



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

BÀI GIẢNG LẬP TRÌNH MẠNG

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

Khoa Khoa học máy tính



Bài 4. Lập trình với giao thức TCP (Transmission Control Protocol)



Giao thức TCP

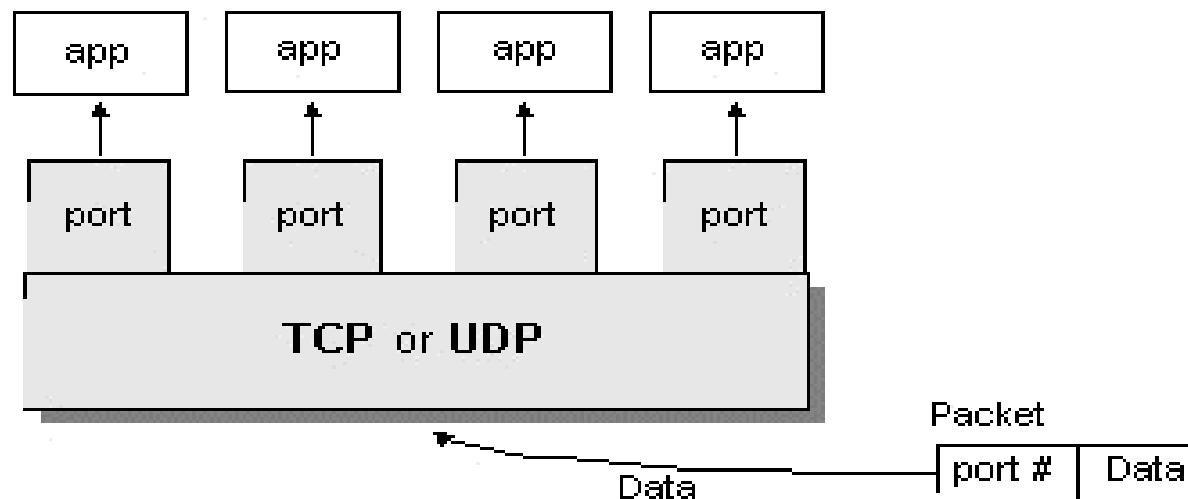
- Là giao thức hướng kết nối (connection – oriented).
 - Truyền thông tin cậy.
 - Thiết lập kênh kết nối.
- Cung cấp một kênh point-to-point cho các ứng dụng yêu cầu truyền thông tin cậy.
- Các giao thức sau là giao thức hướng kết nối:
 - HTTP
 - FTP
 - Telnet





Cổng (Port)

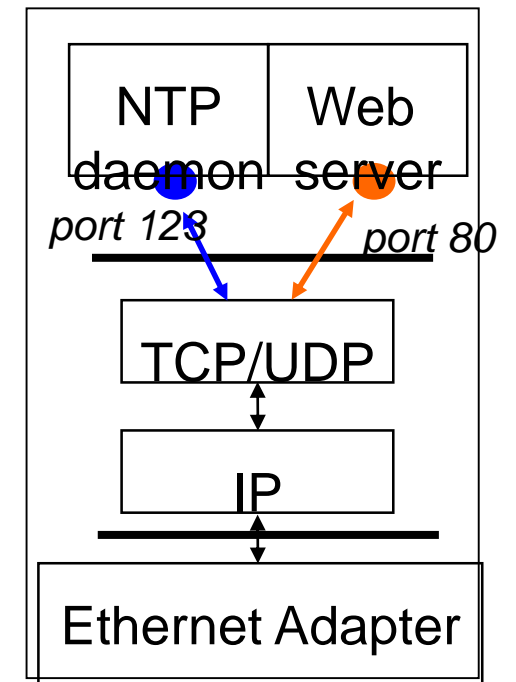
- Là một số đặc biệt
 - được gán cho một tiến trình mạng
- Mỗi tiến trình mạng đều được gán một cổng duy nhất





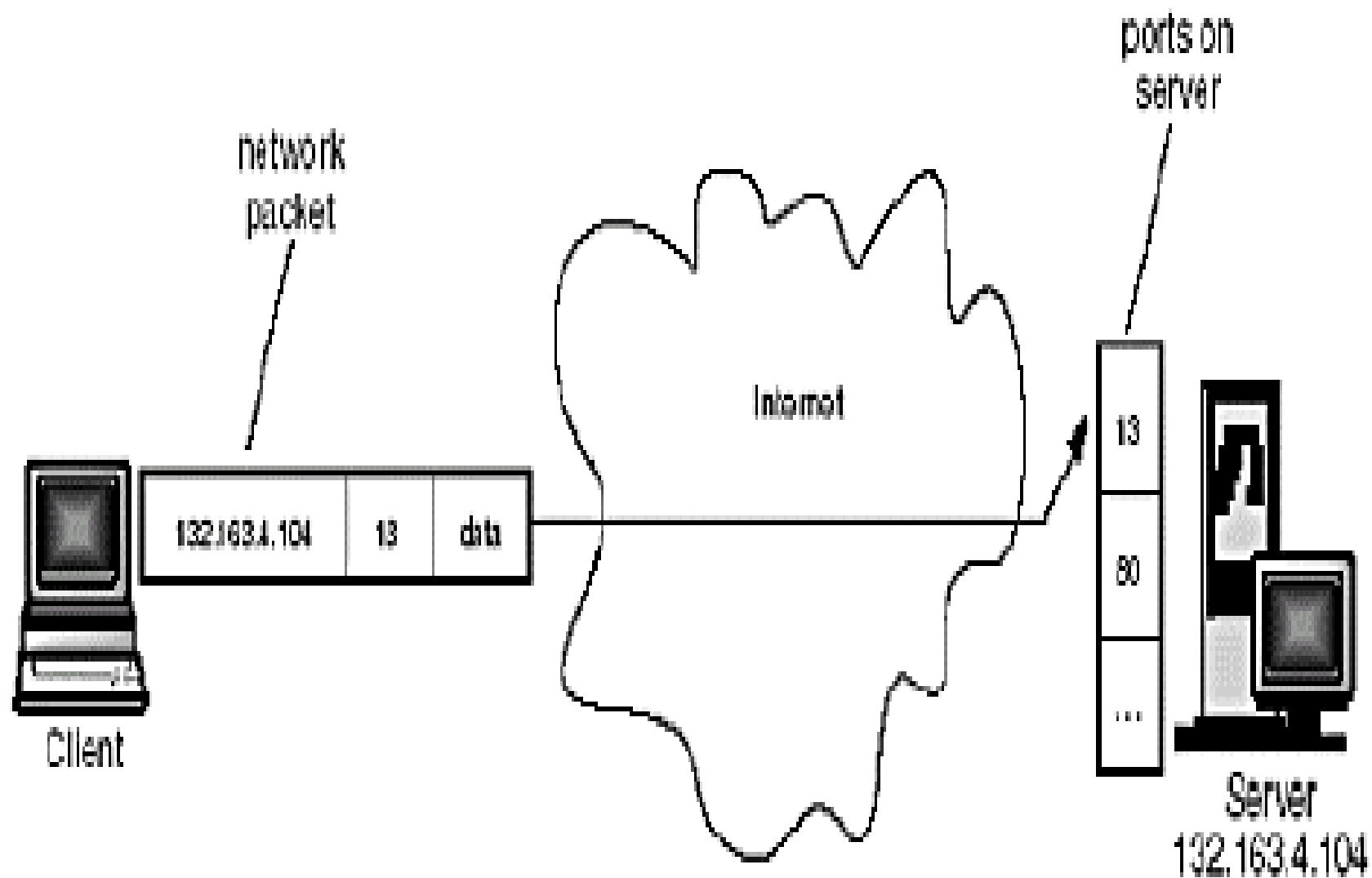
Cổng (Port)

- Số cổng là số 2 bytes
 - Các số 0-1023: dành cho các ứng dụng thông dụng
 - Các số 1024-65535 : là các cổng động
- Các server thường sử dụng các cổng nổi tiếng
 - Mục đích: bất cứ client có thể nhận biết dễ dàng server/service
 - HTTP = 80, FTP = 21, Telnet = 23, ...
- Client thường dùng các cổng động
 - Gán ở thời điểm chạy chương trình





Cổng (port)

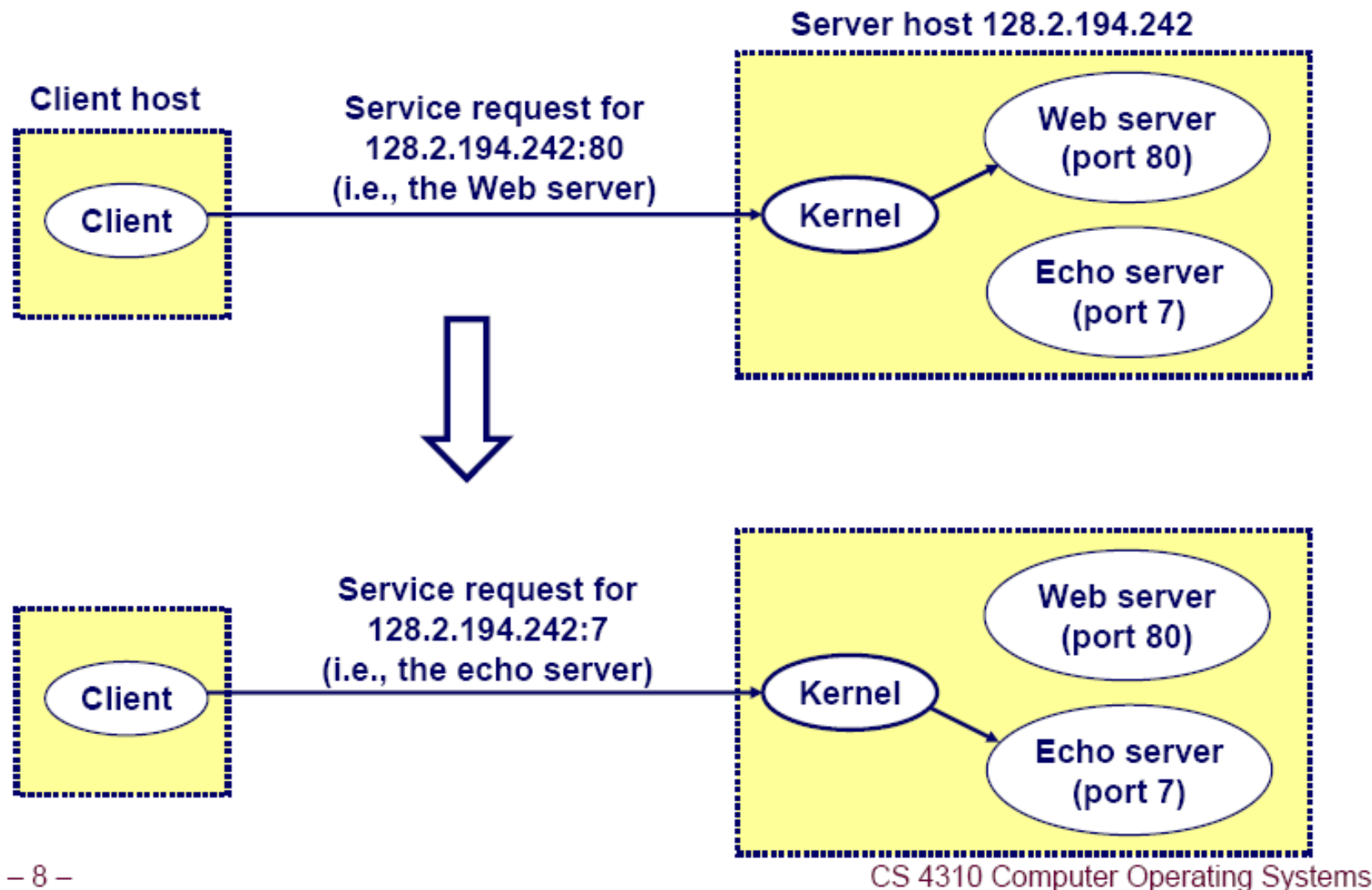




Ví dụ các chương trình mạng sử dụng cổng khác nhau



Sử dụng cổng để nhận biết dịch vụ





Một số cổng nổi tiếng (Well-known)

Port	Protocol	RFC
13	DayTime	RFC 867
7	Echo	RFC 862
25	SMTP (e-mail)	RFC 821 (SMTP)
		RFC 1869 (Extnd SMTP)
		RFC 822 (Mail Format)
		RFC 1521 (MIME)
110	Post Office Protocol	RFC 1725
20	File Transfer Protocol (data)	RFC 959
80	Hypertext Transfer Protocol	RFC 2616



Địa chỉ IP (Addresses)

- Mỗi thực thể trên mạng có một địa chỉ IP duy nhất
 - IPv4: số 32 bits, tạo thành từ 4 octets. Vd: 192.168.12.1
 - Địa chỉ IP được chia thành các lớp sau

0	1	2	3	4	5	6	7	8.	16.	24.	.32			
0	Network ID							Host ID					Class A	
1	0	Network ID						Host ID					Class B	
1	1	0	Network ID						Host ID					Class C
1	1	1	0	Multicast Group ID								Class D		
1	1	1	1	0	RESERVED FOR FUTURE USE							Class E		



Dãy địa chỉ IP theo lớp

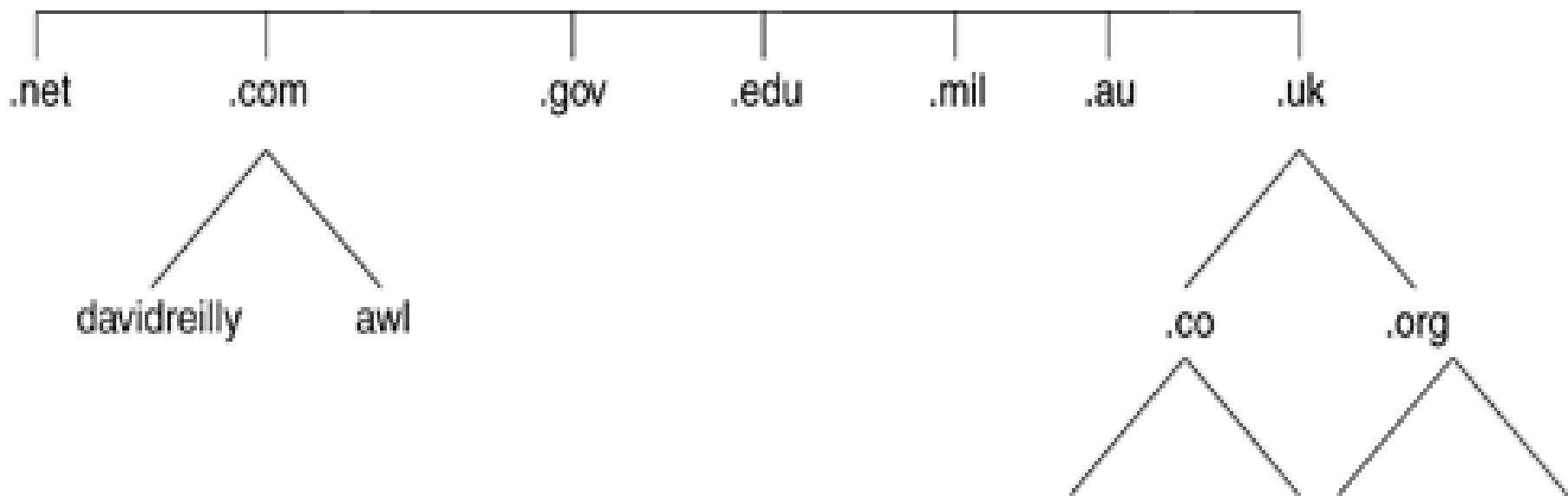
- Lớp A: 0. 0. 0. 0 – 127.255.255.255
- Lớp B: 128.0.0.0 – 191.255.255.255
- Lớp C: 192.0.0.0 – 223.255.255.255
- Lớp D: 224.0.0.0 – 239.255.255.255
- Lớp E: 240.0.0.0 – 247.255.255.255

Địa chỉ 127.0.0.1 chỉ địa chỉ IP máy cục bộ



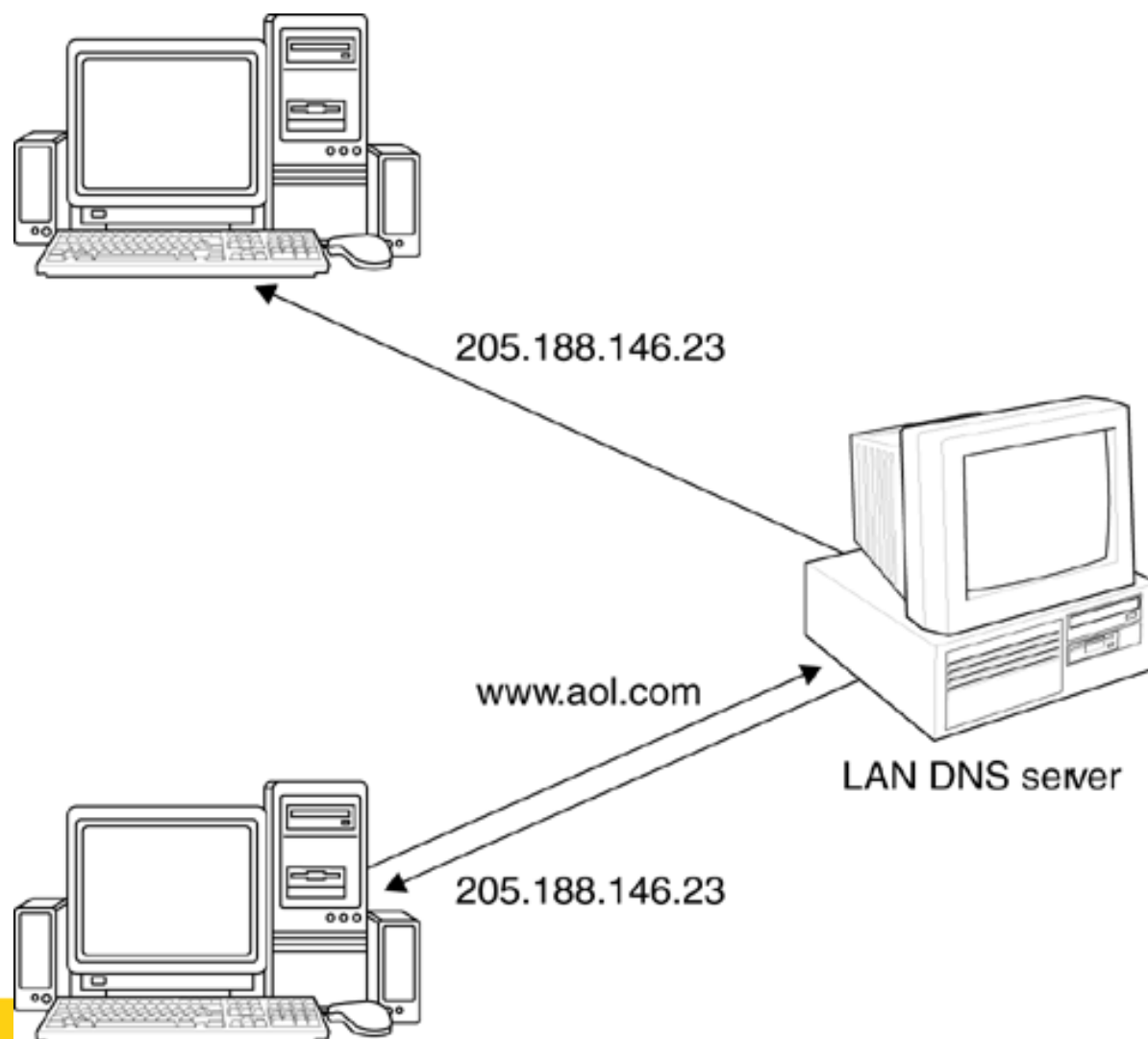
Tên miền (Domain Name)

- Chúng ta thường khó nhớ một số dài hơn là nhớ một tên
 - DNS (Domain Name Server) cung cấp ánh xạ từ địa chỉ sang tên.
 - Ví dụ <ftp.davidreilly.com>, www.davidreilly.com
 - Tên miền cũng được nhóm theo các miền con sau





Ánh xạ tên miền





Socket ?

- Một ứng dụng trên mạng được xác định thông qua
 - Địa chỉ IP duy nhất mà nó chạy trên một hệ thống.
 - Số hiệu cổng riêng được gán cho nó.
- 2 ứng dụng mạng liên lạc được với nhau cần phải thiết lập **kết nối** (**connection**).
 - Mỗi đầu **kết nối** tương ứng với một Socket.



Socket ?

- Vậy,
 - Một socket là một đầu cuối của một sự truyền thông 2 chiều, liên kết giữa hai chương trình chạy trên mạng.
 - Một socket được gán với một số hiệu cổng (port), vì thế tầng giao vận có thể nhận biết ứng dụng mà dữ liệu được chuyển đến.
- Socket cho phép thực hiện các hoạt động sau:
 - Kết nối đến máy ở xa
 - Gửi dữ liệu
 - Nhận dữ liệu
 - Đóng kết nối
 - Gắn với một cổng
 - Lắng nghe dữ liệu đến
 - Chấp nhận kết nối từ máy ở xa trên cổng đã được gán.



Các kiểu socket

- Có 2 kiểu socket
 - Kiểu thứ nhất tương tự như điện thoại kết nối đến một tổng đài. Kiểu này được gọi là kiểu hướng kết nối. TCP socket thuộc kiểu này.
 - Khi sử dụng những socket kết nối TCP, giao thức TCP bảo đảm dữ liệu đến đích an toàn và đầy đủ.
 - Kiểu thứ hai tương tự như hộp thư (mailbox), kiểu này còn gọi là kiểu phi kết nối, tức không giữ kênh kết nối trong quá trình truyền thông. UDP socket thuộc kiểu này (Unreliable Datagram Protocol (UDP)).



State và Stateless

- Stateful :
 - lưu giữ trạng thái giữa các lần kết nối (request/response).
- Stateless
 - Mỗi lần request/response thì cầu nối hủy bỏ. Không giữ trạng thái trước đó.



Các bước tạo một ứng dụng TCP

- Phía Server

- Gán một cổng với *Socket*
- Chờ và lắng nghe yêu cầu kết nối từ client đến
- Chấp nhận kết nối, tạo Socket tương ứng
- Truyền/nhận dữ liệu thông qua các streams in/out của đối đối tượng Socket
- Đóng kết nối

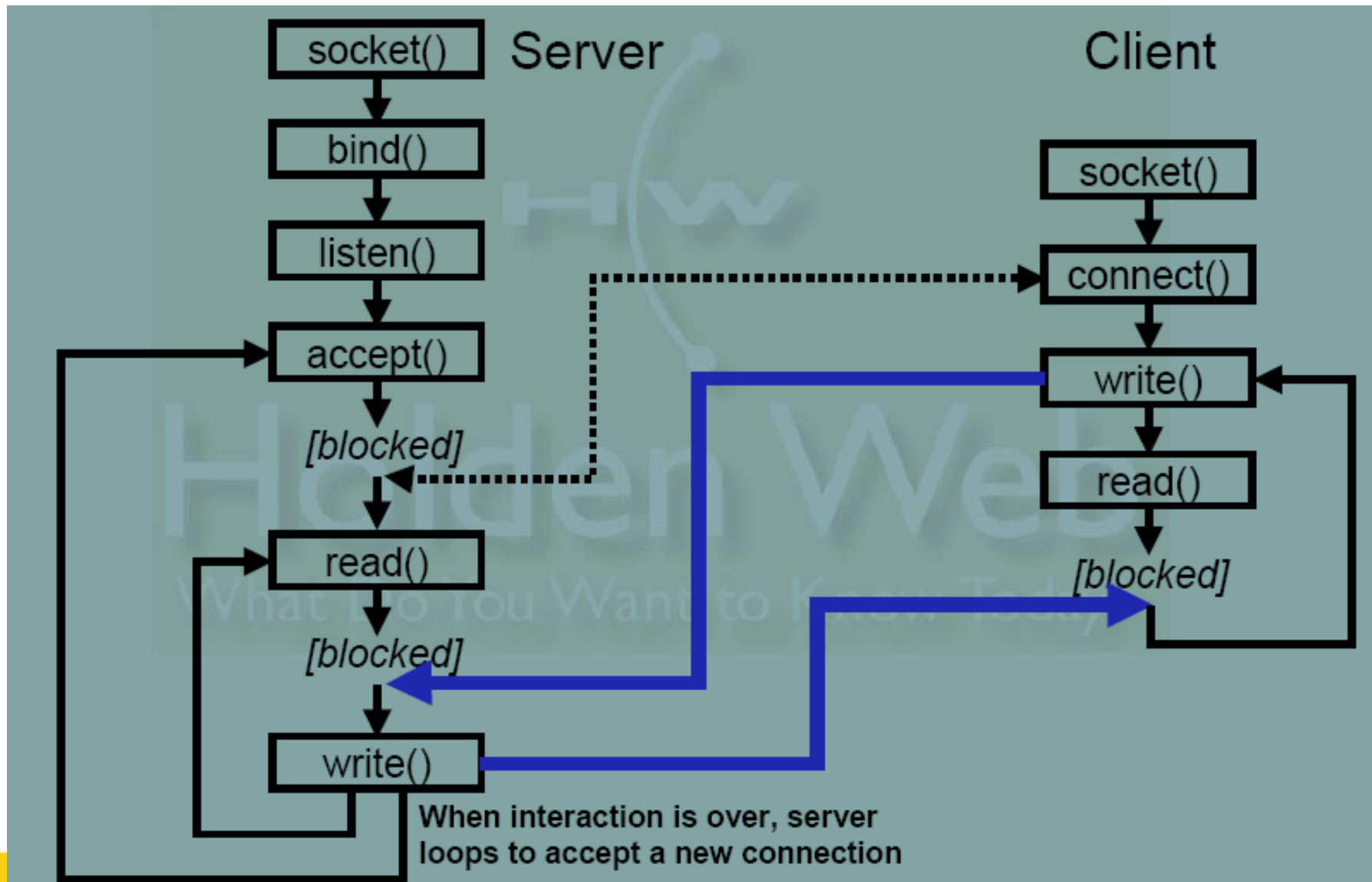
socket → bind → listen → accept → recv/send → close

- Phía Client

- Tạo một TCP socket với địa chỉ IP và số cổng mà chương trình Server đang chạy
- Thiết lập kết nối đến Server
- Trao đổi dữ liệu với Server
- Đóng kết nối

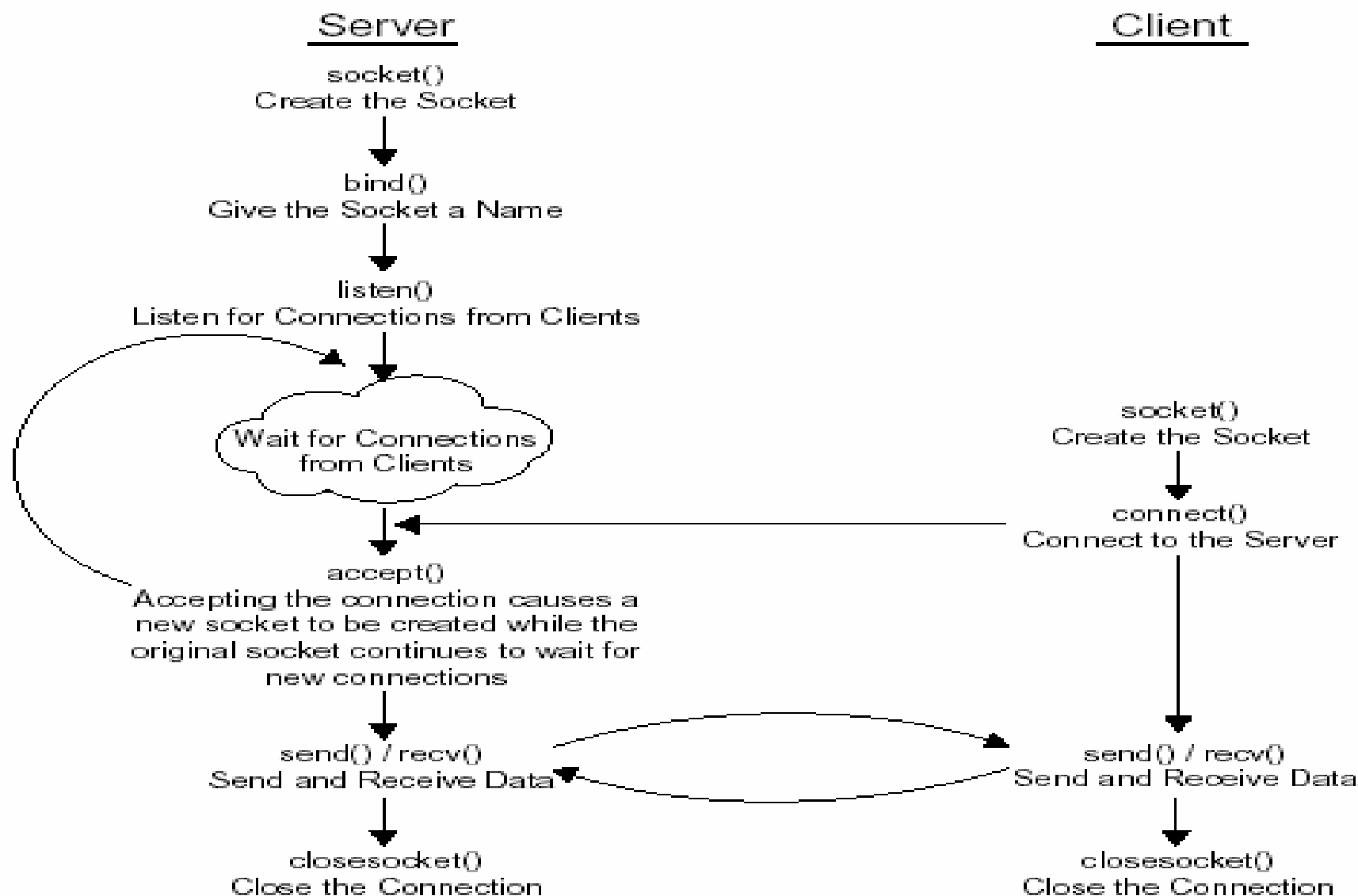
Socket → connect → send/recv → close

Các bước tạo một ứng dụng TCP





Các bước tạo một ứng dụng TCP





Ví dụ các bước tạo ứng dụng TCP với Java

- Phía Server

1: `ServerSocket server = new ServerSocket(7000);`

//Gắn một cổng 7000 với Socket

2: `Socket socket = server.accept();`

//Chờ và Lắng nghe yêu cầu từ client đến

3: nếu có yêu cầu kết nối từ client (hàm `accept()`) thực hiện, tạo ra một socket ở phía server)

4: từ đối tượng socket tham chiếu, tạo ra các stream in/out để trao đổi dữ liệu với client

`InputStream Stream_in = socket.getInputStream();`

`OutputStream Stream_out = socket.getOutputStream();`





Ví dụ các bước tạo ứng dụng TCP với Java

- Phía client

1: `Socket socket = new Socket("192.168.61.15", 7000);`

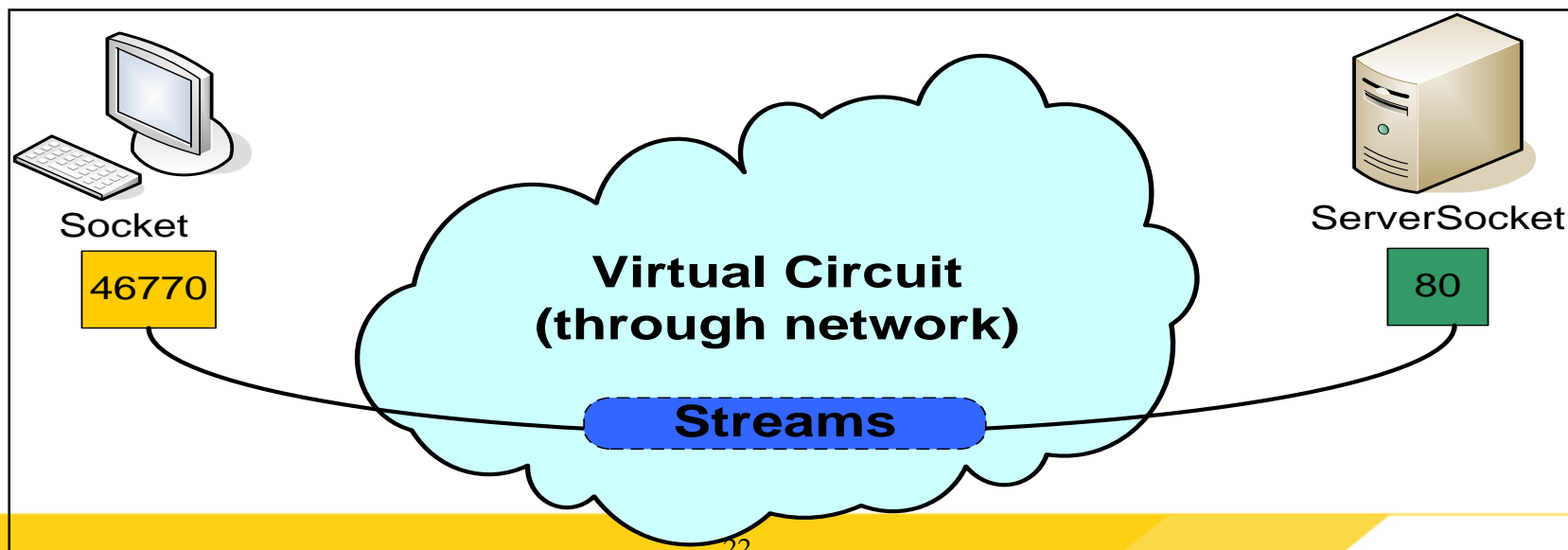
// Tạo một TCP socket với địa chỉ IP và số cổng mà chương trình Server đang chạy

// Nếu Server chấp nhận một kết nối đã được tạo ra, một socket được tạo

2: Từ đối tượng socket đã được tạo, tạo các streams in/out để trao đổi dữ liệu với server

`InputStream Stream_in = socket.getInputStream();`

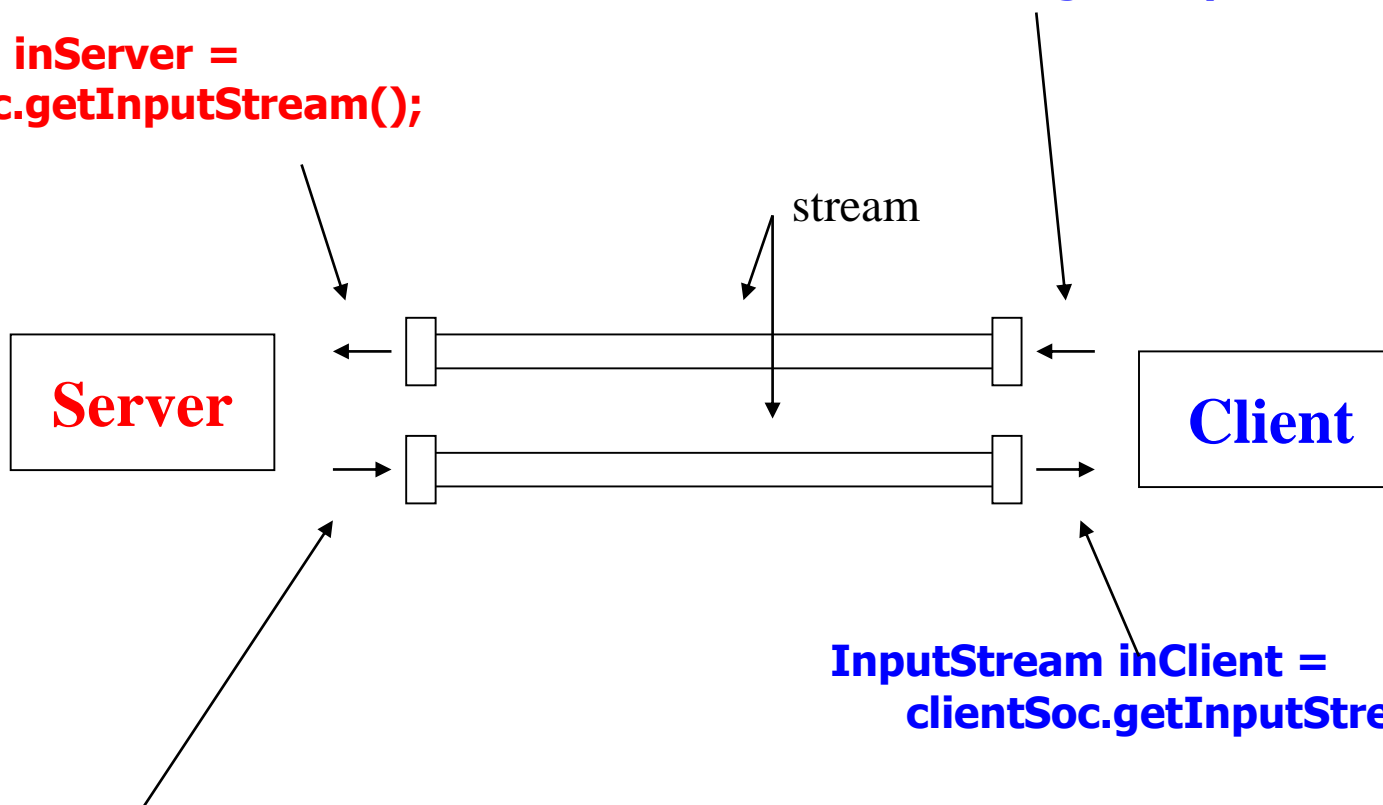
`OutputStream Stream_out = socket.getOutputStream();`



Mô tả quá trình trao đổi dữ liệu giữa client và server

**InputStream inServer =
serverSoc.getInputStream();**

**OutputStream outClient =
clientSoc.getOutputStream();**



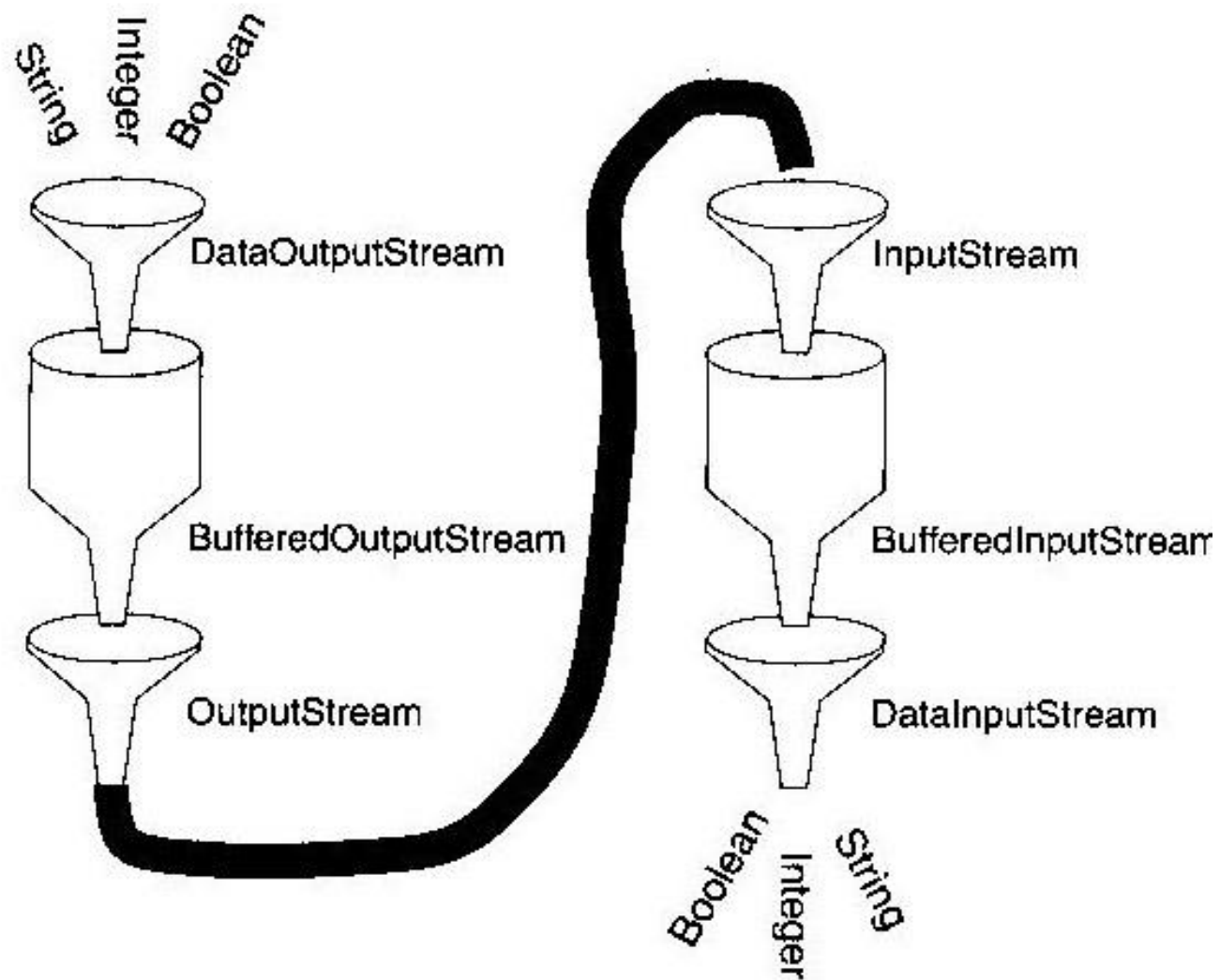
**InputStream inClient =
clientSoc.getInputStream();**

OutputStream outServer = serverSoc.getOutputStream();



Ví dụ chương trình Java : Time Server

```
import java.io.DataOutputStream;
import java.net.*;
import java.util.Date;
public class TimeServer {
    public static void main(String[] args) throws Exception {
        ServerSocket server = new ServerSocket(7000);
        System.out.println("Server is started");
        while(true) {
            Socket socket = server.accept();
            DataOutputStream dos = new
DataOutputStream(socket.getOutputStream());
            String time = new Date().toString();
            dos.writeUTF(time);
            socket.close();
        }
    }
}
```



Chương trình Java: Time client

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;

public class TimeClient {
    public static void main(String[] args) throws Exception {
        Socket socket = new Socket("localhost", 7000);
        DataInputStream din = new
            DataInputStream(socket.getInputStream());
        String time = din.readUTF();
        System.out.println(time);
    }
}
```



Chương trình server đa tuyến (Threaded Server)

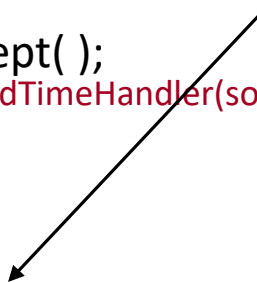
- Một vấn đề xảy ra với ví dụ vừa rồi là chỉ có một client kết nối đến server tại một thời điểm.
- Để cho phép nhiều client có thể kết nối đến server, thì server phải là chương trình đa tuyến. Mỗi tuyến (thread) đảm nhận việc liên lạc với client.
 - Mỗi khi có một client kết nối đến, chương trình server sinh ra một tuyến (thread) để điều khiển việc truyền thông với client



Chương trình server đa tuyến (Threaded Server)

- Đoạn mã chương trình bằng Java

```
Vector v=new Vector();  
while(true){  
    Socket socket = s.accept( );  
    ThreadedTimeHandler oi =new ThreadedTimeHandler(socket).start();  
    v.add(oi);  
}
```



```
class ThreadedTimeHandler extends Thread {  
    Socket socket;  
    public ThreadedTimeHandler(Socket s)  
    {  
        socket = s;  
    }  
    public void run() { ....  
        // đoạn mã truyền nhận dữ liệu ở đây  
    }  
}
```



Bài tập lập trình với TCP Socket

- **Mô hình Client/Server: Chat Room**

- Lập trình một chương trình chat room sử dụng TCP socket.
- Client có giao diện đơn giản hay phức tạp tùy vào bạn.
- Chat Server là một server có khả năng quản lý nhiều clients của chat room.
- Mỗi thông điệp từ một client gửi đến server, server phải có nhiệm vụ gửi đến tất cả các client còn lại, và tất cả client đều hiển thị thông điệp đó lên màn hình.
- Server nên dùng một Vector để lưu trữ tất cả các tiến trình clients. Nhờ Vector này mà nó có thể quản lý và truyền thông điệp đến tất cả các clients trong chat room.
- That's all. Good luck!.



- P2P
 - Viết ứng dụng chat theo mô hình P2P
 - Gợi ý: tham khảo cách hoạt động của Skype
- Viết ứng dụng truyền file
 - Gợi ý: tham khảo cách hoạt động của Napster, Gnutella, Kazaa



Thank your listenning