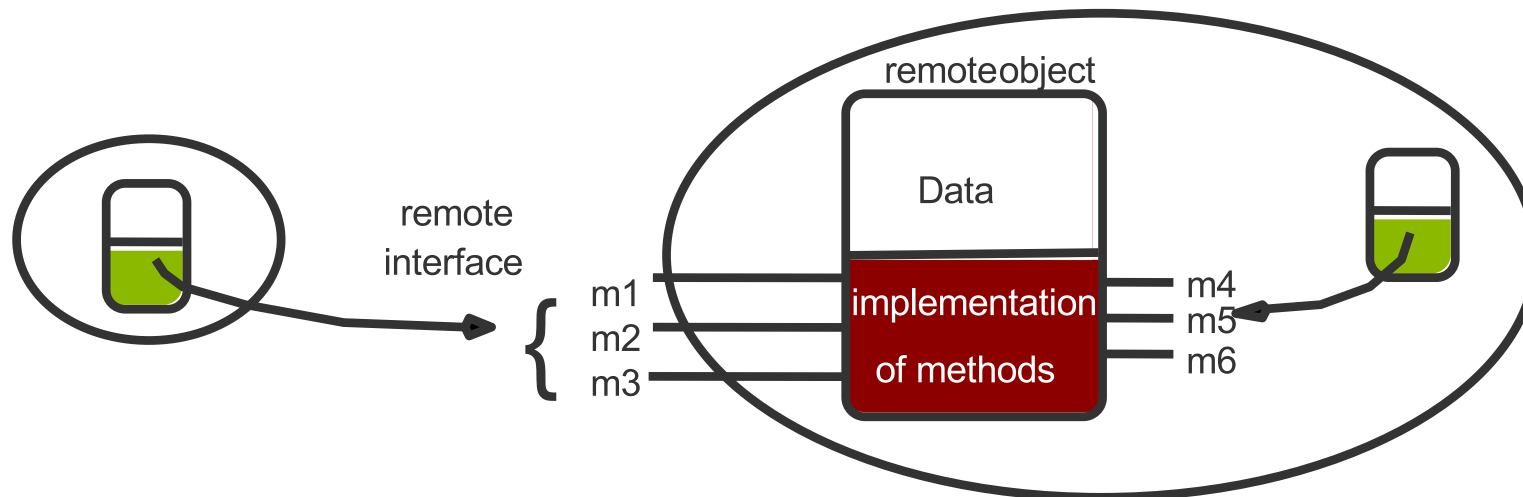


Remote object reference: Other objects can invoke the methods of a remote object if they have access to its remote object reference.

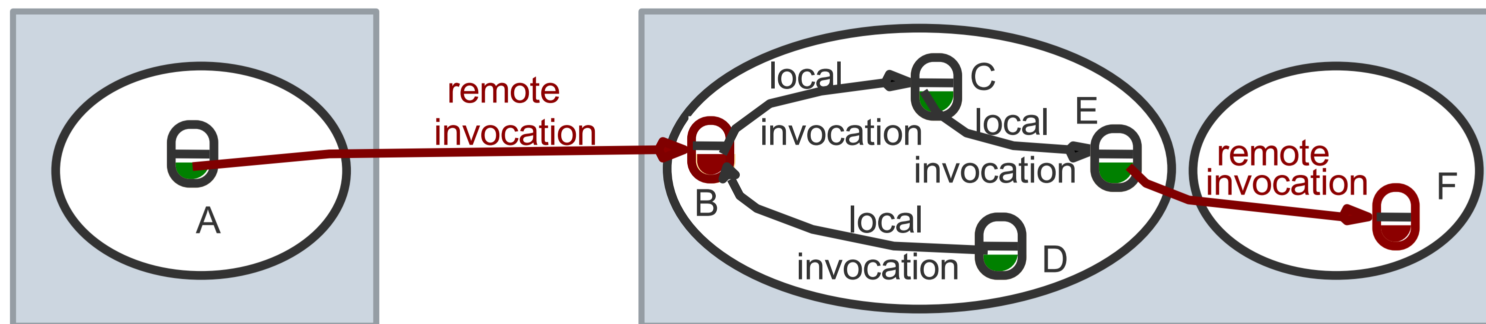
Remote interface: Every remote object has a remote interface that specifies which of its methods can be invoked remotely.

Freie Universität Berlin

A Remote Object and its Remote Interface

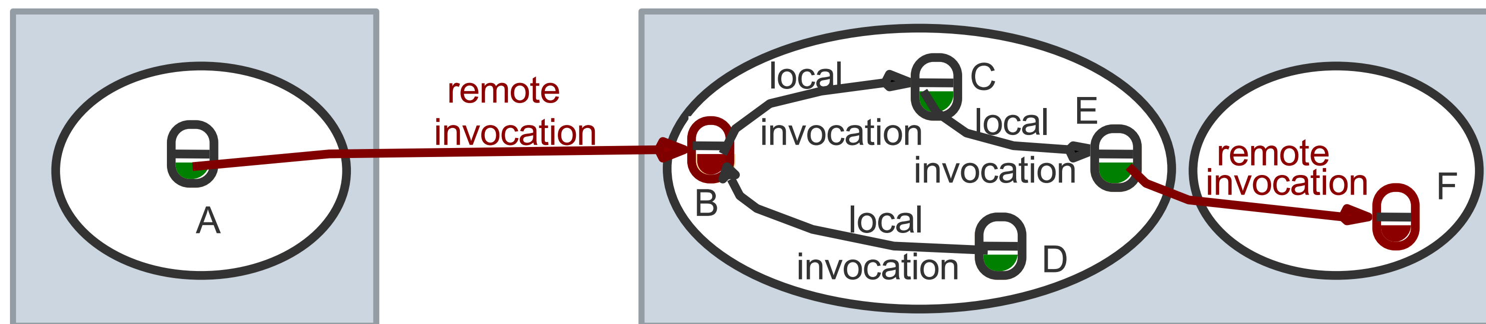


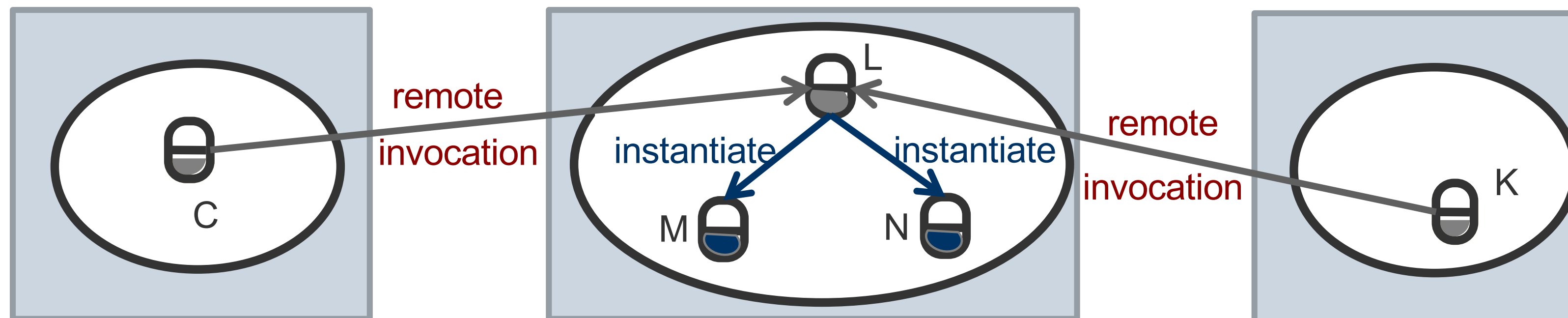
Instantiation of Remote Objects



Freie Universität Berlin

Instantiation of Remote Objects





Exceptions

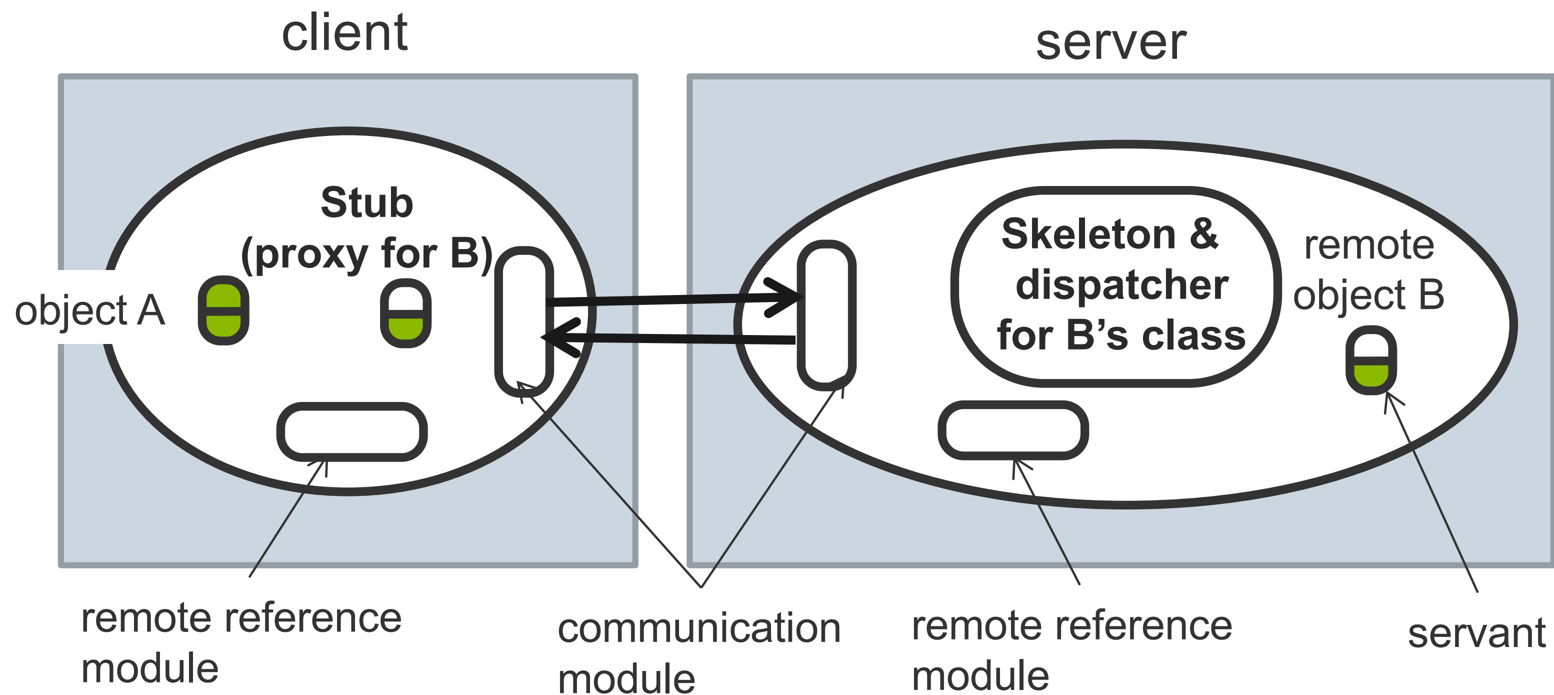
- Any remote invocation may fail for reasons related to the invoked object being in a different process or computer from the invoker.
- Remote method invocation should be able to raise exceptions such as timeouts. Exceptions provide a clean way to deal with error conditions without complicating the code.
- In Java, for example, predefined exceptions are listed here:
  https://docs.oracle.com/en/java/javase/17/docs/specs/rmi/exceptions.html

Freie Universität Berlin

Remote method invocation

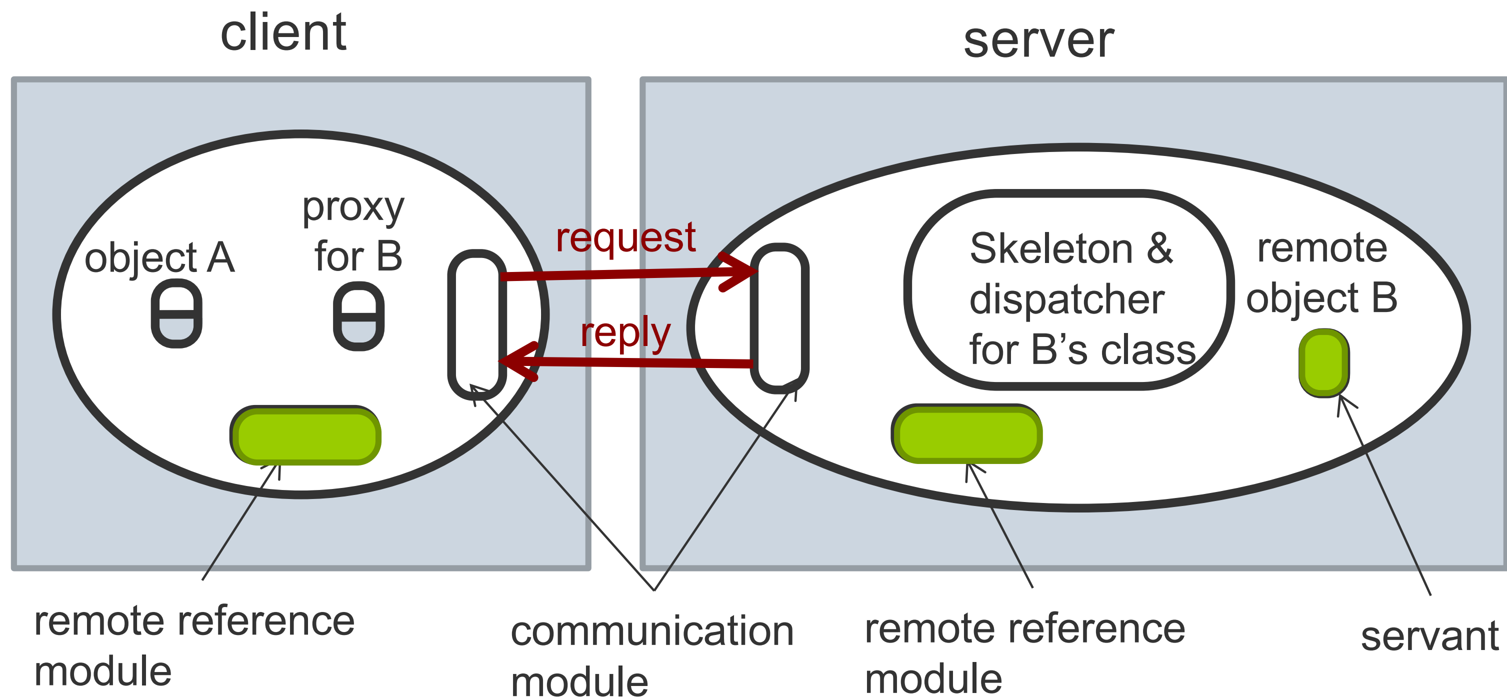
RMI COMPONENTS

Process of Remote Method Invocation



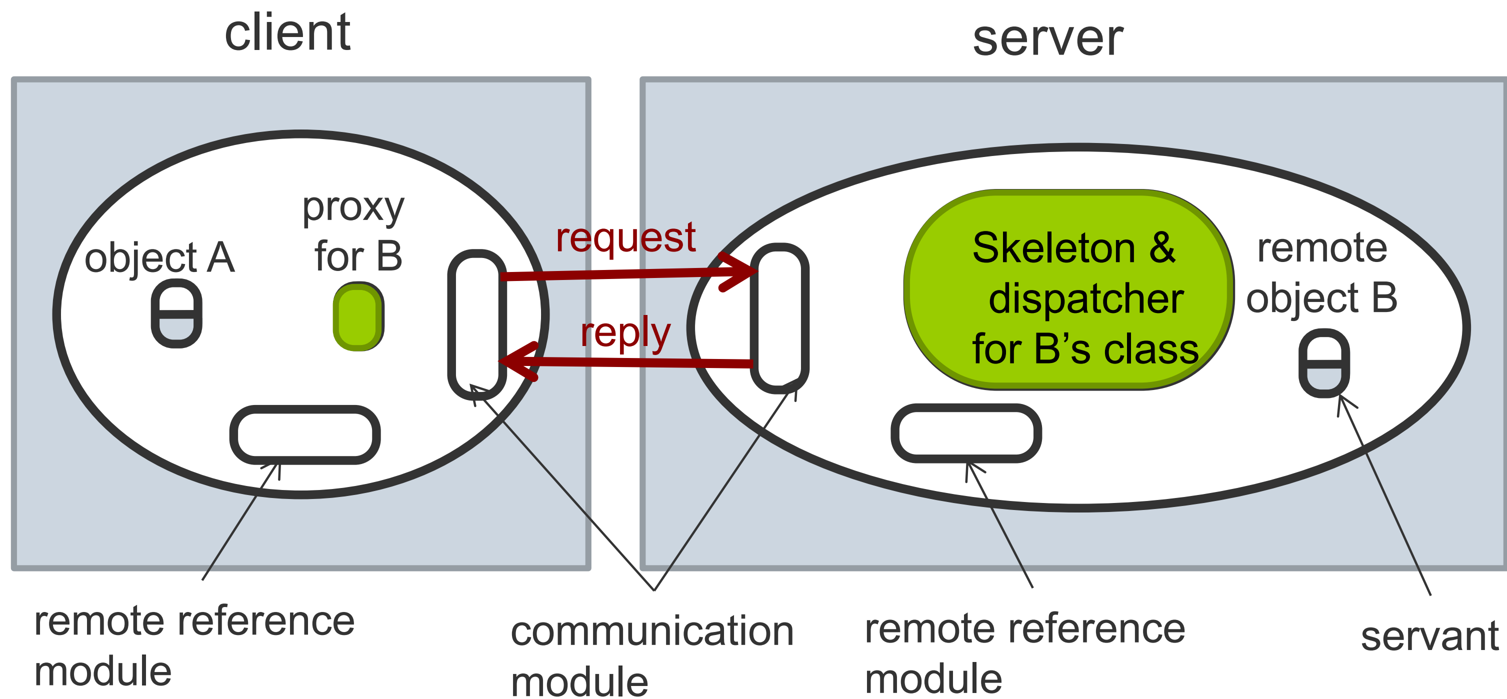
Freie Universität Berlin

Remote Reference Module/Servant

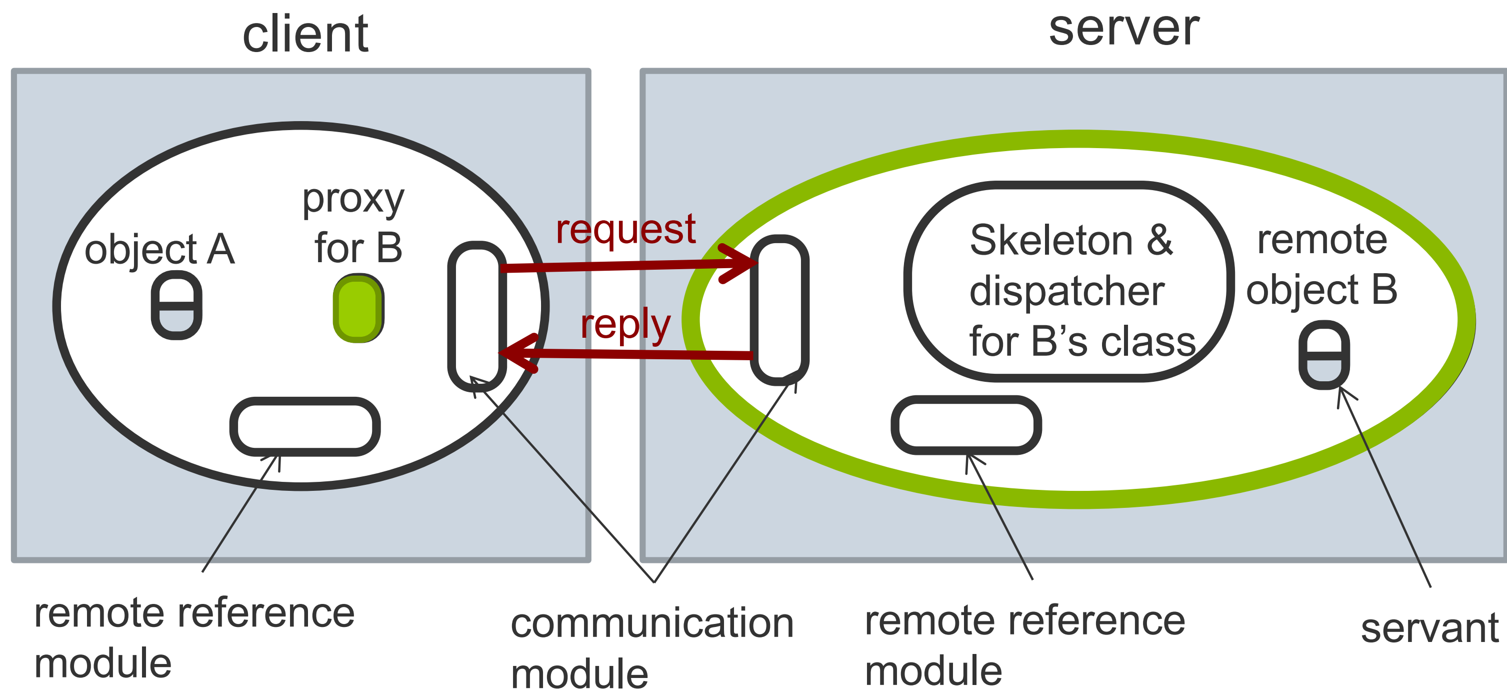


Freie Universität Berlin

RMI Software



Java Distributed Garbage Collection Algorithm

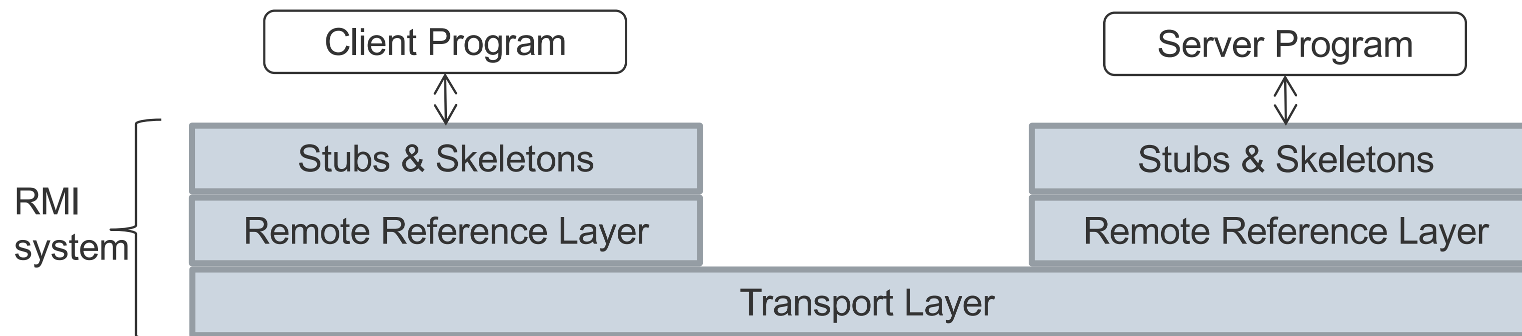


Remote Method Invocation

RMI IMPLEMENTATION

Freie Universität Berlin

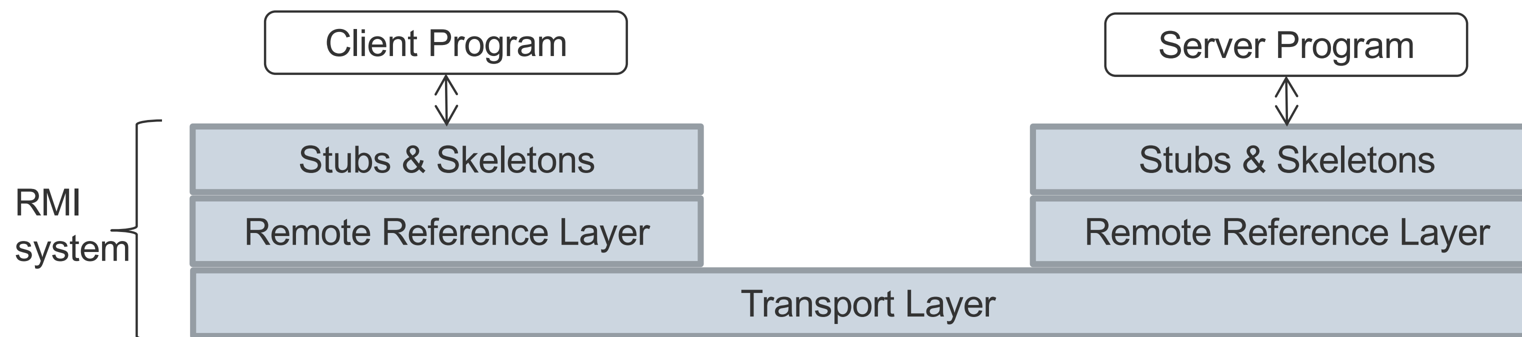
Abstraction Layers in the RMI Implementation



Freie Universität Berlin

Abstraction Layers in the RMI Implementation

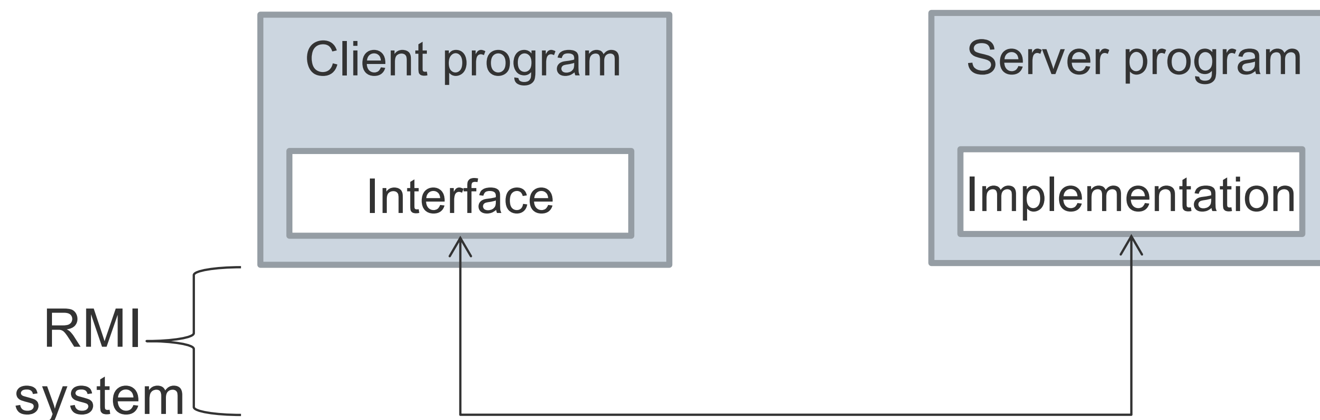
1. **Stub and Skeleton Layer:** Intercepts method calls made by the client to the interface reference variable and redirects these calls to a remote RMI service
2. **Remote Reference Layer:** Interpret and manage references made from clients to the remote service objects
3. **Transport Layer:** Is based on TCP/IP connections between machines in a network. Provides basic connectivity.



Separation of Concerns

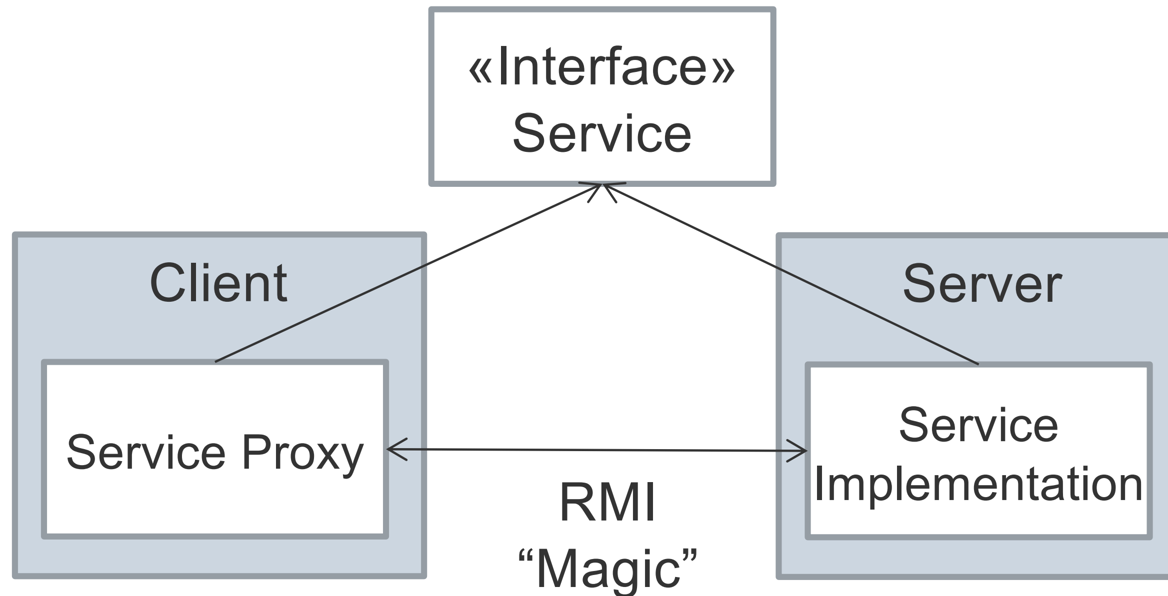
RMI architecture is based on one important principle: *the **definition of behavior** and the **implementation of that behavior** are separate concepts*.

RMI allows the code that defines the behavior and the code that implements the behavior to remain separate and to run on separate JVMs.



Freie Universität Berlin

Implementation of the Interface



Freie Universität Berlin

Server and Client Programs

Server program

- Contains classes for the dispatcher and skeletons, together with the implementations of the classes of all of the servants
- Contains a *initialization* section (responsible for creating and initializing at least one of the servants to be hosted by the server)
- Generally allocates a separate thread for the execution of each remote invocation -> designer of the remote object implementation must allow concurrent executions

Client program

- Contain the classes of the proxies for all of the remote objects that it will invoke
- Require a means of obtaining a remote object reference for at least one of the remote objects held by the server -> binder

Freie Universität Berlin

Naming Remote Objects

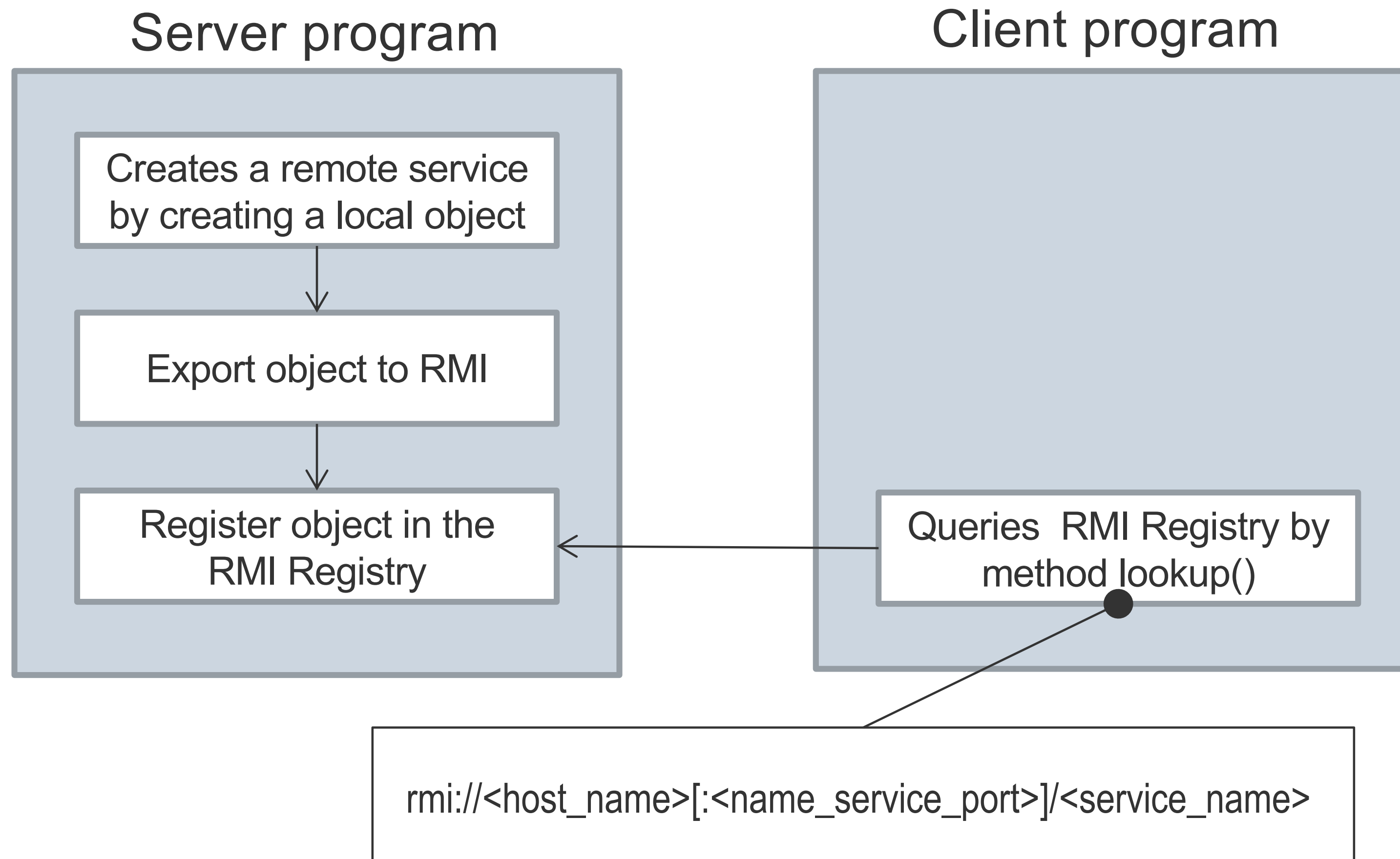
How does a client find a RMI remote service?

RMI includes a simple service called the RMI Registry, *rmiregistry*.

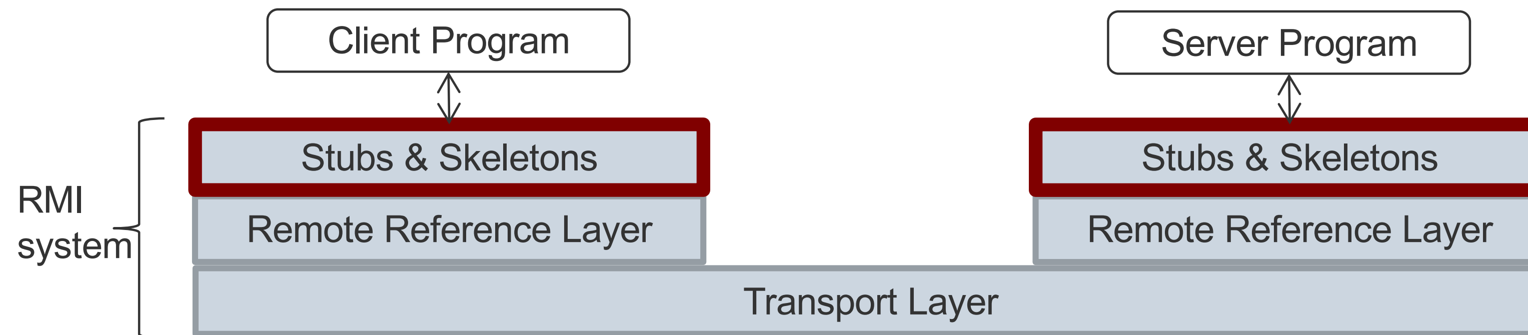
The RMI Registry runs on each machine that hosts remote service objects and accepts queries for services, by default on port 1099.

Freie Universität Berlin

Naming Remote Objects (*cont.*)



Stub and Skeleton Layer



Freie Universität Berlin

Stub and Skeleton Layer

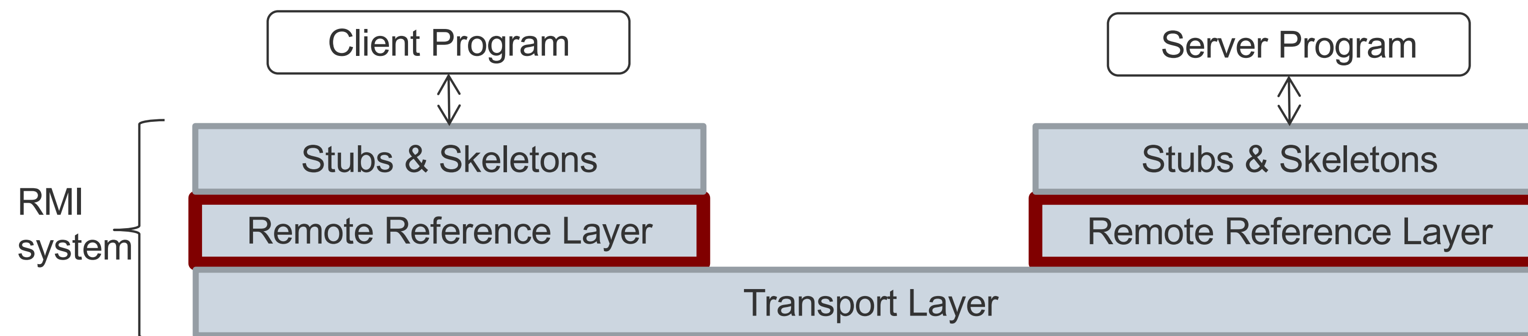
RMI uses the **Proxy design pattern**

- The stub class is the *proxy*
- The remote service implementation class is the *RealSubject*

The Skeleton is a helper class.

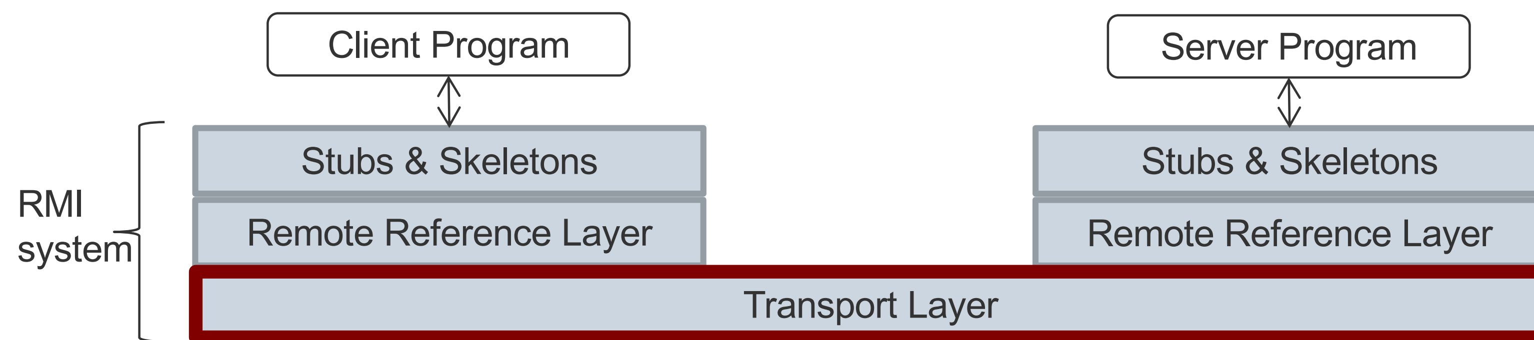
- Carries on a conversation with the stub
- Reads the parameters for the method call → makes the call to the remote service implementation object → accepts the return value → writes the return value back to the stub

Remote Reference Layer

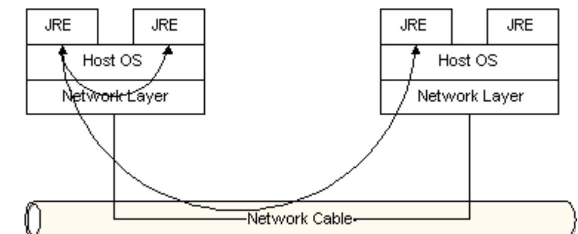


- Defines and supports the invocation semantics of the RMI connection.
- Provides a **RemoteRef** object that represents the link to the remote service implementation object.

Transport Layer



- The Transport Layer makes the connection between JVMs. All connections are stream-based network connections that use TCP/IP.



- On top of TCP/IP, RMI uses a wire level protocol called **Java Remote Method Protocol (JRMP)**. JRMP is a proprietary, stream-based protocol.

Freie Universität Berlin

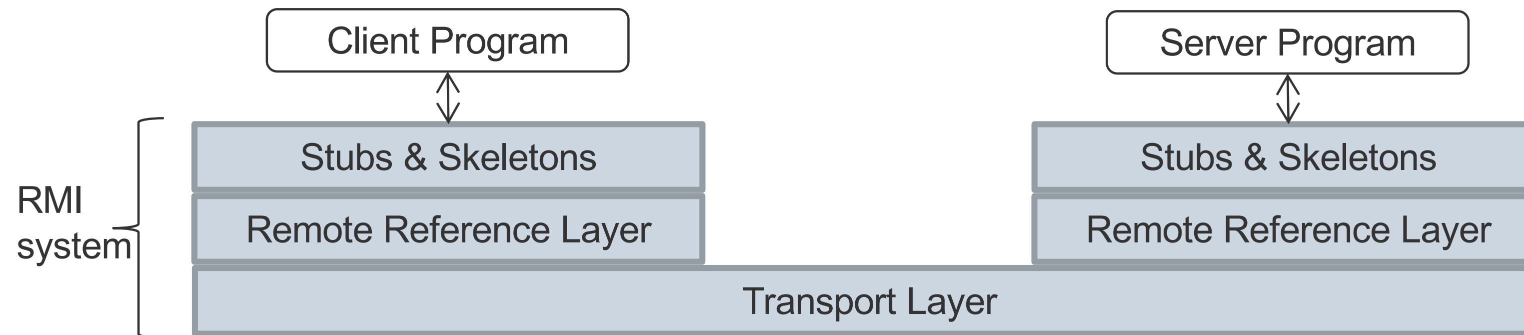
Generation of Classes for Proxies, Dispatcher and Skeleton

Classes for proxies, dispatcher and skeleton are generated automatically by an interface compiler.

Java RMI contains a set of methods offered by a remote object defined as a Java interface that is implemented within the class of the remote object.

Java RMI compiler generates the proxy, dispatcher and skeleton classes from the class remote object.

Abstraction Layers in the RMI Implementation



Freie Universität Berlin

JAVA JMI

Freie Universität Berlin

How to implement a RMI Java application

- Define the remote interface
- Develop the implementation class (remote object)
- Develop the server program
- Develop the client program
- Compile the application
- Execute the application

Defining the Remote Interface

```
import java.rmi.Remote;
import java.rmi.RemoteException;

// Creating Remote interface for our application
public interface Hello extends Remote {
      void printMsg()
                  throws RemoteException;
}
```

Developing the Implementation Class (Remote Object)

```
// Implementing the remote interface
public class ImplExample implements Hello {

// Implementing the interface method
        public void printMsg() {
           System.out.println("This is an example RMI program.");
                }
        }
```

Developing the Server Program

```
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class Server extends ImplExample {
        public Server() {}
        public static void main(String args[]) {
                try {
                // Instantiating the implementation class
                ImplExample obj = new ImplExample();
```

…

Developing the Server Program (*cont.*)

```
        // Exporting the object of implementation class
        // (here we are exporting the remote object to the stub)
        Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);

        // Binding the remote object (stub) in the registry
        Registry registry = LocateRegistry.getRegistry();

        registry.bind("Hello", stub);
        System.err.println("Server ready");
      } catch (Exception e) {
        System.err.println("Server exception: " + e.toString());
        e.printStackTrace();
}     } }
```

Freie Universität Berlin

Developing the Client Program

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class Client {
        private Client() {}
        public static void main(String[] args) {
            try {
                // Getting the registry
                Registry registry = LocateRegistry.getRegistry(null);

                // Looking up the registry for the remote object
                Hello stub = (Hello) registry.lookup("Hello");
```

Freie Universität Berlin

Developing the Client Program (cont.)

```
            // Calling the remote method using the obtained object
            stub.printMsg();
            // System.out.println("Remote method invoked");


         } catch (Exception e) {
            System.err.println("Client exception: " +
                               e.toString());
            e.printStackTrace();
    }     } }
```

Freie Universität Berlin

Compile Application

Open the folder where you have stored all the programs and compile all the Java files:

```
C:\rmi_example>javac *.java
```

Execute Application

Step 1 − Start the **rmi** registry using the following command.

```
C:\rmi_example>start rmiregistry
```

Step 2 − Run the server class file.

```
C:\rmi_example>java Server
Server ready
_
```

Step 3 − Run the client class.

```
C:\rmi_example>java client
```

Verification

```
C:\rmi_example>java Server
Server ready
This is an example RMI program.
```

Freie Universität Berlin

Summary

- The Remote Method Invocation (RMI) is a highly useful API provided in Java that facilitates communication between two separate Java Virtual Machines (JVMs). It allows an object to invoke a method on an object residing in another address space.

- It provides a secure way for applications to communicate with each other. It achieves this functionality using concepts Stub (Client calling object) and Skeleton (Remote object residing on the server).

- RMI is used to build distributed applications by minimizing the complexity of the application in a distributed architecture.

Institute of Computer Science
Department of Mathematics and Computer Science

Freie Universität Berlin

Algorithms and Programming IV

Peer-to-peer computing and networking

Summer Term 2023 | 28.06.2023

Claudia Müller-Birn, Barry Linnert