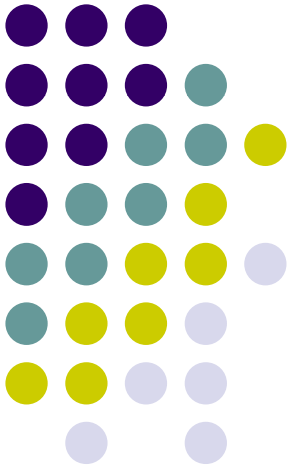
Socket

- Giới thiệu
- Lập trình Socket TCP
- Lập trình Socket UDP
- Lập trình Multicast



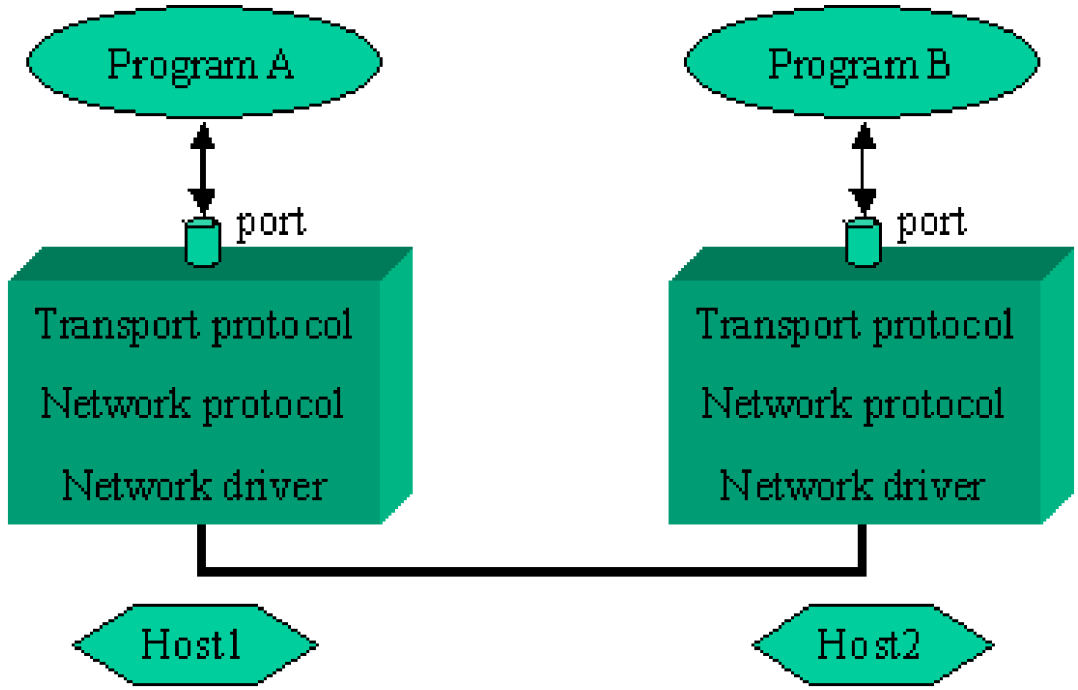
Giới thiệu về Socket

- **Khái niệm về socket**
  - **Góc độ mạng**: Socket là 1 trong 2 điểm cuối của đường nối kết 2 chiều giữa 2 chương trình thực thi trên mạng.

  - **Góc độ người lập trình**: Socket là giao diện lập trình ứng dụng (API) hay bộ thư viện hàm hỗ trợ, dùng để nối kết chương trình ứng dụng với lớp mạng trong hệ thống mạng TCP/IP.
  - Giới thiệu lần đầu dưới hệ điều hành UNIX version 4.3 BSD.

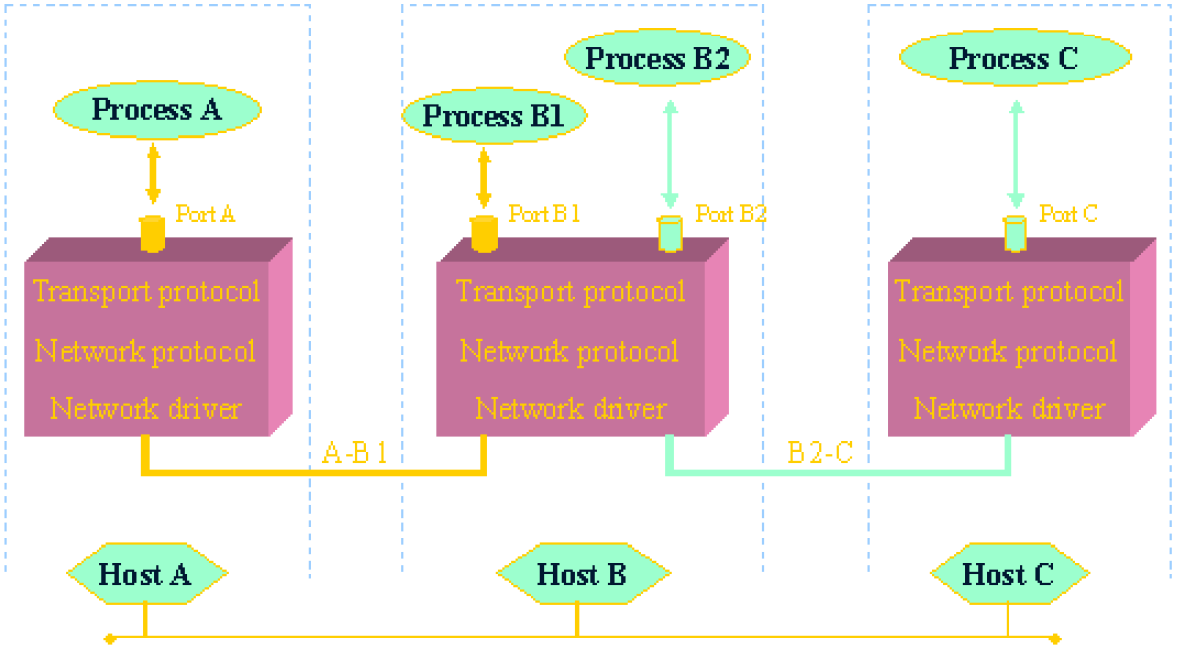
Giới thiệu về Socket

- Phân loại
  - AF_UNIX: giao tiếp giữa các quá trình trong cùng 1 máy.
  - AF_INET: giao tiếp giữa các quá trình trên nhiều máy tính.

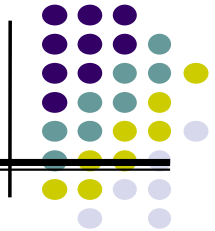


Giới thiệu về Socket

- Cơ chế giao tiếp
  - Một trong hai quá trình phải công bố số hiệu cổng của socket mà mình sử dụng để nhận và gởi dữ liệu.
  - Các quá trình khác có thể giao tiếp với quá trình đã công bố cổng cũng bằng cách tạo ra một socket.



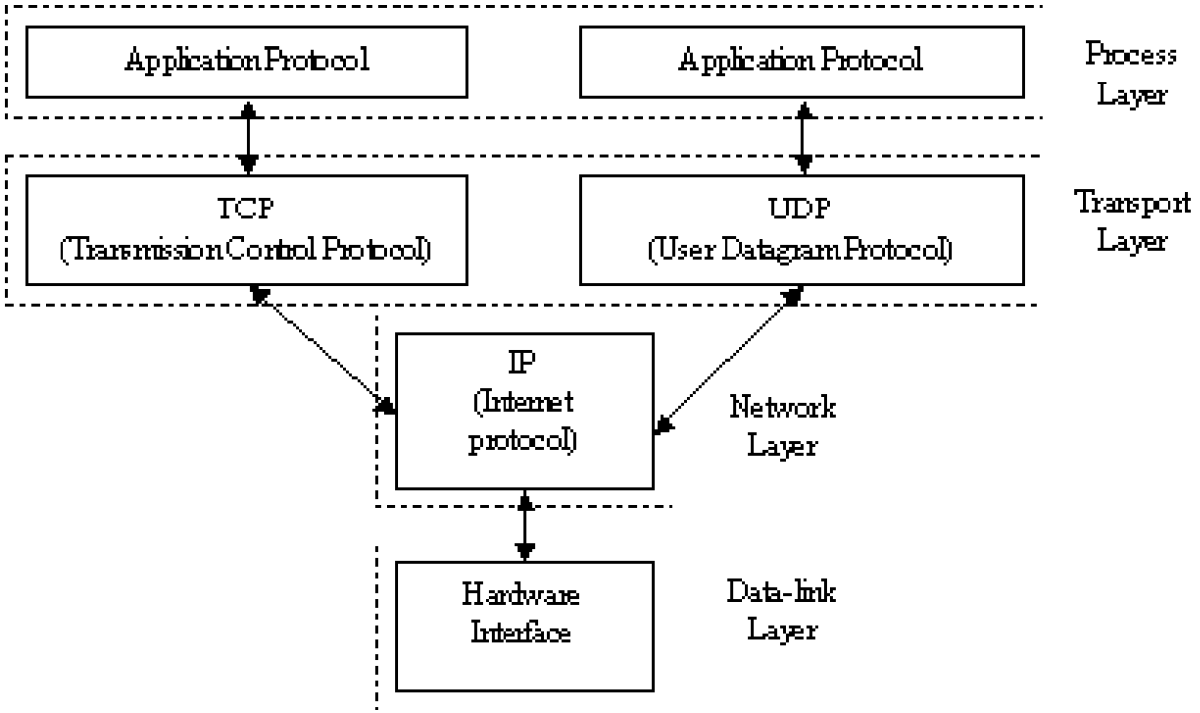
Giới thiệu về Socket



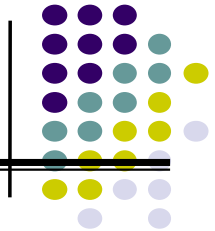
- Cổng (port): là 1 số 16 bit
  - Từ 0 – 1023: cổng hệ thống
  - Từ 1024 – 49151: cổng phải đăng ký (registered port)
  - Từ 49152 – 65535: cổng dùng riêng (private port).
- Một số cổng thông dụng
  - Echo: cổng 7 (TCP, UDP)
  - Web: cổng 80 (TCP)
  - FTP: cổng 21 cho nối kết và 20 cho dữ liệu (TCP)
  - SMTP: cổng 25 (TCP)
  - POP: cổng 110 (TCP)
  - Telnet: cổng 23 (TCP)
  - DNS: cổng 53 (TCP và UDP)
  - SNMP: cổng 161 (UDP)
  - RIP: cổng 520 (UDP)

Giới thiệu về Socket

- Các chế độ giao tiếp
  - TCP (*Transmission Control Protocol*): có nối kết
  - UDP (*User Datagram Protocol*): không nối kết



Giới thiệu về Socket



- So sánh giữa TCP và UDP

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 ...

Giới thiệu về Socket

- So sánh giữa TCP và UDP

Bit 0	Bit 15 Bit 16	Bit 31
Source Port (16)		Destination Port (16)
Sequence Number (32)		
Acknowlegement Number (32)		
Header Length (4) / Reserved (6) / Code Bits (6)		Window (16)
Checksum (16)		Urgent (16)
Options (0 or 32 if any)		
Data (varies)		

20 Bytes

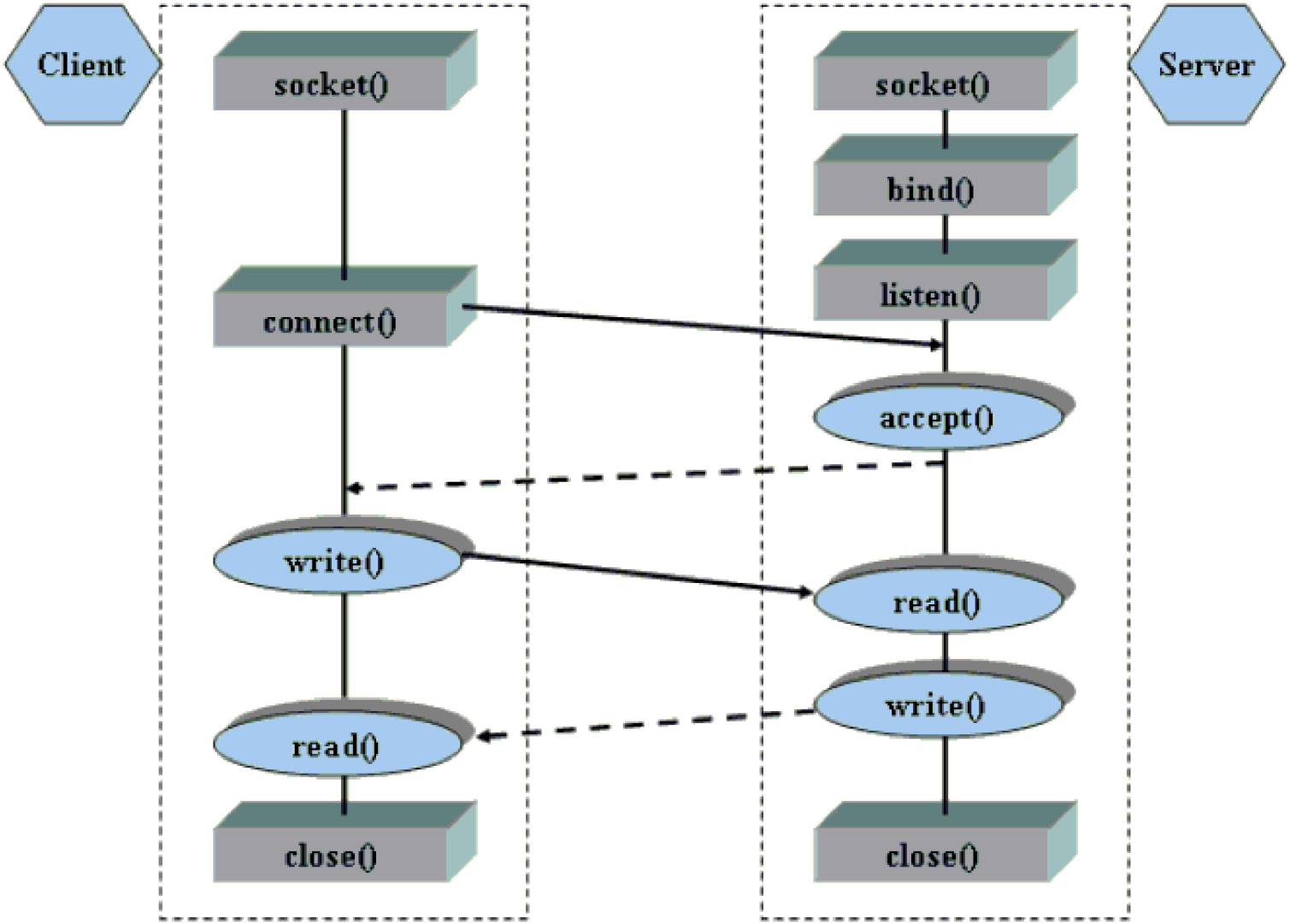
TCP Segment Format

Bit 0	Bit 15 Bit 16	Bit 31
Source Port (16)		Destination Port (16)
Length (16)		Checksum (16)
Data (if any)		

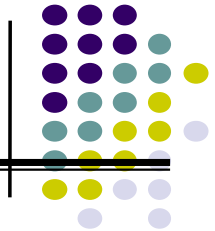
8 Bytes

UDP Segment Format

Socket ở chế độ có nối kết (TCP)

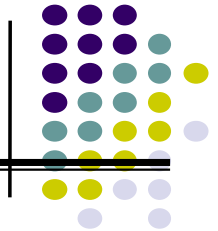


Socket ở chế độ có nối kết (TCP)



- Giao thức ứng dụng
  - Trao đổi thông tin giữa Client và Server phải tuân thủ giao thức của ứng dụng.
  - Nếu theo các Protocol đã định nghĩa sẵn: tham khảo **RFC**.
  - Nếu ứng dụng riêng biệt: tự thiết kế protocol riêng.
- TCP Socket dưới Java
  - Thông qua các lớp trong gói **java.net**
  - Các lớp chính:
    - java.net.Socket: hỗ trợ xây dựng chương trình Client
    - java.net.ServerSocket: hỗ trợ xây dựng chương trình Server

Socket ở chế độ có nối kết (TCP)



- Lớp java.net.Socket
  - Socket(String HostName, int PortNumber) throws IOException: nối kết đến Server có tên là HostName, cổng là PortNumber.

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

Hoặc `Socket s = new Socket("203.162.36.149", 80);`

  - InputStream getInputStream() throws IOException:

trả về 1 InputStream nối với Socket.

  - OutputStream getOutputStream() throws IOException:

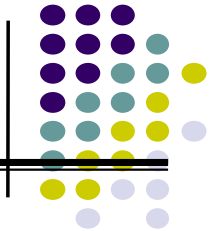
trả về OutputStream nối với Socket.

VD:
```
InputStream is = s.getInputStream();
OutputStream os = s.getOutputStream();
```

  - void close() throws IOException: đóng Socket lại, giải phóng kênh ảo, xóa nối kết giữa Client và Server.

VD: `s.close();`

Socket ở chế độ có nối kết (TCP)

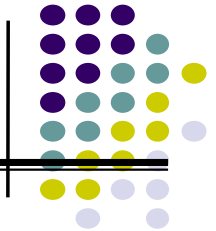


- Lớp java.net.Socket
  - InetAddress getInetAddress(): lấy địa chỉ của máy tính đang nối kết (ở xa).
  - int getPort(): lấy cổng của máy tính đang nối kết (ở xa).
  - InetAddress getLocalAddress(): lấy địa chỉ cục bộ.
  - int getLocalPort(): lấy giá trị cổng cục bộ
  - void setSoTimeout(int timeout) 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.
  - void setKeepAlive(boolean on) throws SocketException:

    quá trình Client muốn giữ nối kết ngay khi nó không gửi thông tin gì cho Server.

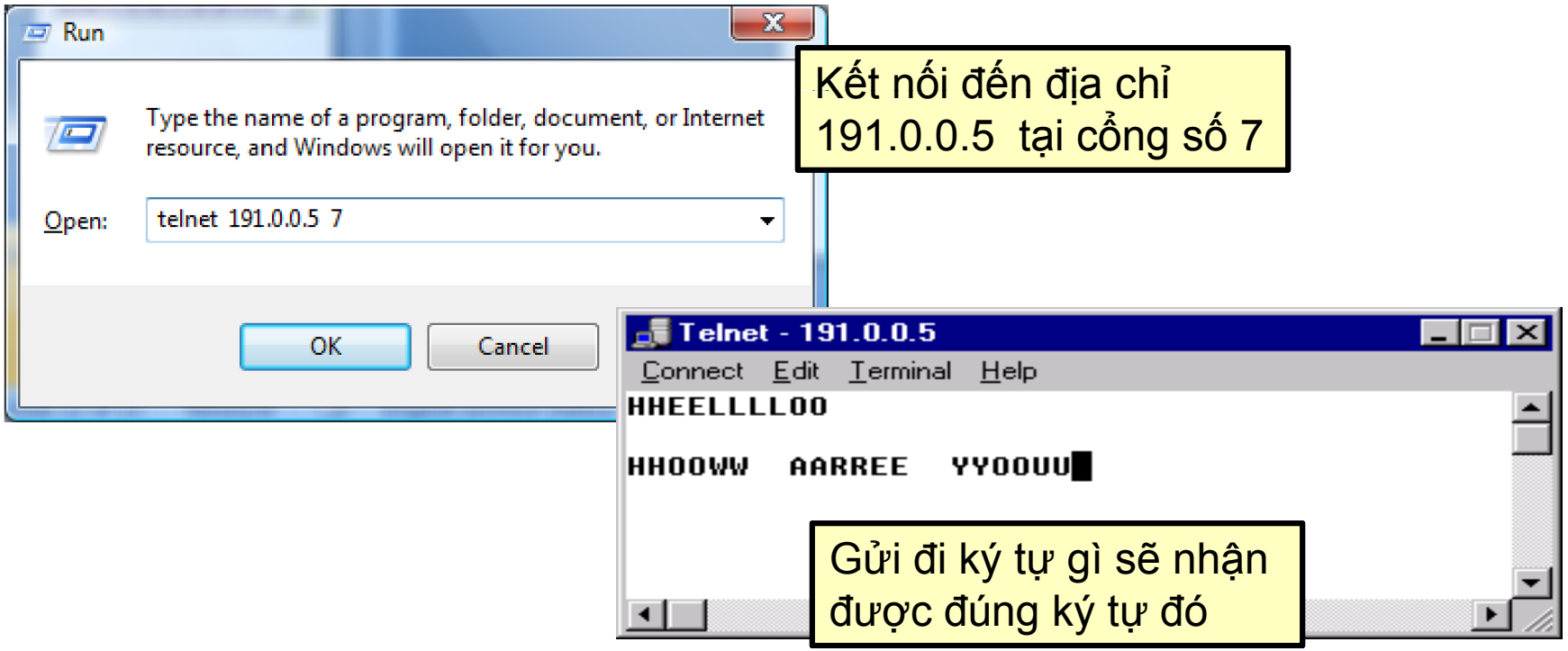
Socket ở chế độ có nối kết (TCP)



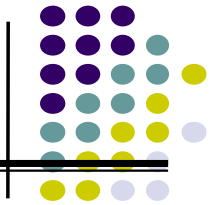
- Xây dựng chương trình Client ở chế độ TCP
  - Mở một socket nối kết đến Server đã biết địa chỉ IP (hay địa chỉ tên miền) và số hiệu cổng.
  - Lấy InputStream và OutputStream gán với Socket.
  - Tham khảo Protocol của dịch vụ để định dạng đúng dữ liệu trao đổi với Server.
  - Trao đổi dữ liệu với Server nhờ vào các InputStream và OutputStream vừa lấy.
  - Đóng Socket trước khi kết thúc chương trình.

Socket ở chế độ có nối kết (TCP)

- Chương trình TCPEchoClient
  - Trên hệ thống UNIX, dịch vụ Echo được thiết kế theo mô hình Client-Server sử dụng Socket cả TCP và UDP.
  - Cổng mặc định dành cho Echo Server là 7.

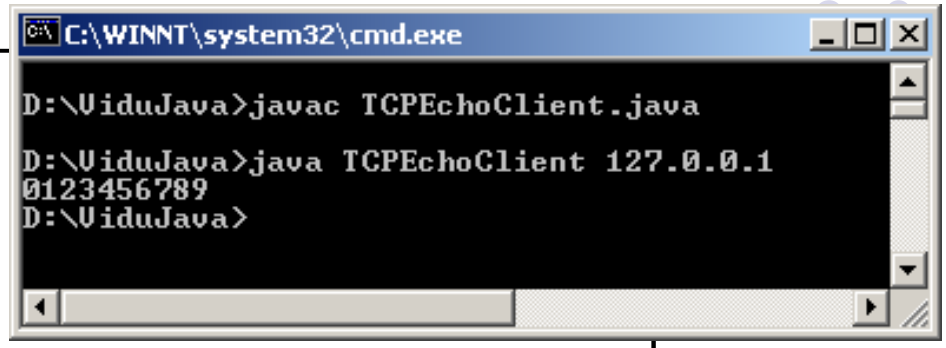


Socket ở chế độ có nối kết (TCP)



```
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
            InputStream is = s.getInputStream();        // Lay InputStream
            OutputStream os = s.getOutputStream(); // Lay OutputStream
            for (int i='0'; i<='9';i++){     // Gui '0'->'9' den EchoServer
                os.write(i);                            // Gui 1 ky tu sang Server
                int ch = is.read();                     // Chi nhan 1 ky tu tu Server
                System.out.print((char)ch);  // In ky tu nhan duoc ra man hinh
            }
        }    //try
        catch(IOException ie){
            System.out.println("Loi: Khong tao duoc socket");
        }    //catch
    }    //main
}
```

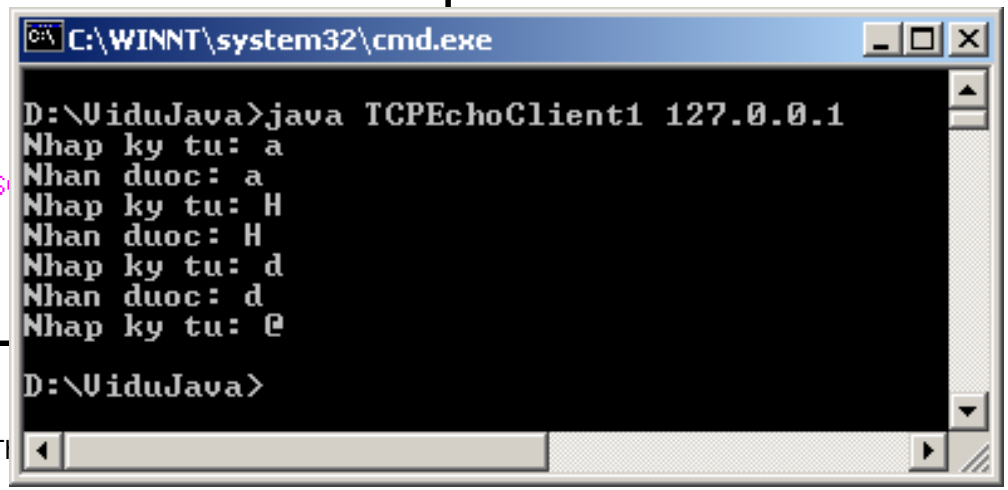


Chương trình TCPClient:
- Gửi qua Server từ 0->9
- Nhận kết quả và hiển thị ra màn hình.

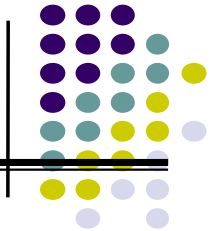
Socket ở chế độ có nối kết (TCP)

Chương trình TCPClient:
- Nhập 1 ký tự từng bàn phím
- Gửi ký tự đó qua Server
- Nhận kết quả và hiển thị ra màn hình.

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



Socket ở chế độ có nối kết (TCP)



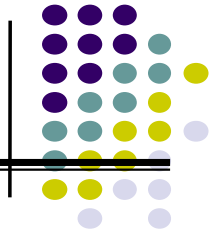
- Lớp java.net.ServerSocket
  - ServerSocket(int PortNumber): tạo một Socket của Server và lắng nghe trên cổng PortNumber.

  VD: `ServerSocket ss = new ServerSocket(7);`
  - Socket accept(): Bị nghẽn cho đến khi có một yêu cầu nối kết từ Client. Chấp nhận cho nối kết, trả về một Socket là một đầu của kênh giao tiếp ảo giữa Server và Client.

  VD: `Socket s = ss.accept();`
  - Server sau đó sẽ lấy InputStream và OutputStream của Socket mới s để giao tiếp với Client:

```
InputStream  is = s.getInputStream();
OutputStream os = s.getOutputStream();
```

Socket ở chế độ có nối kết (TCP)



- Xây dựng chương trình Server ở chế độ TCP

Phục vụ tuần tự

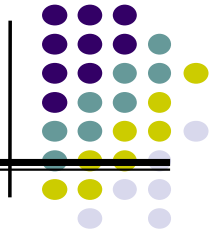
- Tại 1 thời điểm Server chỉ chấp nhận 1 yêu cầu nối kết
- Nếu có các nối kết khác sẽ đưa vào hàng đợi
- Sau khi phục vụ Client đó xong, quay lại phục vụ tiếp Client trong hàng đợi.

Phục vụ song song

- Tại 1 thời điểm Server chấp nhận nhiều yêu cầu nối kết.
- Tất cả yêu cầu nối kết được phục vụ cùng 1 lúc.

- Hiệu quả hơn.
- Cần máy tính đủ mạnh và tài nguyên lớn hơn.

Socket ở chế độ có nối kết (TCP)

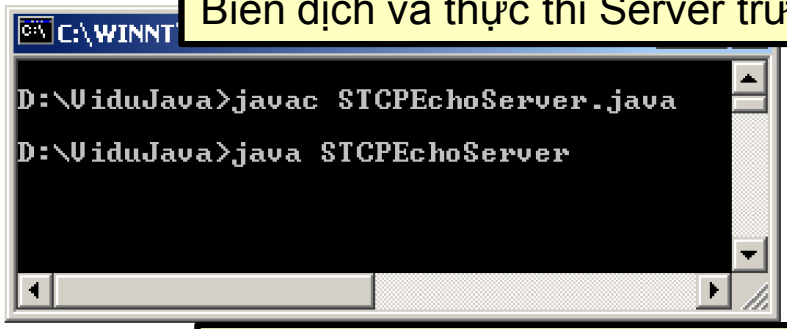


- Chương trình Server **phục vụ tuần tự**
  1. Tạo socket và gán số hiệu cổng cho Server.
  2. Lắng nghe yêu cầu nối kết.
  3. Với một yêu cầu nối kết được chấp nhận thực hiện các bước sau:
     - Lấy InputStream và OutputStream gắn với Socket của kênh ảo vừa được hình thành.
     - Lặp lại công việc sau:
       - Chờ nhận các yêu cầu (công việc).
       - Phân tích và thực hiện yêu cầu.
       - Tạo thông điệp trả lời.
       - Gửi thông điệp trả lời về Client.
       - Nếu không còn yêu cầu hoặc Client kết thúc, đóng Socket và quay lại bước 2 (lắng nghe yêu cầu nối kết tiếp tục).

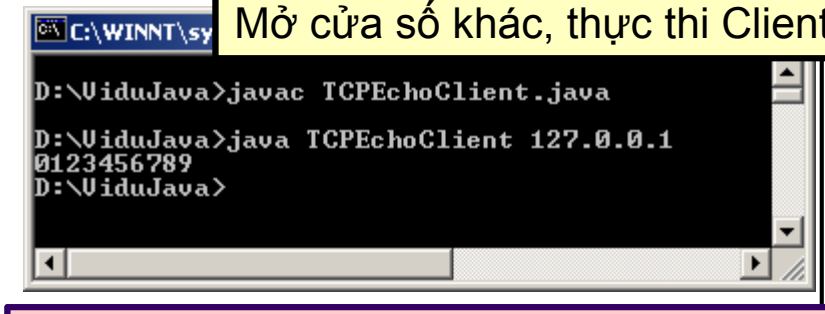
Socket ở chế độ có nối kết (TCP)

- Chương trình STCPEchoServer

Biên dịch và thực thi Server trước



Mở cửa số khác, thực thi Client sau

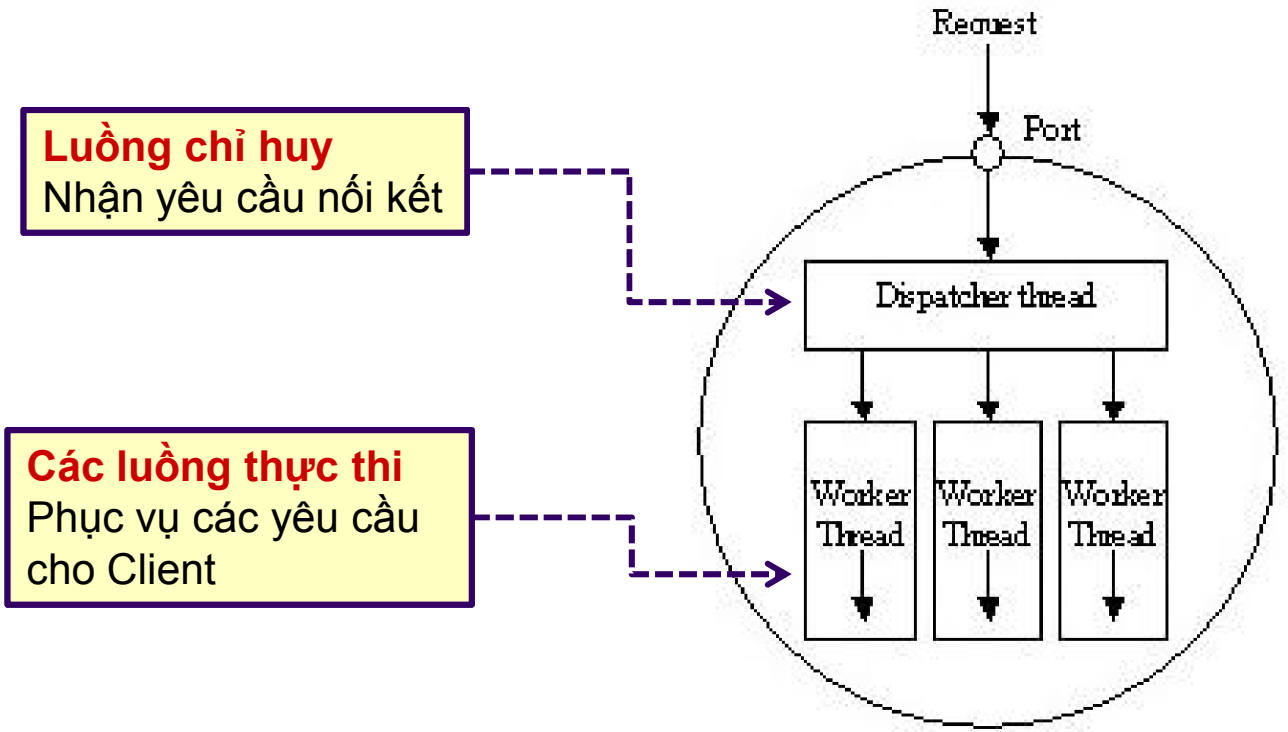


Giả sử Server đang thực thi trên máy tính ở địa chỉ 172.18.213.233, thực thi Client:
java TCPEchoClient 172.18.213.233

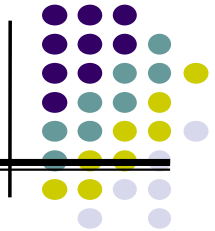
```
import java.net.*;
import java.io.*;
public class STCPEchoServer {
    public final static int defaultPort = 7;
    public static void main(String[] args) {
        try {
            // Tao Server Socket lang nghe tren cong 7
            ServerSocket ss = new ServerSocket(defaultPort);
                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);
        }
    }
}
```

Socket ở chế độ có nối kết (TCP)

- Chương trình 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.

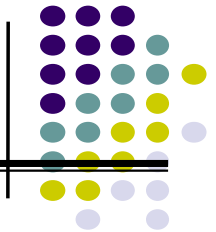


Socket ở chế độ có nối kết (TCP)



- Chương trình Server **phục vụ 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: Lặp lại các công việc sau:
    - Chờ nhận thông điệp yêu cầu của khách hàng.
    - Phân tích và xử lý yêu cầu.
    - Gửi thông điệp trả lời cho khách hàng.
  - Phần 2 sẽ kết thúc khi kênh ảo bị xóa đi.
  - Phần 2 được thiết kế là **1 thread** (*để có thể thực thi song song với phần 1*).

Socket ở chế độ có nối kết (TCP)



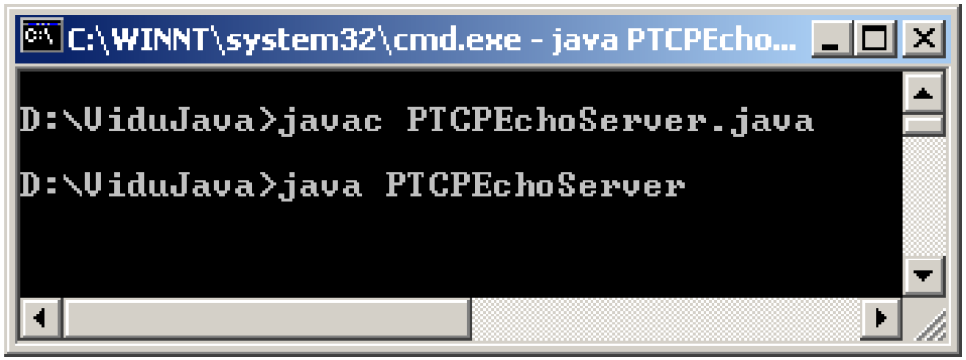
- Chương trình PTCPEchoServer

```
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("Create 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);
         }
    }
};
```

Socket ở chế độ có nối kết (TCP)

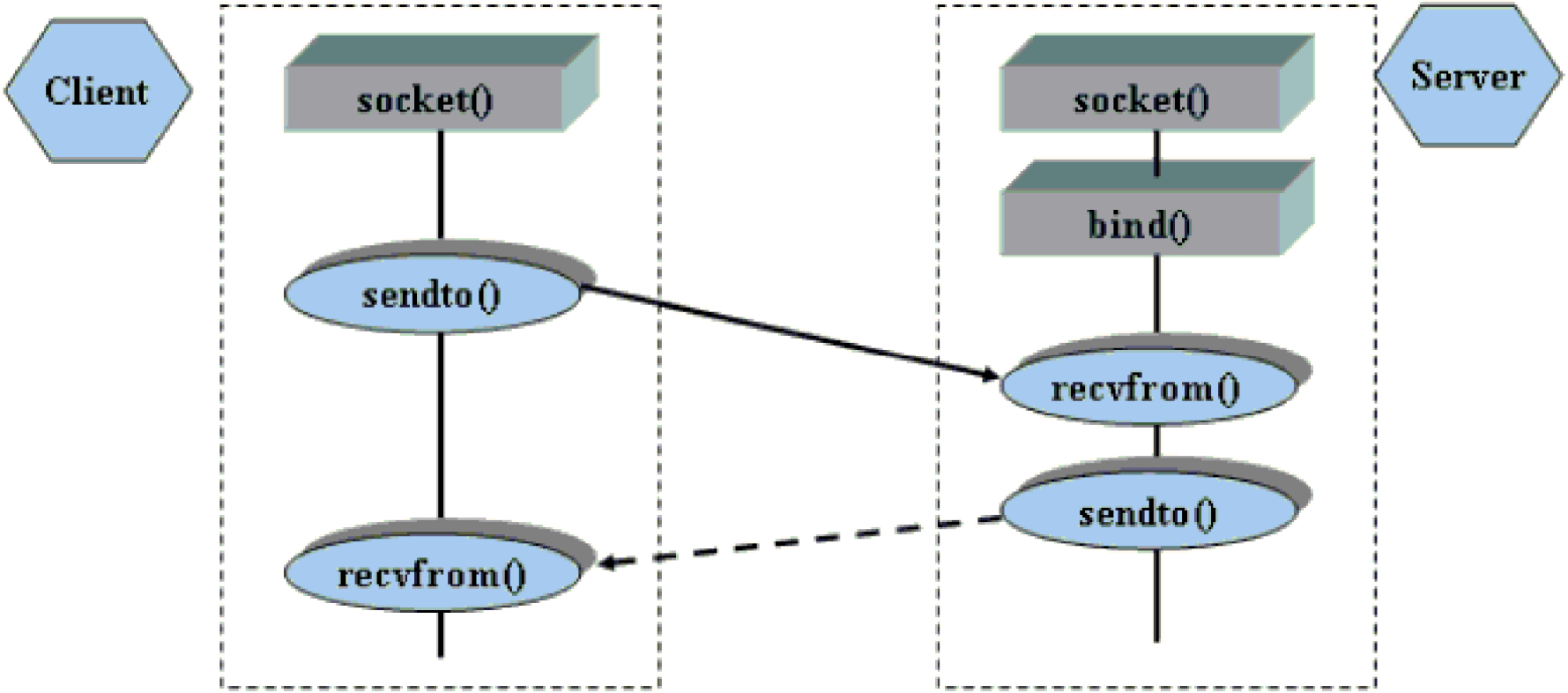
- Chương trình PTCPEchoServer
  - Biên dịch và thực thi Server



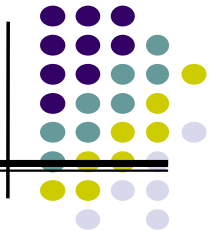
  - Thực thi Client:
    - Mở nhiều cửa sổ khác nhau để thực thi TCPEchoClient1

      Hoặc thực thi trên nhiều máy tính khác nhau.
    - Nhận thấy: PTCPEchoServer có khả năng phục vụ cùng lúc nhiều Client.

Socket ở chế độ không nối kết (UDP)

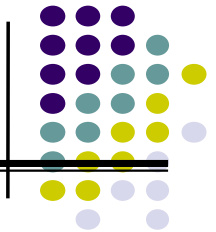


Socket ở chế độ không nối kết (UDP)



- UDP Socket dưới Java
  - Thông qua các lớp trong gói **java.net**
  - Các lớp chính:
    - java.net.DatagramSocket: hỗ trợ xây dựng Socket dạng UDP.
    - java.net.DatagramPacket: gói tin dạng thư tín người dùng (User Datagram) trong giao tiếp giữa Client và Server, gồm:
      - Dữ liệu truyền đi (tối đa khoảng 60.000 byte).
      - Địa chỉ IP của quá trình gửi.
      - Cổng của quá trình gửi.
      - Địa chỉ IP của quá trình nhận.
      - Cổng của quá trình nhận.
  - Cổng của 2 ứng dụng sử dụng TCP và UDP có thể trùng nhau vì chúng thực thi trên 2 không gian khác nhau.

Socket ở chế độ không nối kết (UDP)



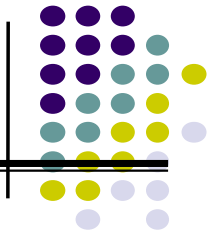
- Lớp java.net.DatagramPacket
  - DatagramPacket (byte[] b, int n)
    - Tạo ra gói tin UDP chứa n bytes dữ liệu đầu tiên của mảng b.
    - Thường dùng cho quá trình nhận để lưu gói nhận về.

VD:
```
byte buff[] = new byte[60000];  // Noi chua du lieu nhan duoc
DatagramPacket  inPacket = new DatagramPacket(buff, buff.length);
```

  - DatagramPacket(byte[] b, int n, InetAddress ia, int port)
    - Tạo ra gói tin UDP chứa dữ liệu (gồm n byte lưu trong mảng b), địa chỉ IP và cổng của máy nhận dữ liệu.

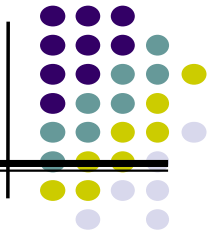
VD:
```
try {  InetAddress   ad = InetAddess.getByName("www.cit.ctu.edu.vn");
       int port = 19;    // Cổng của socket nhận
       String s = "My second UDP Packet";   // Dữ liệu gửi đi
       byte[] b = s.getBytes();   // Đổi chuỗi thành mảng bytes
       DatagramPacket outPacket=new DatagramPacket(b, b.length, ad, port);
   }
catch (UnknownHostException e)  {  System.err.println(e);  }
```

Socket ở chế độ không nối kết (UDP)



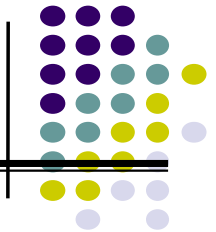
- Lớp java.net.DatagramPacket
  - Lấy thông tin trong gói tin UDP:
    - public synchronized InetAddress getAddress()
    - public synchronized int getPort()
    - public synchronized byte[] getData()
    - public synchronized int getLength()
  - Gán thông tin vào trong gói tin UDP:
    - public synchronized void setAddress(InetAddress ad)
    - public synchronized void setPort(int port)
    - public synchronized void setData(byte[] b)
    - public synchronized void setLength(int len)

Socket ở chế độ không nối kết (UDP)



- Lớp java.net.DatagramSocket
  - DatagramSocket() throws SocketException
    - Tạo Socket theo chế độ không nối kết cho Client
    - Cổng được gán ngẫu nhiên.
  - DatagramSocket(int port) throws SocketException
    - Tạo Socket theo chế độ không nối kết cho Server
    - Cổng phục vụ có giá trị là port.
  - void send(DatagramPacket dp) throws IOException
    - Gửi đi gói tin dp
  - synchronized void receive(DatagramPacket dp) throws IOException
    - Chờ nhận 1 gói tin UDP.
    - Quá trình sẽ bị nghẽn cho đến khi có dữ liệu đến.

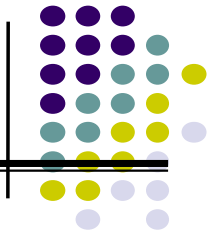
Socket ở chế độ không nối kết (UDP)



Chương trình UDPEchoClient

```
import java.net.*;
import java.io.*;
public class UDPEchoClient {
    public final static int serverPort = 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("Syntax: java UDPClient HostName"); return;
             }
            DatagramSocket ds = new DatagramSocket();      // Tao DatagramSocket
            InetAddress  server = InetAddress.getByName(args[0]);   // Dia chi Server
            while(true) {
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                String theString = br.readLine();          // Nhap tu ban phim
                byte[] data = theString.getBytes();      // Doi chuoi ra mang bytes
                // Tao goi tin
                DatagramPacket dp = new DatagramPacket(data,data.length,server, serverPort);
                ds.send(dp); // Gui goi tin sang Server
                byte[] buffer = new byte[6000];      // Tao vung dem cho goi tin de nhan
                // 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);
        }
    }
}
```

Socket ở chế độ không nối kết (UDP)

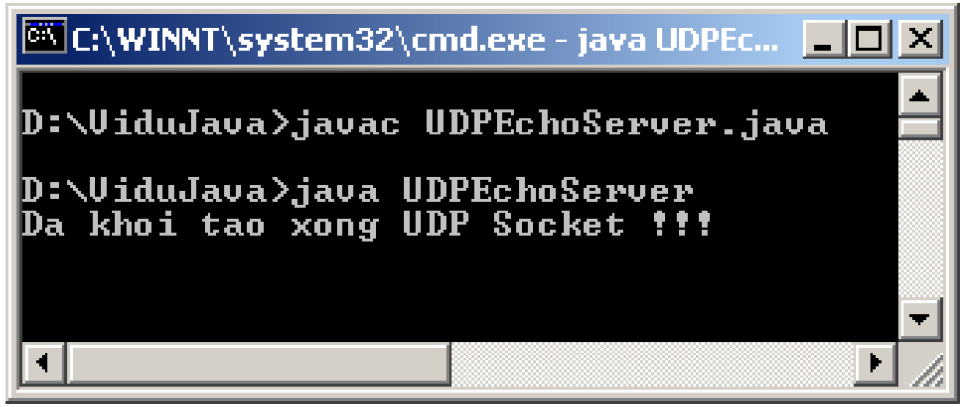


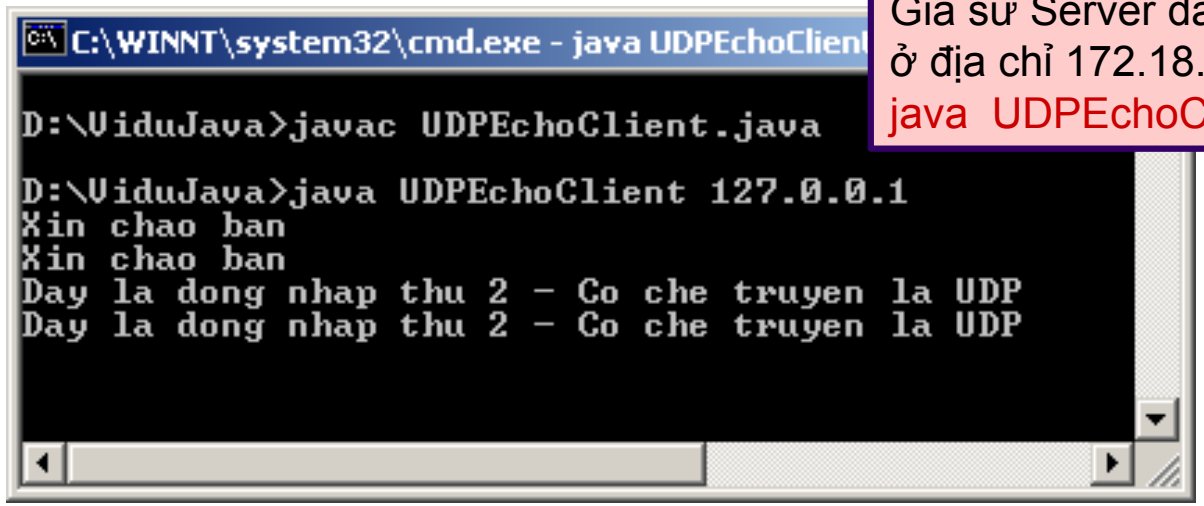
Chương trình UDPEchoServer

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

Socket ở chế độ không nối kết (UDP)

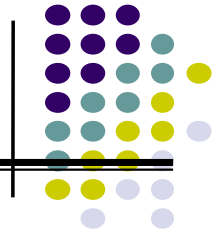
- Biên dịch và thực thi





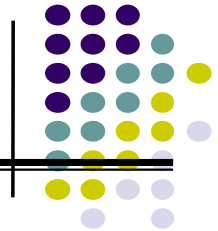
Giả sử Server đang thực thi trên máy tính ở địa chỉ 172.18.213.233, thực thi Client:
java UDPEchoClient 172.18.213.233

Lập trình multicast



- Khái niệm
  - Unicast: 1 máy tính gửi và chỉ 1 máy tính nhận.
  - Multicast: liên lạc theo nhóm
    - Gửi quảng bá, nhưng chỉ đến 1 nhóm các máy tính cho trước.
    - Thuộc địa chỉ lớp D: 224.0.0.0 - 239.255.255.255
    - Địa chỉ 224.0.0.1 là địa chỉ dành riêng.
      - Ping 224.0.0.1: tất cả các máy tính hỗ trợ multicast sẽ trả lời.
  - Ứng dụng của multicast:
    - Game nhiều người chơi
    - Giải thuật vạch đường (Routing Protocol)
    - Ứng dụng mà đối tượng cùng nhận chung 1 loại thông tin.

Lập trình multicast

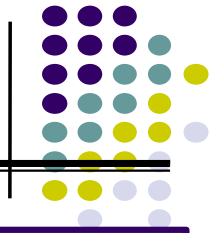


- Lập trình Multicast dùng Java
  - Sử dụng lớp java.net.MulticastSocket
    - Là 1 DatagramSocket (UDP)
    - Gia nhập (joining) vào 1 nhóm các máy tính multicast.
    - Một 1 máy tính gửi gói tin đến nhóm, các thành viên trong nhóm sẽ nhận được gói tin đó.

VD: // Gia nhập 1 nhóm multicast ở địa chỉ 228.5.6.7

```
InetAddress group = InetAddress.getByName("228.5.6.7");
MulticastSocket  s = new MulticastSocket(6789);
s.joinGroup(group);
// Thoát ra khỏi nhóm multicast
s.leaveGroup(group);
```

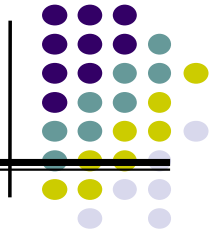
Lập trình multicast



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 multicast là **230.0.0.1**

```
import java.io.*;
import java.net.*;
import java.util.Date;
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,address, 9013);
                // Gui 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"); }
    }
}
```

Lập trình multicast



```
import java.io.*;
import java.net.*;
import java.util.Date;
public class MulticastTimeClient {
    public static void main(String args[]) {
        try {
            // Tao Socket theo dang Multicast tren cong 9013
            MulticastSocket socket = new MulticastSocket(9013);
            InetAddress address = InetAddress.getByName("230.0.0.1");
            socket.joinGroup(address);  // Tham gia vao nhom dia chi 230.0.0.1
            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);
            socket.leaveGroup(address);  // Roi khoi nhom dia chi lop D la 230.0.0.1
            socket.close();
        }
        catch(Exception e)
          { System.out.println("Co loi khi tao va thuc thi Client"); }
    }
  }
```

Có thể thử nghiệm chương trình bằng cách thực thi chương trình Client đồng thời trên nhiều máy tính có địa chỉ IP khác nhau.