# BÀI 12.
# LẬP TRÌNH SOCKET TRONG JAVA

1

## Nội dung

- Cơ bản về mạng máy tính
- URL
- Lập trình socket trong Java

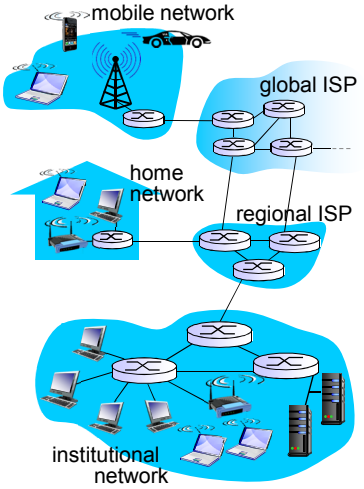
2

# 1. CƠ BẢN VỀ MẠNG MÁY TÍNH

3

## Mạng máy tính là gì?

- Tập hợp các máy tính kết nối với nhau dựa trên một kiến trúc nào đó để có thể trao đổi dữ liệu
  - Máy tính: máy trạm, máy chủ, bộ định tuyến
  - Kết nối bằng một phương tiện truyền
  - Theo một kiến trúc mạng




4

# Mạng máy tính là gì?

- Phương tiện truyền: đường truyền vật lý:
  - Hữu tuyến: cáp đồng, cáp quang
  - Vô tuyến: sóng hồng ngoại, sóng radio
- Kiến trúc mạng:
  - Hình trạng mạng: cách thức các máy tính kết nối bằng đường truyền vật lý với nhau
  - Giao thức mạng: cách thức các máy tính trao đổi dữ liệu với nhau như thế nào?
- Hoạt động cơ bản trên hệ thống mạng máy tính: truyền thông tin từ máy tính này sang máy tính khác
  - Tương tự như con người trao đổi thư tín qua hệ thống bưu điện
  - Máy nguồn: gửi dữ liệu
  - Máy đích: nhận dữ liệu

5

# Phân loại mạng máy tính

- Mạng cá nhân (PAN – Personal Area Network)
  - Phạm vi kết nối: vài chục mét
  - Số lượng người dùng: một vài người dùng
  - Thường phục vụ cho cá nhân
- Mạng cục bộ (LAN – Local Area Network):
  - Phạm vi kết nối: vài ki-lô-mét
  - Số lượng người dùng: một vài đến hàng trăm nghìn
  - Thường phục vụ cho cá nhân, hộ gia đình, tổ chức

6

# Phân loại mạng máy tính

- Mạng đô thị (MAN – Metropolitian Area Network)
  - Phạm vi kết nối: hàng trăm ki-lô-mét
  - Số lượng người dùng: hàng triệu
  - Phục vụ cho thành phố, khu vực
- Mạng diện rộng (WAN – Wide Area Network)
  - Phạm vi kết nối: vài nghìn ki-lô-mét
  - Số lượng người dùng: hàng tỉ
  - GAN – Global Area Network: phạm vi toàn cầu (Ví dụ: Internet)

7

# Trao đổi thông tin giữa các nút mạng

- Dữ liệu được tổ chức như thế nào?
- Định danh – đánh địa chỉ: Phân biệt các máy với nhau trên mạng?
- Tìm đường đi cho dữ liệu qua hệ thống mạng như thế nào?
- Làm thế nào để phát hiện lỗi dữ liệu (và sửa)?
- Làm thế nào để dữ liệu gửi đi không làm quá tải đường truyền, quá tải máy nhận?
- Làm thế nào để chuyển dữ liệu thành tín hiệu?
- Làm thế nào để biết dữ liệu đã tới đích?...

→ Phân chia nhiệm vụ cho các thành phần, tổ chức các thành phần thành các tầng (layer)

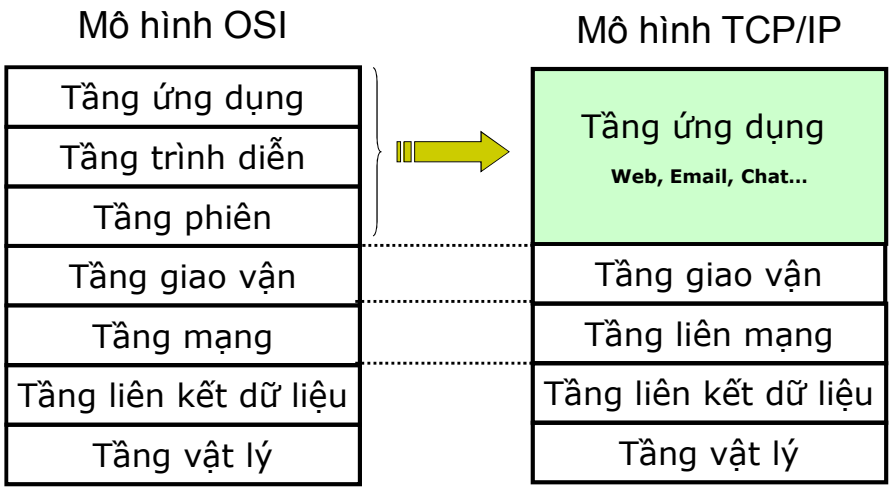
8

# Phân tầng

- Mỗi tầng:
  - Có thể có một hoặc nhiều chức năng
  - Triển khai dịch vụ để thực hiện các chức năng
    - Cung cấp dịch vụ cho tầng trên
    - Sử dụng dịch vụ tầng dưới
    - Độc lập với các tầng còn lại
  - Mỗi dịch vụ có thể có một hoặc nhiều cách triển khai khác nhau, cho phép tầng trên lựa chọn dịch vụ phù hợp
- Lợi ích:
  - Dễ dàng thiết kế, triển khai
  - Dễ dàng tái sử dụng
  - Dễ dàng nâng cấp

9

# Mô hình OSI và mô hình TCP/IP




10

# Định danh trên kiến trúc phân tầng

- Tầng ứng dụng : tên miền định danh cho máy chủ cung cấp dịch vụ
  - Tên miền: chuỗi ký tự dễ nhớ với người dùng. Thiết bị mạng không dùng tên miền khi truyền tin
  - Ví dụ: mps.gov.vn (máy chủ Web của Bộ CA)
- Tầng giao vận: số hiệu cổng định danh cho các dịch vụ khác nhau
  - Số hiệu cổng: từ 1-65535
  - Ví dụ: Web-80, DNS-53, Email(SMTP-25, POP-110, IMAP-143)
- Tầng mạng: địa chỉ IP định danh cho các máy trạm, máy chủ, bộ định tuyến
  - Có thể dùng trong mạng nội bộ và mạng Internet
  - Địa chỉ IPv4: 4 số có giá trị từ 0-255, các nhau bởi 1 dấu '.'
  - Ví dụ: 123.30.9.222 (máy chủ Web của Bộ CA)
- Tầng liên kết dữ liệu: địa chỉ MAC định danh cho các máy trạm, máy chủ, thiết bị mạng
  - Chỉ dùng trong mạng nội bộ

11

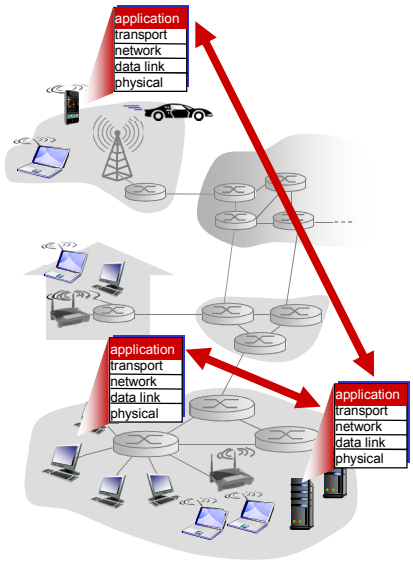
# Mô hình TCP/IP – Tầng ứng dụng

- Cung cấp dịch vụ mạng cho người dùng
- Phối hợp hoạt động của chương trình client và chương trình server
  - Client: cung cấp giao diện cho người dùng
  - Server: đáp ứng dịch vụ
- Một số dịch vụ tiêu biểu: Web, Email, Lưu trữ và chia sẻ file (FTP)...
- Mô hình cung cấp dịch vụ:
  - Client/Server
  - Ngang hàng
  - Mô hình lai

12

# Ứng dụng mạng

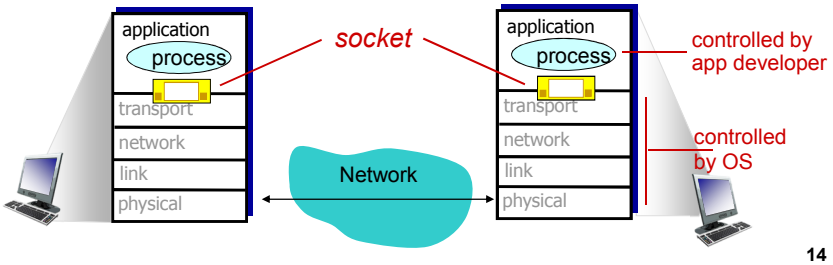
- Hoạt động trên các hệ thống đầu cuối (end system)
- Cài đặt giao thức ứng dụng để cung cấp dịch vụ
- Gồm có 2 tiến trình giao tiếp với nhau qua môi trường mạng:
  - Client: cung cấp giao diện NSD, gửi thông điệp yêu cầu dịch vụ
  - Server: cung cấp dịch vụ, trả thông điệp đáp ứng
- Ví dụ: Web
  - Web browser (trình duyệt Web): Chrome, Firefox…
  - Web server: Apache, Tomcat…




13

# Giao tiếp giữa các tiến trình ứng dụng

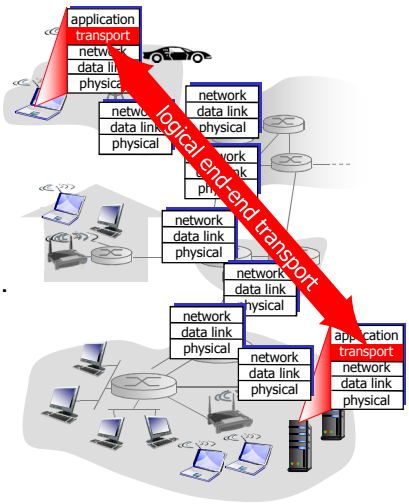
- Socket: dịch vụ mà tầng giao vận cung cấp cho tiến trình trên ứng dụng sử dụng để trao đổi dữ liệu
- Socket được định danh bởi: Địa chỉ IP, Số hiệu cổng
  - Ví dụ: 202.191.56.65:80 cho web server của SoICT
- Tiến trình server sử dụng socket có địa chỉ cố định để chờ yêu cầu của client gửi đến




14

# Tầng giao vận

- Được cài đặt trên các hệ thống cuối
- Cung cấp dịch vụ để các ứng dụng mạng trao đổi dữ liệu
- Hai dạng dịch vụ giao vận
  - Tin cậy, hướng liên kết, e.g TCP
  - Không tin cậy, không liên kết, e.g. UDP
- Đơn vị truyền: datagram (UDP), segment (TCP)




15

# Thông số của liên kết

- Mỗi một liên kết tạo ra trên tầng giao vận để vận chuyển dữ liệu cho tiến trình tầng ứng dụng của 2 nút mạng được xác định bởi bộ 5 thông số (5-tuple):
  - Địa chỉ IP nguồn
  - Địa chỉ IP đích

  } Tầng mạng
  - Số hiệu cổng nguồn
  - Số hiệu cổng đích
  - Giao thức (TCP/UDP,...)

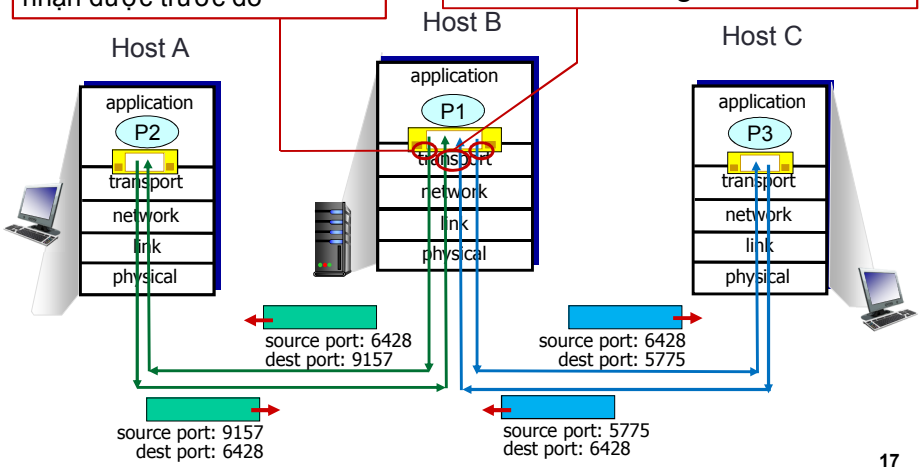
  } Tầng giao vận

16

# UDP socket trên ứng dụng mạng

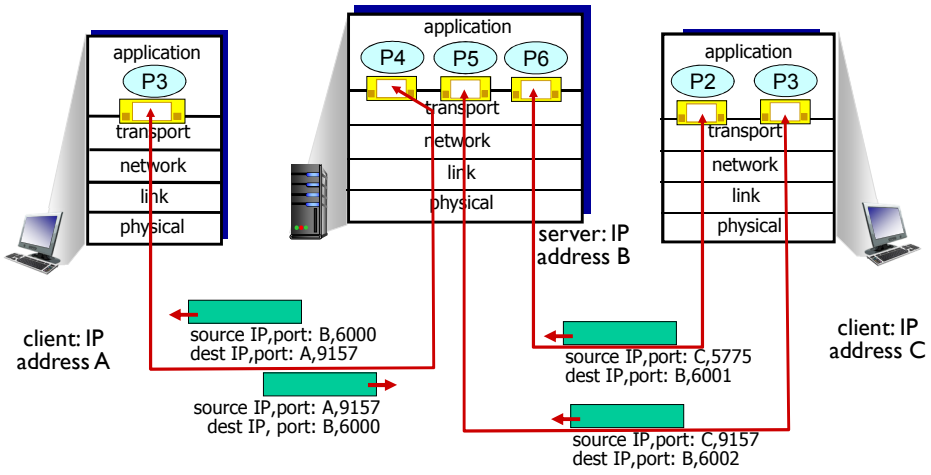
Gửi dữ liệu tới đúng tiến trình đích bằng địa chỉ IP nguồn và cổng nguồn trên dữ liệu nhận được trước đó

Nhận dữ liệu: Dựa trên số hiệu cổng đích trên bức tin để đưa dữ liệu đến đúng socket



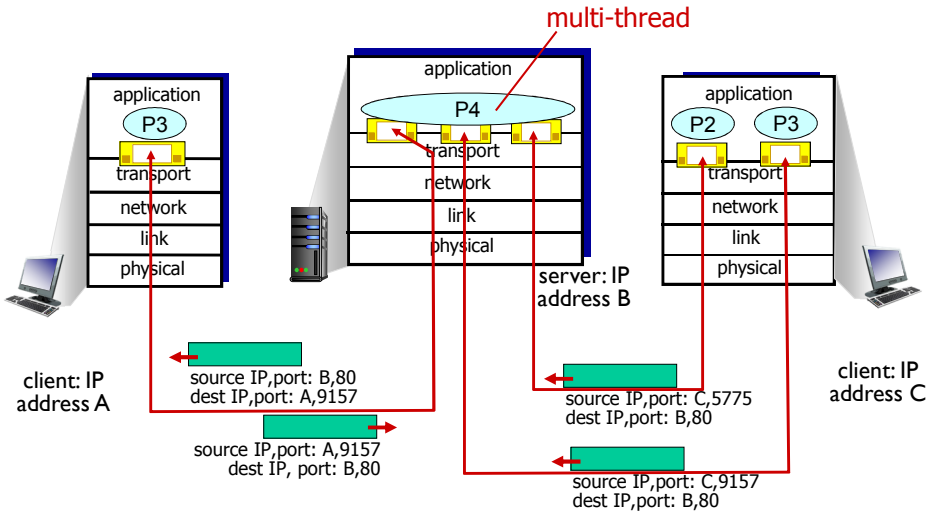

17

# TCP socket trên ứng dụng mạng




18

# TCP socket trên ứng dụng mạng




19

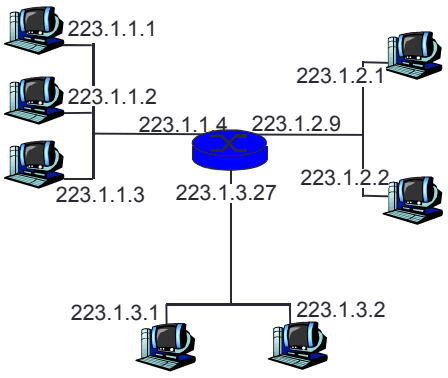
# Mô hình TCP/IP - Tầng liên mạng

- Cung cấp các cơ chế để kết nối các hệ thống mạng với nhau (internetworking)
  - Mạng của các mạng
- Giao thức IP : Internet Protocol
  - Định danh: sử dụng địa chỉ IP để gán cho các nút mạng (máy trạm, máy chủ, bộ định tuyến)
  - Khuôn dạng dữ liệu
- Định tuyến(chọn đường): tìm các tuyến đường tốt nhất qua hệ thống trung gian để gửi thông tin
- Chuyển tiếp: quyết định gửi dữ liệu qua tuyến đường nào

20

# Địa chỉ IP (IPv4)

- *Địa chỉ IP:* Một số 32-bit để định danh cổng giao tiếp mạng trên nút đầu cuối (PC, server, smart phone), bộ định tuyến
- Mỗi địa chỉ IP được gán cho một cổng duy nhất
- Địa chỉ IP có tính duy nhất trong mạng

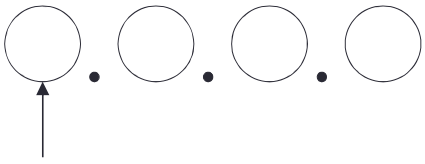



223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

21

# Biểu diễn địa chỉ IPv4

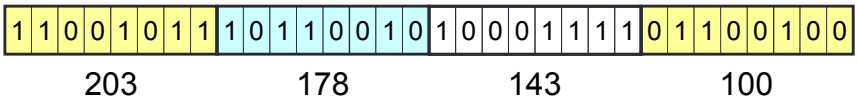


8 bits
0 – 255 integer

Ví dụ:

| | |
|---|---|
| 203.178.136.63 | o |
| 259.12.49.192 | x |
| 133.27.4.27 | o |

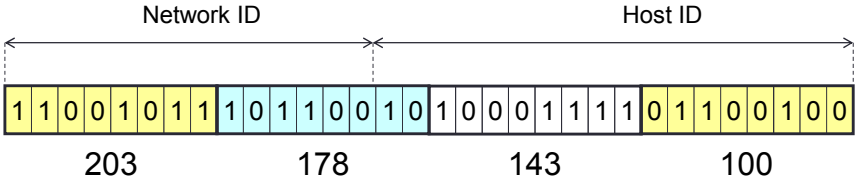
Sử dụng 4 phần 8 bits để miêu tả một địa chỉ 32 bits




22

# Địa chỉ IPv4

- Địa chỉ IP có hai phần
  - Host ID – phần địa chỉ máy trạm
  - Network ID – phần địa chỉ mạng




- Làm thế nào biết được phần nào là cho máy trạm, phần nào cho mạng?
  - Phân lớp địa chỉ
  - Không phân lớp – CIDR

23

# Các dạng địa chỉ

- Địa chỉ mạng (Network Address):
  - Định danh cho một mạng
  - Tất cả các bit phần HostID là 0
- Địa chỉ quảng bá (Broadcast Address)
  - Địa chỉ dùng để gửi dữ liệu cho tất cả các máy trạm trong mạng
  - Tất cả các bit phần HostID là 1
- Địa chỉ máy trạm
  - Gán cho một cổng mạng
- Địa chỉ nhóm (Multicast address): định danh cho nhóm

24

## 2. LỚP INETADDRESS



## java.net.InetAddress

- Chứa thông tin địa chỉ Internet của một nút mạng
- Các phương thức:
  - `static InetAddress getByName(String host)`: tạo một đối tượng `InetAddress` chứa thông tin nút mạng có tên `host`
  - `static InetAddress[] getAllByName(String host)`
  - `static InetAddress getByAddress(byte[] addr)`: tạo một đối tượng `InetAddress` chứa thông tin nút mạng có địa chỉ nằm trong mảng `addr`
  - `static InetAddress getByAddress(String host, byte[] addr)`
  - `static InetAddress getLocalHost()`: tạo đối tượng `InetAddress` chứa thông tin máy trạm cục bộ
  - `static InetAddress getLoopbackAddress()`: tạo đối tượng `InetAddress` chứa thông tin máy trạm cục bộ

# Các phương thức

- `public String getHostName():`
  - Phương thức này trả về một xâu biểu diễn tên (hostname) của một đối tượng `InetAddress`.
  - Nếu máy không có hostname, thì nó sẽ trả về địa chỉ IP của máy này dưới dạng một xâu ký tự.
- `public byte[] getAddress() :`
  - Trả về một địa chỉ IP dưới dạng một mảng các byte.
- `public boolean isReachable(int timeout)`
  - Kiểm tra kết nối tới máy trạm có địa chỉ trong đối tượng InetAddress với timeout xác định

27

# Ví dụ 1

```
try{
   InetAddress host =
       InetAddress.getByName("www.microsoft.com");
    System.out.println(host);
    if(!host.isReachable(10000))
       System.out.println("Could not connect this host");
}
catch(UnknownHostException e)
{
    System.println("Could not find this host");
}
```

28

# Ví dụ 2

```
try{
   byte[] addr = {8, 8, 8, 8};
    InetAddress otherHost =
                    InetAddress.getByAddress(addr);
    if(!otherHost.isReachable(10000))
       System.out.println("Could not connect this hos")
}
catch(UnknownHostException e)
{
    System.println("Could not find this host");
}
```

29

## 3. TRUY CẬP TÀI NGUYÊN QUA URL

30

# Địa chỉ URL

- Uniform Resource Locator: định vị tài nguyên trên mạng
- Cấu trúc:

*protocol://hostname:port/directory/URI*

*protocol:* Giao thức điều khiển truy cập tài nguyên

*hostname:* Địa chỉ nút mạng vật lý (tên miền, địa chỉ IP)

*port*: Số hiệu cổng dịch vụ (có thể không cần nếu server sử dụng cổng chuẩn)

*directory*: đường dẫn thư mục

*URI*: định danh tài nguyên (thường là tên file)

31

# Lớp `java.net.URL`

- Nằm tại java.net.URL để thực hiện các thao tác định vị tài nguyên
- Đối tượng URL bao gồm các thuộc tính:
  - Giao thức (protocol),
  - Hosname
  - Cổng (port)
  - Đường dẫn (path)
  - Tên tập tin (filename)
  - Mục tài liệu (document section)

32

# Các constructor của URL

`public URL(String url) throws MalformedURLException`

- Chỉ chứa xâu địa chỉ URL (như dạng website)

- Ví dụ

```
try{
    URL u = new URL("http://www.sun.com/index.html");
}
catch(MalformedURLException e)
{
    System.err.println(e);
}
```

33

# Các constructor của URL

`public URL(String protocol, String host, String file) throws MalformedURLException`

- URL xây dựng từ các xâu phân biệt xác định giao thức, hostname, và tệp tin.
- Cổng mặc định cho giao thức sẽ được sử dụng.

- Ví dụ

```
try{
       URL u = new URL("http","/www.sun.com","index.html");
}
catch(MalformedURLException e){
   System.err.println(e);
}
```

34

# Các constructor của URL

- `public URL(String protocol, String host, int port, String file) throws MalformedURLException`
  - Bổ sung tham số cổng.
- Ví dụ

```
try{
   URL u = new URL("http","/www.sun.com",80,"index.html");
}
catch(MalformedURLException e){
   System.err.println(e);
}
```

35

# Các constructor của URL

- `public URL(URL u, String s) throws MalformedURLException`
  - URL tuyệt đối từ URL tương đối;
- Ví dụ

```
URL u1,u2;
try{
    URL u1= new URL("http://www.macfaq.com/index.html");
    URL u2 = new URL(u1,"vendor.html");
}
catch(MalformedURLException e)
{
    System.err.println(e);
}
```

36

# Các phương thức

- `public String getProtocol()`
  - Trả về một xâu ký tự biểu diễn phần giao thức của URL
- `public String getHost()`
  - Trả về một xâu ký tự biểu diễn phần hostname của URL.
- `public int getPort()`
  - Trả về một số nguyên kiểu int biểu diễn số hiệu cổng có trong URL.
- `public int getDefaultPort()`
  - Phương thức getDefautlPort() trả về số hiệu cổng mặc định cho giao thức của URL
- `public String getFile()`
  - Trả về một xâu ký tự chứa đường dẫn tới tài nguyên

37

# Các phương thức (tiếp)

- `public final InputStream openStream()`
  - Đọc dữ liệu từ tài nguyên đã truy cập được
  - Dữ liệu nhận từ luồng này là dữ liệu thô của một tệp tin mà URL tham chiếu (mã ASCII nếu đọc một tệp văn bản, mã HTML nếu đọc một tài liệu HTML, một ảnh nhị phân nếu ta đọc một file ảnh).
- `public URLConnection openConnection()`
  - Trả vể một URLConnection  kết nối với tài nguyên

38

# Ví dụ

```
url = new
      URL("http://soict.hust.edu.vn/~tungbt/index.htm");
try(BufferedReader br = new BufferedReader(new
             InputStreamReader(url.openStream()))){
   String line;
   while((line = br.readLine()) != null)
       System.out.println(line);
}
catch(IOException e){
   System.out.println(e.getMessage());
}
```

39

# Lớp URLConnection

- Cung cấp thêm nhiều phương thức hơn để tương tác với tài nguyên
- Các lớp con: HttpURLConnection, HttpsURLConnection
- Các phương thức:
  - `InputStream getInputStream()`: đọc dữ liệu từ URLConnection
  - `OutputStream getOutputStream()`: ghi dữ liệu lên URLConnection
- Đọc thêm về URLConnection và các lớp con trong tài liệu của Java

40

## 4. LẬP TRÌNH SOCKET

41

## Socket là gì?

- Socket là công cụ để các ứng dụng trao đổi dữ liệu
- Trên kiến trúc phân tầng của hệ thống mạng, socket là dịch vụ mà tầng giao vận cung cấp cho tầng ứng dụng
- Hai dạng socket:
  - TCP socket
  - UDP socket
- Ứng dụng sử dụng TCP Socket:
  - Lớp `Socket`: sử dụng để client và server trao đổi dữ liệu
  - Lớp `ServerSocket`: sử dụng tại ứng dụng server để nghe yêu cầu kết nối từ client
- Ứng dụng sử dụng UDP Socket:
  - Lớp `DatagramSocket`: truyền tải dữ liệu
  - Lớp `DatagramPacket`: xử lý dữ liệu

42

# XÂY DỰNG ỨNG DỤNG SỬ DỤNG TCP SOCKET

43

## Lớp `Socket`

- Các phương thức khởi tạo: sau khi được khởi tạo, socket tự động gửi yêu cầu kết nối tới server
  - `Socket(InetAddress serverAddr, int)`: tạo socket kết nối tới cổng `port` trên server có địa chỉ `serverAddr`
  - `Socket(String hostname, int port)`: tạo socket kết nối tới cổng `port` trên server có tên `hostname`
  - `Socket (String host, int port, InetAddress interface, int localPort)`: tạo socket có địa chỉ xác định
  - `Socket (InetAddress host, int port, InetAddress interface, int localPort)`: tạo socket có địa chỉ xác định

44

## Lớp `Socket`–Các phương thức

- `InetAddress getInetAddress()`: lấy địa chỉ của bên kia
- `int getPort()`: lấy số hiệu cổng của bên kia
- `int getLocalPort()`: lấy số hiệu cổng trên nút mạng cục bộ mà ứng dụng đang chạy
- `InetAddress getLocalAddress()`: lấy địa chỉ trên nút mạng cục bộ
- `InputStream getInputStream()`: trả về luồng InputStream để nhận dữ liệu
- `OutputStream getOutputStream()`: trả về luồng OutputStream để gửi dữ liệu
  - Gửi-nhận dữ liệu trên socket giống vào-ra trên file

45

## Lớp `Socket`–Các phương thức

- `void setSoTimeout(int timeout)`: thiết lập thời gian chờ
- `void setSendBufferSize(int size)`: thiết lập kích thước bộ đệm gửi
- `void setReceiveBufferSize(int size)`: thiết lập kích thước bộ đệm nhận
- `void close()`: đóng socket cả 2 chiều gửi-nhận
- `void shutdownInput()`: đóng socket chiều nhận
- `void shutdownOutput()`: đóng socket chiều gửi
- `boolean isInputShutdown()`: kiểm tra tình trạng đóng chiều nhận
- `boolean isOutputShutdown()`: kiểm tra tình trạng đóng chiều gửi

46

## Lớp `ServerSocket`

- Các phương thức khởi tạo: sau khi được khởi tạo, ServerSocket tự động thiết lập trạng thái chờ yêu cầu tạo kết nối TCP từ client gửi tới
- `ServerSocket(int port)`: tạo ServerCoket với số hiệu cổng `port` chỉ định
  - Ngoại lệ sinh ra khi cổng đã được sử dụng hoặc sử dụng một cổng trong dải cổng chuẩn 0-1023.
- `ServerSocket(int port, int max)`: tạo ServerSocket với số kết nối tối đa chấp nhận là `max`
- `ServerSocket(int port, int max, InetAddress addr)`: tạo ServerSoket với địa chỉ IP và số hiệu cổng chỉ định

47

## Lớp `ServerSocket`-Các phương thức

- `Socket accept()`: chấp nhận yêu cầu xin kết nối từ client, trả về một đối tượng `Socket` để trao đổi dữ liệu với client đó.
- `void close()`: đóng ServerSocket
- `void setSoTimeout(int timeout)`: thiết lập thời gian chờ
- `void setSendBufferSize(int size)`: thiết lập kích thước bộ đệm gửi
- `void setReceiveBufferSize(int size)`: thiết lập kích thước bộ đệm nhận
- `InetAddress getInetAddress()`
- `int getLocalPort()`

48

# Các bước xây dựng ứng dụng server

- B1: Khởi tạo một đối tượng `ServerSocket` để nghe yêu cầu kết nối từ client

Sử dụng một vòng lặp để thực hiện các bước 2-5:

- B2: Gọi phương thức accept() để chấp nhận. Phương thức này trả về một đối tượng `Socket` để trao đổi dữ liệu với client.
- B3: sử dụng phương thức `getInputStream()` và `getOutputStream()` lấy các luồng vào-ra để trao đổi dữ liệu với client
- B4: Trao đổi dữ liệu với client
- B5: Đóng socket kết nối với client
- B6: Đóng SeverSocket

49

# Ví dụ - TCPEchoServer

```
public class TCPEchoServer {
   public final static int DEFAULT_PORT = 5000;
   public static void main(String[] args) {
      try(ServerSocket servSocket = new
              ServerSocket(DEFAULT_PORT)){
         while (true){
            Socket connSocket = servSocket.accept();
            System.out.println("Accepted client:" +
              connSocket.getInetAddress().getHostAddress());
            try(BufferedReader in = new BufferedReader(new
               InputStreamReader(connSocket.getInputStream()));
               PrintWriter out = new PrintWriter(new
               OutputStreamWriter(connSocket.getOutputStream()))
            ){
```

50

# Ví dụ - TCPEchoServer (tiếp)

```
                String message;
                while((message = in.readLine()) != null){
                    System.out.println("Receive from client:"
                            + message);
                    out.println(message);
                    out.flush();
                }
                System.out.println("Client has stopped sending
                                        data!");
            }catch (IOException e){
                System.out.println(e.getMessage());
            }
        }
    }catch (IOException e){
        System.out.println(e.getMessage());
    }
  }
}
```

51

# Các bước xây dựng ứng dụng client

- B1: Khởi tạo một đối tượng Socket để kết nối với server
- B2: sử dụng phương thức `getInputStream()` và `getOutputStream()` lấy các luồng vào-ra để trao đổi dữ liệu với server
  - Gửi-nhận dữ liệu với Socket giống vào-ra trên file
  - Cần sử dụng linh hoạt các loại luồng vào-ra trong Java
- B3: Trao đổi dữ liệu với server
- B4: Đóng socket

52

# Ví dụ - TCPEchoClient

```
public class TCPEchoClient {
   public static void main(String[] args) {
       try(Socket clientSocket = new Socket("localhost", 5000);
           BufferedReader user = new BufferedReader(new
                                 InputStreamReader(System.in));
           BufferedReader in = new BufferedReader(new
              InputStreamReader(clientSocket.getInputStream()));
           PrintWriter out = new PrintWriter(new
              OutputStreamWriter(clientSocket.getOutputStream()))
       ){
          String message;
          while(true){
              System.out.print("Send to server: ");
              message = user.readLine();
              if(message.length() == 0)
                 break;
```

53

# Ví dụ - TCPEchoClient (tiếp)

```
              out.println(message);
              out.flush();
              String reply;
              reply = in.readLine();
              System.out.println("Reply from Server:" + reply);
          }
          clientSocket.close();
       }catch (IOException e){
          System.out.println(e.getMessage());
       }
   }
}
```

54

# Truyền các đối tượng qua socket

- Sử dụng luồng `ObjectOutputStream` để gửi đối tượng qua socket
  - Phương thức `writeObject(Object o)`
- Sử dụng luồng `ObjectInputStream` để nhận đối tượng từ socket
  - Phương thức `Object readObject()`
- Đối tượng truyền trên socket phải thuộc lớp được triển khai từ giao diện `Serializable`
  - Các lớp của Java định nghĩa đều triển khai từ `Serializable`

55

# Ví dụ - Student

```
public class Student implements Serializable {
    private String id;
    private String name;

    public Student(String id, String name) {
        this.id = id;
        this.name = name;
    }

    public String getId() {
        return id;
    }

    public String getName() {
        return name;
    }
}
```

56

# Ví dụ - Server

```
public class Server {
   public final static int DEFAULT_PORT = 5000;
   private static void communicate(Socket connSocket){
        try(ObjectInputStream in = new
                ObjectInputStream(connSocket.getInputStream())
        ){
           Student student;
           try{
                while((student = (Student) in.readObject()) != null)
                   System.out.println("Received: " +
                                        student.getName());
           }catch (ClassNotFoundException e) {
                System.out.println("Invalid data from client!");
           }catch(IOException e){
                System.out.println("Client stopped sending data!");
           }
        }catch(IOException e){
        System.out.println("Cannot communicate to client!");
        }
   }
```

57

# Ví dụ - Server (tiếp)

```
   public static void main(String[] args) {
        try(ServerSocket lisSocket = new
                        ServerSocket(DEFAULT_PORT)
        ){
           System.out.println("Server started!");
           while(true){
                Socket connSocket = lisSocket.accept();
                communicate(connSocket);
           }
        }catch (IOException e) {
           System.out.println("Cannot start server on port 5000!");
        }
   }
}
```

58

# Ví dụ - Client

```
public class Client {
    public static void main(String[] args) {
        try(Socket clientSocket = new Socket("localhost", 5000);
            BufferedReader user = new BufferedReader(new
                                  InputStreamReader(System.in));
            ObjectOutputStream out = new
                ObjectOutputStream(clientSocket.getOutputStream())
        ){
           while(true){
                String id;
                System.out.println("Student's ID: ");
                id = user.readLine();
                if(id.length() == 0){
                   System.out.println("Stopped sending data to
                                       server!");
                   break;
                }
```

59

# Ví dụ - Client (tiếp)

```
                String name;
                System.out.println("Student's name: ");
                name = user.readLine();

                Student student = new Student(id, name);
                out.writeObject(student);
                out.flush();
           }
           clientSocket.close();
        }catch(IOException e){
                System.out.println("Cannot connect to server!");
        }
    }
}
```

60

# Đa luồng trong lập trình socket

- Với các ví dụ trên → 1 server tại một thời điểm chỉ xử lí một client → không thể đáp ứng nhiều yêu cầu một lúc
- Sử dụng đa luồng để khắc phục nhược điểm
  - Khi có một client kết nối đến server (accept trả về)
  - Tạo ra một luồng để xử lí công việc với client đó

61

# Ví dụ - TCPEchoThread

```
public class TCPEchoThread implements Runnable{
   private Socket socket;
   public EchoThread(Socket s){
       socket = s;
   }
   public void run(){
       try(BufferedReader in = new BufferedReader(new
              InputStreamReader(socket.getInputStream()));
           PrintWriter out = new PrintWriter(new
              OutputStreamWriter(socket.getOutputStream()))
       ){
          String message;
```

62

# Ví dụ - EchoThread(tiếp)

```
            while((message = in.readLine()) != null){
                System.out.println("Receive from client:" +
                                    message);
                out.println(message);
                out.flush();
            }
            System.out.println("Client has stopped sending
                                    data!");
            socket.close();
        }catch (IOException e){
            System.out.println(e.getMessage());
        }
    }
}
```

63

# Ví dụ - MultiThreadTCPEchoServer

```
public class MultiThreadTCPEchoServer {
    public final static int DEFAULT_PORT = 5000;

    public static void main(String[] args) {
        try(ServerSocket servSocket = new
                    ServerSocket(DEFAULT_PORT)
        ){
            while(true){
                Runnable t = new
                       TCPEchoThread(servSocket.accept());
                new Thread(t).start();
            }
        }catch(IOException e){
            System.out.println(e.getMessage());
        }
    }
}
```

64

## XÂY DỰNG ỨNG DỤNG SỬ DỤNG UDP SOCKET



## Lớp DatagramPacket

- Cung cấp các phương thức để nhận và thiết lập:
  - Địa chỉ nguồn, đích
  - Nhận và thiết lập các thông tin về cổng nguồn và đích
  - Nhận và thiết lập độ dài dữ liệu.
- DatagramPacket sử dụng các phương thức khởi tạo khác nhau tùy thuộc vào gói tin được sử dụng để gửi hay nhận dữ liệu.

# Khởi tạo DatagramPacket để nhận

- `DatagramPacket(byte[] buf, int length)`
  - Khởi tạo DatagramPacket để nhận dữ liệu có kích thước length và lưu trên buf
  - Chú ý: length ≤ buf.length()
- `DatagramPacket(byte[] buf,int offset, int length)`
  - Lưu dữ liệu trên buf từ vị trí offset
  - Chú ý: length ≤ buf.length()−offset

67

# Khởi tạo DatagramPacket để gửi

- `DatagramPacket(byte[] buf, int length, InetAddress address, int port)`
  - `buf`: chứa dữ liệu cần gửi
  - `length`: kích thước dữ liệu
  - `port`: cổng nhận
  - `address`: nút mạng nhận
- `DatagramPacket(byte[] buf, int offset, int length, InetAddress address, int port)`
  - `offset`: ví trí bắt đầu của dữ liệu cần gửi trên `buf`

68

## Các phương thức

- `InetAddress getAddress()`
- `int getPort()`
- `byte[] getData()`: trả về mảng byte chứa dữ liệu
- `int getLength()`
- `void setAddress(InetAddress iaddr)`
- `void setPort(int iport)`
- `void setData(byte[] buf)`

69

## Lớp DatagramSocket

- DatagramSocket được sử dụng để gửi và nhận các gói tin UDP
- Các phương thức khởi tạo:
  - `DatagramSocket(int port)`: khởi tạo với cổng chỉ định
  - `DatagramSocket(int port, InetAddress laddr)`: khởi tạo với cổng và địa chỉ IP chỉ định
- Các phương thức trao đổi dữ liệu:
  - `void connect(InetAddress address, int port)`: kết nối với một socket khác để trao đổi dữ liệu
  - `void receive(DatagramPacket p)`:nhận dữ liệu
  - `void send(DatagramPacket p)`: gửi dữ liệu
  - `void close()`: đóng socket
  - `void setSoTimeout(int timeout)`: thiết lập thời gian chờ (milisecond)

70

# Ví dụ: UDPEchoServer

```
public class UDPEchoServer {
   public static void main(String[] args) {
       try(DatagramSocket servSocket = new DatagramSocket(5050)){
          System.out.println("Server started");
          servSocket.setSoTimeout(10000);
          byte[] buff = new byte[1024];
          while(true){
              try{
                 DatagramPacket in = new DatagramPacket(buff,
                            buff.length);
                 servSocket.receive(in);
                 DatagramPacket out = new
                     DatagramPacket(in.getData(), in.getLength(),
                     in.getAddress(),in.getPort());
                 servSocket.send(out);
              }
```

71

# Ví dụ: UDPEchoServer(tiếp)

```
              catch(SocketTimeoutException e){
                 System.out.println("Timeout!");
              }catch(IOException e){
                 System.out.println("Cannot communicate to
                                        client");
              }
          }
       }catch(SocketException e){
          System.out.println("Cannot start server!");
       }
   }
}
```

72

# Ví dụ: UDPEchoClient

```
public class UDPEchoClient {
   public static void main(String[] args) {
       try(DatagramSocket cliSocket = new DatagramSocket(5051)){
          try(BufferedReader user = new BufferedReader(new
                              InputStreamReader(System.in))
          ){
               String message;
               while(true){
               InetAddress servAddr =
                      InetAddress.getByName("localhost");
               System.out.print("Send to server: ");
               message = user.readLine();
               if(message.length() == 0) break;
               byte[] buff = message.getBytes("UTF-8");
               DatagramPacket out = new DatagramPacket(buff,
                                    buff.length,servAddr, 5050);
               cliSocket.send(out);
```

73

# Ví dụ: UDPEchoClient(tiếp)

```
               DatagramPacket in = new DatagramPacket(buff,
                                                   buff.length);
               cliSocket.receive(in);
               message = new String(buff, 0, buff.length,"UTF-8");
               System.out.println("From server " + ": " +
                                                        message);
          }
       }catch(IOException e){
          System.out.println("Cannot communicate to server!");
       }
       }catch(SocketException e){
          System.out.println("Cannot start client!);
       }
   }
}
```

74