Java RMI

thangld@uit.edu.vn
Khoa Mạng máy tính và Truyền thông
Đại học Công nghệ Thông tin

Nội dung

- Tổng quan
- Khái niệm gọi hàm từ xa
- Giới thiệu RMI
- Cài Đặt Hệ thống dịch vụ
- Truyền tham số trong RMI
- Load RMI Stubs từ xa
- Dynamic Class Loading

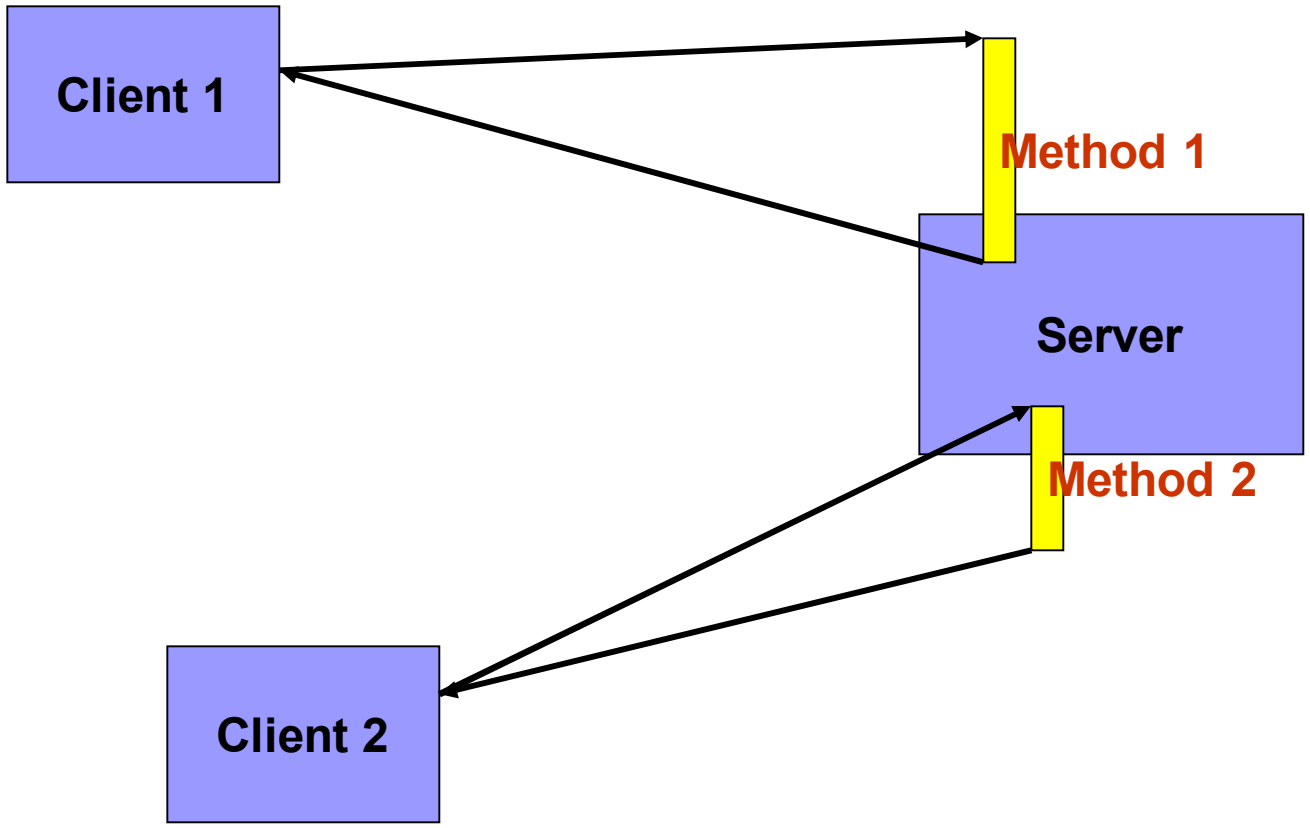
Tổng quan [1]

- Triển khai phương thức/hàm cục bộ
  - Thông thường, trong ứng dụng, sau khi khởi tạo lớp thường, chúng ta có thể triển khai các phương thức trên lớp đó
  - Các lớp thường trong ứng dụng phân tải trên cùng một không gian bộ nhớ và thực thi trên cùng một máy tính

Tổng quan [2]

- Triệu gọi phương thức/hàm từ xa
  - Một tương đương thực thi trên máy A gọi thực hiện các phương thức của các đối tượng Lang thực thi trên máy B hoặc các máy tính khác
  - Phương thức Lấy triệu gọi từ xa, chủy trên máy tính ở xa; do Ló, sử dụng các tài nguyên của máy tính ở xa
  - Đang dụng trên máy A có thWphải chờ phương thức thực hiện xong và nhận kết quả trả vQtừ phương thức Ló

Triệu gọi hàm từ xa



Kỹ thuật gọi hàm từ xa

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

RPC

- K: thuật triển gọi hàm/thủ tục trên các máy tính ở xa
- Cho phép triển khai trên các hệ thống máy tính khác nhau
- Có thể cài đặt bằng nhiều ngôn ngữ lập trình khác nhau
- Hệ ng thủ tục

CORBA

- ṇ Hե tΓ truy vЬ̈n Ľ́ i tể ng trên các máy tính ˂ xa
- S˙ d̤ ng ngôn ng̤ Ľ̉ nh nghọa giao tiЛp theo kiẆu Ľ́ i tể ng (Interface Definition Languages)
- ṇ ˛ c lԁp v˛ i h̉ th́ ng máy tính và ngôn ng̤ lԁp trình
- Hẹ ng Ľ́ i tể ng

RMI [1]

- K: thuật triển gọi phương thức từ các đối tượng trên các máy tính xa
- Nền tảng tính toán phân tán trên Java
- Hệ đối tượng
- RMI tích hợp mô hình phân tán đối tượng vào ngôn ngữ lập trình Java

RMI [2]

- RMI cho phép m¸ t chƠng trình nƠm gi tham chiЉu ĽЉn m¸ t Ľ´ i tƠ ng trên m¸ t h th´ ng ˂ xa Ľ ng th i cho phép tri u g i các phƠng th c c a Ľ´ i tƠ ng Ľó
- Các RMI client giao tiЉp v i các Ľ´ i tƠ ng ˂ xa thông qua *published interface* c a Ľ´ i tƠ ng
- RMI cung cЬp d ch v Ľ nh v Ľ´ i tƠ ng t xa *non-persistent*

RMI [3]

- KiЛ]n trúc Client-Server
  - Server nдm giị Ľ´ i tЄ̉ ng phụ c vụ
  - Client nдm giị mộ t RMI Stub dùng ĽЖtruy cдp ĽЛ]n Ľ´ i tЄ̉ ng ` phía Server

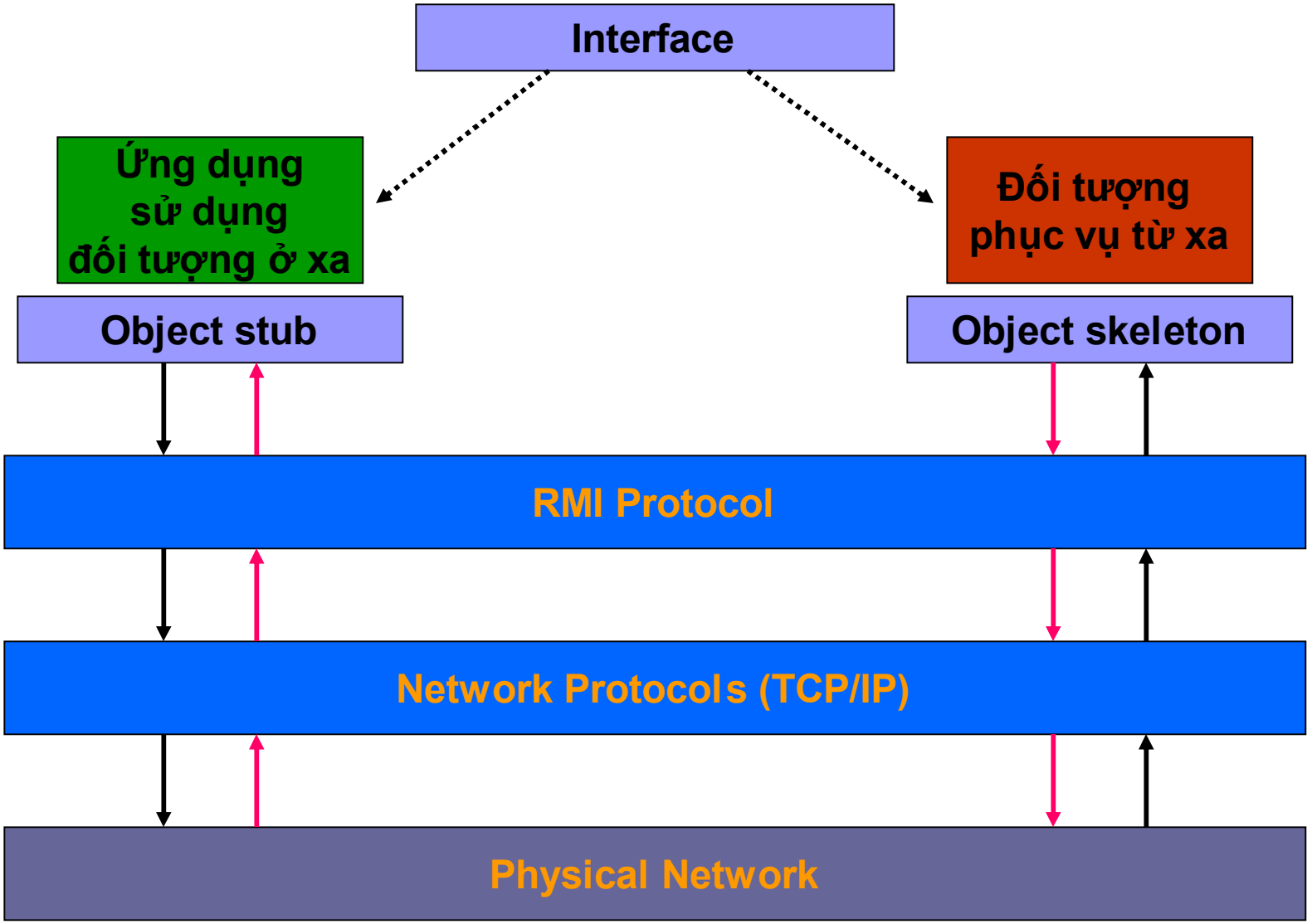
RMI Stubs

- ng óng vai trò là m t proxy c a Ľ i t ng xa trên h th ng client
- Cung cЬp ch ch Ľóng gói các Ľ i g i ph ng th c t xa và g i Ľ n h th ng server

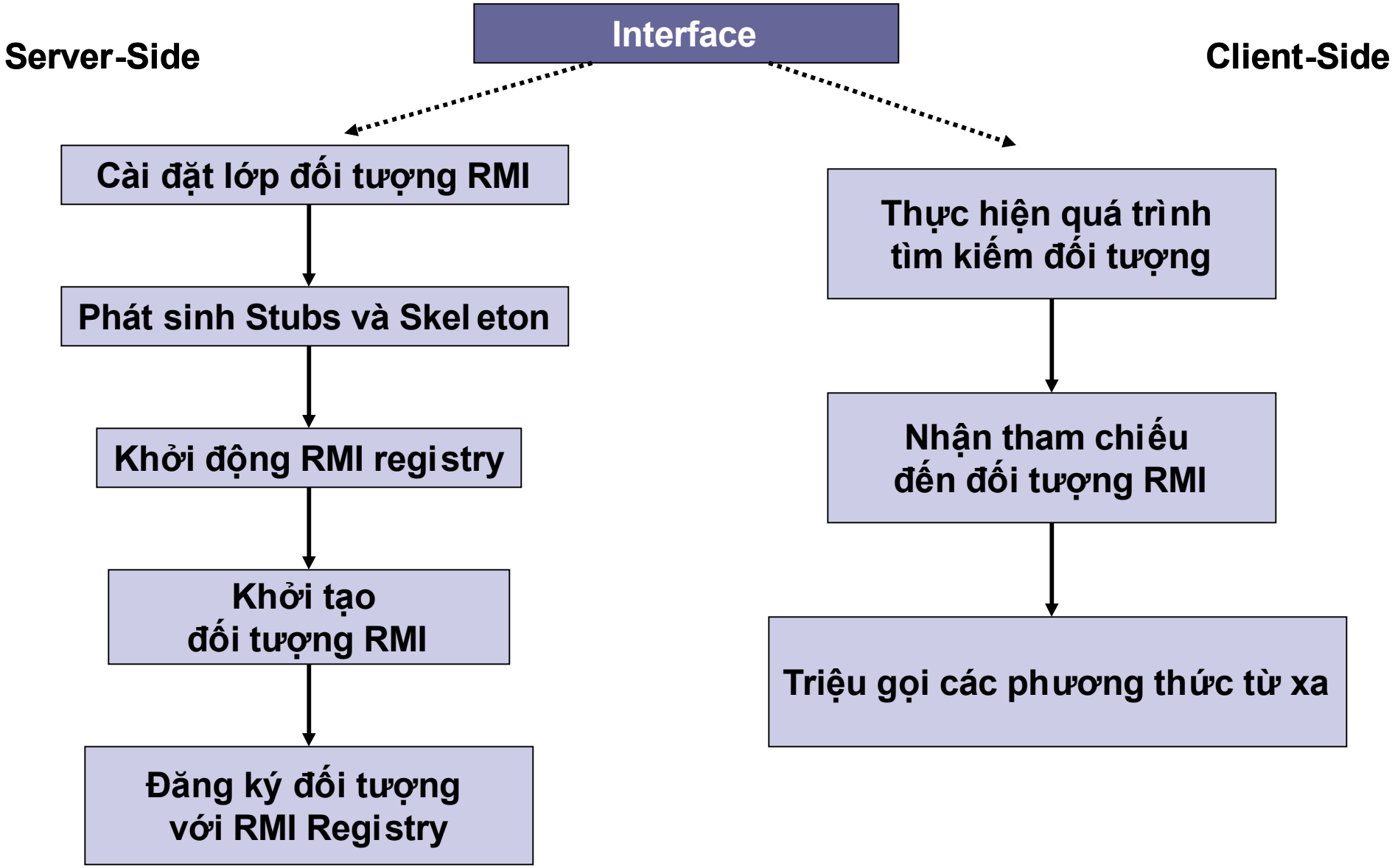
RMI Skeletons

- T◆ng tác tr c ti/p v i các RMI Stubs trên h th ng client
- Nh◄n các l i tri u g i ph◆ng th c t xa, trích xuЫt d li u và th c hi n tri u g i L i t◆ng ph c v

Kiến trúc RMI



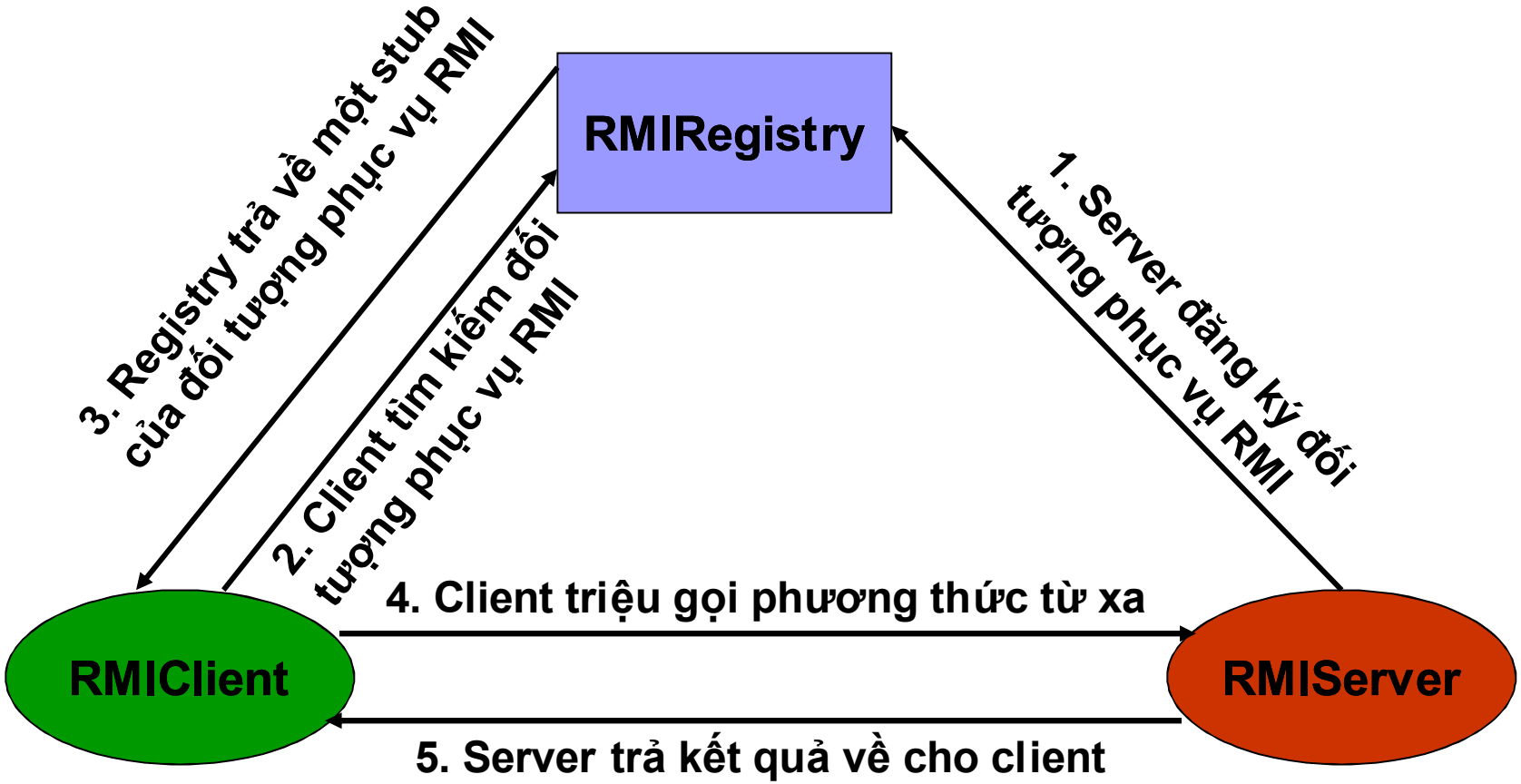
Qui trình RMI



Ứng dụng RMI

- Cài LHtm t ng d ng th c hi n các phép toán ch bFn (c ng, tr , nhân, chia) t xa
- Các thành phFn:
  - n i t ng RMI
  - Ch-ơng trình Server
  - Ch-ơng trình Client

Mô hình ứng dụng RMI



Đối tượng RMI

- Interface
  - *Khai báo tất cả các phương thức cần "export" để phục vụ từ xa*
- Implementation
  - Một class cài đặt interface trên phía server
- RMI Stub
- RMI Skeleton

Interface

- PhЧm vi truy xuЫt: *public*
- DЖn xuЫt tҿ interface Remote
- Khai báo các phѲӨng thѿc phѾc vѾ tҿ xa. TЫt cГ các phѲӨng thѿc này phГi khai báo *throws RemoteExcepti on*

Ví dụ

```
import java.rmi.*;

public interface iCalculator extends Remote {
   long Add(int x, int y) throws RemoteException;
   long Sub(int x, int y) throws RemoteException;
   long Div(int x, int y) throws RemoteException;
   long Mul(int x, int y) throws RemoteException;
}
```

Implementation

- Class phải được thiết kế cho phép truy cập từ xa:
  - Thi công interface đã tạo ở trên (Remote)
  - Thường kế thừa class giao tiếp RMI (UnicastRemoteObject)
- Thường chỉ hỗ trợ gọi phương thức từ xa
- Cài đặt các phương thức RMI
- Tất cả các phương thức đều phải khai báo *throws RemoteException*

Ví dụ

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;

public class CalculatorImpl
             extends UnicastRemoteObject
                              implements iCalculator {
  public CalculatorImpl() throws RemoteException {
  }
  public long Add(int x, int y)
                              throws RemoteException {
      return x + y;
  }
  ...
}
```

Stub & Skeleton

- ḥ Ể c tЧo ra t̤ chƠng trình tỉ n ích rmic
  - *rmic <tên_class_implemet ation>*
- JDK1.5+, v̤ i RMI v1.2
  - *rmic* cḧ tЧo ra Stub

Server

- CFn có RMI Registry trE c khi th c thi chEhng trình Server
  - Th c thi chEhng trình *rmiregistry.exe*
  - Kh i tÜo RMI Registry c c b trong chEhng trình Server
- Kh i tÜo L´ i tE ng RMI
- Export L´ i tE ng nJu cFn thiJt
- ņŁng ký L´ i tE ng v i RMI Registry và ch ph c v các yêu cFu RMI

Ví dụ [1]

```
import java.io.*;
import java.rmi.*;
import java.rmi.registry.LocateRegistry;

public class CalculatorServer {
  public static void main(String arg[])
                                  throws Exception {
      Registry reg =
            LocateRegistry.createRegistry(1099);
      CalculatorImpl cal = new CalculatorImpl();
      iCalculator calRef = (iCalculator)
            UnicastRemoteObject.exportObject(cal);
      reg.bind("Calculator", calRef);
      System.out.println("RMI Object ready...");
```

Ví dụ [2]

```
        BufferedReader rdr = new BufferedReader(
                             new InputStreamReader(
                                 System.in));
        while(true) {
          System.out.println("Type EXIT: shutdown");
          if ("EXIT".compareIgnoreCase(
                                 rdr.readLine()) == 0)
              break;
        }
        reg.unbind("Calculator");
        UnicastRemoteObject.unexportObject(cal);
    }
}
```

Bind/Unbind

- Bind và unbind Lí tế ng RMI

#1

```
Naming.bind("rmi://localhost:1099/RC", obj);
Naming.unbind("rmi://localhost:1099/RC");
```

#2

```
[Registry].bind("RC", obj);
[Registry].unbind("RC");
```

Client

- Truy vấn RMI Registry để tìm kiếm đối tượng RMI
- Triệu gọi phương thức từ xa
- Nhận và xử lý kết quả trả về (nếu có)

Ví dụ

```
import java.rmi.*;

public class CalculatorClient {
  public static void main(String arg[])
                                  throws Exception {
      iCalculator objRef = (iCalculator)
            Naming.lookup("rmi:///Calculator");
      long s = objRef.Add(5, 7);
      System.out.println("Result = " + s);
  }
}
```

Lookup

- Lookup L´ i tế ng RMI

#1

```
Naming.lookup(“rmi://localhost:1099/RC”);
```

#2

```
Registry reg = getRegistry(“localhost”, 1099);
reg.lookup(“RC”);
```

Truyền tham số trên JVM

- KiWu dụ liệu chỉ số: truyQn tham trị (giá trị của tham số Lể c sao chép)
- Các Lối tể ng Lể c cBp phát trên vùng nhớ Heap.
- ợi tể ng: truyQn bằng tham chiJu.

Tham trị trong RMI

- KiWu dụ liễu chỉ : giá trị của tham số Lể c sao chép từ client-JVM sang server -JVM và ngể c lỵi.
- ặ i tể ng
  - JVMs không chia sẻ vùng nhớ Heap
  - Thi công interface Serializable
  - Toàn bộ nội dung/giá trị của Lặ i tể ng Lể c sao chép và serialize LWtruyQn Li

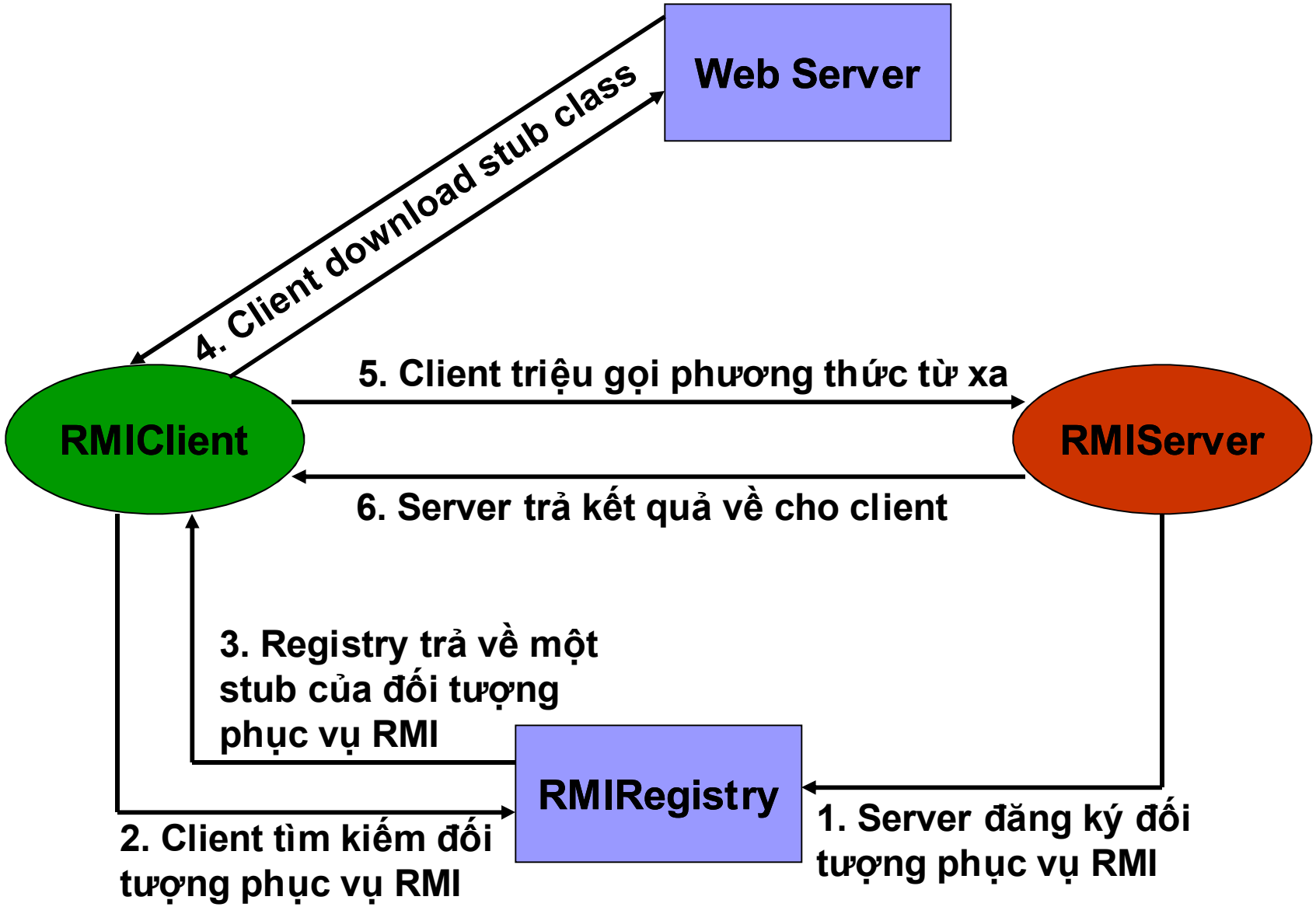
Tham chiếu trong RMI [1]

- Áp dụng cho lớp đối tượng
- Tham chiếu lớn lớp đối tượng cục bộ truy vấn qua RMI Registry
- Tham chiếu lớn lớp đối tượng cục bộ truyền vào trong phương thức của đối tượng tham số
- Tham chiếu lớn lớp đối tượng cục bộ trả về đối tượng kết quả của phương thức

Tham chiếu trong RMI [2]

- Đối tượng
  - Thi công interface Remote
  - Được export trước khi truyền lại
  - Phải biên dịch Stub và Skeleton cho Đối tượng

Loading Stub từ xa [1]



Loading Stub từ xa [2]

- Thiết lập security

```
System.setSecurityManager(
                new RMISecurityManager());
```

- Java command line parameter
  - `-Djava.rmi.server.codebase=`
    `http://web.rmi.com/export/`
  - `-Djava.rmi.server.codebase=`
    `"http:// web1.rmi.com/mylib1.jar`
    `http://web2.rmi.com/mylib2.jar"`

Loading Stub từ xa [3]

- Java command line parameter
  - `-Djava.security.policy=java.policy`
- File java.policy

```
grant {
    permission java.security.AllPermission;
};
```

Dynamic Class Loading

- Load class

```
  Class loadedClass =  null;
/*1*/ loadedClass = RMIClassLoader.loadClass(
                    "<URL>", "<Class_Name>");
/*2*/ loadedClass = Class.forName("<Class_Name>");
/*3*/ loadedClass = ClassLoader
                    .getSystemClassLoader()
                         .loadClass("<Class_Name>");
  loadedClass.newInstance();
```

<URL>:

```
file:///D:/Classes/export/
http://web.rmi.com/...
```

Q&A