ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

BÀI GIẢNG
LẬP TRÌNH MẠNG

PGS.TS.Huỳnh Công Pháp; Nguyễn Anh Tuấn; Lê Tân;
Nguyễn Thanh Cẩm;Hoàng Hữu Đức

Khoa Khoa học máy tính

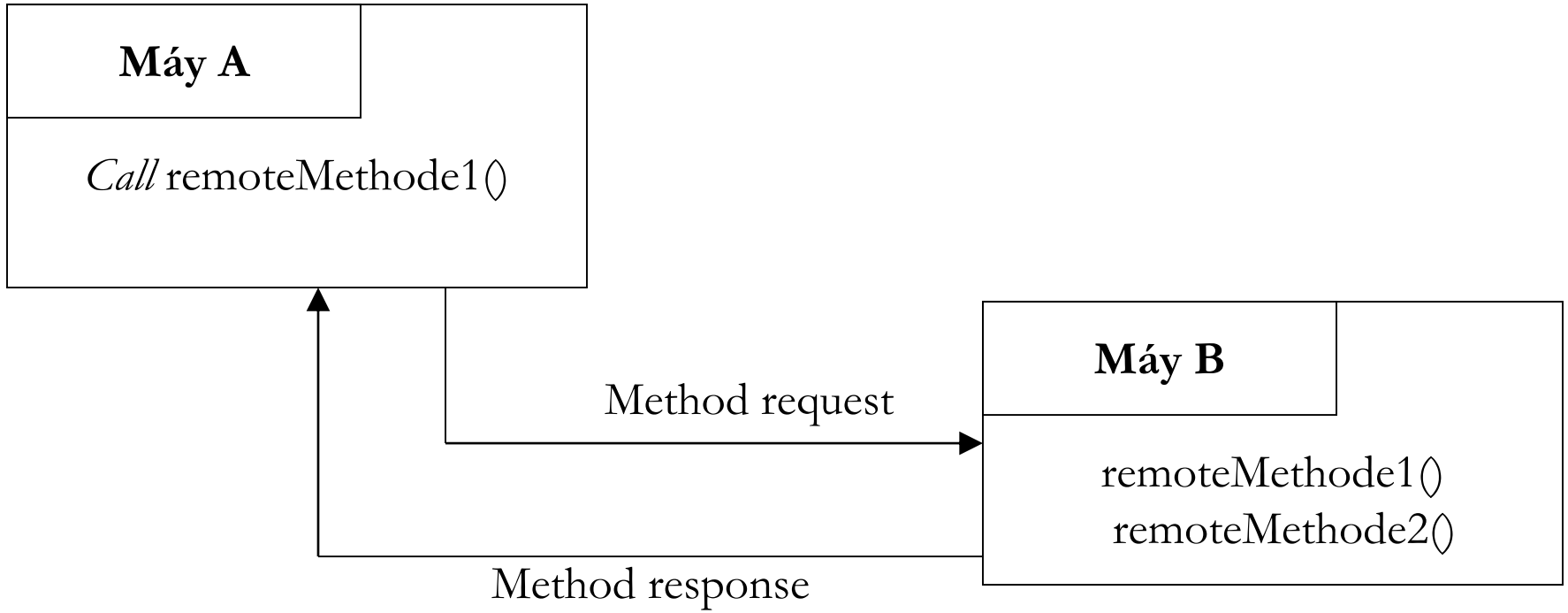
Bài 8. Gọi hàm và thủ tục từ xa

3 kỹ thuật phổ biến:

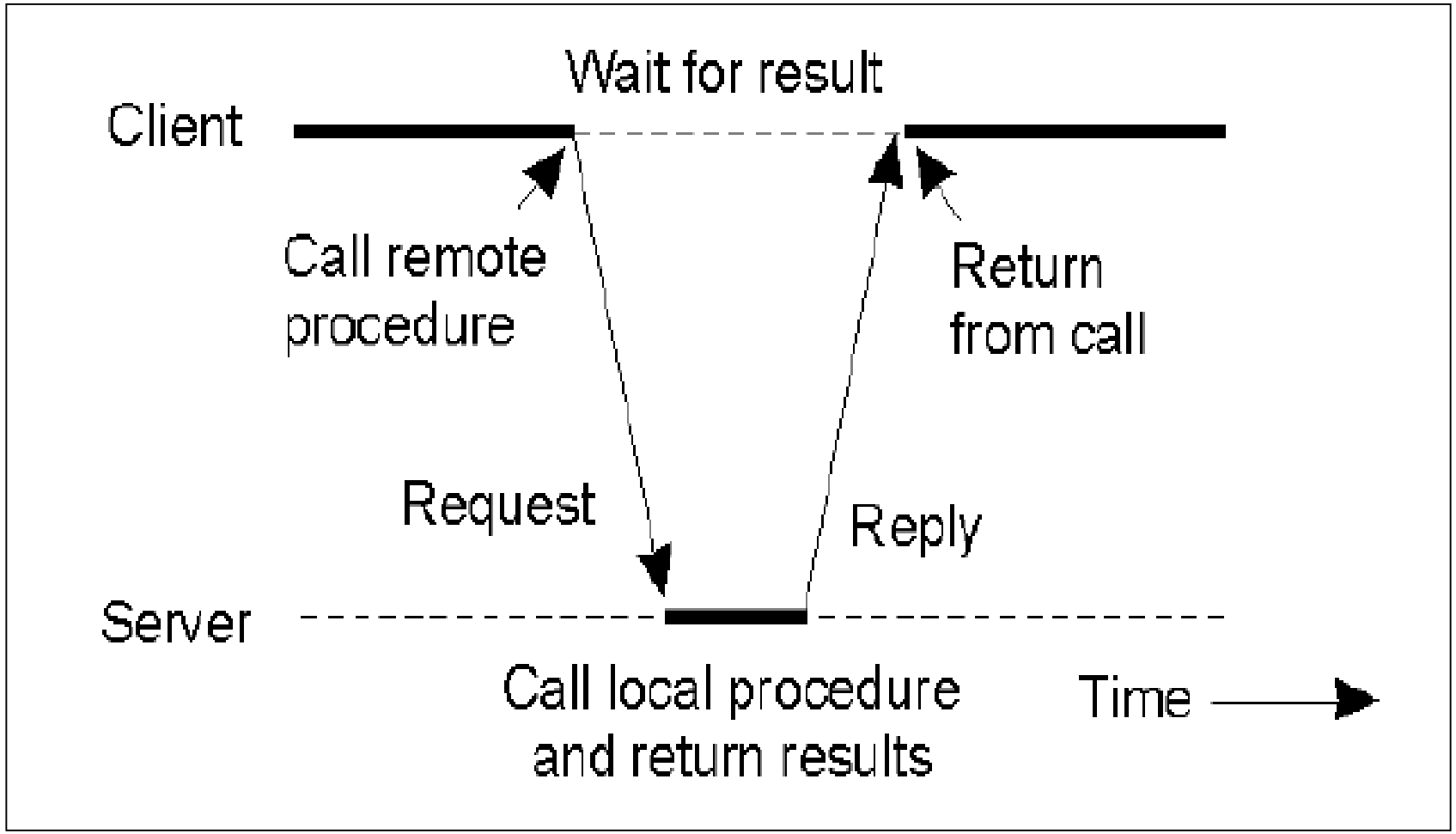
- Remote Procedure Calls (RPC)
- Remote Method Invocation (RMI)
- Common Object Request Broker Architecture (CORBA)

Tổng quan

- Một trong những kỹ thuật độc đáo tạo các chương trình phân tán client/server là cho phép
  - một ứng dụng trên một máy có thể gọi các phương thức ở các ứng dụng trên máy khác trên mạng.
- Công nghệ này hết sức quan trọng cho việc phát triển các hệ thống lớn, bởi vì khả năng phân bố tài nguyên và xử lý bài toán trên nhiều máy khác nhau.



Nguyên lý gọi hàm và phương thức từ xa giữa chương trình client và server.



Sự khác với gọi hàm và thủ tục cục bộ

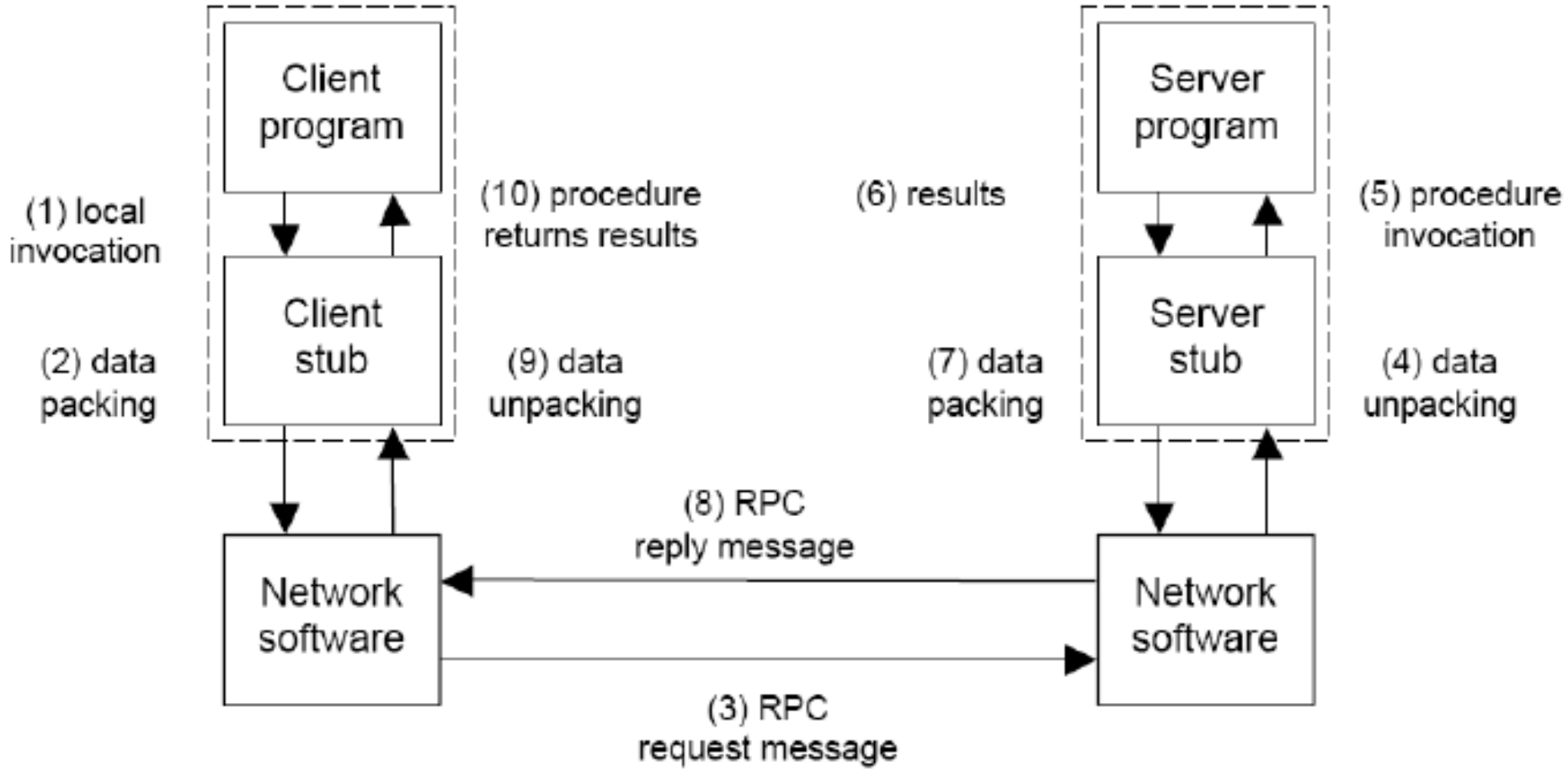
- Độ trể tương đối lớn hơn
- Không thể truyền con trỏ đến các cấu trúc dữ liệu
  - client và server hoạt động ở các không gian địa chỉ khác nhau
- Môi trường xử lý khác nhau (mở files, mở sockets, etc)
- Chế độ báo lỗi khác nhau

So sánh kỹ thuật RMI với RPC

- RMI chỉ hỗ trợ ứng dụng viết bằng Java.
- RMI làm việc với đối tượng, và cho phép các phương thức chấp nhận và trả lại các đối tượng cũng như kiểu dữ liệu cơ bản.

- RPC hỗ trợ đa ngôn ngữ
- RPC, không hỗ trợ khái niệm đối tượng.
- Các thông điệp truyền đến dịch vụ RPC được biểu diễn bởi ngông ngữ External Data Representation (XDR).
- Chỉ có các kiểu dữ liệu có thể định nghĩa bởi XDR có thể truyền, chủ yếu là kiểu cơ bản, không cho phép đối tượng được truyền.

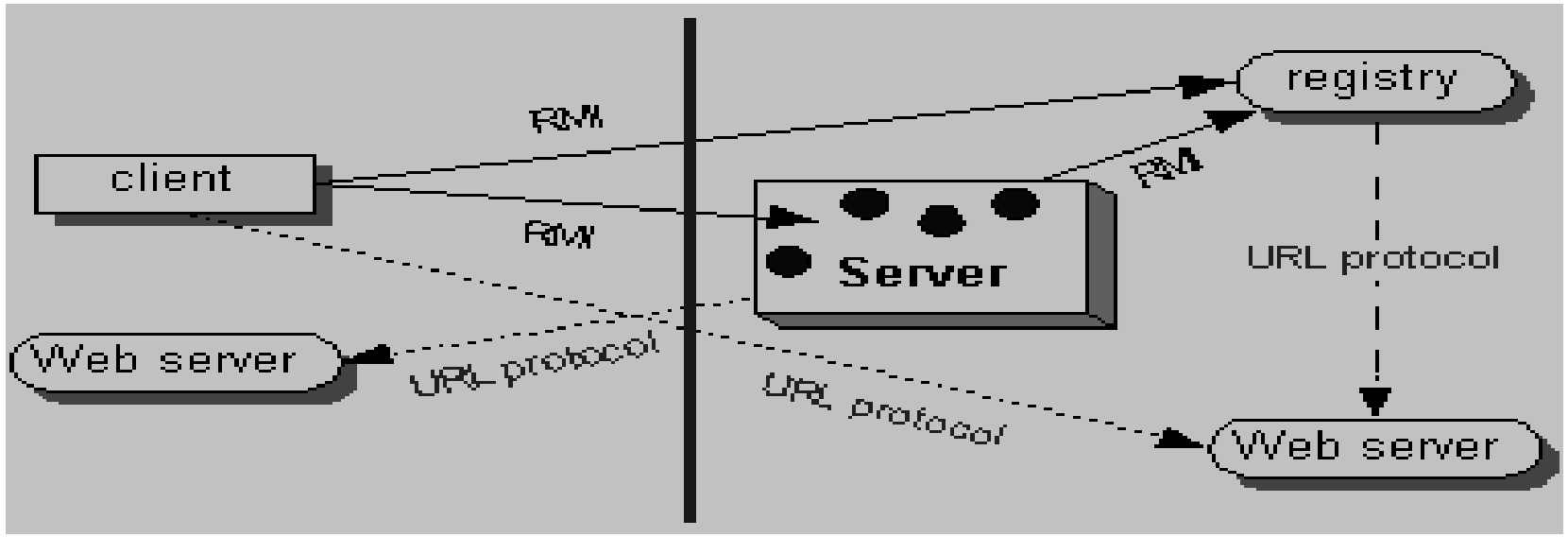
Các bước gọi thủ tục từ xa RPC



Lập trình RMI

RMI

- RMI cho phép các đối tượng đang thực thi trên một trạm có khả năng gọi các phương thức của các đối tượng khác đang chạy trên các trạm khác.
- Sự gọi các phương thức trên các máy khác được gọi là sự gọi phương thức từ xa (remote method invocation).
- Các phương thức ở xa thực hiện việc tính toán và trả lại kết quả cho các hàm cục bộ.
- RMI là công nghệ Java cho phép tạo các ứng dụng phân tán.



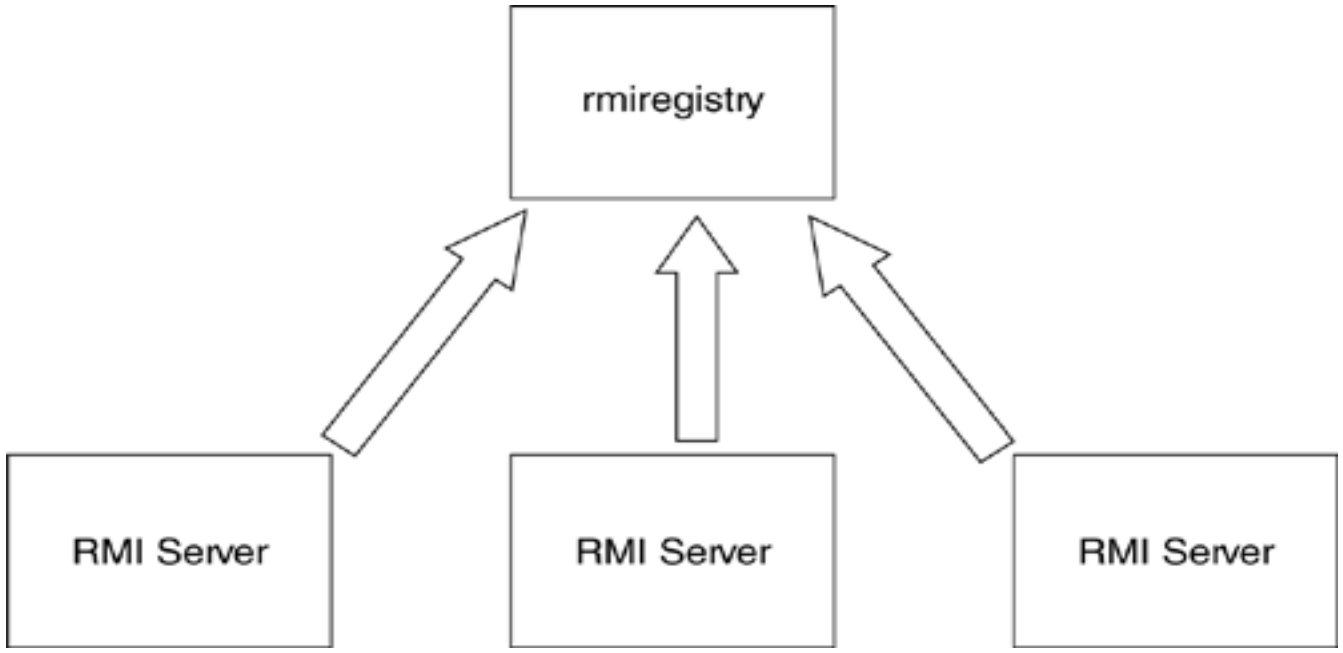
- Java cung cấp một số interface và lớp cho phép tạo ứng dung phân tán RMI
- Các đối tượng chứa các phương thức có thể được gọi bởi các máy ảo khác được gọi là **đối tượng từ xa** (**remote objects)**

Đối tượng cục bộ và đối tượng từ xa

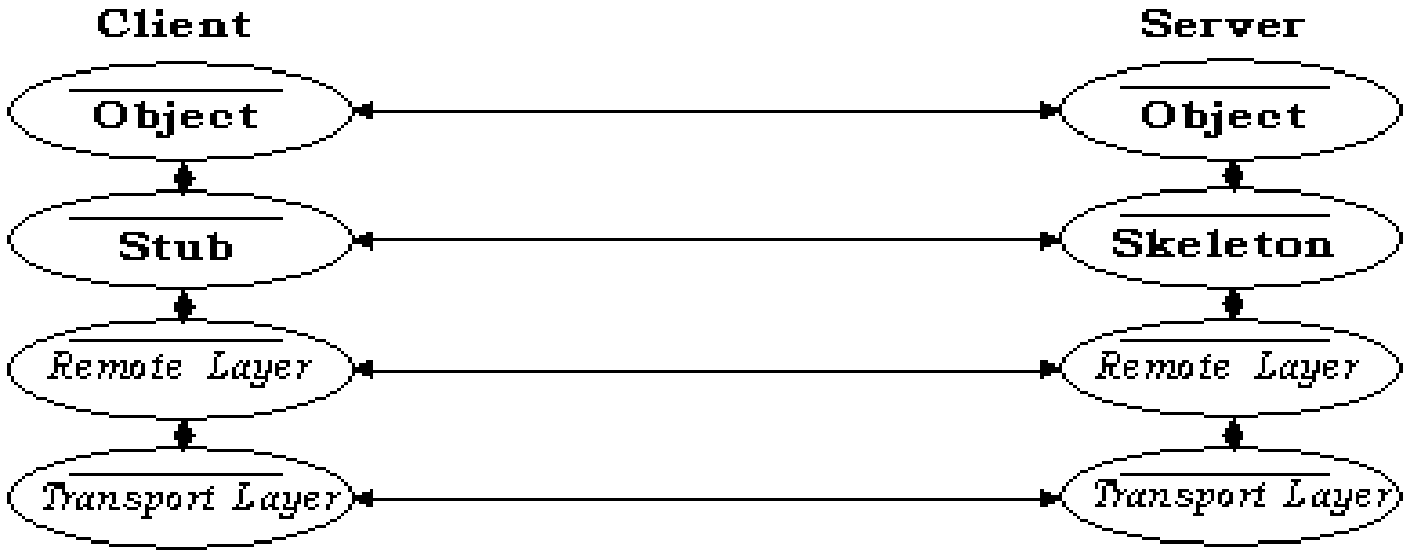
- Đối tượng cục bộ (Local objects) là các đối tượng được thực thi trên một trạm client.
- Đối tượng từ xa (**remote objects)** là các đối tượng được thực thi ở các máy server ở xa.
- RMI sử dụng các stubs và skeletons
- **stub** là một đối tượng cục bộ hoạt động như proxy cục bộ
- **Skeleton** là sự đại diện cho đối tượng từ xa được đặt trên cùng một trạm.
- Stubs và Skeletons truyền thông với nhau thông qua tầng Remote Reference
- Stubs liên lạc với skeleton sử dụng TCP.
- Đối tượng gọi phương thức ở xa gọi là đối tượng client.
- Đối tượng chứa phương thức được gọi từ xa gọi là đối tượng **Server.**
  - Đối tượng client và server không hoạt động trực tiếp với nhau

Cách hoạt động của RMI

- Các server RMI phải đăng ký với dịch vụ tra cứu (lookup service, Remote Registry Server), để cho phép các clients tìm thấy chúng.
- Dịch vụ tra cứu này trong Java là *rmiregistry. Dịch vụ* này hoạt động như một tiến trình tách biệt và cho phép các ứng dụng đăng ký các dịch vụ RMI.
- Sau khi một server đã đăng ký, sẽ chờ các yêu cầu RMI từ các client.



Sự phân lớp của RMI



- Đối tượng client gọi các phương thức của stub
- Stub sử dụng tầng remote để liên lạc với skeleton
- Tầng remote sử dụng tầng transport để thiết lập kết nối giữa client và server

Lập trình RMI

- Lập trình RMI liên quan đến 2 bước:
  - Lập trình RMI trên trạm ớ xa
  - Lập trình RMI trên trạm cục bộ

Các bước lập trình RMI trên trạm ở xa

- **Bước 1**

  Tạo một remote interface:
  - Phải là public và phải extend từ interface tên là Remote
  - Khai báo các hàm sẽ được gọi từ xa. Các hàm phải throw RemoteException
- **Bước 2**

  Tạo một lớp hiện thực remote interface:
  - Lớp này cũng phải extends lớp UnicastRemoteObject để kế thừa các chức năng đối tượng từ xa
  - Lớp này tạo và khởi tạo đối tượng remote object
  - Hiện thực tất cả các methods đã định nghĩa trong remote interface
- Tạo hàm **main( )** thực thi lớp ở xa

Các bước lập trình RMI trên trạm ớ xa (tt)

- **Bước 3**

  Tạo các lớp stub và skeleton sử dụng lệnh **rmic**.

- **Bước 4**

  Chép remote interface và file stub sinh ra đến trạm client.

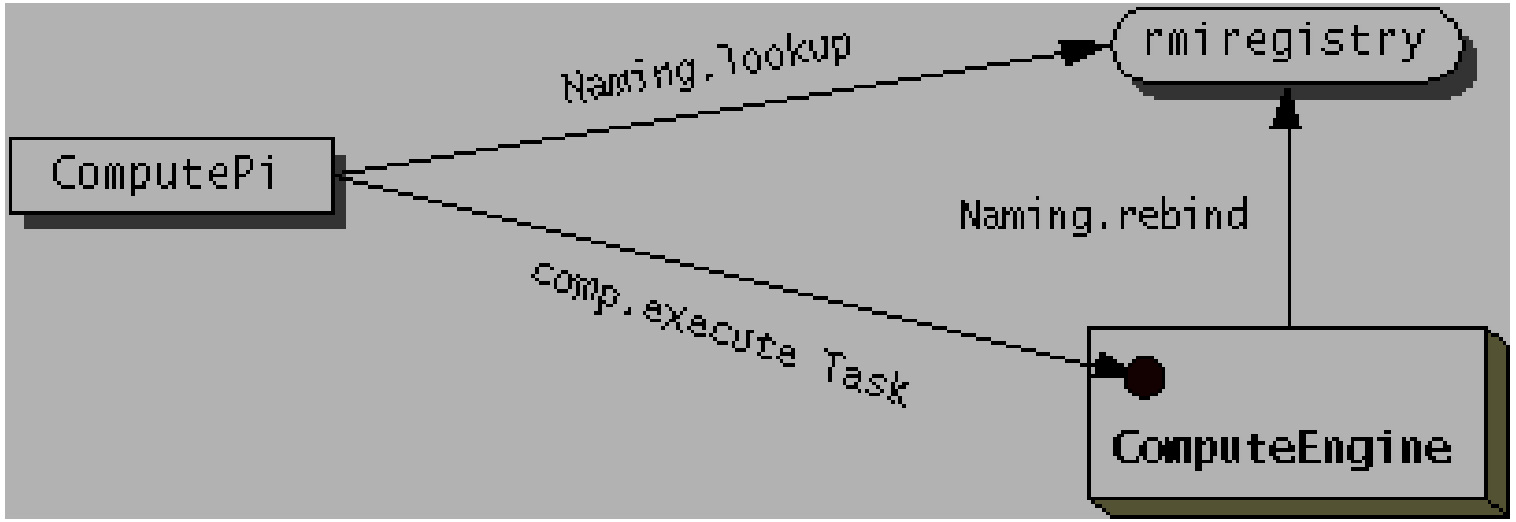
- **Bước 5**

  Khởi động Remote Registry Server sử dụng lệnh **start rmiregistry.**

- **Bước 6**

  Thực thi chương trình dùng lệnh **java** để tạo một đối tượng của lớp. Đối tượng từ xa phải được đăng ký với Remote Registry Server

Lập trình RMI trên trạm cục bộ



- Sau khi RMI được lập trình ở máy từ xa, remote server phải được chạy.
- Tạo một chương trình client để gọi đối tượng từ xa, và hiển thị kết quả

Mã Java: tạo remote interface

```
import java.rmi.*;
public interface RMICalcul extends Remote
{
public int add (int a, int b) throws RemoteException;
public int sub(int a, int b) throws RemoteException;
}
```

Tạo một lớp hiện thực remote interface

```
import java.rmi.*;
public class RMICalculimpl extends UnicastRemoteObject implements RMICalcul
{
    public RMICalculimpl() throws RemoteException
    {
    }
    public int add (int a, int b) throws RemoteException
    {
    return (a+b);
    }
    public int sub(int a, int b) throws RemoteException
    {
    return (a-b);
    }
}
```

Tạo các lớp Stub và Skeleton

- Tại cửa sổ dòng lệnh gõ:
  - rmic RMICalculimpl
  - sẽ có 2 files sinh ra cùng thư mục chứa RMICalculimpl :
    - RMICalculimpl_Stub.class
    - RMICalculimpl_Skeleton.class
  - Chép interface RMICalcul và RMICalculimpl_Stub.class đến máy client.

Tạo một RMI Server

```
import java.rmi.*;
import java.rmi.server.*;
public class CalculServer
{
        public static void main(String args[])
        {
         try
         {
                        // Load the service
           RMICalculimpl CalculService = new RMICalculimpl();
           String registration = "rmi://localhost/RMICalcul";

                        // Register with service so that clients can find us
           Naming.rebind( registration, CalculService);
         }
         catch (Exception e)
         {
           System.err.println ("Error - " + e);
         }
}
```

Tạo một RMI Client

```
public class CalculClient
{
        public static void main(String args[])
        {
         try
          {
          String registration = "rmi://RegistryServerName/RMICalcul";
                // Lookup the service in the registry, and obtain a remote service
          Remote remoteService = Naming.lookup (registration );
                // Cast to a RMICalcul interface
          RMICalcul calculService = (RMICalcul) remoteService;
                // call remote  method :add (..), sub
          System.out.println ("sum = "+calculService.add(3,5));
          System.out.println ("sub = "+calculService.sub(3,5));
         }
        catch (Exception e)
         {
           System.out.println ("Error - " + e);
         }
        }
}
```

Chạy chương trình CalculServer, CalculClient

1. Tại cửa sổ dòng lệnh gõ: **start** rmiregistry
2. Dịch RMICalcul, RMICalculimpl, CalculServer và chạy CalculServer
3. Dịch và chạy CalculClient

Lập trình CORBA

SV tự tìm hiểu

Bài tập gọi hàm và thủ tục từ xa

- Xây dựng chương trình chat bằng RMI

Thank you listening