Chương 4
SOCKET

Mục đích

Chương này nhằm giới thiệu về cách thức xây dựng ứng dụng Client-Server trên mạng TCP/IP theo cả hai chế độ Có nối kết (TCP) và Không nối kết (UDP).

Yêu cầu

Sau khi hoàn tất chương này, bạn có thể:

- Giải thích được Socket là gì, vai trò của số hiệu cổng (Port) và địa chỉ IP trong cơ chế Socket.
- Phân biệt được sự khác biệt của hai loại Protocol TCP và UDP.
- Trình bày được các bước xây dựng một chương trình Client-Server sử dụng Socket làm phương tiện giao tiếp trong cả hai chế độ TCP và UDP.
- Liệt kê các lớp hỗ trợ lập trình Socket của Java.
- Xây dựng được các chương trình Client sử dụng Socket ở chế độ có nối kết bằng ngôn ngữ Java.

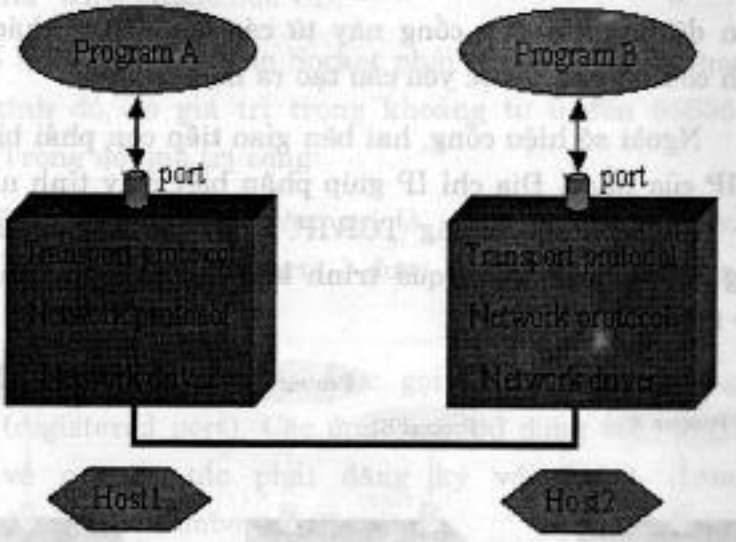
- Xây dựng được các chương trình Server sử dụng Socket ở chế độ có nối kết phục vụ tuần tự và phục vụ song song bằng ngôn ngữ Java.
- Xây dựng được các chương trình Client-Server sử dụng Socket ở chế độ không nối kết bằng ngôn ngữ Java.
- Xây dựng được các chương trình Client-Server liên lạc với nhau theo dạng Multicast (liên lạc nhóm) bằng ngôn ngữ Java.
- Tự xây dựng được các Protocol mới cho ứng dụng TCP/IP của mình.

4.1 GIỚI THIỆU VỀ SOCKET

4.1.1 Giới thiệu

Dưới góc độ người lập trình, Socket là một giao diện lập trình ứng dụng (API-Application Programming Interface). Nó được giới thiệu lần đầu tiên trong ấn bản UNIX - BSD 4.2. dưới dạng các hàm hệ thống theo cú pháp ngôn ngữ C (socket(), bind(), connect(), send(), receive(), read(), write(), close() ,...). Ngày nay, Socket được hỗ trợ trong hầu hết các hệ điều hành như MS Windows, Linux và được sử dụng trong nhiều ngôn ngữ lập trình khác nhau: như C, C++, Java, Visual Basic, Visual C++, . . .

Socket cho phép thiết lập các kênh giao tiếp mà hai đầu kênh được đánh dấu bởi hai cổng (port). Thông qua các cổng này một quá trình có thể nhận và gởi dữ liệu với các quá trình khác.



Hình 4.1 – Mô hình Socket

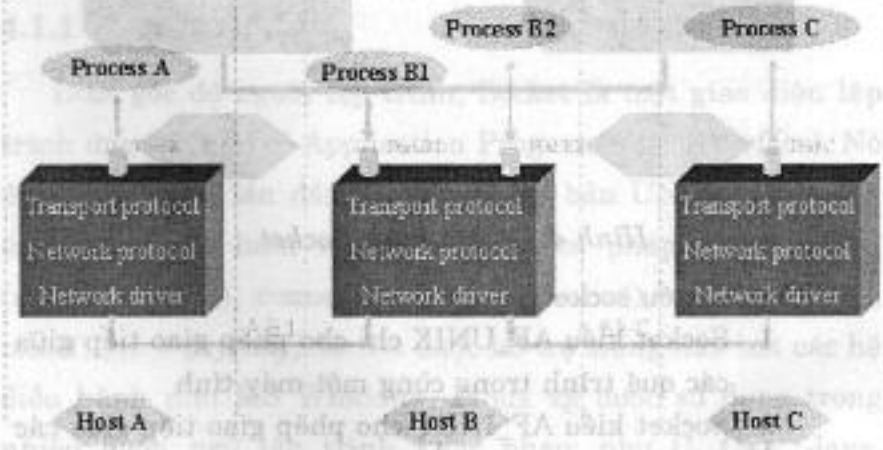
Có hai kiểu socket:

1. Socket kiểu AF_UNIX chỉ cho phép giao tiếp giữa các quá trình trong cùng một máy tính
2. Socket kiểu AF_INET cho phép giao tiếp giữa các quá trình trên những máy tính khác nhau trên mạng TCP/IP.

4.1.2 Số hiệu cổng (Port Number) của socket

Để có thể thực hiện các cuộc giao tiếp, một trong hai quá trình phải công bố số hiệu cổng của socket mà mình sử dụng. Mỗi cổng giao tiếp thể hiện một địa chỉ xác định trong hệ thống. Khi quá trình được gán một số hiệu cổng, nó có thể nhận dữ liệu gởi đến cổng này từ các quá trình khác. Quá trình còn lại cũng được yêu cầu tạo ra một socket.

Ngoài số hiệu cổng, hai bên giao tiếp còn phải biết địa chỉ IP của nhau. Địa chỉ IP giúp phân biệt máy tính này với máy tính kia trên mạng TCP/IP. Trong khi số hiệu cổng dùng để phân biệt các quá trình khác nhau trên cùng một máy tính.



Hình 4.2 – Cổng trong Socket

Trong hình trên, địa chỉ của quá trình B1 được xác định bằng 2 thông tin: (Host B, Port B1):

Địa chỉ máy tính có thể là địa chỉ IP (dạng 203.162.36.149) hay là địa chỉ theo dạng tên miền (chẳng hạn như www.cit.ctu.edu.vn).

Số hiệu cổng gán cho Socket phải duy nhất trên phạm vi máy tính đó, có giá trị trong khoảng từ 0 đến 65535 (16 bits). Trong đó, giá trị cổng:

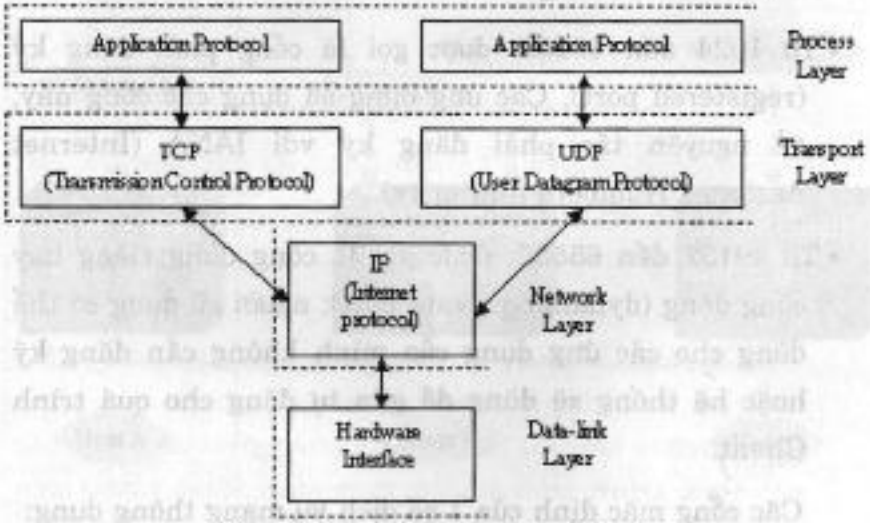
- Từ 0 đến 1023: được gọi là cổng hệ thống (common hay well-known ports) được dành riêng cho các quá trình của hệ thống.
- Từ 1024 đến 49151: được gọi là cổng phải đăng ký (registered port). Các ứng dụng sử dụng các cổng này, về nguyên tắc phải đăng ký với IANA (Internet Assigned Numbers Authority).
- Từ 49152 đến 65535: được gọi là cổng dùng riêng hay cổng động (dynamic-private port): người sử dụng có thể dùng cho các ứng dụng của mình không cần đăng ký hoặc hệ thống sẽ dùng để gán tự động cho quá trình Client.

Các cổng mặc định của 1 số dịch vụ mạng thông dụng:

Số hiệu cổng	Quá trình hệ thống
7	Dịch vụ Echo
21	Dịch vụ FTP
23	Dịch vụ Telnet
25	Dịch vụ E-mail (SMTP)
80	Dịch vụ Web (HTTP)
110	Dịch vụ E-mail (POP)

4.1.3 Các chế độ giao tiếp

Xét kiến trúc của hệ thống mạng TCP/IP



Hình 4.3 – Bộ giao thức TCP/IP

Tầng vận chuyển giúp chuyển tiếp các thông điệp giữa các chương trình ứng dụng với nhau. Nó có thể hoạt động theo hai chế độ:

- Giao tiếp có nối kết, nếu sử dụng giao thức TCP
- Hoặc giao tiếp không nối kết, nếu sử dụng giao thức UDP

Socket là giao diện giữa chương trình ứng dụng với tầng vận chuyển. Nó cho phép ta chọn giao thức sử dụng ở tầng vận chuyển là TCP hay UDP cho chương trình ứng dụng của mình.

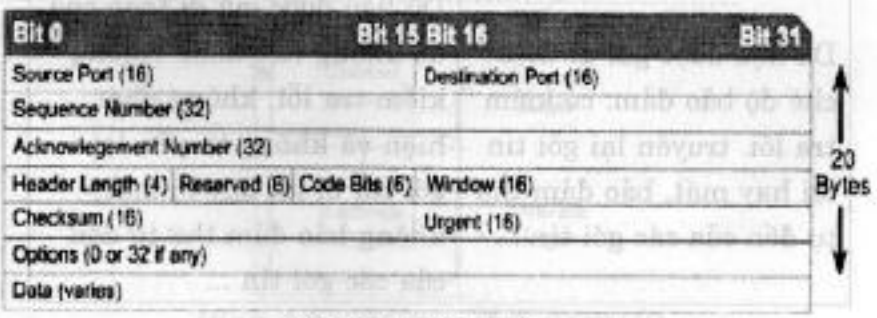
Bảng sau so sánh sự khác biệt giữa hai chế độ giao tiếp có nối kết và không nối kết:

Có nối kết (TCP)	Không nối kết (UDP)
Tồn tại kênh giao tiếp ảo giữa 2 quá trình	Không tồn tại kênh giao tiếp ảo giữa 2 quá trình
Dữ liệu được gửi đi theo chế độ bảo đảm: có kiểm tra lỗi. truyền lại gói tin lỗi hay mất, bảo đảm thứ tự đến của các gói tin ...	Dữ liệu được gửi đi theo chế độ không bảo đảm: Không kiểm tra lỗi, không phát hiện và không truyền lại gói tin bị lỗi hay bị mất, không bảo đảm thứ tự đến của các gói tin ...

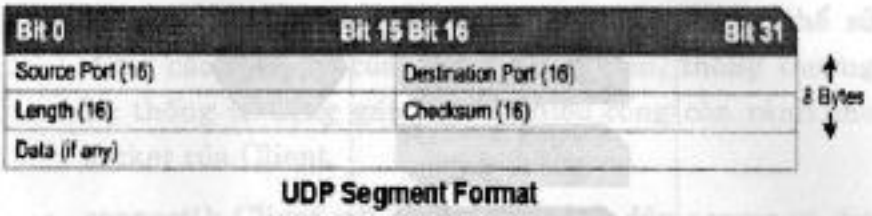
Dữ liệu chính xác, Tốc độ truyền chậm	Dữ liệu không chính xác, tốc độ truyền nhanh
Thích hợp cho các ứng dụng cần độ chính xác cao: truyền file, thông tin điều khiển ...	Thích hợp cho các ứng dụng cần tốc độ, không cần chính xác cao: truyền âm thanh, hình ảnh ...

Cả 2 giao thức TCP và UDP đều phân dữ liệu ra thành các gói tin. Tuy nhiên TCP có thêm vào những tiêu đề (Header) vào trong gói tin để cho phép truyền lại những gói tin thất lạc và tập hợp các gói tin lại theo đúng thứ tự. UDP không cung cấp tính năng này, nếu 1 gói tin bị thất lạc hoặc bị lỗi, nó sẽ không được truyền lại, và thứ tự đến đích của các gói tin cũng không được bảo đảm giống như thứ tự lúc nó được gửi đi. Bù lại, về tốc độ, UDP sẽ truyền nhanh xấp xỉ gấp 3 lần TCP.

Dưới đây là các cấu trúc của 2 dạng đơn vị dữ liệu sử dụng trong TCP và UDP.



TCP Segment Format



Hình 4.4 – Cấu trúc các segment của TCP và UDP

Về cơ bản, so với gói tin UDP, gói tin TCP sẽ có thêm các thành phần quan trọng như: số thứ tự gói tin (sequence number) và số thứ tự báo nhận (acknowlegement number).

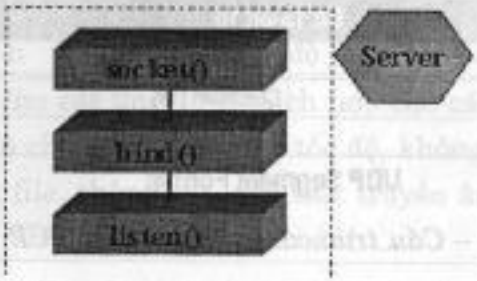
4.2 XÂY DỰNG ỨNG DỤNG CLIENT-SERVER VỚI SOCKET

Socket là phương tiện hiệu quả để xây dựng các ứng dụng theo kiến trúc Client-Server. Các ứng dụng trên mạng Internet như Web, Email, FTP là các ví dụ điển hình.

Phần này trình bày các bước cơ bản trong việc xây dựng các ứng dụng Client-Server sử dụng Socket làm phương tiện giao tiếp theo cả hai chế độ: Có nối kết và không nối kết.

4.2.1 Mô hình Client-Server sử dụng Socket ở chế độ có nối kết (TCP)

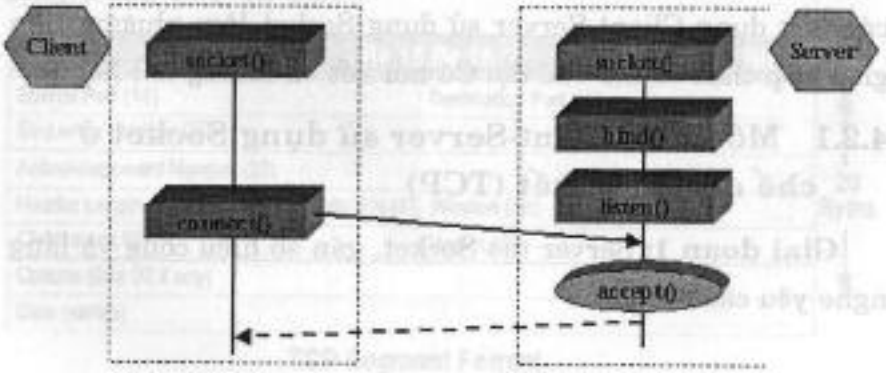
Giai đoạn 1: Server tạo Socket, gán số hiệu cổng và lắng nghe yêu cầu nối kết.



- socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
- bind(): Server yêu cầu gán số hiệu cổng (port) cho socket.
- listen(): Server lắng nghe các yêu cầu nối kết từ các client trên cổng đã được gán.

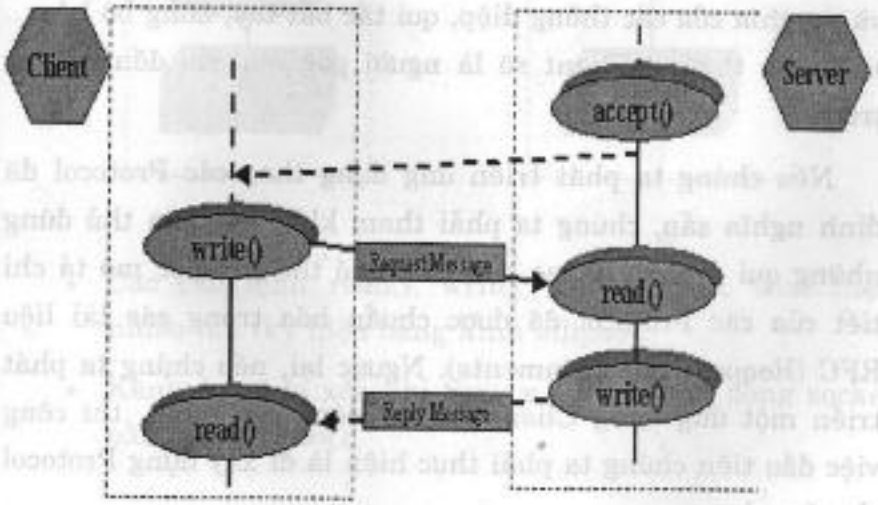
Server sẵn sàng phục vụ Client.

Giai đoạn 2: Client tạo Socket, yêu cầu thiết lập một nối kết với Server.



- socket(): Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho socket của Client.
- connect(): Client gởi yêu cầu nối kết đến server có địa chỉ IP và Port xác định.
- accept(): Server chấp nhận nối kết của client, khi đó một kênh giao tiếp ảo được hình thành, Client và server có thể trao đổi thông tin với nhau thông qua kênh ảo này.

Giai đoạn 3: Trao đổi thông tin giữa Client và Server.

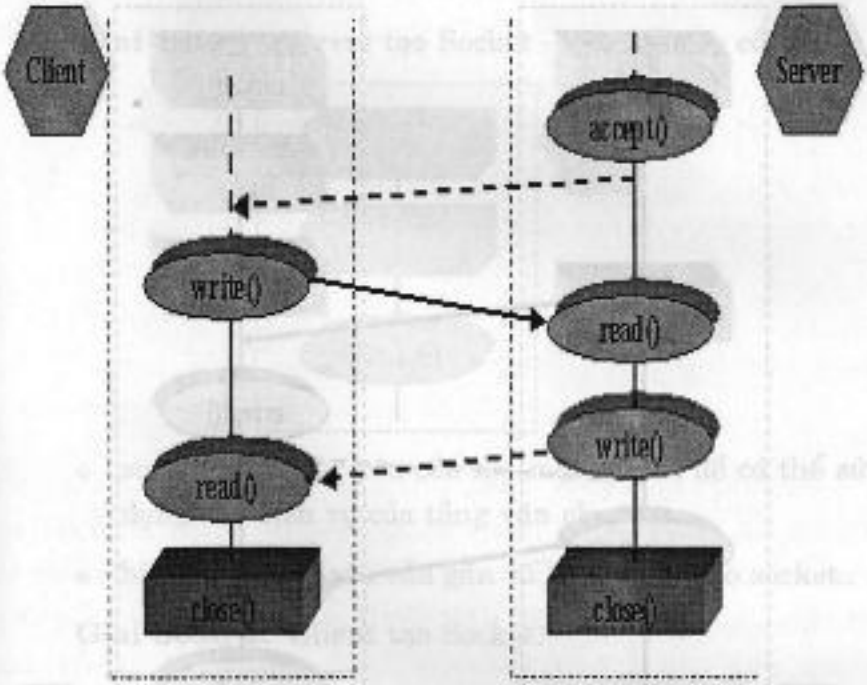


- Sau khi chấp nhận yêu cầu nối kết, thông thường server thực hiện lệnh read() và nghẽn cho đến khi có thông điệp yêu cầu (Request Message) từ client gởi đến.
- Server phân tích và thực thi yêu cầu. Kết quả sẽ được gởi về client bằng lệnh write().
- Sau khi gởi yêu cầu bằng lệnh write(), client chờ nhận thông điệp kết quả (Reply Message) từ server bằng lệnh read().

Trong giai đoạn này, việc trao đổi thông tin giữa Client và Server phải tuân thủ giao thức của ứng dụng (Dạng thức và ý nghĩa của các thông điệp, qui tắc bắt tay, đồng bộ hóa,...). Thông thường Client sẽ là người gởi yêu cầu đến Server trước.

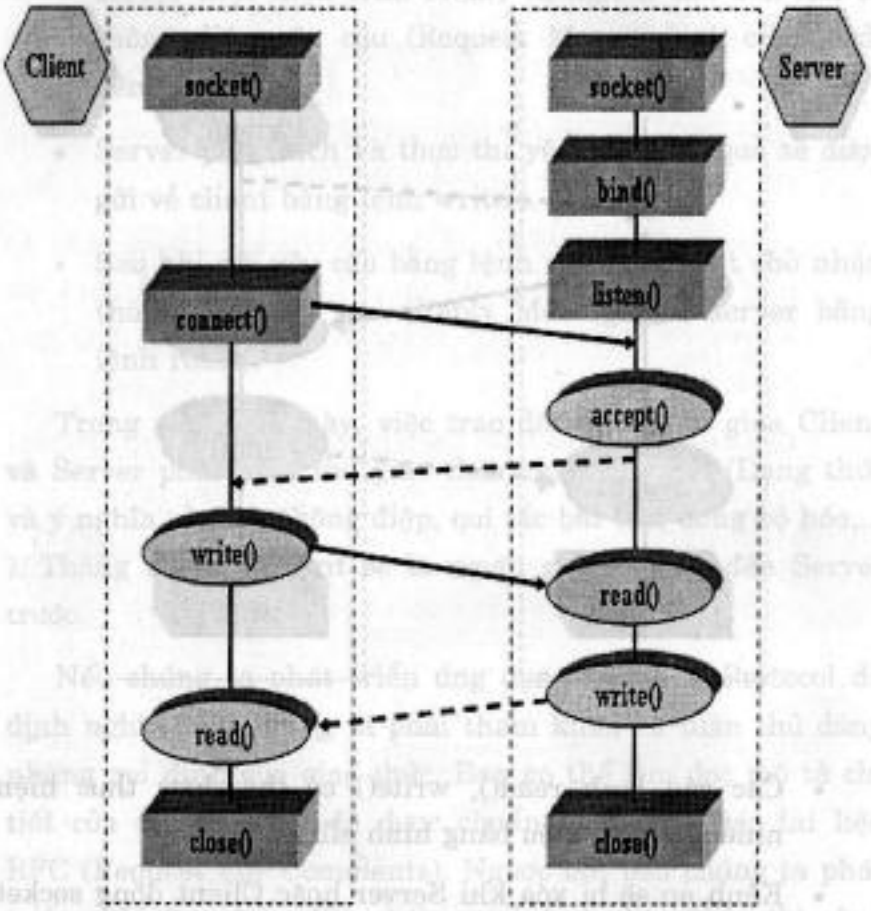
Nếu chúng ta phát triển ứng dụng theo các Protocol đã định nghĩa sẵn, chúng ta phải tham khảo và tuân thủ đúng những qui định của giao thức. Bạn có thể tìm đọc mô tả chi tiết của các Protocol đã được chuẩn hóa trong các tài liệu RFC (Request For Comments). Ngược lại, nếu chúng ta phát triển một ứng dụng Client-Server riêng của mình, thì công việc đầu tiên chúng ta phải thực hiện là đi xây dựng Protocol cho ứng dụng.

Giai đoạn 4: Kết thúc phiên làm việc.



- Các câu lệnh read(), write() có thể được thực hiện nhiều lần (ký hiệu bằng hình ellipse).
- Kênh ảo sẽ bị xóa khi Server hoặc Client đóng socket bằng lệnh close().

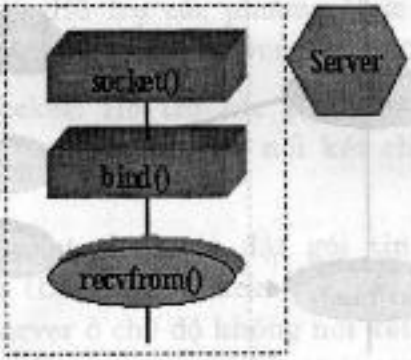
Như vậy toàn bộ tiến trình diễn ra như sau:



Hình 4.5 – Mô hình Client-Server sử dụng Socket TCP

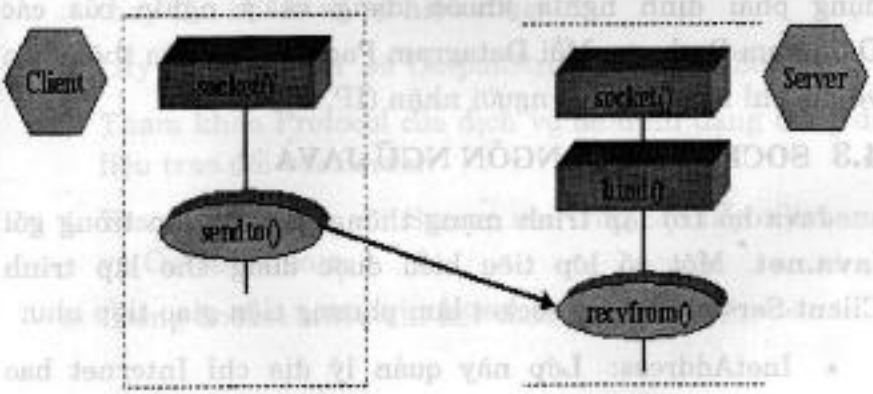
4.2.2 Mô hình Client-Server sử dụng Socket ở chế độ không nối kết (UDP)

Giai đoạn 1: Server tạo Socket - gán số hiệu cổng.

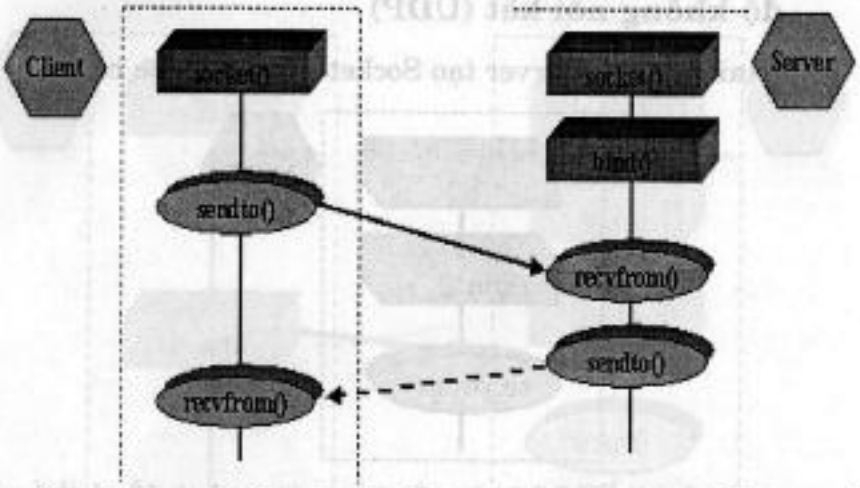


- socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
- bind(): Server yêu cầu gán số hiệu cổng cho socket..

Giai đoạn 2: Client tạo Socket.



Giai đoạn 3: Trao đổi thông tin giữa Client và Server.



Sau khi tạo Socket xong, Client và Server có thể trao đổi thông tin qua lại với nhau thông qua hai hàm sendto() và recvfrom(). Đơn vị dữ liệu trao đổi giữa Client và Server là các **Datagram Package** (Gói tin thư tín). Protocol của ứng dụng phải định nghĩa khuôn dạng và ý nghĩa của các Datagram Package. Mỗi Datagram Package có chứa thông tin về địa chỉ người gởi và người nhận (IP, Port).

4.3 SOCKET DƯỚI NGÔN NGỮ JAVA

Java hỗ trợ lập trình mạng thông qua các lớp trong gói **java.net**. Một số lớp tiêu biểu được dùng cho lập trình Client-Server sử dụng socket làm phương tiện giao tiếp như:

- InetAddress: Lớp này quản lý địa chỉ Internet bao

gồm địa chỉ IP và tên máy tính.

- Socket: Hỗ trợ các phương thức liên quan đến Socket cho chương trình Client ở chế độ có nối kết.
- ServerSocket: Hỗ trợ các phương thức liên quan đến Socket cho chương trình Server ở chế độ có nối kết.
- DatagramSocket: Hỗ trợ các phương thức liên quan đến Socket ở chế độ không nối kết cho cả Client và Server.
- DatagramPacket: Lớp cài đặt gói tin dạng thư tín người dùng (Datagram Packet) trong giao tiếp giữa Client và Server ở chế độ không nối kết.

4.3.1 Xây dựng chương trình Client ở chế độ có nối kết

Các bước tổng quát:

1. Mở một socket nối kết đến server đã biết địa chỉ IP (hay tên miền) và số hiệu cổng.
2. Lấy InputStream và OutputStream gán với Socket.
3. Tham khảo Protocol của dịch vụ để định dạng đúng dữ liệu trao đổi với Server.
4. Trao đổi dữ liệu với Server nhờ vào các InputStream và OutputStream.
5. Đóng Socket trước khi kết thúc chương trình.

4.3.1.1 Lớp java.net.Socket

Lớp Socket hỗ trợ các phương thức cần thiết để xây dựng các chương trình client sử dụng socket ở chế độ có nối kết. Dưới đây là một số phương thức thường dùng để xây dựng Client:

public Socket(String HostName, int PortNumber) throws IOException

Phương thức này dùng để nối kết đến một server có tên là HostName, cổng là PortNumber. Nếu nối kết thành công, một kênh ảo sẽ được hình thành giữa Client và Server.

- HostName: Địa chỉ IP hoặc tên logic theo dạng tên miền.
- PortNumber: có giá trị từ 0 ..65535

Ví dụ: Mở socket và nối kết đến Web Server của khoa Công Nghệ Thông Tin, Đại Học Cần Thơ:

```
Socket s = new Socket("www.cit.ctu.edu.vn",80);
Hoặc: Socket s = new
Socket("203.162.36.149",80);
```

public InputStream getInputStream()

Phương thức này trả về InputStream nối với Socket. Chương trình Client dùng InputStream này để nhận dữ liệu từ Server gởi về.

Ví dụ: Lấy InputStream của Socket s:

```
InputStream is = s.getInputStream();
```

public OutputStream getOutputStream()

Phương thức này trả về OutputStream nối với Socket. Chương trình Client dùng OutputStream này để gởi dữ liệu cho Server.

Ví dụ: Lấy OutputStream của Socket s:

```
OutputStream os = s.getOutputStream();
```

public close()

Phương thức này sẽ đóng Socket lại, giải phóng kênh ảo, xóa nối kết giữa Client và Server.

Ví dụ: Đóng Socket s:

```
s.close();
```

Một số các phương thức hỗ trợ khác:

- **public InetAddress getInetAddress()**:lấy địa chỉ của máy tính đang nối kết (ở xa)
- **public int getPort()**: lấy cổng của máy tính đang nối kết (ở xa)
- **public InetAddress getLocalAddress()**: lấy địa chỉ cục bộ tại máy tính đang tạo Socket.

- **public int getLocalPort():** lấy cổng cục bộ tại máy tính đang tạo Socket.
- **public void setSoTimeout(int timeout) throws SocketException:** Khi đang nghẽn (blocked) trên hàm read(), sau 1 thời gian timeout tính bằng mili giây mà 1 Client không gửi yêu cầu gì (request), Server sẽ quẳng ra 1 ngoại lệ (SocketException) trên Socket đó. Điều này, cũng có thể được đặt cho quá trình Client khi không nhận trả lời (response) từ Server.
- **public void setKeepAlive(boolean on) throws SocketException:** Quá trình Client muốn giữ nối kết ngay khi nó không gửi thông tin gì cho Server.

4.3.1.2 Chương trình TCPEchoClient

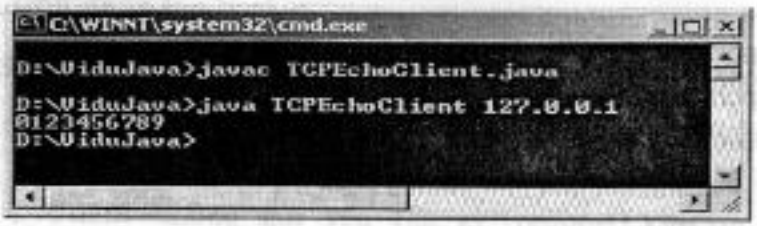
Trên hệ thống UNIX, Dịch vụ Echo được thiết kế theo kiến trúc Client-Server sử dụng Socket làm phương tiện giao tiếp. Cổng mặc định dành cho Echo Server là 7, bao gồm cả hai chế độ có nối kết và không nối kết.

Chương trình TCPEchoClient sẽ nối kết đến EchoServer ở chế độ có nối kết, lần lượt gởi đến Echo Server 10 ký tự từ '0' đến '9', chờ nhận kết quả trả về và hiển thị chúng ra màn hình.

Vi du 1:

```
import java.io.*;
import java.net.*;
public class TCPEchoClient {
    public static void main(String args[]) {
        try {
            Socket s = new Socket(args[0],7);// Noi ket den Server
            // Lay InputStream
            InputStream is = s.getInputStream();
            //Lay OutputStream
            OutputStream os = s.getOutputStream();
            // Gui '0'-> '9' den EchoServer
            for (int I = '0'; I<= '9'; I++)
                os.write(I);          //Gui 1 ky tu sang Server
                int ch = is.read();       // Chi nhan 1 ky tu tu Server
                //In ky tu nhan  duoc ra man hinh
                System.out.print((char)ch);
            }
        }    //try
        catch(IOException ie) {
            System.out.println("Loi: Khong tao duoc socket");
        }    //catch
    }  //mai
}
```

Biên dịch và thực thi chương trình như sau:



Chương trình này nhận một đối số là địa chỉ IP hay tên miền của máy tính mà ở đó Echo Server đang chạy. Trong hệ thống mạng TCP/IP mỗi máy tính được gán một địa chỉ IP cục bộ là **127.0.0.1** hay có tên là **localhost**. Trong ví dụ trên, chương trình Client nối kết đến Echo Server trên cùng máy với nó.

Trong môi trường mạng thực tế, nếu tồn tại 1 Server Unix (hoặc Linux) trên hệ thống mạng, chúng ta có thể dùng chương trình TCPEchoClient trên để kết nối đến Server đó (có địa chỉ IP thực) và thử nghiệm dịch vụ Echo Service.

Ví dụ 2: Chương trình này sẽ cho phép người sử dụng nhập từng ký tự từ bàn phím, sau đó mới gửi ký tự đó đến EchoServer và hiển thị kết quả trả về. Chương trình sẽ kết thúc (đóng nối kết) khi người dùng nhập 1 ký tự đặc biệt là '@'.

```
import java.io.*;
import java.net.*;
public class TCPEchoClient1{
    public static void main(String args[]) {
        try {    // Noi ket den Server
        Socket s = new Socket(args[0],7);
        // Lay InputStream
        InputStream is = s.getInputStream();
        //Lay OutputStream
        OutputStream os = s.getOutputStream();
        while(true) {
            System.out.print("Nhap ky tu:");
            // Nhap 1 ky tu tu ban phim
            int ch = System.in.read();
            if(ch == '@') break;
            // Neu ky tu la 2 thi thoat khoi CT
            System.in.skip(2);      // Bo qua 2 ky tu \r va \n
            os.write(ch);       // Gui ky tu qua Server
            int ch1 = is.read();  // Nhan ky tu tu Server gui ve
            //In ra man hinh
            System.out.println("Nhan duoc:" + (char)ch1);
        }
        s.close();          // Dong noi ket
    }    //try
    catch(IOException ie) {
        System.out.println("Loi: Khong tao duoc socket");
    }    //catch
}  //main
}
```

Kết quả hiển thị như sau:

4.3.2 Xây dựng chương trình Server ở chế độ có nối kết

4.3.2.1 Lớp java.net.ServerSocket

Lớp ServerSocket hỗ trợ các phương thức cần thiết để xây dựng các chương trình Server sử dụng socket ở chế độ có nối kết. Dưới đây là một số phương thức thường dùng để xây dựng Server:

public ServerSocket(int PortNumber);

Phương thức này tạo một Socket với số hiệu cổng là PortNumber mà sau đó Server sẽ lắng nghe trên cổng này.

Ví dụ: Tạo TCP socket cho Server với số hiệu cổng là 7:

```
ServerSocket ss = new ServerSocket(7);
```

public Socket accept()

Phương thức này lắng nghe yêu cầu nối kết của các Client. Đây là một phương thức hoạt động ở chế độ nghẽn. Nó sẽ bị nghẽn cho đến khi có một yêu cầu nối kết của client

gởi đến.

Khi có yêu cầu nối kết của Client gởi đến, nó sẽ chấp nhận yêu cầu nối kết, trả về một Socket là một đầu của kênh giao tiếp ảo giữa Server và Client yêu cầu nối kết.

Ví dụ: Socket ss chờ nhận yêu cầu nối kết:

```
Socket s = ss.accept();
```

Server sau đó sẽ lấy InputStream và OutputStream của Socket mới **s** để giao tiếp với Client.

4.3.2.2 Xây dựng chương trình Server phục vụ tuần tự

Một Server có thể được cài đặt để phục vụ các Client theo hai cách: phục vụ tuần tự hoặc phục vụ song song.

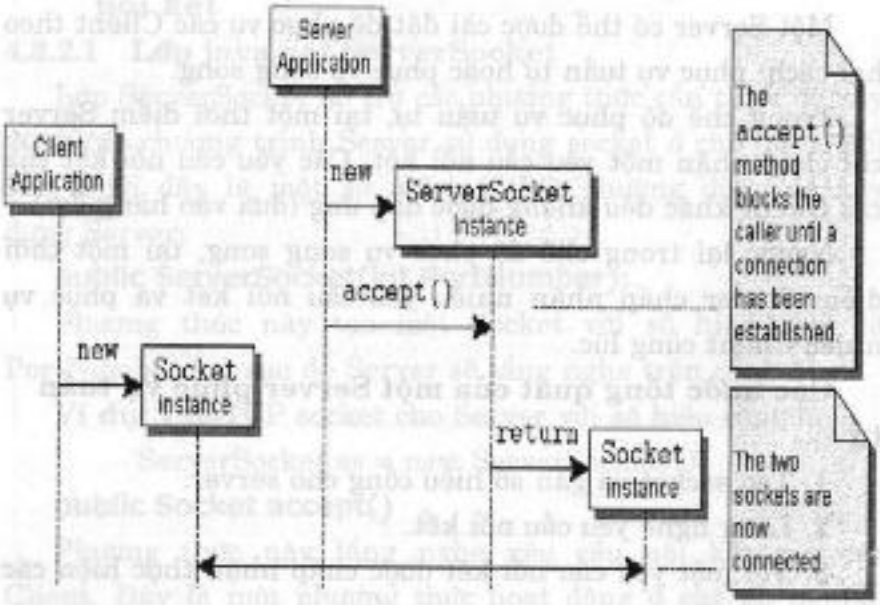
Trong chế độ phục vụ tuần tự, tại một thời điểm Server chỉ chấp nhận một yêu cầu nối kết. Các yêu cầu nối kết của các Client khác đều không được đáp ứng (đưa vào hàng đợi).

Ngược lại trong chế độ phục vụ song song, tại một thời điểm Server chấp nhận nhiều yêu cầu nối kết và phục vụ nhiều Client cùng lúc.

Các bước tổng quát của một Server phục vụ tuần tự

1. Tạo socket và gán số hiệu cổng cho server.
2. Lắng nghe yêu cầu nối kết.
3. Với một yêu cầu nối kết được chấp nhận thực hiện các bước sau:

- Lấy InputStream và OutputStream gắn với Socket của kênh ảo vừa được hình thành.
- Lặp lại công việc sau:
  - Chờ nhận các yêu cầu (công việc).
  - Phân tích và thực hiện yêu cầu.
  - Tạo thông điệp trả lời.
  - Gởi thông điệp trả lời về Client.
  - Nếu không còn yêu cầu hoặc Client kết thúc, đóng Socket và quay lại bước 2.



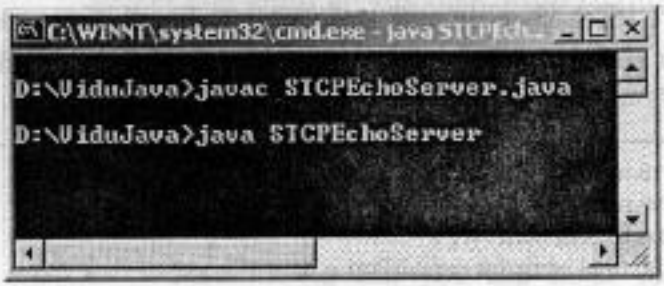
Hình 4.6 – Mô hình TCPServer phục vụ song song

4.3.2.3 Chương trình STCPEchoServer

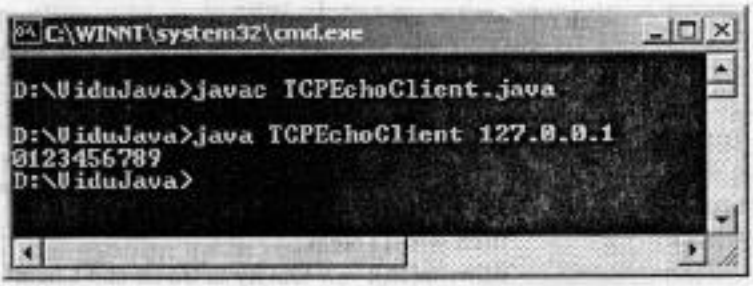
STCPEchoServer cài đặt một Echo Server phục vụ tuần tự ở chế độ có nối kết. Server lắng nghe trên cổng mặc định số 7.

```
import java.io.*;
import java.net.*;
public class TCPEchoServer {
    public final static int defaultPort = 7;
    public static void main(String args[]) {
        try {   // Tao Server Socket lang nghe tren cong 7
            ServerSocket ss = new ServerSocket(defaultPort);
                while (true) {
                    try {
                        Socket s = ss.accept(); // Chap nhan 1 client noi ket
                        OutputStream os = s.getOutputStream();
                        inputStream is = s.getInputStream();
                        int ch = 0;
                        while (true) {
                            ch = is.read();  //Nhan ky tu tu Client
                            if(ch == -1) break;
                            os.write(ch);    // Gui ky tu tra lai cho Client
                            }
                            s.close();
                        }catch (IOException e){
                            System.err.println("Connection Error:" + e);
                        }
                    }
            }catch (IOException e){
                System.err.println("Server Creation Error:" + e);
            }
    }
}
```

Biên dịch và thực thi chương trình theo cách sau:



Sau đó, thực thi chương trình Echo Client trên 1 cửa sổ khác để kiểm tra:



Hai chương trình này có thể nằm trên hai máy khác nhau. Trong trường hợp đó khi thực hiện chương trình TCPEchoClient phải chú ý nhập đúng địa chỉ IP của máy tính đang chạy chương trình STCPEchoServer thay vì dùng địa chỉ cục bộ là 127.0.0.1 để thử nghiệm.

Ví dụ: Echo Server đang thực thi tại máy tính có địa chỉ là 172.18.213.233, thực thi Client như sau:

java TCPEchoClient 172.18.213.233

Ta có thể xem địa chỉ IP của một máy tính Windows bằng lệnh **ipconfig**.

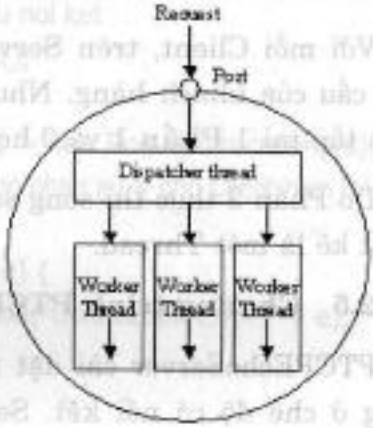
4.3.2.4 Server phục vụ song song

Các bước tổng quát của một Server phục vụ song song

Server phục vụ song song gồm 2 phần thực hiện song song nhau:

- Phần 1: Xử lý các yêu cầu nối kết.
- Phần 2: Xử lý các thông điệp yêu cầu từ khách hàng.

Có cấu trúc như hình sau, trong đó **Phần 1** là (Dispatcher Thread), **Phần 2** là các (Worker Thread)



Hình 4.7 - Server ở chế độ song song

Phần 1: Lặp lại các công việc sau:

- Lắng nghe yêu cầu nối kết của khách hàng.
- Chấp nhận một yêu cầu nối kết.

- Tạo kênh giao tiếp ảo mới với khách hàng.
- Tạo Phần 2 để xử lý các thông điệp yêu cầu của khách hàng.

Phần 2: Lập lại các công việc sau:

- Chờ nhận thông điệp yêu cầu của khách hàng.
- Phân tích và xử lý yêu cầu.
- Gởi thông điệp trả lời cho khách hàng.

Phần 2 sẽ kết thúc khi kênh ảo bị xóa đi.

Với mỗi Client, trên Server sẽ có một **Phần 2** để xử lý yêu cầu của khách hàng. Như vậy tại một thời điểm bất kỳ luôn tồn tại 1 **Phần 1** và 0 hoặc nhiều **Phần 2**.

Do Phần 2 thực thi song song với Phần 1 cho nên nó được thiết kế là một Thread.

4.3.2.5 Chương trình PTCPEchoServer

PTCPEchoServer cài đặt một Echo Server phục vụ song song ở chế độ có nối kết. Server lắng nghe trên cổng mặc định là 7. Chương trình này gồm 2 lớp:

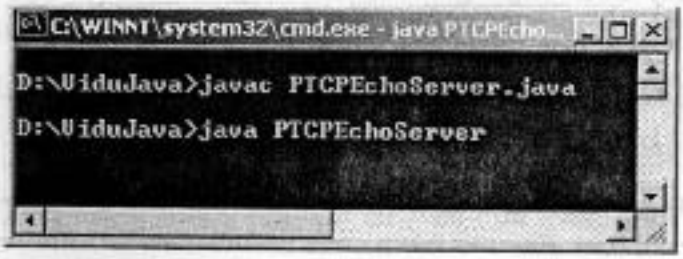
- Lớp TCPEchoServer, cài đặt các chức năng của Phần 1 - xử lý các yêu cầu nối kết của TCPEchoClient.
- Lớp RequestProcessing, là một Thread cài đặt các chức

năng của Phần 2 - Xử lý các thông điệp yêu cầu.

```
public class PTCPEchoServer {
    public final static int defaultPort = 7;  // Cong mac dinh
    public static void main(String[]args) {
        try {
            // Tao socket cho Server
            ServerSocket ss = new ServerSocket(defaultPort);
            while(true) {
                try {
                    // Lang nghe yeu cau noi ket
                    Socket s = ss.accept();
                    // Tao phan xu ly
                    RequestProcessing rp = new RequestProcessing(s);
                    rp.start(); // Khoi dong phan xu ly cho Client hien tai
                    }
                     catch (IOException e) {
                         System.out.println("Connection Error:" + e);
                     }
                }
            }
            catch (IOException e) {
                System.err.println("Creat Socket Error:" + e);
            }
        }
}
```

```
class RequestProcessing extends Thread
{     private Socket s;
      public RequestProcessing(Socket s1){
          s=s1;
      }
      public void run() {
          try {
              OutputStream os = s.getOutputStream();
              InputStream is = s.getInputStream();
              Int ch = 0;
              While(true) {
                    ch = is.read();      // Nhan ky tu tu Client
                    if(ch == -1) break;
                    os.write(ch)    // gui ky tu tra lai cho Client
               }
              s.close();
          }
          catch (IOException e) {
              System.err.println("Processing Error:" + e);
          }
    }
};
```

Biên dịch và thực thi chương trình như sau:



Sau đó mở thêm 2 của sổ DOS khác để thực thi chương trình TCPEchoClient nối kết tới PTCPEchoServer. Ta sẽ nhận thấy rằng PTCPEchoServer có khả năng phục vụ đồng thời nhiều Client.

4.3.3 Xây dựng chương trình Client - Server ở chế độ không nối kết

Khi sử dụng socket, ta có thể chọn giao thức UDP cho lớp vận chuyển. UDP viết tắt của User Datagram Protocol, cung cấp cơ chế vận chuyển không bảo đảm và không nối kết trên mạng IP, ngược với giao thức vận chuyển tin cậy, có nối kết TCP.

Cả giao thức TCP và UDP đều phân dữ liệu ra thành các gói tin. Tuy nhiên TCP có thêm vào những tiêu đề (Header) vào trong gói tin để cho phép truyền lại những gói tin thất lạc và tập hợp các gói tin lại theo thứ tự đúng đắn. UDP không cung cấp tính năng này, nếu một gói tin bị thất lạc hoặc bị lỗi, nó sẽ không được truyền lại, và thứ tự đến đích của các gói tin cũng không giống như thứ tự lúc nó được gởi đi.

Tuy nhiên, về tốc độ, UDP sẽ truyền nhanh gấp 3 lần TCP. Cho nên chúng thường được dùng trong các ứng dụng đòi hỏi thời gian truyền tải ngắn và không cần tính chính xác cao, ví dụ truyền âm thanh, hình ảnh . . .

Mô hình client - server sử dụng lớp ServerSocket và Socket ở trên sử dụng giao thức TCP. Nếu muốn sử dụng mô hình client - server với giao thức UDP, ta sử dụng hai lớp java.net.**DatagramSocket** và java.net.**DatagramPacket**.

DatagramSocket được sử dụng để truyền và nhận các DatagramPacket. Dữ liệu được truyền đi là một mảng những byte, chúng được gói vào trong lớp DatagramPacket. Chiều dài của dữ liệu tối đa có thể đưa vào DatagramPacket là khoảng hơn 60.000 byte (phụ thuộc vào dạng đường truyền). Ngoài ra DatagramPacket còn chứa địa chỉ IP và cổng của quá trình gởi và nhận dữ liệu.

Cổng trong giao thức TCP và UDP có thể trùng nhau. Trên cùng một máy tính, bạn có thể gán cổng 20 cho socket dùng giao thức TCP và cổng 20 cho socket sử dụng giao thức UDP.

4.3.3.1 Lớp DatagramPacket

Lớp này dùng để đóng gói dữ liệu gởi đi. Dưới đây là các phương thức thường sử dụng để thao tác trên dữ liệu truyền / nhận qua DatagramSocket.

public DatagramPacket(byte[] b, int n)

- Là phương thức khởi tạo, cho phép tạo ra một DatagramPacket chứa n bytes dữ liệu đầu tiên của

mảng b. (n phải nhỏ hơn chiều dài của mảng b)

- Phương thức trả về một đối tượng thuộc lớp DatagramePacket

Ví dụ: Tạo DatagramPacket để nhận dữ liệu:

```
byte buff[] = new byte[60000];  // Nơi chứa dữ liệu nhận được
DatagramPacket inPacket = new Datagrampacket (buff, buff.length);
```

public DatagramPacket(byte[] b, int n, InetAddress ia, int port)

- Phương thức này cho phép tạo một DatagramPacket chứa dữ liệu và cả địa chỉ của máy nhận dữ liệu.
- Phương thức trả về một đối tượng thuộc lớp DatagramePacket

Ví dụ: Tạo DatagramPacket chứa chuỗi "My second UDP Packet", với địa chỉ máy nhận là www.cit.ctu.edu.vn, cổng của quá trình nhận là 19:

```
try {//Địa chỉ Internet của máy nhận
    InetAddress ia = InetAddess.getByName("www.cit.ctu.edu.vn");
     int port = 19; //   Cổng của socket nhận
     String s = "My second UDP Packet";   // Dữ liệu gởi đi
```

```
byte[] b = s.getBytes();   // Đổi chuỗi thành mảng bytes
  // Tạo gói tin gởi đi
  DatagramPacket outPacket = new DatagramPacket(b,
b.length, ia, port);
}
catch (UnknownHostException e) {
  System.err.println(e);
}
```

Các phương thức lấy thông tin trên một **DatagramPacket** nhận được

Khi nhận được một **DatagramPacket** từ một quá trình khác gởi đến, ta có thể lấy thông tin trên **DatagramPacket** này bằng các phương thức sau:

- public synchronized() InternetAddress getAddress() : Địa chỉ máy gởi
- public synchronized() int getPort() : Cổng của quá trình gởi
- public synchronized() byte[] getData() : Dữ liệu từ gói tin
- public synchronized() int getLength() : Chiều dài của dữ liệu trong gói tin

- Các phương thức đặt thông tin cho gói tin gởi

Trước khi gởi một DatagramPacket đi, ta có thể đặt thông tin trên DatagramPacket này bằng các phương thức sau:

- public synchronized() void setAddress(InternetAddress dis) : Đặt địa chỉ máy nhận.
- public synchronized() void setPort(int port) : Đặt cổng quá trình nhận
- public synchronized() void setData(byte buffer[]) : Đặt dữ liệu gởi
- public synchronized() void setLength(int len) : Đặt chiều dài dữ liệu gởi
- Lớp DatagramSocket

Lớp này hỗ trợ các phương thức sau để gởi / nhận các DatagramPacket

public DatagramSocket() throws SocketException

- Tạo Socket kiểu không nối kết cho Client. Hệ thống tự động gán số hiệu cổng chưa sử dụng cho socket.

Ví dụ: Tạo một socket không nối kết cho Client:

```
try{
    DatagramSocket ds = new DatagramSocket();
} catch(SocketException se) {
    System.out.print("Create DatagramSocket Error: "+se);
}
```

public DatagramSocket(int port) throws SocketException

- Tạo Socket kiểu không nối kết cho Server với số hiệu cổng được xác định trong tham số (port).

Ví dụ: Tạo một socket không nối kết cho Server với số hiệu cổng là 7:

```
try{
    DatagramSocket dp = new DatagramSocket(7);
 } catch(SocketException se) {
    System.out.print("Create DatagramSocket Error: "+se);
 }
```

public void send(DatagramPacket dp) throws IOException

- Dùng để gởi một DatagramPacket đi.

Ví dụ: Gởi chuỗi "Day la du lieu can gui", cho quá trình ở địa chỉ www.cit.ctu.edu.vn, cổng nhận là 19:

```
try {
    DatagramSocket ds = new DatagramSocket();  //Tạo Socket
    //Địa chỉ Internet của máy nhận
   InetAddress ia = InetAddess.getByName("www.cit.ctu.edu.vn");
```

```
        int port = 19; // Cổng của quá trình nhận
        String s = "Day la du lieu can gui";  // Dữ liệu cần gửi
        byte[] b = s.getBytes();  // Đổi sang mảng bytes
        // Tạo gói tin
        DatagramPacket outPacket = new DatagramPacket(b,
b.length, ia, port);
        ds.send(outPacket);     // Gửi gói tin đi
    }
    catch (IOException e) {
       System.err.println(e);
    }
```

public synchronized void receive(DatagramPacket dp) throws IOException

- Chờ nhận một DatagramPacket. Quá trình sẽ bị nghẽn cho đến khi có dữ liệu đến.

Ví dụ: Nhận 1 DatagramPacket. Hiển thị thông tin của gói tin ra màn hình.

```
try {
    DatagramSocket ds = new DatagramSocket(); //Tạo Socket
    byte[] b = new byte[60000]; // Nơi chứa dữ liệu nhận được
```

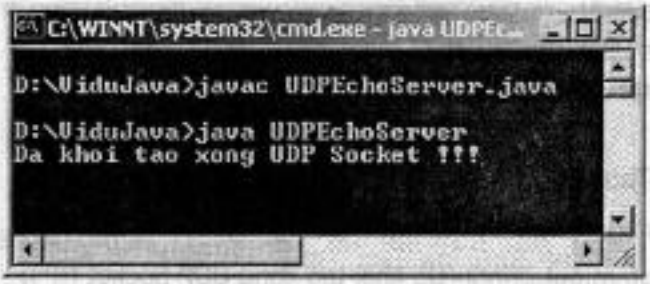
```
        // Tạo gói tin
        DatagramPacket inPacket = new DatagramPacket(b, b.length);
        ds.receive(inPacket); // Chờ nhận gói tin
        // Hiển thị nội dung của gói tin ra màn hình
        System.out.println("Cong cua qua trinh gui: " +
inPacket.getPort());
        System.out.println("IP cua qua trinh gui: " +
inPacket.getAddress().toString());
        System.out.println("Du lieu ben trong goi tin: ");
        System.out.println( new String(inPacket.getData(), 0,
inPacket.getLength()));
      }
     catch (IOException e) {
        System.err.println(e);
  }
```

4.3.3.2 Chương trình UDPEchoServer

Chương trình UDPEchoServer cài đặt Echo Server ở chế độ không nối kết, cổng mặc định là 7. Chương trình chờ nhận từng gói tin, lấy dữ liệu ra khỏi gói tin nhận được và gởi ngược dữ liệu đó về Client.

```
import java.net.*;
import java.io.*;
public class UDPEchoServer {
   public static void main(String[] args) {
      try {
        // Tao Socket voi cong la 7
        DatagramSocket ds = new DatagramSocket(7);
        System.out.println(Da khoi tao xong UDP Socket !!!");
       // Vung dem chua du lieu cho goi tin nhan
        Byte[] buffer = new byte[60000];
       While(true) {      // Tao goi tin nhan
            DatagramPacket in = new DatagramPacket(buffer,
    buffer, length);
            ds.receive(in);  // Cho nhan goi tin gui den
            // Lay du lieu khoi goi tin nhan
            String str = new String(in.getData(),0,in.getLength());
            // Tao goi tin goi chua du lieu vua nhan duoc
            DatagramPaket out = new DatagramPaket(str.getByte(),
                  in.getLength(),in.getAddress(),in.getPort());
           ds.send(out);
         }
       }catch (IOException e) { System.err.println(e); }
   }
}
```

Biên dịch và thực thi chương trình như sau



4.3.3.3 Chương trình UDPEchoClient

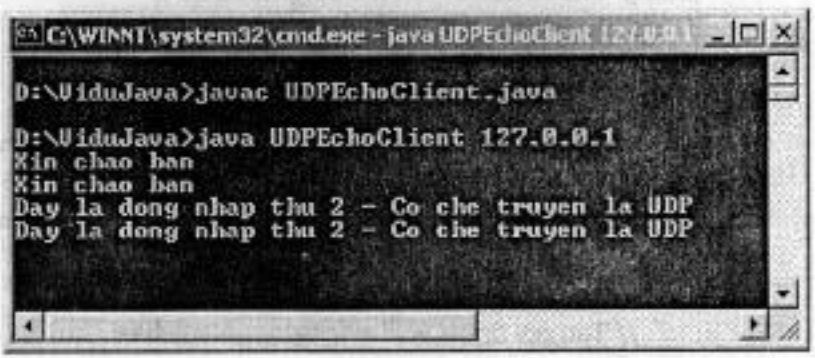
Chương trình này cho phép người sử dụng nhận các chuỗi từ bàn phím, gởi chuỗi sang EchoServer ở chế độ không nối kết ở cổng số 7, chờ nhận và in dữ liệu từ Server gởi về ra màn hình.

Lưu chương trình sau vào tập tin UDPEchoClient.java

```
import java.net.*;
import java.io.*;
public class UDPEchoClient {
    public final static int severPort = 7;   // Cong phuc vu cua Echo
Server
    public static void main(String[] args) {
        try {
          if (args.length == 0) // Kiem tra tham so la dia chi cua Server
             { System.out.print(Systax: java UDPClient HostName");
               return;  }
```

```
        // Tao DatagramSocket
        DatagramSocket ds = new DatagraSocket();
        // Dia chi Server
        InetAddress server =InetAddress.getByName(args[0]);
         While (true) {
            BufferedReader br = new BufferedReader(new
      InputStreamReader(System.in));
            String theString = br.readLine();// Nhap tu ban phim
            // Doi chuoi ra mang bytes
            Byte[] data = theString.getBytes();
            // Tao goi tin
            DatagramPacket dp= new
      DatagramPacket(data,data.length, server, serverPort);
            ds.send(dp); // gui goi tin sang server
            //Tao vung dem cho goi tin de nhan
            byte [] buffer=new byte [6000];
            // Tao goi tin de nhan
            DatagramPacket incoming = new
    DatagramPacket(buffer, buffer.length);
            ds.receive(incoming);//Cho nhan goi tin tra loi tu Server
            // Hien thi noi dung goi tin nhan duoc ra man hinh
              System.out.println(new
           String(incoming.getData(),0,incoming.getLength()));
        }
      }
      catch (IOException e) {
        System.err.println(e);
      }
   }
}
```

Biên dịch và thực thi chương trình như sau:



Chú ý, khi thực hiện chương trình UDPEchoClient phải đưa vào đối số là địa chỉ của máy tính đang thực thi chương trình UDPEchoServer. Trong ví dụ trên, Server và Client cùng chạy trên một máy nên địa chỉ của UDPEchoServer là ***localhost*** (hay 127.0.0.1). Nếu UDPEchoServer chạy trên máy tính khác thì khi thực thi, ta phải biết được địa chỉ IP của máy tính đó và cung cấp vào đối số của chương trình. Chẳng hạn, khi UDPEchoServer đang phục vụ trên máy tính ở địa chỉ 172.18.250.211, ta sẽ thực thi UDPEchoClient theo cú pháp sau:

```
java UDPEchoClient 172.18.250.211
```

4.3.4 Xây dựng chương trình client - server nối kết theo dạng multicast (truyền theo nhóm)

4.3.4.1 Multicast

Tất cả các cơ chế Socket TCP và UDP đã trình bày phía trước đều gọi là unicast, nghĩa là giao tiếp chỉ diễn ra giữa 1 máy tính gửi và 1 máy tính nhận.

Multicast (tạm dịch là liên lạc theo nhóm) là việc gửi quảng bá (broadcast) nhưng chỉ đến 1 nhóm các máy tính ở cùng 1 địa chỉ cho trước. Địa chỉ multicast là địa chỉ lớp D được xác định trong khoảng từ 224.0.0.0 đến 239.255.255.255. Địa chỉ 224.0.0.0 là địa chỉ dành riêng nên không được sử dụng. Do đó, nếu ta gõ lệnh **ping 224.0.0.1** thì tất cả các máy tính hỗ trợ multicast sẽ trả lời.

Một gói tin multicast sẽ có khả năng được gửi cho toàn bộ mạng (flood), không hạn chế chỉ gửi đến chỉ 1 địa chỉ máy nhận duy nhất. Điều này có thể được sử dụng trong game nhiều người chơi, trong những giải thuật vạch đường (Routing Protocol) khi các router muốn cập nhật thông tin với nhau hoặc trong những ứng dụng mà đối tượng là nhiều thiết bị hay máy tính cùng nhận chung 1 loại thông tin.

4.3.4.2 Multicast trong Java

Java hỗ trợ multicast thông qua lớp java.net.MulticastSocket

Một Multicast Socket là 1 DatagramSocket (UDP) có khả năng gia nhập (joining) vào 1 nhóm các máy tính multicast trên mạng. Khi một máy tính nào gửi 1 thông điệp đến nhóm thì tất cả các máy tính trong đó đều nhận được.

Ví dụ:

Gia nhập vào 1 nhóm multicast có địa chỉ 228.5.6.7 bằng đoạn chương trình:

```
InetAddress group = InetAddress.getByName("228.5.6.7");
MulticastSocket s = new MulticastSocket(6789);
s.joinGroup(group);
```

Thoát ra khỏi nhóm multicast bằng cách:

```
s.leaveGroup(group);
```

4.3.4.3 Ví dụ về Multicast trong Java

Cài đặt 1 dịch vụ tên là Time Service phục vụ trên cổng 9013, dùng để gửi thông tin về thời gian đến nhóm khách hàng ở địa chỉ 230.0.0.1. Thực ra, chương trình Server vẫn phục vụ theo dạng Socket UDP chỉ khác là sẽ gửi gói tin Datagram đến địa chỉ lớp D.

Chương trình MulticastTimeServer như sau:

```
import java.io.*;
import java.net.*;
Public class MulticastTimeServer {
    public static void main(String args[]) {
        try {
            DatagramSocket socket = new DatagramSocket(1213);
            while (true) {
                // Tao du lieu can gui la ngay va gio hien tai
                String date = new Date().toString();
                byte buffer[] = date.getBytes();
                InetAddress address =
                InetAddress.getByName("230.0.0.1");
            // Tao ra goi tin gui den dia chi 230.0.0.1 va cong 9013
                DatagramPacket packet = new
                DatagramPacket(buffer, buffer.length, ddress, 9013);
              // Goi du lieu den cac Client
                Socket.send(packet);
              System.out.println("Vua moi gui xong goi tin vao luc" +
            date);
              Thread.sleep(5000);  // 5s se gui 1 lan
              }
        }
        catch(Exception e)
            { System.out.println("Co loi khi tao va thuc thi
            Socket"); }
    }
}
```

Biên dịch và thực thi chương trình như sau:

Chương trình MulticastTimeClient như sau:

```
import java.io.*;
import java.net.*;
import java.util.Date;
public class MulticastTimeClient {
    public static void main(String args[]) {
        try {
            // Tao Socket the dang Multicast tren cong 9013
            MulticastSocket socket = new MulticastSocket(9013);
```

```
            InetAddress address =
InetAddress.getByname("230.0.0.1");
            // Tham gia vao nhom dia chi 230.0.0.1
            socket.joinGroup(address);
            byte message[] = new byte [256];
            DatagramPacket packet = new
DatagramPacket(message, message.length);
            // Nhan goi tin tu Server
            socket.receive(packet);
            String time =new String(packet.getData());
            System.out.println("Hien nay la:" + time);
            // Roi khoi nhom dia chi lop D la 230.0.0.1
            socket.leaveGroup(address);
            socket.close();
        }
        catch(Exception e)
            { System.out.println("Co loi khi tao va thuc thi Client");
            }
    }
}
```

Ta có thể thử nghiệm chương trình bằng cách thực thi Client đồng thời trên nhiều máy tính có địa chỉ IP khác nhau, ta nhận thấy, các Client này sẽ nhận được dữ liệu như nhau từ Server.

Biên dịch và thực thi chương trình như sau:

4.4 BÀI TẬP ÁP DỤNG

Chủ đề 1: Client ở chế độ có nối kết

- **Mục đích:**

Viết các chương trình Client nối kết đến các server theo các Protocol chuẩn.

- **Yêu cầu**

Sinh viên thực hiện các bài tập sau

- **Bài 1** : Viết chương trình nhận đối số là một URL. Nối kết đến Web Server trong URL nhận được, lấy trang web về và in ra màn hình theo dạng textfile (html).

Chủ đề 2: Client - Server chế độ có nối kết

- **Mục đích:**

- Viết các chương trình Client -Server theo chế độ có nối kết.

- **Yêu cầu**

Sinh viên thực hiện các bài tập sau, với mỗi bài tập hãy thiết kế một Server phục vụ ở chế độ tuần tự và một Server phục vụ ở chế độ song song.

- **Bài 1**: Viết chương trình theo mô hình Client-Server sử dụng dụng Socket ở chế độ có nối kết. Trong đó :

  + Server làm nhiệm vụ đọc một ký tự số từ '0' đến '9'.

  (Ví dụ : nhận số 0 : trả về "khong" , 1 : trả về "một" ; ... ... 9 : trả về "chín", nếu nhận ký tự khác số thì trả về "Không phải số nguyên").

+ Client sẽ nhập vào 1 ký tự, gửi qua Server, nhận kết quả trả về từ Server và thể hiện lên màn hình

- **Bài 2**: Viết chương trình theo mô hình Client-Server sử dụng Socket ở chế độ có nối kết.

Trong đó :

+ Server sẽ nhận các yêu cầu là một chuỗi có khuôn dạng như sau:

"OP Operant1 Operant2\n"

Trong đó:

- OP là một ký tự chỉ phép toán muốn thực hiện: '+','-', '*', '/'.

- Operant1, Operant2 là đối số của phép toán.

- Các thành phần trên cách nhau bởi 1 ký tự trắng ' '.

- Kết thúc yêu cầu bằng ký tự xuống dòng '\n'.

Mỗi khi server nhận được một thông điệp nó sẽ thực hiện phép toán:

Operant1 OP Operant2 để cho ra kết quả, sau đó đổi kết quả thành chuỗi và gởi về Client.

+ Client cho phép người dùng nhập các phép toán muốn tính theo cách thức thông thường. Ví dụ: 100+200. Client tạo ra thông điệp yêu cầu theo đúng dạng do Server qui định, mô tả về phép toán muốn Server thực thi, rồi gởi sang Server, chờ nhận kết quả trả về và in ra màn hình.

Chủ đề 3: Client-Server ở chế độ không nối kết

- **Mục đích**:
  - Viết các chương trình Client -Server theo chế độ không nối kết.
- **Yêu cầu**
  - **Bài 1** : Viết chương trình Talk theo chế độ không nối kết. Cho phép hai người ngồi trên hai máy tính có thể tán gẫu (chat) với nhau.

Chương 5

RPC và RMI

Mục đích

Chương này nhằm giới thiệu cách thức xây dựng các ứng dụng phân tán bằng các cơ chế gọi thủ tục từ xa (RPC - Remote Procedure Call và RMI - Remote Method Invocation)

Yêu cầu

Sau khi hoàn tất chương này, bạn có thể:

- Định nghĩa được ứng dụng phân tán là gì.
- Trình bày được kiến trúc của một ứng dụng phân tán xây dựng theo cơ chế gọi thủ tục từ xa (RPC).
- Trình bày được kiến trúc của một ứng dụng phân tán (hay còn gọi Ứng dụng đối tượng phân tán) xây dựng theo cơ chế RMI của Java.
- Trình bày được các cơ chế liên quan khi xây dựng một ứng dụng theo kiểu RMI.
- Trình bày được cơ chế vận hành của một ứng dụng theo kiểu RMI.

- Giải thích được vai trò rmiregistry server.
- Liệt kê được các lớp của java hỗ trợ xây dựng các ứng dụng kiểu RMI.
- Trình bày chi tiết các bước khi xây dựng một ứng dụng theo kiểu RMI.
- Biên soạn, biên dịch và thực thi thành công chương trình minh họa Hello.
- Phân tích, thiết kế và cài đặt được các chương trình theo cơ chế RMI để giải quyết các vấn đề cụ thể.

5.1 LỜI GỌI THỦ TỤC XA (RPC- REMOTE PROCEDURE CALL)

5.1.1 Giới thiệu

Lời gọi thủ tục xa là một cơ chế cho phép một chương trình có thể gọi thực thi một thủ tục (hay hàm) trên một máy tính khác. Trong chương trình lúc này, tồn tại hai loại thủ tục: thủ tục cục bộ và thủ tục ở xa.

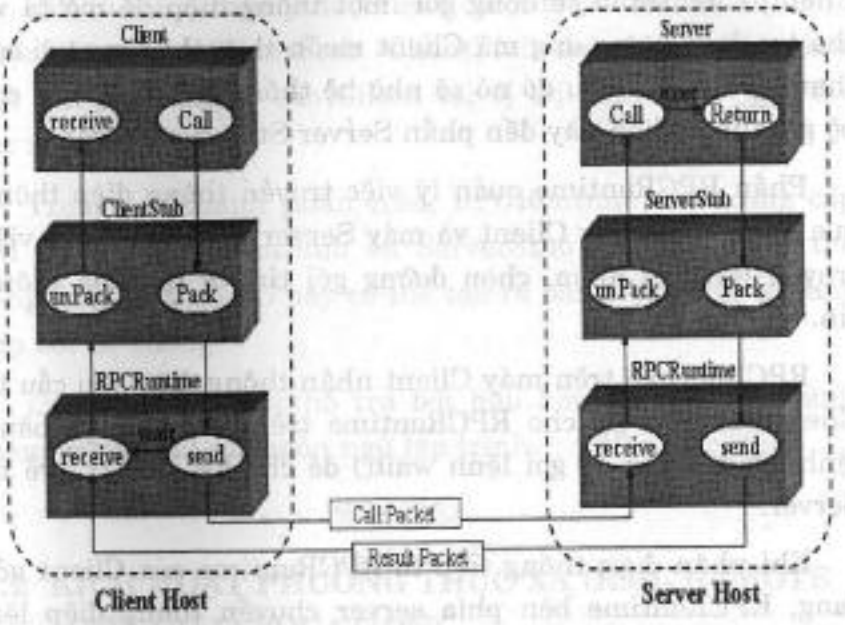
- Thủ tục cục bộ (local method): là thủ tục được định nghĩa, cài đặt và thực thi tại máy của chương trình.
- Thủ tục ở xa (remote method): là thủ tục được định nghĩa, cài đặt và thực thi trên một máy tính khác.

Cú pháp giữa lời gọi thủ tục cục bộ và ở xa thì giống nhau. Tuy nhiên, khi một thủ tục ở xa được gọi đến, một thành phần của chương trình gọi là **Stub** sẽ chuyển hướng để kích hoạt một thủ tục tương ứng nằm trên một máy tính khác với máy của chương trình gọi. Đối với người lập trình, việc gọi thủ tục xa và thủ tục cục bộ thì giống nhau về mặt cú pháp. Đây chính là cơ chế cho phép đơn giản hóa việc xây dựng các ứng dụng Client-Server. Trong hệ thống RPC, Server chính là máy tính cung cấp các thủ tục ở xa cho phép các chương trình trên các máy tính khác gọi thực hiện. Client chính là các chương trình có thể gọi các thủ tục ở xa trong quá trình tính toán của mình.

Một Client có thể gọi thủ tục ở xa của nhiều hơn một máy tính. Như vậy sự thực thi của chương trình Client lúc này không còn gói gọn trên một máy tính của Client mà nó trải rộng trên nhiều máy tính khác nhau. Đây chính là mô hình của ứng dụng phân tán (Distributed Application).

5.1.2 Kiến trúc của chương trình Client-Server cài đặt theo cơ chế lời gọi thủ tục xa

Một ứng dụng Client-Server theo cơ chế RPC được xây dựng gồm có sáu phần như sơ đồ dưới đây:



Hình 5.1 Kiến trúc chương trình kiểu RPC

Phần Client là một quá trình người dùng, nơi khởi tạo một lời gọi thủ tục từ xa. Mỗi lời gọi thủ tục ở xa trên phần Client sẽ kích hoạt một thủ tục cục bộ tương ứng nằm trong phần Stub của Client.

Phần ClientStub cung cấp một bộ các hàm cục bộ mà phần Client có thể gọi. Mỗi một hàm của ClientStub đại diện cho một hàm ở xa được cài đặt và thực thi trên Server.

Mỗi khi một hàm nào đó của ClientStub được gọi bởi

Client, ClientStub sẽ đóng gói một thông điệp để mô tả về thủ tục ở xa tương ứng mà Client muốn thực thi cùng với các tham số nếu có. Sau đó nó sẽ nhờ hệ thống RPCRuntime cục bộ gởi thông điệp này đến phần Server Stub của Server.

Phần RPCRuntime quản lý việc truyền thông điệp thông qua mạng giữa máy Client và máy Server. Nó dảm nhận việc truyền lại, báo nhận, chọn đường gói tin và mã hóa thông tin.

RPCRuntime trên máy Client nhận thông điệp yêu cầu từ ClientStub, gởi nó cho RPCRuntime trên máy Server bằng lệnh send(). Sau đó gọi lệnh wait() để chờ kết quả trả về từ Server.

Khi nhận được thông tiệp từ RPCRuntime của Client gởi sang, RPCRuntime bên phía server chuyển thông điệp lên phần ServerStub.

ServerStub mở thông diệp ra xem, xác định hàm ở xa mà Client muốn thực hiện cùng với các tham số của nó. ServerStub gọi một thủ tục tương ứng nằm trên phần Server.

Khi nhận được yêu cầu của ServerStub, Server cho thực thi thủ tục được yêu cầu và gởi kết quả thực thi được cho ServerStub.

ServerStub đóng gói kết quả thực trong một gói tin trả lời, chuyển cho phần RPCRuntime cục bộ để nó gởi sang RPCRuntime của Client .

RPCRuntime bên phía Client chuyển gói tin trả lời nhận được cho phần ClientStub. ClientStub mở thông điệp chứa kết quả thực thi về cho Client tại vị trí phát ra lời gọi thủ tục xa.

Trong các thành phần trên, RPCRuntime được cung cấp bởi hệ thống. ClientStub và ServerStub có thể tạo ra thủ công (phải lập trình) hay có thể tạo ra bằng các công cụ cung cấp bởi hệ thống.

Cơ chế RPC được hỗ trợ bởi hầu hết các hệ điều hành mạng cũng như các ngôn ngữ lập trình.

5.2 KÍCH HOẠT PHƯƠNG THỨC XA (RMI- REMOTE METHOD INVOCATION)

5.2.1 Giới thiệu

RMI là một sự cài đặt cơ chế RPC trong ngôn ngữ lập trình hướng đối tượng Java. Hệ thống RMI cho phép một đối tượng chạy trên một máy ảo Java này có thể kích hoạt một phương thức của một đối tượng đang chạy trên một máy ảo Java khác. Đối tượng có phương thức được gọi từ xa gọi là các đối tượng ở xa (Remote Object).

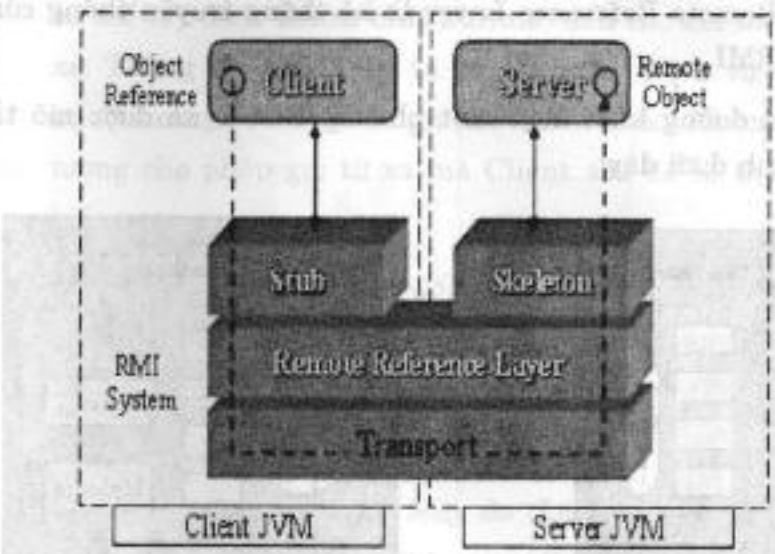
Một ứng dụng RMI thường bao gồm 2 phần phân biệt: Một chương trình Server và một chương trình Client.

- Chương trình Server tạo một số các Remote Object, tạo các **tham chiếu** (reference) đến chúng và chờ những chương trình Client kích hoạt các phương thức của các Remote Object này.
- Chương trình Client lấy một **tham chiếu** đến một hoặc nhiều Remote Object trên Server và kích hoạt các phương thức từ xa thông qua các tham chiếu.

Một chương trình Client có thể kích hoạt các phương thức ở xa trên một hay nhiều Server. Tức là sự thực thi của chương trình được trải rộng trên nhiều máy tính. Đây chính là đặc điểm của các ứng dụng phân tán. Nói cách khác, RMI là cơ chế để xây dựng các ứng dụng phân tán dưới ngôn ngữ Java.

5.2.2 Kiến trúc của chương trình Client-Server theo cơ chế RMI

Kiến trúc một chương trình Client-Server theo cơ chế RMI được mô tả như hình dưới đây:



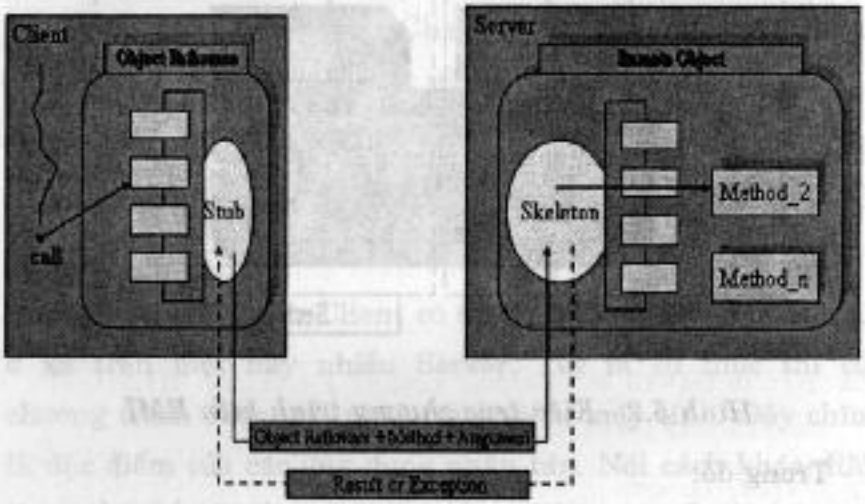
Hình 5.2 - Kiến trúc chương trình kiểu RMI

Trong đó:

- Server là chương trình cung cấp các đối tượng có thể được gọi từ xa.
- Client là chương trình có tham chiếu đến các phương thức của các đối tượng ở xa trên Server.
- Stub chứa các tham chiếu đến các phương thức ở xa trên Server.
- Skeleton đón nhận các tham chiếu từ Stub để kích hoạt phương thức tương ứng trên Server.

- Remote Reference Layer là hệ thống truyền thông của RMI.

Con đường kích hoạt một phương thức ở xa được mô tả như hình dưới đây:



Hình 5.3 Cơ chế hoạt động của RMI

5.2.3 Các cơ chế liên quan trong một ứng dụng đối tượng phân tán

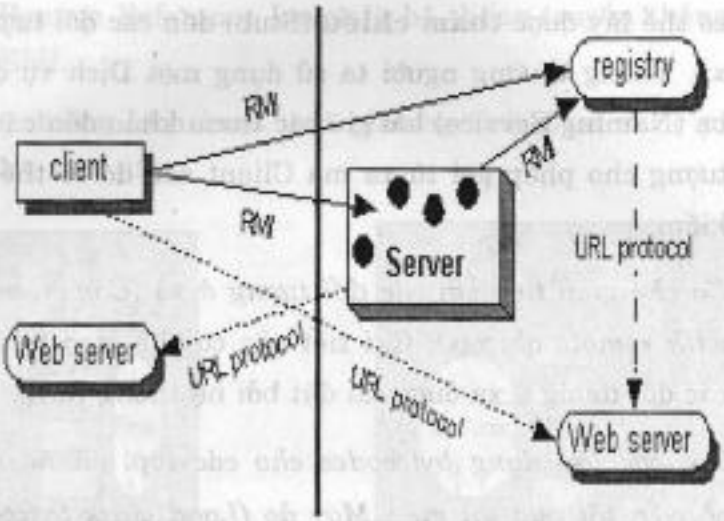
Trong một ứng dụng phân tán cần có các cơ chế sau:

- *Cơ chế định vị đối tượng ở xa (Locate remote objects)*: Cơ chế này xác định cách thức mà chương trình Client

có thể lấy được **tham chiếu** (Stub) đến các đối tượng ở xa. Thông thường người ta sử dụng một Dịch vụ danh bạ (Naming Service) lưu giữ các tham khảo đến các đối tượng cho phép gọi từ xa mà Client sau đó có thể tìm kiếm.

- *Cơ chế giao tiếp với các đối tượng ở xa (Communicate with remote objects)*: Chi tiết của cơ chế giao tiếp với các đối tượng ở xa được cài đặt bởi hệ thống RMI.
- *Tải các lớp dạng bytecodes cho các lớp mà nó được chuyển tải qua lại giữa Máy ảo (Load class bytecodes for objects that are passed around)*: Vì RMI cho phép các chương trình gọi phương thức từ xa trao đổi các đối tượng với các phương thức ở xa dưới dạng các tham số hay giá trị trả về của phương thức, nên RMI cần có cơ chế cần thiết để tải mã Bytecodes của các đối tượng từ máy ảo này sang máy ảo khác.

Hình dưới đây mô tả một ứng dụng phân tán dưới RMI sử dụng dịch vụ danh bạ để lấy các tham khảo của các đối tượng ở xa.



Hình 5.4 Vai trò của dịch vụ tên

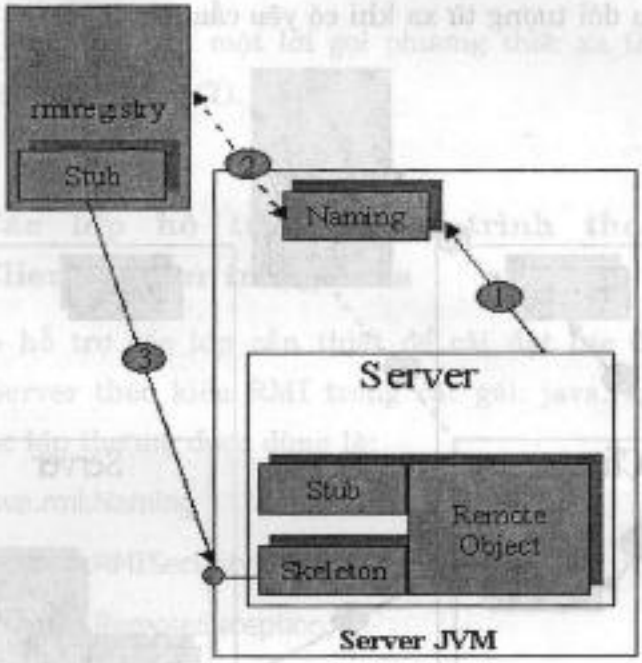
Trong đó:

- Server đăng ký tên cho đối tượng có thể được gọi từ xa của mình với Dịch vụ danh bạ (Registry Server).
- Client tìm đối tượng ở xa thông qua tên đã được đăng ký trên Registry Server (looks up) và tiếp đó gọi các phương thức ở xa.

Hình minh họa cũng cho thấy cách thức mà hệ thống RMI sử dụng một WebServer sẵn có để truyền tải mã bytecodes của các lớp qua lại giữa Client và Server.

5.2.4 Cơ chế vận hành của của một ứng dụng Client-Server theo kiểu RMI

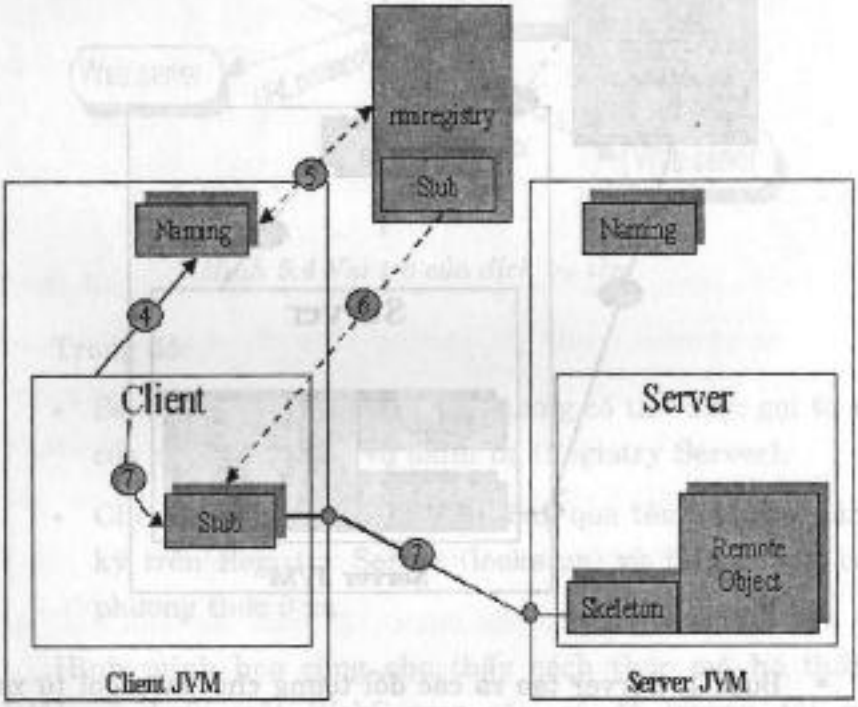
Tiến trình vận hành của một ứng dụng Client-Server theo kiểu RMI diễn ra như sau:



- Bước 1: Server tạo ra các đối tượng cho phép gọi từ xa cùng với các Stub và Skeleton của chúng.
- Bước 2: Server sử dụng lớp Naming để đăng ký tên cho

một đối tượng từ xa (1).

- Bước 3: Naming đăng ký Stub của đối tượng từ xa với Registry Server (2).
- Bước 4: Registry Server sẵn sàng cung cấp tham thảo đến đối tượng từ xa khi có yêu cầu (3).



- Client yêu cầu Naming định vị đối tượng xa qua tên đã được đăng ký (phương thức lookup) với dịch vụ tên (4).

- Naming tải Stub của đối tượng xa từ dịch vụ tên mà đối tượng xa đã đăng ký về Client (5).
- Cài đặt đối tượng Stub và trả về tham khảo đối tượng xa cho Client (6).
- Client thực thi một lời gọi phương thức xa thông qua đối tượng Stub (7).

5.2.5 Các lớp hỗ trợ chương trình theo kiểu Client-Server trong Java

Java hỗ trợ các lớp cần thiết để cài đặt các ứng dụng Client-Server theo kiểu RMI trong các gói: java.rmi. Trong số đó các lớp thường được dùng là:

- java.rmi.Naming
- java.rmi.RMISecurityManager
- java.rmi.RemoteException;
- java.rmi.server.RemoteObject
- java.rmi.server.RemoteServer
- java.rmi. server.UnicastRemoteObject

5.3 XÂY DỰNG MỘT ỨNG DỤNG PHÂN TÁN VỚI RMI

Xây dựng một ứng dụng phân tán bằng cơ chế RMI gồm các bước sau:

1. Thiết kế và cài đặt các thành phần của ứng dụng.
2. Biên dịch các chương trình nguồn và tạo ra Stub và Skeleton.
3. Tạo các lớp có thể truy xuất từ mạng cần thiết.
4. Khởi tạo ứng dụng

5.3.1 Thiết kế và cài đặt các thành phần của ứng dụng.

Đầu tiên bạn phải xác định lớp nào là lớp cục bộ, lớp nào là lớp được gọi từ xa. Nó bao gồm các bước sau:

- *Định nghĩa các giao diện cho các phương thức ở xa (remote interfaces)*: Một remote interface mô tả các phương thức mà nó có thể được kích hoạt từ xa bởi các Client. Đi cùng với việc định nghĩa Remote Interface là việc xác định các lớp cục bộ làm tham số hay giá trị trả về của các phương thức được gọi từ xa.
- *Cài đặt các đối tượng từ xa (remote objects)*: Các Remote Object phải cài đặt cho một hoặc nhiều

Remote Interfaces đã được định nghĩa. Các lớp của Remote Object class cài đặt cho các phương thức được gọi từ xa đã được khai báo trong Remote Interface và có thể định nghĩa và cài đặt cho cả các phương thức được sử dụng cục bộ. Nếu có các lớp làm đối số hay giá trị trả về cho các phương thức được gọi từ xa thì ta cũng định nghĩa và cài đặt chúng.

- *Cài đặt các chương trình Client*: Các chương trình Client có sử dụng các Remote Object có thể được cài đặt ở bất kỳ thời điểm nào sau khi các Remote Interface đã được định nghĩa.

5.3.2 Biên dịch các tập tin nguồn và tạo Stubs và Skeleton

Giai đoạn này gồm 2 bước: Bước thứ nhất là dùng chương trình biên dịch javac để biên dịch các tập tin nguồn như các remote interface, các lớp cài đặt cho các remote interface, lớp server, lớp client và các lớp liên quan khác. Kế tiếp ta dùng trình biện dịch rmic để tạo ra stub và skeleton cho các đối tượng từ xa từ các lớp cài đặt cho các remote interface.

5.3.3 Tạo các lớp có thể truy xuất từ mạng

Tạo một tập tin chứa tất cả các file có liên quan như các remote interface stub, các lớp hỗ trợ mà chúng cần thiết phải tải về Client và làm cho tập tin này có thể truy cập đến thông qua một Web server.

5.3.4 Thực thi ứng dụng

Thực thi ứng dụng bao gồm việc thực thi rmiregistry server, thực thi server, và thực thi client.

Tóm lại các công việc phải làm là:

- Tạo giao diện (interface) khai báo các phương thức được gọi từ xa của đối tượng.
- Tạo lớp cài đặt (implement) cho giao diện đã được khai báo.
- Viết chương trình Server.
- Viết chương trình Client.
- Dịch các tập tin nguồn theo dạng RMI để tạo ra các lớp tương ứng và stub cho client, skeleton cho server.
- Khởi động dịch vụ registry.
- Thực hiện chương trình Server.
- Thực thi chương trình Client.

5.3.5 Ví dụ minh họa

Trong ví dụ này chúng ta định nghĩa một phương thức String sayHello() được gọi từ xa. Mỗi khi phương thức này được kích hoạt nó sẽ trả về chuỗi "Hello World" cho Client gọi nó.

Dưới đây là các bước để xây dựng ứng dụng:

- **Bước 01: Tạo giao diện (interface) khai báo các phương thức được gọi từ xa của đối tượng.**
  - Cú pháp tổng quát:

```
import java.rmi.Remote;
   import java.rmi.RemoteException;
   public interface InterfaceName extends Remote {
      ReturnType remoteMethodOne() throws
   RemoteException;
      ReturnType remoteMethodTwo() throws
   RemoteException;
      . . .
   }
```

  - Định nghĩa remote interface có tên là HelloItf, có phương thức được gọi từ xa là String sayHello() như sau:

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

public interface HelloItf extends Remote {
    public String sayHello() throws RemoteException;
      }
```

Lưu chương trình này vào tập tin HelloItf.java

- **Bước 02: Tạo lớp cài đặt (implements) cho giao diện đã được khai báo:**

- Cú pháp tổng quát:

```
import java.rmi. server.UnicastRemoteObject;
  import java.rmi.RemoteException;
  public class RemoteClass extends UnicastRemoteObject
  implements      InterfaceName {
     public RemoteClass() throws RemoteException {
        super();
        ..... // Implement of Method
     }
     public ReturnType remoteMethodOne() throws
  RemoteException {
        ..... // Implement of Method
     }
     public ReturnType remoteMethodTwo() throws
  RemoteException {
        ..... // Definition of Method
     }
  }
```

- Định nghĩa lớp cài đặt có tên là Hello cài đặt cho remote interface HelloItf

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class Hello extends UnicastRemoteObject implements
HelloItf {
    public Hello() throws RemoteException {
        super();
    }
    // Cai dat ham cho phep goi tu xa
    public String sayHello() {
        return "Hello RMI";
    }
}
```

Lưu chương trình này vào tập tin Hello.java

- **Bước 03: Viết chương trình Server:**
  - Cú pháp tổng quát:

```
import java.rmi.Naming;
    import java.rmi.RemoteException;
    import java.rmi.RMISecurityManager;
    public class ServerName {
        public static void main(String args[]) {
            If (System.getSecurityManager() == null) { //
    Cài đặt cơ chế bảo mật
```

```
            System.setSecurityManager(new
RMISecurityManager());
        }
        try {
            // Tạo các đối tượng từ xa
            RemoteClass remoteObject = new
RemoteClass();

            // Đăng ký tên cho các đối tượng từ xa
            Naming.rebind("RegistryName",
remoteObject);
            ...
        }
        catch (Exception e) {
            System.out.println("Error: . . ." + e);
        }
    }
}
```

- Tạo server có tên HelloServer chứa một đối tượng từ xa obj thuộc lớp cài đặt Hello. Đăng ký tên cho đối tượng obj là HelloObject

```
import java.rmi.*;
import java.net.MalformedURException;
public class HelloServer {
    public static void main(String[] args) {
            // Cai dat co che bao mat
        if(System.getSecurityManager() == null)
          System.setSecurityManager(new RMISecurityManager());
        try {
          // Tao ra doi tuong cho phep goi ham tu xa
          Hello obj = new Hello();
          System.out.println("Tao duoc doi tuong cho phep tu xa");
            // Dang ky doi tuong
            Naming.rebind("HelloObject", obj);
            System.out.println("Da dang ky doi tuong de goi tu xa
thanh cong !!!");
        }
        catch(RemoteException e)
          {System.out.println("Loi trong khi khoi tao doi tuong tu xa");
          }
        catch(MalformedURLException e)
        {
          System.out.println("Loi trong khi dang ky doi tuong tu xa");
        }
    }
}
```

Lưu chương trình này vào tập tin HelloServer.java

- **Bước 04: Viết chương trình Client:**
  - Cú pháp tổng quát:

```
import java.rmi.Naming;
        import java.rmi.RemoteException;
        public class Client {
           public static void main(String args[]) {
              String remoteObjectURL =
        "rmi://NameServer/RegistryName";
              Interfacename object = null;
              try {
                 object =
        (InterfaceName)Naming.lookup(remoteObjectURL);
                 object.remoteMethodOne();
              ...
              }
              catch (Exception e) {
                 System.out.println(" Error: "+ e);
              }
           }
        }
```

  - Tạo client có tên là HelloClient, tìm đối tượng HelloObject trên rmiregistry chẳng hạn tại địa chỉ 172.18.211.160. Gọi phương thức sayHello() và in kết quả trả về ra màn hình.

```
import java.rmi.*;
import java.net.MalformedURException;
public class HelloClient {
    public static void main(String[] args) {
      try {
          // Do tim doi tuong tu xa
          HelloItf ref = (HelloItf) Naming.lookup
       ("rmi://172.18.211.160/HelloObject");
          System.out.println("Tao duoc doi tuong cho phep tu xa");
          // Goi ham tren doi tuong
          String result = ref.sayHello();
          System.out.println("Ket qua nhan duoc la:" + result);
        }
        catch(NotBoundException e)
          { System.out.println("Khong tim thay doi tuong goi tu xa");
          }
        catch(MalformedURLException e)
            { System.out.println(" Sai trong dinh dang URL");}
        catch(RemoteException e)
            { System.out.println(" Loi trong khi goi ham tu xa");}
    }
}
```

Lưu chương trình vào tập tin HelloClient.java

- **Bước 05: Dịch các tập tin nguồn theo dạng RMI để tạo ra các lớp tương ứng và stub cho client, skeleton cho server:**
  - Cú pháp tổng quát:

javac InterfaceName.java RemoteClass.java Server.java Client.java

(Tạo ra các lớp InterfaceName.class RemoteClass.class Server.class Client.class)

rmic RemoteClass

(Tạo ra các lớp cho Skeleton và Stub: **RemoteClass_Skel.class** và **RemoteClass_Stub.class**)

- Biên dịch các lớp trong Hello:

javac Hello.java HelloItf.java HelloServer.java HelloClient.java

rmic Hello.class

- **Bước 06: Khởi động dịch vụ rmiregistry**
  - Cú pháp tổng quát:

start rmiregistry [port]

Cổng mặc định là 1099.

- Khởi động dịch vụ rmiregistry trên cổng mặc định như sau:

Khi đó rmiregistry server sẽ chạy trên một cửa sổ mới, giữ nguyên cửa sổ này, **không đóng nó lại**.

- **Bước 07: Thực hiện chương trình Server**
  - Cú pháp tổng quát:

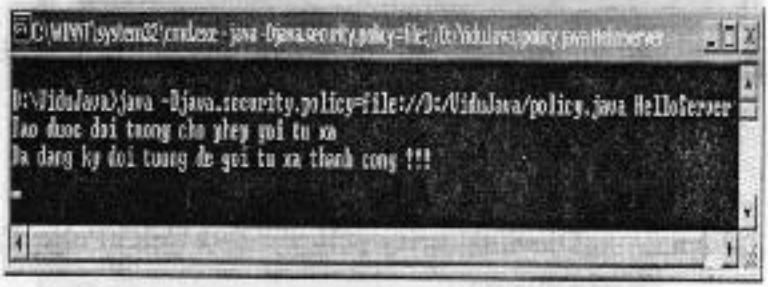
Java -Djava.security.policy =UrlOfPolicyFile ServerName

Trong đó UrlOfPolicyFile là địa chỉ theo dạng URL của tập tin mô tả chính sách về bảo mật mã nguồn của Server (policy file). Nó qui định "ai" (chương trình, máy tính, quá trình trên) sẽ có quyền download các tập tin của nó trong đó có stub. Để đơn giản trong phần này ta cho phép tất cả mọi người đều có quyền download các tập tin của Server. Khi triển khai các ứng dụng thật sự ta phải có các chính sách bảo mật nghiêm ngặt hơn (Tham khảo tài liệu về Security của Java). File policy có dạng như sau:

```
grant {
        // Allow everything for now
        permission java.security.AllPermission;
        };
```

Lưu nội dung trên vào tập tin có tên **policy.java**

- Thực thi HelloServer với địa tập tin plolicy nằm ở thư mục D:\ViduJava\;



- **Bước 08: Thực thi chương trình Client:**
  - Cú pháp tổng quát

java ClientName

  - Thực thi HelloClient với địa chỉ của rmiregistry đưa vào trong tham số

Để thực thi được chương trình HelloClient chỉ cần có hai class nằm cùng thư mục với nó là HelloItf.class và Hello_Stub.class.

BÀI TẬP ÁP DỤNG

- **Mục đích:**

Xây dựng ứng dụng phân tán theo cơ chế RMI.

- **Yêu cầu**

Sinh viên thực hiện các bài tập sau:

- **Bài 1** : Xây dựng một ứng dụng phục vụ việc bán vé máy bay cho các đại lý phân tán ở các tỉnh thành khác nhau. Ứng dụng này có các lớp sau:
  - Lớp chuyến bay: Đại diện cho một chuyến bay
    - Có các thuộc tính: Số hiệu chuyến bay, Ngày giờ bay, Nơi đi, Nơi đến, Thời gian bay, Tổng số ghế, Số lượng ghế đã bán, Số lượng ghế còn trống.
    - Các phương thức trên một chuyến bay: phương thức xem thông tin về chuyến bay, phương thức mua vé, phương thức trả vé. Để phục vụ cho nhiều đại lý các phương thức trên thuộc loại được gọi từ xa.
    - Lớp Server, tạo ra nhiều chuyến bay và duy trì nó để cho phép các đại lý thực hiện các giao dịch trên chuyến bay cụ thể.
    - Client là chương trình cho phép mỗi đại lý được quyền xem thông tin về chuyến bay, mua vé, trả vé theo yêu cầu.

Phụ lục

GỢI Ý GIẢI BÀI TẬP

Chương 2

Chủ đề 1:

Bài 3

```
import java.io.*;
public class DoiChuHoa {
   public static void main(String[] args) {
      try {
            BufferedReader  keyboard = new
        BufferedReader(new InputStreamReader(System.in));
         System.out.print("Nhap vao 1 chuoi ky tu: ");
         String  str = keyboard.readLine();
         str = str.toUpperCase();
         System.out.println("Doi thanh chu Hoa: " + str);
      }
      catch(IOException e) {
         System.out.println("Loi khi nhap xuat chuoi");
      }
   }
}
```

Bài 4

```
import java.io.*;
public class SoNguyenTo {
   public static boolean LaSoNguyenTo(int  x) {
      for(int i=x/2; i>=2; i--)
         if (x % i == 0)
            return false;
      return true;
   }
   public static void main(String[] args) {
      try {
            BufferedReader  keyboard = new BufferedReader(new
         InputStreamReader(System.in));
         System.out.print("Nhap vao 1 so nguyen: ");
         String  str = keyboard.readLine();
         int  n = Integer.valueOf(str).intValue();
         if(n<=0)
            System.out.println("Khong phai la so nguyen duong");
         else
            if(SoNguyenTo.LaSoNguyenTo(n))
               System.out.println(n + " la so nguyen to");
            else
               System.out.println(n + " khong la so nguyen to");
      }
      catch(NumberFormatException e) {
         System.out.println("Khong phai la so nguyen");
      }
      catch(IOException e) {
         System.out.println("Loi khi nhap xuat chuoi");
      }
   }
}
```

Bài 5

```
import java.io.*;
public class PTB2 {
   public static void main(String[] args) {
      try {
         BufferedReader  keyboard = new BufferedReader(new
         InputStreamReader(System.in));
         System.out.print("Nhap a: ");
         String  str = keyboard.readLine();
         float  a = Float.valueOf(str).floatValue();
         System.out.print("Nhap b: ");
         str = keyboard.readLine();
         float  b = Float.valueOf(str).floatValue();
         System.out.print("Nhap c: ");
         str = keyboard.readLine();
         float  c = Float.valueOf(str).floatValue();
         if(a==0)   // Phuong trinh bac 1
            if(b==0)
               if(c==0)
               System.out.println("Phuong trinh co vo so nghiem");
               else
                  System.out.println("Phuong trinh vo nghiem");
            else
              System.out.println("Phuong trinh co 1 nghiem la " + (-
      c/b) );
         else {
            float delta = b*b - 4*a*c;
            if(delta<0)
               System.out.println("Phuong trinh vo nghiem");
            else
               if(delta==0)
                  System.out.println("Phuong trinh co nghiem kep
```

```
            x1 = x2 = " +  -b/(2*a) );
               else {
                        System.out.println("Phuong trinh co 2
                        nghiem phan biet:");
                        System.out.println("  x1 = " + (-b +
                        Math.sqrt(delta)) / (2*a) );
                        System.out.println("  x2 = " + (-b  -
                        Math.sqrt(delta)) / (2*a) );
               }
         }
      }
      catch(NumberFormatException e) {
            System.out.println("Nhap he so khong phai la so
            nguyen");
      }
      catch(IOException e) {
         System.out.println("Loi khi nhap xuat chuoi");
      }
   }
}
```

Bài 6 :

```
import java.io.*;
public class TinhTong {
   public static void main(String[] args) {
      try {
            BufferedReader  keyboard = new BufferedReader(new
         InputStreamReader(System.in));
         System.out.print("Nhap n: ");
         String  str = keyboard.readLine();
         int  n = Integer.valueOf(str).intValue();
```

```
            int  tong =0;
            for(int i=1; i<=n ; i++)
               tong += i;
            System.out.println("Tong tu 1 den " + n + " = " +     tong);
         }
         catch(NumberFormatException e) {
            System.out.println("Khong phai la so nguyen");
         }
         catch(IOException e) {
            System.out.println("Loi khi nhap xuat chuoi");
         }
      }
   }
```

Bài 7

```
import java.io.*;
public class TinhTong1 {
   public static void main(String[] args) {
      try {
            BufferedReader  keyboard = new BufferedReader(new
         InputStreamReader(System.in));
         System.out.print("Nhap so luong phan tu trong day so: ");
         String  str = keyboard.readLine();
         int  n = Integer.valueOf(str).intValue();
         float  ds[] = new float[n];
         float ketqua = 0;
         // Nhap gia tri cho day so
         for(int i=0; i<n; i++) {
```

```
            try {
               System.out.print("Nhap phan tu thu " + (i+1) + " : ");
               str = keyboard.readLine();
               ds[i] = Float.valueOf(str).floatValue();
            }
            catch(NumberFormatException e) {
               System.out.println("Khong phai la so thuc");
            }
            catch(IOException e) {
                     System.out.println("Loi khi nhap xuat chuoi");
            }
         }
         for(int i=0; i<n; i++)
            if(ds[i]>0)
               ketqua += ds[i];
             System.out.println("Tong cac so duong trong day la: "
         + ketqua);

      }
      catch(NumberFormatException e) {
         System.out.println("Khong phai la so nguyen");
      }
      catch(IOException e) {
         System.out.println("Loi khi nhap xuat chuoi");
      }
   }
}
```

Chủ đề 2:

Bài 1

```
import java.io.*;
class Diem {
   private int x;
   private int y;
   public Diem() {
      x = y = 0;
   }
   public Diem(int h, int t) {
      x = h; y= t;
   }
   public void Gan(int h, int t) {
      x= h; y=t;
   }
   public void Nhap() {
      try {
            BufferedReader keyboard = new BufferedReader(new
         InputStreamReader(System.in));
         System.out.print("  Nhap hoanh do: ");
         String str = keyboard.readLine();
         x = Integer.valueOf(str).intValue();
         System.out.print("  Nhap tung do: ");
         str = keyboard.readLine();
         y = Integer.valueOf(str).intValue();
      }
      catch(NumberFormatException e) {
         System.out.println("Nhap sai dinh dang");
      }catch(IOException e) {
         System.out.println("Co loi trong khi nhap");
      }
   }
```

```
    public void In() {
        System.out.println( "(" + x + "," + y + ")" );
    }
    public double  KhoangCach() {
        return Math.sqrt( x*x + y*y );
    }

    public static void main(String[] args) {
        Diem  a = new Diem(2,7);
        Diem b = new Diem();
        System.out.print("Diem A co toa do "); a.In();
        System.out.println("\nNhap gia tri diem B");
        b.Nhap();
        System.out.println("Khoang cach tu B den goc toa do: "
                           +  b.KhoangCach());
    }
}
```

Bài 2 :

```
import java.io.*;
class PhanSo  {
    private int  tu;
    private int  mau;
    public PhanSo() {
        tu = 0; mau = 1;
    }
    public PhanSo(int  t, int  m) {
        tu = t; mau = m;
    }
```

```
public void Nhap() {
   while (true) {
      try {
          BufferedReader  keyboard = new BufferedReader(new
      InputStreamReader(System.in));
         System.out.print("  Nhap tu so: ");
         String str = keyboard.readLine();
         tu = Integer.valueOf(str).intValue();
         System.out.print("  Nhap mau so: ");
         str = keyboard.readLine();
         mau = Integer.valueOf(str).intValue();
         if(mau!=0)    break;
         System.out.println("Mau so = 0 - Nhap lai !!!");
      }
      catch(NumberFormatException  e) {
         System.out.println("Nhap sai dinh dang");
      }
      catch(IOException  e) {
         System.out.println("Co loi trong khi nhap");
      }
   }
}

public void In() {
   if(tu==0)
      { System.out.print(0); return; }
   if(mau==1)
      { System.out.print(tu); return; }
   if(tu*mau<0)
      System.out.print("-");
   System.out.print( Math.abs(tu) + "/" + Math.abs(mau) );
}
```

```
    public void  NghichDao() {
        if(tu==0)
            System.out.print("Khong the nghich dao duoc");
        else {
            int temp = tu;
            tu = mau;
            mau = temp;
        }
    }
    public float  GiaTriThuc() {
        return (float)tu/mau;
    }
    public void RutGon() {
        int  min = Math.min(tu, mau);
        for(int i=min; i>=2; i--)
            if (tu%i==0 && mau % i ==0)
                { tu /= i; mau /= i; return; }
    }

    public  PhanSo  Cong (PhanSo  a) {
        PhanSo  kq = new PhanSo(tu*a.mau + a.tu*mau,
mau*a.mau);
        kq.RutGon();
        return kq;
    }

    public  PhanSo  Tru (PhanSo  a) {
        PhanSo  kq = new PhanSo(tu*a.mau - a.tu*mau,
mau*a.mau);
        kq.RutGon();
        return kq;
    }
```

```
    public PhanSo Nhan (PhanSo a) {
        PhanSo kq = new PhanSo(tu*a.tu, mau*a.mau);
        kq.RutGon();
        return kq;
    }
    public PhanSo Chia (PhanSo a) {
        If(a.tu==0) {
            System.out.println("Khong the chia cho phan so 0");
            return new PhanSo();
        }
        PhanSo kq = new PhanSo(tu*a.mau , mau*a.tu);
        kq.RutGon();
        return kq;
    }
    public static void main(String[] args) {
        PhanSo a = new PhanSo();
        PhanSo b = new PhanSo();
        System.out.println("Nhap phan so a");
        a.Nhap();
        System.out.println("Nhap phan so b");
        b.Nhap();
        a.In(); System.out.print(" + ");
        b.In(); System.out.print(" = "); a.Cong(b).In();
    }
}
```

Chủ đề 3:

Bài 1

```
class  NguoiSX extends Thread {
    private  KhoHang  kho;
   int    max; // So luong san pham san xuat toi da
   public   NguoiSX(KhoHang  kh, int  m)
         {  kho=kh; max = m; }
   public  void sanXuat() {
      int sl= (int) (Math.random()*max);
      if(sl==0)   return;
        System.out.println("San xuat duoc " + sl + "  san pham");
        System.out.println("Trong kho dang co " +
kho.laySoLuong()+ "  san pham");
        int  slnhap = kho.nhapThem(sl);
        System.out.println(" Nhap them " + slnhap + "  san pham");
        System.out.println("     Trong kho dang co " +
kho.laySoLuong()+ "  san pham");
         if(kho.hetCho()) {
            System.out.println("--------  Da het cho trong kho - Tam
         ngung san xuat  -------");
         try{   wait();  }
               catch (Exception  e) {}
           }
     }
     public  void  run()  {
        while(true) {
           sanXuat();
           try{  sleep((int) (Math.random() + 3000)) ;   }
           catch (Exception  e) {}
        }
     }
 }
```

```
class NguoiTD extends Thread
{    private  KhoHang kho;
     int max; // So luong san pham tieu dung toi da
     int tongso;     // tong so hang hoa khach co nhu cau
     public  NguoiTD(KhoHang  kh, int  m)
     {  kho= kh; tongso =0; max= m; }

     public  void tieuDung() {
        int sl= (int) (Math.random()*max);
        if(sl==0) return;
        System.out.println("Nhu cau them " + sl + "  san pham");
        tongso += sl;
        System.out.println("Tong nhu cau " + tongso + "  san
pham");
         int tdduoc = kho.tieuThu(tongso);
        System.out.println("     Da tieu dung duoc " + tdduoc + "
san pham");
        tongso -= tdduoc;
        System.out.println("     Tong nhu cau con lai " + tongso +
"  san pham");
        System.out.println();
     }

     public  void  run () {
        while(true) {
           tieuDung();
           try
              {  sleep((int) (Math.random() + 3000)) ;   }
             catch (Exception  e) {}
           }
       }
  }
```

```
class KhoHang
  {  private int soluong;// so luong san pham hien co trong kho
     private int    succhua;   // suc chua toi da cua kho
     private boolean cosanpham; // dang co hang
     private boolean controng;     // co`n cho tro'ng trong kho de
them hang
     public   KhoHang(int  sc) {
        soluong=0; succhua = sc;
        cosanpham=false;  controng=true;
        }
     public int  laySoLuong() { return soluong; }
     public boolean  hetCho() {
        return (soluong==succhua);
     }
     public  synchronized  Int tieuThu(int m) {
        while (cosanpham==false) {
              try
                 {  wait ();  }
              catch ( Exception  e ) {}
           }
        int  ketqua = m;
        if(soluong > m)
            soluong -= m;
        else {
           ketqua = soluong;
           soluong = 0;
           cosanpham=false;
         }
        controng = true; notify();
        return ketqua;
     }
```

```
    public synchronized int nhapThem( int n) {
        while (controng==false) {
            try
              { wait (); }
            catch ( Exception e ) {}
          }
        int ketqua = n;
        if( succhua - soluong > n)
          soluong += n;
        else {
          ketqua = succhua - soluong;
          soluong = succhua;
          controng=false;
        }
        cosanpham= true; notify();
        return ketqua;
      }
}

public class SanXuatTieuDung
  { public static void main(String[] args)
      {  KhoHang kho= new KhoHang(7);
         NguoiSX sx1= new NguoiSX (kho, 6);
         NguoiSX sx2= new NguoiSX (kho, 6);
         NguoiTD td1= new NguoiTD (kho, 5);
         NguoiTD td2= new NguoiTD (kho, 5);
         sx1.start (); td1.start(); sx2.start(); td2.start();
      }
  }
```

CHƯƠNG 4

Chủ đề 1

Bài 1

```
import java.io.*;
import java.net.Socket;
public class SimpleWebClient {
   public static void main(String args[])
   {
    BufferedReader bp = new BufferedReader (new
InputStreamReader(System.in));
    try
      {  System.out.print("Nhap vao dia chi Web can truy cap :");
       String diachi = bp.readLine();
       // Noi ket voi Web Server o dia chi tren
       Socket s = new Socket(diachi, 80);
       // Tao cac luong nhap xuat den Server Web
       InputStream is = s.getInputStream();
       OutputStream os = s.getOutputStream();
           BufferedReader br = new BufferedReader (new
       InputStreamReader(is));
       PrintWriter   pw = new PrintWriter(os);
       // Nhap vao ten file HTML can truy xuat :
          System.out.print("Nhap vao path den 1 file HTML : ");
       String  path = bp.readLine();
       String  yeucau = "GET /" + path + " HTTP/1.0\r\n";
          // Gui yeu cau den Server theo cau lenh cua protocol
       HTTP
       pw.println(yeucau); pw.flush();
       // Nhan du lieu ve
       String ketqua;
       while ((ketqua = br.readLine()) != null) // Khi het file
```

```
        {  System.out.println(ketqua);
           // Hoac kiem tra khi ket thuc trang Web
           if ( ketqua.indexOf("</HTML>") != -1 )
              break;
        }
        // Ket thuc noi ket
        br.close(); pw.close(); s.close();
      }
    catch (IOException ie) { System.out.println("Loi noi ket hoac
nhap xuat"); }
  }
}
```

Chủ đề 2:

Bài 1

```
import java.net.*;
import java.io.*;
public class ClientDocSo {
   public static void main(String[] args)  {
      try {
         Socket  s = new Socket(args[0], 2008);
         InputStream  is  =s.getInputStream();
         OutputStream os  =s.getOutputStream();
         while(true) {
            System.out.print("Nhap 1 ky tu so : ");
            int  i = System.in.read();
            os.write(i); System.in.skip(2);
            if(i=='@')  break;
            // Nhan ve n ky tu tu Server
            byte  b[]  = new byte[50];
```

```
            int n = is.read(b);
            String ketqua = new String(b, 0, n);
            System.out.println("Doc la: " + ketqua);
         }
      }
      catch(IOException e) {
         System.out.println("Co loi khi noi ket den Server");
      }
   }
}
import java.net.*;
import java.io.*;
public class ServerDocSo {
   public static void main(String[] args) {
      try {
      ServerSocket ss = new ServerSocket(2008);
      System.out.println("Da khoi tao xong Server Socket !!!");
      while(true) {
         try {
            Socket s = ss.accept();
             System.out.println("  Co 1 Client (dia chi " +
             s.getInetAddress().toString() +
                        " , cong " + s.getPort() + ")  noi ket den");
            InputStream is =s.getInputStream();
            OutputStream os =s.getOutputStream();
            while(true) {
               int ch = is.read();
               if(ch=='@') break;
               // Xu ly yeu cau cua Client (Cach thu 1)
               String ketqua="";
```

```
                switch(ch) {
                    case '0':  ketqua = "Khong";  break;
                    case '1':  ketqua = "Mot";    break;
                    case '2':  ketqua = "Hai";    break;
                    case '3':  ketqua = "Ba";   break;
                    case '4':  ketqua = "Bon";    break;
                    case '5':  ketqua = "Nam";    break;
                    case '6':  ketqua = "Sau";    break;
                    case '7':  ketqua = "Bay";    break;
                    case '8':  ketqua = "Tam";    break;
                    case '9':  ketqua = "Chin";   break;
                    default:    ketqua="Khong phai la so";
                }
                // Gui ket qua ve cho Client
                os.write(ketqua.getBytes());
            }
            System.out.println("  Dong noi ket voi Client (" +
    s.getInetAddress().toString() +" , " + s.getPort() + ")");
            s.close();
          }
          catch(IOException  e) {
            System.out.println("Co loi khi phuc vu 1 Client");
          }
        }
      }
      catch (IOException  e)  {
        System.out.println("Co loi khi tao Server Socket");
      }
    } // main
} // class
```

```
import java.net.*;
import java.io.*;
public class ServerDocSoSongSong {
   public static void main(String[] args) {
      try {
         ServerSocket ss = new ServerSocket(2008);
         System.out.println("Da khoi tao xong Server Socket");
         while(true) {
            try {
               Socket s = ss.accept();
               System.out.println("  Co 1 Client (dia chi " +
s.getInetAddress().toString() +" , cong " + s.getPort() + ")  noi ket
den");
               DocSo t1 = new DocSo(s);
               t1.start();
            }
            catch(IOException e) {
               System.out.println("Co loi khi phuc vu 1 Client");
            }
         }
      }
      catch (IOException e) {
         System.out.println("Co loi khi tao Server Socket");
      }
   } // main
} // class

class DocSo extends Thread
{ Socket s;
   public DocSo(Socket s1) {
      s = s1;
   }
```

```
    public void run() {
        try {
        InputStream is =s.getInputStream();
        OutputStream os =s.getOutputStream();
        while(true) {
                int ch = is.read();
                if(ch=='@') break;
                // Xu ly yeu cau cua Client (cach thu 2)
                String ketqua="";
                String doc[] = {"Khong", "Mot", "Hai", "Ba", "Bon",
"Nam", "Sau", "Bay", "Tam", "Chin"};
                if(ch<'0' || ch>'9')
                   ketqua = "Khong phai la so";
                else
                   ketqua=doc[ch - '0'];
                // Gui ket qua ve cho Client
                os.write(ketqua.getBytes());
            } //  while
            System.out.println("  Dong noi ket voi Client (" +
     s.getInetAddress().toString() +" , " + s.getPort() + ")");
                s.close();
        }
        catch(IOException e) {
           System.out.println("Co loi khi phuc vu Client");
        }
    } // run
}; // class
```

Bài 2

```
import java.net.*;
import java.io.*;
public class ClientLamToan {
   public static void main(String[] args) {
      try {
         Socket s = new Socket(args[0], 2000);
            // Tao ra kenh giao tiep ao
         InputStream in = s.getInputStream();
         OutputStream out = s.getOutputStream();
         BufferedReader key = new BufferedReader(new
         InputStreamReader(System.in));
         BufferedReader br = new BufferedReader(new
         InputStreamReader(in));
         PrintWriter pw = new PrintWriter(out);
            // Gui va nhan du lieu
         while(true) {
               System.out.print("Nhap bieu thuc tinh toan :");
               String str = key.readLine();
               if(str.equals("EXIT")) {
                  pw.println(str); pw.flush();
                  break;
               }
               // Xu ly de gui theo dung dinh dang
               String chuoi1="", chuoi2="";
               char pheptoan=' ';
               for(int i=0;i<str.length(); i++)
                  if(str.charAt(i)=='+' || str.charAt(i)=='-' ||
                  str.charAt(i)=='*' || str.charAt(i)=='/')
                     { chuoi1 = str.substring(0, i);
                        chuoi1= chuoi1.trim();
                        chuoi2 = str.substring(i+1);
```

```
                    chuoi2 = chuoi2.trim();
                    pheptoan = str.charAt(i);
                    break;
                    }
                // Tao ra chuoi de gui cho Server
            String  yeucau = "" + pheptoan + " " + chuoi1 + " " +
          chuoi2;
            // gui yeu cau qua Server
            pw.println(yeucau); pw.flush();
            // Nhan tu Server
            String  str1 = br.readLine();
            System.out.println("Ket qua  tu Server: " + str1);
          }
            s.close();
        }
       catch(IOException  e) {
          System.out.println("Khong tao duoc  Socket!!!");
       }
    }
}
//-------------------------
import java.net.*;
import java.io.*;
////////////////////////////////  Lop phuc vu cho qua trinh noi ket
//////////////
public class  ServerLamToan {
   public static void main(String[] args) {
      try {
        ServerSocket  ss = new ServerSocket(2000);
        System.out.println("Da khoi tao xong Server !!! ");
        int count=0;
         while(true) {
```

```
            Socket  s = ss.accept();
            System.out.println("Co 1 Client noi ket");
            PhucVu  pv = new PhucVu(s);
            pv.start();
            System.out.println("Da cho Thread thu " + (++count) +
" phuc vu");
          }
       }
       catch(IOException  e) {
          System.out.println("Khong tao duoc Server Socket!!!");
       }
   }
}
////////////////////////////////////  Lop dung de phuc vu cho 1
Client  ////////////////////
class PhucVu extends Thread
{  Socket  s;
   public PhucVu(Socket  s1) {
        s = s1;
   }
   public void run() {
      try{
            // Tao ra kenh giao tiep ao
            InputStream     in  = s.getInputStream();
            OutputStream  out = s.getOutputStream();
            BufferedReader  br = new BufferedReader(new
     InputStreamReader(in));
            PrintWriter        pw = new PrintWriter(out);
            // Gui va nhan du lieu
            while(true) {
               String  str = br.readLine();
               if(str.equals("EXIT"))   break;      // Xu ly phep toan
```

```
            char pheptoan = str.charAt(0);
            int vitri = str.lastIndexOf(' ');
            String chuoi1 = str.substring(2,vitri);
            String chuoi2 = str.substring(vitri+1);
            try {
               int x1 = Integer.valueOf(chuoi1).intValue();
               int x2 = Integer.valueOf(chuoi2).intValue();
               int ketqua=0; String str1 = "";
               switch(pheptoan) {
                  case '+': ketqua = x1 + x2; break;
                  case '-': ketqua = x1 - x2; break;
                  case '*': ketqua = x1 * x2;
               }
               if(pheptoan=='/')
                  str1 = str1 + (float)x1/x2;
               else
                  str1 = str1 + ketqua;
                  pw.println(str1); pw.flush();
               }
               catch(NumberFormatException e) {
                  pw.println(new String("Khong phai la so - Khong
      tinh toan duoc !"));
                  pw.flush();
               }
         }
         System.out.println("Dong ket noi voi Client");
         s.close();
      } catch(IOException e) {
        System.out.println("Co loi trong khi thuc thi Client");
      }
   }
}
```

Chủ đề 3:

Bài 1

```
import java.net.*;
import java.io.*;
class  UDPChatClient {
   public static void main(String[] args) {
      try {
         // Tao UDP Socket
         DatagramSocket  ds = new DatagramSocket();
         // Tao datagram de nhan
         byte b[] = new byte[1000];
         DatagramPacket  nhan = new DatagramPacket(b, 1000);

         // Nhap dia chi cua Server tu ban phim
         BufferedReader  keyboard = new BufferedReader(new
InputStreamReader(System.in));
         System.out.print("Nhap vao dia chi cua Server: ");
         String address = keyboard.readLine();
         while(true) {
               //  Nhap tu ban phim
               System.out.print("My computer: ");
               String  str1 = keyboard.readLine();
               // Kiem tra dieu kien de thoat
               if(str1.equals("EXIT"))  break;
               // Tao ra datagram de gui
               byte  b1[] = str1.getBytes();
               int port = 30000;
      InetAddress  server_address =
   InetAddress.getByName(address);
                  DatagramPacket  gui = new DatagramPacket(b1,
b1.length, server_address, port); // Gui datagram qua Server
                  ds.send(gui);
```

```
                // Nhan datagram tu Server
                ds.receive(nhan);
                // Hien thi noi dung cua Server gui qua
                String  str = new String( nhan.getData(), 0,
            nhan.getLength() );
                System.out.println("  Remote computer: " + str);

            }
        }
        catch(IOException  e) {
            System.out.println("Co loi khi tao Socket UDP");
        }
    }
}

import java.net.*;
import java.io.*;
class  UDPChatServer {
    public static void main(String[] args)  {
        try {
            // Tao UDP Socket tren cong 30000
            DatagramSocket  ds = new DatagramSocket(30000);
            System.out.println("---------  Da khoi tao Server ------  ");
            // Nhan datagram tu Client
            byte b[] = new byte[1000];
            DatagramPacket  nhan = new DatagramPacket(b, 1000);
            BufferedReader  keyboard = new BufferedReader(new
        InputStreamReader(System.in));
            while(true) {
                ds.receive(nhan);
                    // Hien thi noi dung cua Client gui qua
              String  str=new String(nhan.getData(),0, nhan.getLength());
```

```
                String  remote_add = nhan.getAddress().toString();
                System.out.println(remote_add + ": " + str);
                //  Nhap tu ban phim
                System.out.print("  My Computer: ");
                String  str1 = keyboard.readLine();
                // Kiem tra dieu kien de thoat
                if(str1.equals("EXIT"))  break;
                // Tao ra datagram de gui
                byte  b1[] = str1.getBytes();
                DatagramPacket  gui = new DatagramPacket(b1,
              b1.length, nhan.getAddress(), nhan.getPort());
            // Gui datagram qua Client
            ds.send(gui);
          }
       }
       catch(IOException  e) {
          System.out.println("Loi khi tao Socket UDP");
       }
    }
}
```

CHƯƠNG 5

Bài 1

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import java.io.*;
class ChuyenBay   extends  UnicastRemoteObject
                        implements ChuyenBayItf
{   private String  sohieu, ngay, gio, noidi, noiden;
    private int  thoigian, soghe, soghedaban;
    // Ham xay dung
```

```
    public  ChuyenBay() throws RemoteException {
       super();
       sohieu = ngay = gio = noidi = noiden = "";
       thoigian = soghe = soghedaban = 0;
    }
   // Ham cuc bo chi duoc goi ben trong Server
   public void nhapThongTin() {
      try {
      BufferedReader  key = new BufferedReader(new
InputStreamReader(System.in));
      System.out.print("Nhap so hieu chuyen bay: ");
     sohieu = key.readLine();
      System.out.print("Nhap ngay bay: "); ngay = key.readLine();
      System.out.print("Nhap gio bay: "); gio = key.readLine();
      System.out.print("Nhap noi di: "); noidi = key.readLine();
      System.out.print("Nhap noi den: "); noiden = key.readLine();
      System.out.print("Nhap thoi gian bay: ");
      String temp = key.readLine();
      thoigian = Integer.valueOf(temp).intValue();
      System.out.print("Nhap tong so ghe: ");
      temp = key.readLine();
      soghe = Integer.valueOf(temp).intValue();
      }
      catch(IOException  e)
      { System.out.println("Loi ve nhap xuat !");}
   }

  // Ham cho phep goi tu xa tu` cac Client
   public String  xemThongTin() {
        String ketqua = "";
        ketqua = "Chuyen bay so hieu " + sohieu + "\n";
        ketqua += "    Gio khoi hanh: " + gio + "\n";
```

```
        ketqua += "   Ngay: " + ngay + "\n";
        ketqua += "   Lo trinh: " + noidi + " - " + noiden + "\n";
        ketqua += "   Thoi gian bay: " + thoigian + "\n";
        ketqua += "   Tong so ghe: "  + soghe + "\n";
        ketqua += " So ghe con trong:  " + (soghe - soghedaban);
        return ketqua;
    }

    public boolean  muaVe(int  sove) {
       // Kiem tra so ve
       if (soghe-soghedaban>=sove) {
          soghedaban += sove;
          return true;
       }
       else
          return false;
    }

    public boolean  traVe(int sove) {
       soghedaban -= sove;
       return true;
    }
}

//---------------------------------------
import java.rmi.*;
public interface  ChuyenBayItf extends  Remote
{   // Khai bao cac phuong thuc cho phep goi tu xa
    public String  xemThongTin() throws RemoteException;
    boolean  muaVe(int  sove)  throws RemoteException;
    boolean  traVe(int sove) throws RemoteException;
}
```

```
import java.rmi.*;
import java.io.*;
class Client {
   public static void main(String[] args) {
      try {
         BufferedReader  keyboard = new BufferedReader(new
InputStreamReader(System.in));
         System.out.print("Nhap vao dia chi cua Server: ");
         String server_add = keyboard.readLine();
         // Do tim doi tuong tu xa
         ChuyenBayItf  c1 = (ChuyenBayItf) Naming.lookup("rmi://"
         + server_add + "/chuyenbay1");
         ChuyenBayItf  c2 = (ChuyenBayItf) Naming.lookup("rmi://"
         + server_add + "/chuyenbay2");
         //  Goi ham tren doi tuong
         System.out.println("1." + c1.xemThongTin());
         System.out.println("2." + c2.xemThongTin());
         System.out.print("Chon chuyen bay de mua ve (1 hoac 2): ");
         String temp = keyboard.readLine();
         int stt = Integer.valueOf(temp).intValue();
         System.out.print("Nhap so luong ve can mua: ");
          temp = keyboard.readLine();
         int sove = Integer.valueOf(temp).intValue();
         if(stt==1)
            c1.muaVe(sove);
         if(stt==2)
            c2.muaVe(sove);
```

```
            // Co the thu them phuong thuc tra ve hoac mua ve them
        }
        catch(Exception e)
        { System.out.println("Co loi khi thuc thi"); }
    }
}
//-----------------------------------------------
import java.rmi.*;
import java.net.MalformedURLException;
class Server {
    public static void main(String[] args) {
        // Kiem tra co che bao mat cho RMI
        if(System.getSecurityManager()==null)
            System.setSecurityManager(new RMISecurityManager());
        // Tao doi tuong goi tu xa
        try {
            ChuyenBay  cb1 = new ChuyenBay();
            ChuyenBay  cb2 = new ChuyenBay();
            // Nhap thong tin cho cac chuyen bay
            System.out.println("---- Nhap thong tin cho chuyen bay thu
nhat ----");
            cb1.nhapThongTin();
            System.out.println("---- Nhap thong tin cho chuyen bay thu
hai ----");
            cb2.nhapThongTin();
            // Dang ky doi tuong cho phep goi tu xa
            Naming.rebind("chuyenbay1", cb1);
            Naming.rebind("chuyenbay2", cb2);
            System.out.println("Da dang ky xong doi tuong cho goi tu
xa !!!");
        }
```

```
        catch(RemoteException  e) {
            System.out.println("Co loi khi tao doi tuong");
        }
        catch(MalformedURLException e)
        { System.out.println("Khong dang ky duoc doi tuong"); }

    }
}
```

Mục lục

GIÁO TRÌNH

LẬP TRÌNH TRUYỀN THÔNG

Giá: 40.000đ