CHƯƠNG III

KỸ THUẬT XÂY DỰNG ỨNG DỤNG MẠNG PHÍA SERVER

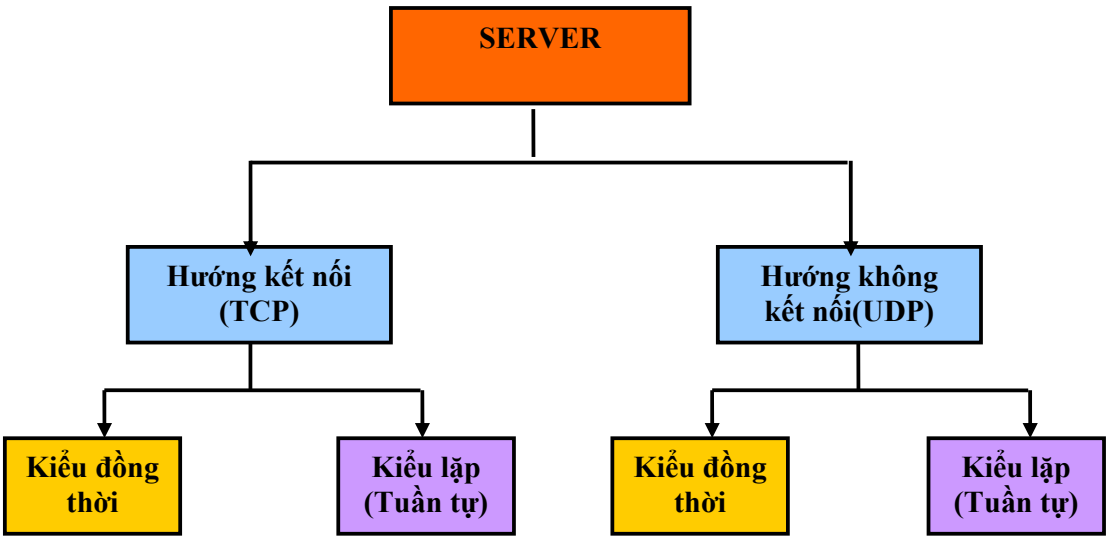
I. GIỚI THIỆU VỀ CÁC KIỂU SERVER

Trong mô hình client/server, chương trình server đóng vai trò phục vụ yêu cầu gửi tới từ chương trình client. Chương trình server có thể phục vụ một hoặc nhiều client đồng thời hoặc phục vụ kiểu lặp.

Server có thể phân thành các loại sau:

- Server chạy chế độ đồng thời hướng không kết nối(TCP)
- Server chạy chế độ lặp hướng không kết nối(TCP)
- Server chạy chế độ đồng thời hướng không kết nối(UDP)
- Server chạy chế độ lặp hướng không kết nối(UDP)

và sự phân loại này được thể hiện như hình 3.



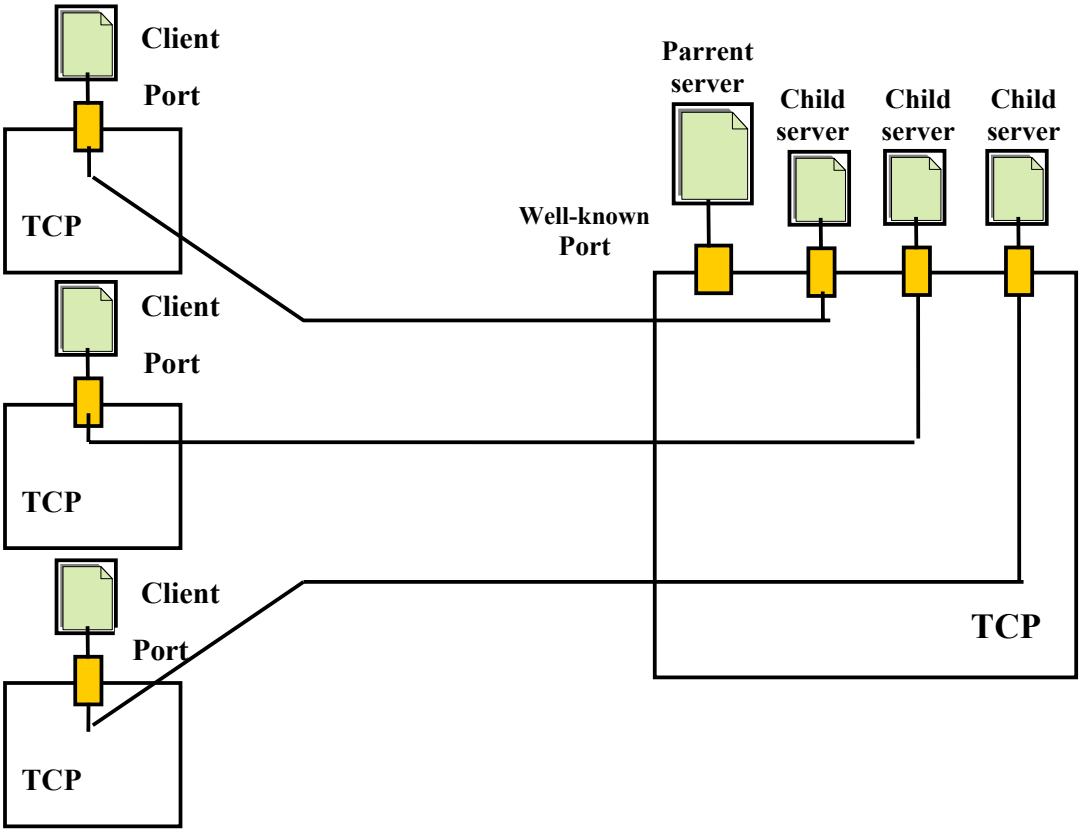
Hình 3.1. Các kiểu server

Trong các kiểu server này, kiểu server đồng thời hướng kết nối và server kiểu lặp hướng không kết nối được sử dụng phổ biến. Chính vì vậy chúng ta chỉ tập trung vào xét 2 kiểu server này.

1. Server chạy chế độ đồng thời hướng kết nối

Đây là loại server chuẩn, sử dụng giao thức truyền thông TCP. Server này có thể phục vụ nhiều client đồng thời. Kết nối được thiết lập giữa server với mỗi client và kết nối được duy trì hoạt động cho đến khi toàn bộ luồng được xử lý, cuối cùng kết nối được kết thúc. Server hướng kết nối đồng thời không thể chỉ sử dụng một cổng đã biết bởi mỗi kết nối cần một địa chỉ cổng và có nhiều kết nối sẽ được thiết lập tại cùng thời điểm. Chính vì vậy server phải sử dụng nhiều cổng và nó chỉ sử dụng một cổng biết rõ trước. Khi khởi tạo, server sẽ thực hiện mở thụ động tại cổng biết rõ đó và đặt ở trạng thái nghe tín hiệu đến kết nối từ client. Mỗi khi có một client thiết lập kết nối với server qua cổng đó, server sẽ sinh ra các server con với một số cổng khác để phục vụ client đó. Còn server chính sẽ tiếp tục đặt ở trạng thái nghe tín hiệu kết nối khác.Server cũng có thể sử dụng bộ đệm cho mỗi kết nối. Các segment truyền từ client

tới sẽ được cất vào bộ đệm phù hợp và sẽ được phục vụ đồng thời bởi server. Mô hình hoạt động của server này được thể hiện như hình 3.2.



Hình 3.2. Mô hình server phục vụ đồng thời hướng kết nối

2. Server chạy chế độ lặp hướng không kết nối

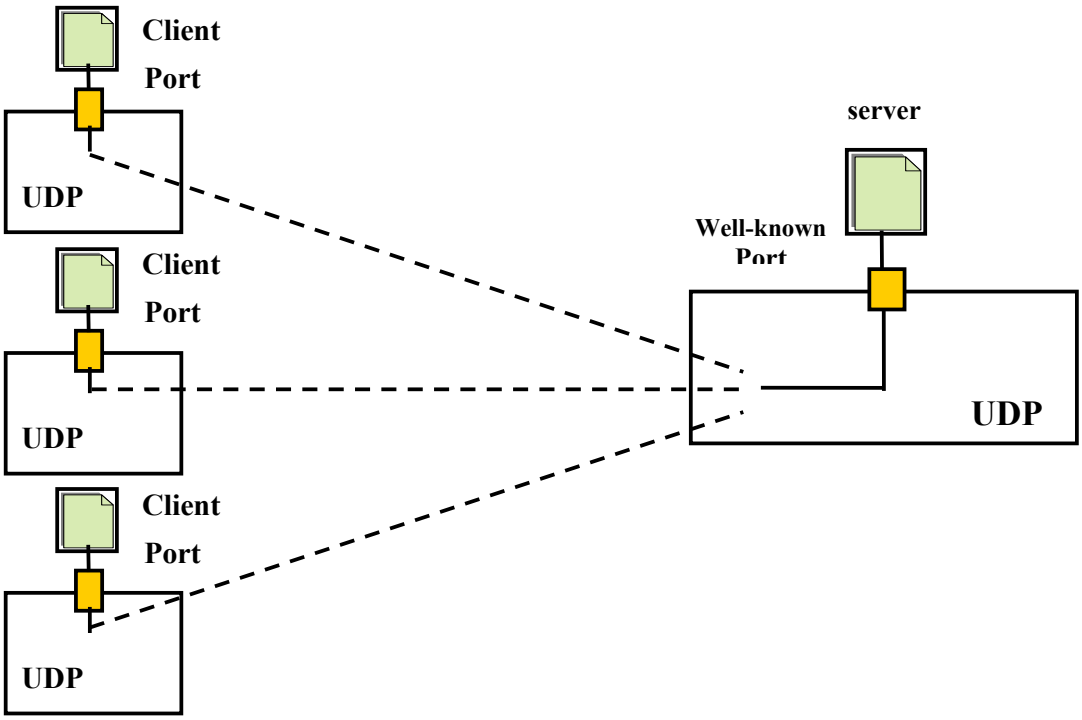
Server kiểu lặp hướng không kết nối thường sử dụng giao thức UDP. Trong kiểu server này, tại mỗi thời điểm nó chỉ xử lý một yêu cầu. Server lấy yêu cầu từ UDP, xử lý yêu cầu và trả đáp ứng về cho UDP để gửi về client. Khi các client gửi gói tin đến sẽ được chứa trong hàng đợi để chờ phục vụ. Các gói tin này có thể đi tới từ một client hoặc nhiều client và server sẽ thực hiện sử lý tuần tự từng yêu cầu theo trật tự trong hàng đợi. Mô hình hoạt động của server này được thể hiện như hình 3.3.

II. XÂY DỰNG CHƯƠNG TRÌNH SERVER PHỤC VỤ NHIỀU CLIENT HƯỚNG KẾT NỐI

1. Giới thiệu

Để cài đặt chương trình server TCP phục vụ nhiều client đồng thời, trong lập trình mạng có 2 kỹ thuật phổ biến:

- Xây dựng chương trình đa tiến trình: Trong chương trình này tiến trình cha sẽ sinh ra tiến trình con mỗi khi có một client gửi yêu cầu tới server. Nhược điểm của kỹ thuật lập trình này là không tận dụng hiệu quả CPU. Vì khi chương trình chạy, nó phải sử dụng cơ chế ngắt để chuyển từ tiến trình này sang tiến trình khác, nên CPU sẽ rảnh rỗi trong quá trình này.



Hình 3.3. Mô hình server kiểu lặp hướng không kết nối

trong quá trình chuyển tiến trình đó. Kỹ thuật này được sử dụng phổ biến trong các ngôn ngữ lập trình C/C^{++}(Linux, Unix), VC^{++}...

- Xây dựng chương trình đa luồng(đa tiểu trình): Chương trình server kiểu này sẽ sinh ra một luồng mới mỗi khi có một client gửi yêu cầu tới đòi phục vụ. Kiểu chương trình này khi chạy tận dụng hiệu quả CPU vì chương trình không có việc chuyển từ tiến trình này sang tiến trình khác. Kỹ thuật lập trình này được các ngôn ngữ lập trình phổ biến hiện nay hỗ trợ mạnh mẽ như VC^{++}, Java, .NET...Sau đây chúng ta sẽ lướt qua kỹ thuật lập trình đa luồng trong java và sử dụng để xây dựng chương trình server đáp ứng nhiều kết nối đồng thời với giao thức truyền thông TCP.

2. Kỹ thuật lập trình đa luồng trong Java(MultiThread)

Một luồng là một thuộc tính duy nhất của Java. Nó là đơn vị nhỏ nhất của đoạn mã có thể thi hành được mà thực hiện một công việc riêng biệt. Ngôn ngữ Java và máy ảo Java cả hai là các hệ thống đươc phân luồng. Java hổ trợ đa luồng, mà có khả năng làm việc với nhiều luồng. Một ứng dụng có thể bao hàm nhiều luồng. Mỗi luồng được đăng ký một công việc riêng biệt, mà chúng được thực thi đồng thời với các luồng khác.

Đa luồng giữ thời gian nhàn rỗi của hệ thống thành nhỏ nhất. Điều này cho phép bạn viết các chương trình có hiệu quả cao với sự tận dụng CPU là tối đa. Mỗi phần của chương trình được gọi một luồng, mỗi luồng định nghĩa một đường dẫn khác nhau của sự thực hiện. Đây là một thiết kế chuyên dùng của sự đa nhiệm.

Trong sự đa nhiệm, nhiều chương trình chạy đồng thời, mỗi chương trình có ít nhất một luồng trong nó(luồng chính). Một vi xử lý thực thi tất cả các chương trình. Cho dù nó có thể xuất hiện mà các chương trình đã được thực thi đồng thời, trên thực tế bộ vi xử lý nhảy qua lại giữa các luồng.

Cấu trúc của một chương trình đa luồng gồm một luồng chính(main) và các luồng con. Luồng chính được khởi tạo ngay khi chương trình chạy và nó có đặc điểm:

- Là luồng sinh ra các luồng con
- Là luồng kết thúc sau cùng.

Mỗi luồng trong chương trình Java được đăng ký cho một quyền ưu tiên. Máy ảo Java không bao giờ thay đổi quyền ưu tiên của luồng. Quyền ưu tiên vẫn còn là hằng số cho đến khi luồng bị ngắt.

Mỗi luồng có một giá trị ưu tiên nằm trong khoảng của Thread.MIN_PRIORITY(Giá trị 1), và Thread.MAX_PRIORITY(giá trị 10). Mỗi luồng phụ thuộc vào một nhóm luồng, và mỗi nhóm luồng có quyền ưu tiên của chính nó. Mỗi luồng mặc định có mức độ ưu tiên bằng 5. Mỗi luồng mới thừa kế quyền ưu tiên của luồng mà tạo ra nó.

Để hỗ trợ lập trình đa luồng, java có giao diện Runnable và lớp Thread, ThreadGroup thuộc gói *java.lang(*gói mặc định)). Các phương thức của lớp Thread như bảng sau:

Phương thức	Mô tả
Enumerate(Thread t)	Sao chép tất cả các luồng hiện hành vào mảng được chỉ định từ nhóm của các luồng, và các nhóm con của nó.
getName()	Trả về tên của luồng
isAlive()	Kiểm tra một luồng có còn tồn tại (sống)
getPriority()	Trả về quyền ưu tiên của luồng
setName(String name)	Đặt tên của luồng là tên mà luồng được truyền như là một tham số
join()	Đợi cho đến khi luồng kết thúc
resume()	Chạy lại một luồng
sleep()	Tạm dừng một luồng sau khoảng thời gian nào đó
start()	Khởi tạo một luồng, thực chất là gọi thi hành phương thức run
run()	Điểm vào của một luồng(tương tự phương thức main())

Mỗi luồng con trong chương trình java có điểm vào là phương thức run() là phương thức của giao diện Runnable hoặc lớp Thread.

```
public void run()
{
//Khối lệnh của luồng
}
```

Chu kỳ sống của luồng được thể hiện như hình 3.4.

Để tạo một luồng mới:

Để tạo luồng mới có 2 cách khai báo:

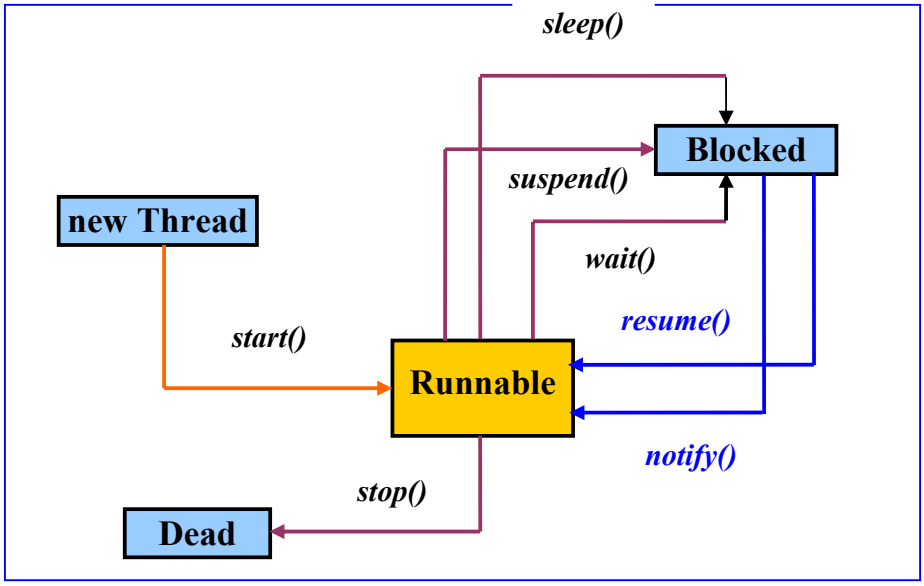
- Cách 1: Khai báo một lớp kế thừa lớp Thread, từ đó cài đặt mã lệnh thực thi của luồng vào phương thức run() bằng cách khai báo nạp chồng phương thức run.

Ví dụ:

Viết chương trình sinh ra 10 luồng, mỗi luồng in ra số thứ tự của luồng.

```
//TestThread.java
class NewThread extends Thread
{
```

```
private  int  count;
//Khai bao cau tu
NewThread(int count)
{
      super();
      this.count=count;
      start();
}
public void run()
{
System.out.println("Luong thu:"+count);
}}
class TestThread{
public static void main(String[] args)
{
int  i=0;
while(i<10)
{
new  NewThread(i);
i++;
}
}}
```



Hình 3.4. Chu kỳ sống của luồng(thread)

- Cách 2: Khai báo lớp thực thi giao diện Runnable. Lớp này cho phép tạo ra đối tượng Thread và cài đặt phần thân cho phương thức run() của giao diện. Ví dụ viết lại chương trình trên, chương trình chỉ khác phần khai báo lớp NewThread.

  //TestThread.java

```
class NewThread   implêmnts  Runnable
{
private  int  count;
//Khai bao cau tu
NewThread(int count)
{
      Thread  t=new  Thread();
      this.count=count;
public void start()
{  run(); }
public void run()
{
System.out.println("Luong thu:"+count);
}}
```

Một vấn đề quan trọng khác là vấn đề đồng bộ. Để giải quyết vấn đề này Java sử dụng một cơ chế đặc biệt gọi là Monitor.

3. Xây dựng chương trình server phục vụ nhiều client đồng thời hướng kết nối

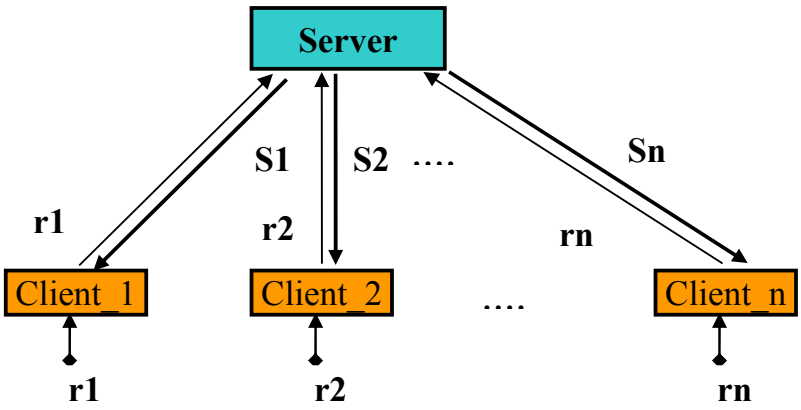
Để minh hoạ kỹ thuật này, chúng ta tiến hành xây dựng một chương trình ví dụ:

Hãy viết chương trình server phục vụ nhiều client đồng thời sử dụng giao thức truyền thông TCP. Chương trình cho phép nhận bán kính đường tròn gửi đến từ các client, tính diện tích hình tròn, hiển thị tên, địa chỉ IP, số cổng, bán kính r, diện tích của client tương ứng. Sau đó trả kết quả về cho client.

a) Chạy chương trình sử dụng trình telnet

b) Viết chương trình client.

Để viết chương trình này chúng ta sẽ sử dụng kỹ thuật đa luồng trong Java. Mỗi khi một chương trình client gửi yêu cầu kết nối đến, server sẽ sinh ra một luồng mới để phục vụ kết nối đó. Sau khi phục vụ xong kết nối nào thì luồng đó được giải phóng. Mô hình xây dựng chương trình thể hiện như hình 3.5.



Hình 3.5. Mô hình client/server của bài toán

3.1. Chương trình client

Chương trình client thực hiện các công việc sau:

- Gửi kết nối tới server
- Nhập bán kính r từ bàn phím
- Gửi bán kính tới server
- Nhận kết quả trả về và hiển thị
- Kết thúc chương trình

```
//areaClient.java
import  java.io.*;
import  java.net.*;
class areaClient{
public static void main(String[] args)
{
//Khai bao bien
Socket  cl=null;
BufferedReader  inp=null;//luong nhap
PrintWriter     outp=null.//luong xuat
BufferedReader  key=null;//luong nhap tu ban phim
String ipserver= "127.0.0.1";//Chuoi dia chi server
int     portserver=3456; //dia chi cong server
String   r; //ban kinh r la chuoi so
//Tao socket va ket noi toi server
try{
cl=new Socket(ipserver,portserver);
//tao luong nha/xuatp kieu ky tu cho socket
inp=new BufferedReader(new InputStreamReader(cl.getInputStream()));
outp=new PrintWriter(cl,getOutputStream(),true);
//tao luong nhap tu ban phim
key=new BufferedReader(new InputStreamReader(System.in));
//Nhap ban kinh r tu ban phim
System.out.print("r=");
r=key.readLine().trim();
//gui r toi server
outp.println(r);
//Nhan dien tich tra ve tu server va hien thi
System.out.println("Area:"+inp.readLine());
//ket thuc chuong trinh
if(inp!=null)
inp.close();
if(key!=null)
key.close();
if(outp!=null)
outp.close();
if(cl!=null)
cl.close();
}
cacth(IOException  e)
{
System.out.println(e);
}
}
}
```

3.2. Chương trình server

Chương trình server phục vụ nhiều client thực hiện các công việc sau:

- Khởi tạo đối tượng ServerSocket và nghe tại số cổng 3456.
- Thực hiện lặp lại các công việc sau:
  - ❖ Nhận kết nối mới, tạo socket mới
  - ❖ Phát sinh một luồng mới và nhận socket
  - ❖ Nhận bán kính gửi tới từ client
  - ❖ Tính diện tích
  - ❖ Hiển thị số thứ tự luồng, tên, địa chỉ IP, số cổng, bán kính r, diện tích của client
  - ❖ Gửi diện tích về cho client
  - ❖ Kết thúc luồng

```
//AreaThreadServer.java
import  java.io.*;
import  java.net.*;
//Khai báo lớp NewThread cho phép tạo ra luồng mới
class NewThread   extends     Thread
{
   private   int count;
   private  Socket  cl=null;
   private BufferedReader  inp=null;//luong nhap
   private  PrintWriter     outp=null;//luong xuat
   NewThread(Socket   cl, int  count)
   {
   super();//Truy xuất cấu tử lớp Thread
   this.cl=cl;
   this.count=count;
   start();
   }
//cai dat phuong thuc run-Luong moi
public  void   run()
{
try{
//tao luong nhap /xuat cho socket cl
inp=new  BufferedReader(new  InputStreamReader(cl.getInputStream()));
outp=new PrintWriter(cl.getOutputStream(),true);
//Doc ban kinh gui toi tu client
double  r=Double.parseDouble(inp.readLine().trim());
// lay dia chi client
InetAddress  addrclient=cl.getInetAddress();
//lay so cong phia client
int   portclient=cl.getPort();
//Tinh dien tich
double  area=3.14*r*r;
//Hien thi
System.out.println("Luong                                    thu:"+count+",
client:"+addrclient.getHostName()+
              ", ip:"+addrclient.getHostAddress()+",port:"+portclient+
              ", r="+r+",area:"+area);
//Gui dien tich ve cho client tuong ung
outp.println(area);
//ket thuc luong
```

```
inp.close();
outp.close();
cl.close();
}
catch(IOException e)
{
System.out.println(e);
}
}
}
//Chuong trinh server
class   AreaThreadServer{
public static void main(String[] args)
{
//Khai bao bien
int   count;
ServerSocket  svr=null;
Socket   cl=null;
int portserver=3456;
try{
svr=new ServerSocket(portserver);
count=0;
while(true){
cl=svr.accept();
new  NewThread(cl, count);
count++;
}
}
catch(IOException   e)
{
System.out.println(e);
}
}
}
```

3.3. Dịch và chạy chương trình

Dịch chương trình:

Mở cửa sổ lệnh và đến thư mục chứa chương trình client và server, thực hiện biên dịch chương trình:

javac areaClient.java [Enter]

javac AreaThreadServer.java [Enter]

Chạy chương trình:

❖ ***Chạy chương trình với trình telnet:***

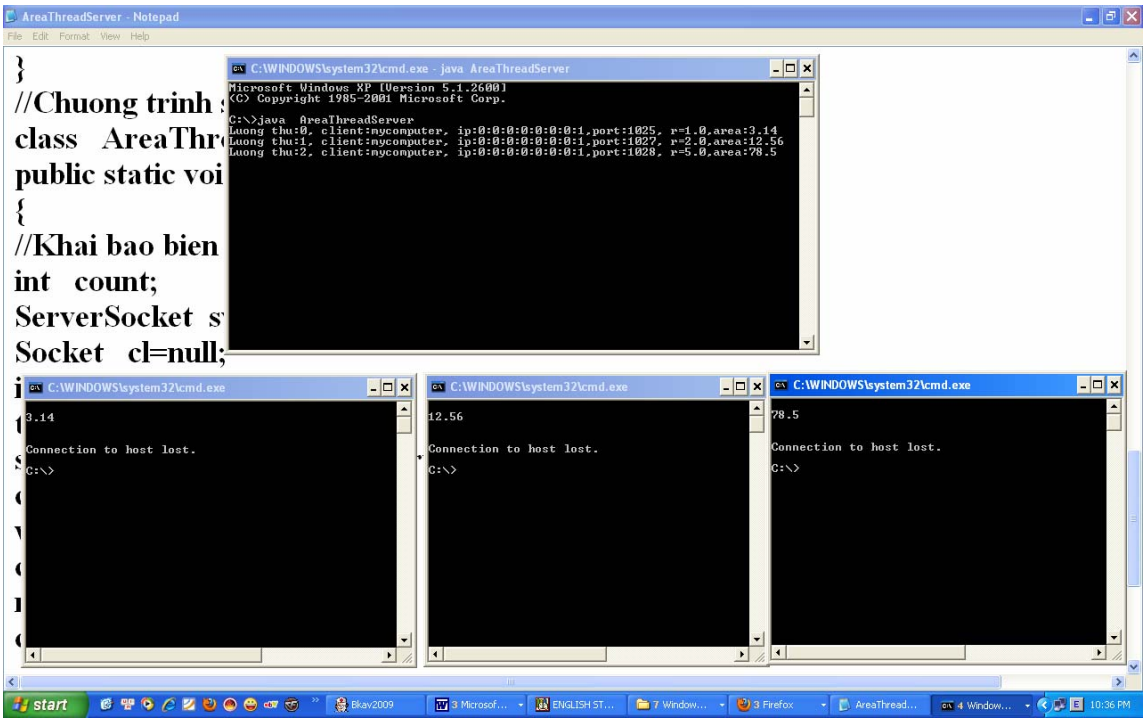
- Mở 1 cửa sổ lệnh, chạy chương trình server:

java AreaThreadServer [Enter]

- Giả sử mở 3 cửa sổ, mỗi cửa sổ là chạy một chương trình client sử dụng trình telnet được chạy với cú pháp sau:

telnet localhost 3456 [Enter]

Kết quả chạy chương trình thể hiện như cửa sổ hình 3.6.



Hình 3.6. Kết quả chạy chương trình với trình telnet

❖ Chạy chương trình với chương trình client:

Thay vì chạy trình telnet, sử dụng chương trình client areaClient. Chương trình chạy trong các cửa sổ với cú phát sau:

java areaClient [Enter]

❖ Chạy chương trình trên mạng cục bộ:

Bước 1: Sửa lại chương trình client trong cấu lệnh new Socket(....,....) với địa chỉ ipserver là địa chỉ của máy trạm trên đó chạy chương trình server. Sau đó dịch lại chương trình.

Bước 2: Copy chương trình server tới máy có địa chỉ dùng để sửa ở bước 1 và chạy chương trình.

Bước 3: Copy chương trình client đã dịch ở bước 1 tới các máy tính khác trên mạng và thực hiện chạy chương trình client đó.

Bước 4: Nhập giá trị bán kính r từ của sổ client, quan sát kết quả chạy chương trình trên client và server.

III. KẾT LUẬN

Trong chương 3 này chúng ta đã khảo sát các kiểu chương trình server, khảo sát kỹ thuật lập trình đa luồng và ứng dụng nó vào xây dựng chương trình server phục vụ nhiều client đồng thời. Cuối cùng chúng ta đã xây dựng một chương trình ví dụ đơn giản để minh hoạ kỹ thuật xây dựng server. Từ chương trình ví dụ, sinh viên có thể sửa chương trình để ứng dụng nhiều bài toán thực tế như bài toán tra cứu tuyển sinh, bài toán nhập dữ liệu từ xa, bài toán tra cứu thời tiết ... mà có kết nối với các cơ sở dữ liệu như Access, SQL hoặc Oracle. Các kỹ thuật lập trình mạng này sẽ được củng cố hơn ở các chương tiếp theo.