LẬP TRÌNH MẠNG

TCP Socket

hungdn@ptit.edu.vn

Nội dung

- Giới thiệu
- Một số thao tác với host
- Lập trình ứng dụng mạng với TCP - Socket

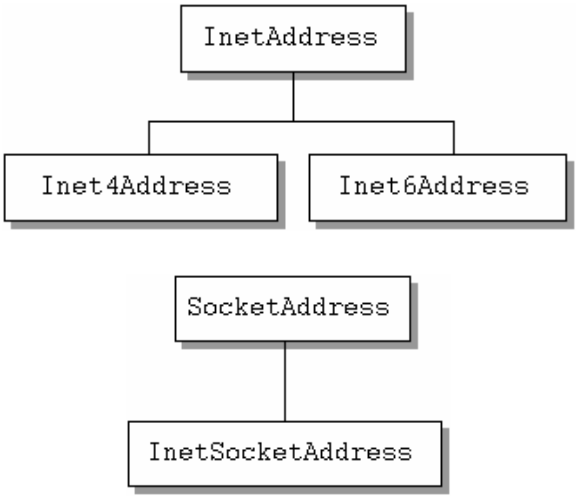
1. Giới thiệu

Giới thiệu

- **Lập trình mạng sử dụng TCPSocket và UDPSocket sử dụng ngôn ngữ Java**
  - Java.net
  - Java.nio
- **Java NIO là một API cung cấp khả năng nhập xuất dữ liệu, truyền dữ liệu qua mạng**
  - Java 1.4
- **Các gói quan trọng nhất trong gói java.net gồm 6 lớp**
  - InetAddress
  - URI/URL
  - ServerSocket
  - Socket
  - DatagramPacket
  - DatagramSocket

Một số thao tác với máy trạm

- **Lớp InetAddress là lớp quan trọng cung cấp hầu hết các thao tác khi làm việc với host (node) trên mạng**
  - Lấy thông tin
  - Phân loại địa chỉ
  - Kiểm tra trạng thái
  - ...
- **InetAddress có 2 lớp kế thừa**
  - Inet4Address
  - Inet6Address
- **Lớp SocketAddress**
  - InetSocketAddress



Lớp InetAddress – một số phương thức

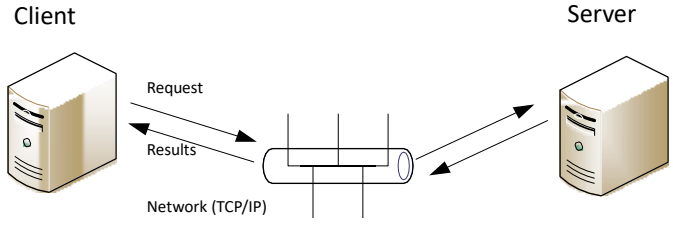
Tóm tắt các phương thức của lớp InetAddress	
boolean	equals(Object obj) So sánh đối tượng với đối tượng obj
byte[]	getAddress() Trả về địa chỉ IP chứa trong đối tượng InetAddress dạng mảng byte
static InetAddress[]	getAllByName(String host) Trả về mảng địa chỉ của tất cả các máy trạm có cùng tên trên mạng
static InetAddress	getByAddress(byte[] addr) Trả về đối tượng InetAddress tương ứng với địa chỉ IP truyền cho phương thức dưới dạng mảng byte
static InetAddress	getByAddress(String host,byte[] addr) Tạo đối tượng InetAddress dựa trên tên và địa chỉ IP
static InetAddress	getByName(String host) Xác định địa chỉ IP của máy trạm từ tên của máy trạm(host)
String	getCanonicalHostName() Lấy tên miền của địa chỉ IP
String	getHostAddress() Trả về địa chỉ IP chứa trong đối tượng INetAddress là chuỗi dạng a.b.c.d
String	getHostName() Trả về tên máy trạm
static InetAddress	getLocalHost() Lấy đối tượng InetAddress của máy cục bộ

URL/URI

- **URI là chuỗi các ký tự có quy tắc để xác định một tài nguyên trên mạng**
- **Cú pháp URI**
  - *scheme:scheme-specific-part*
- **Trong đó, scheme**
  - data
  - file
  - ftp
  - http
  - mailto
  - magnet
  - telnet
  - urn
- **URL là dạng thông dụng nhất của URI (Uniform Resource Identifier), ngoài xác định tài nguyên, URL thường chỉ ra cách để truy cập/làm việc tài nguyên**
- **Cú pháp URL**
  - *protocol://userInfo@host:port/path?query#fragment*
- ***URL tương đối vs URL tuyệt đối***
- **Java cung cấp lớp**
  - java.net.URL
  - Java.net.URLConnection

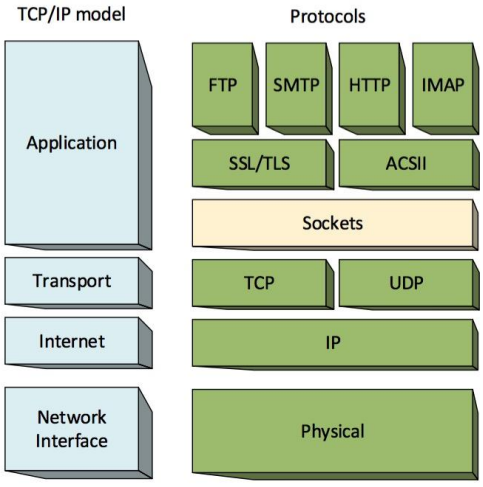
Client/Server (re)

- Mức độ liên lạc mạng đơn giản nhất
- Bao gồm: Server, Client, và môi trường kết nối
  - Client: máy tính chạy chương trình tạo yêu cầu dịch vụ
  - Server: máy tính chạy chương trình trả lời yêu cầu dịch vụ



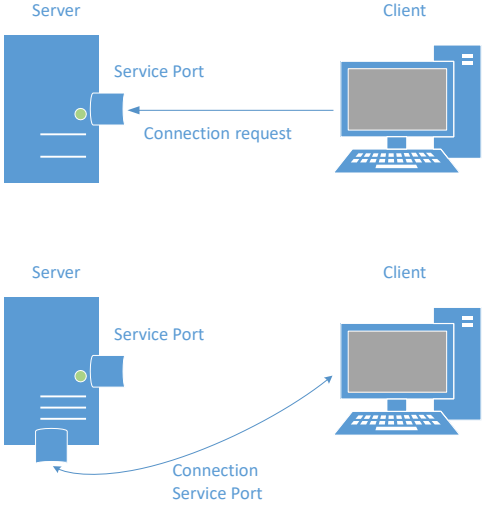
Socket (re)

- **Tầng thấp nhất nằm tiếp giáp tầng Giao vận**
- **Thiết lập một điểm đầu cuối cho đường kết nối 2 chiều**
- **Cung cấp giao tiếp → lập trình liên lac mạng.**
- **Tương tự vào ra file, socket được xử lý tương tự file.**
- **Độc lập ngôn ngữ lập trình**



Socket (re)

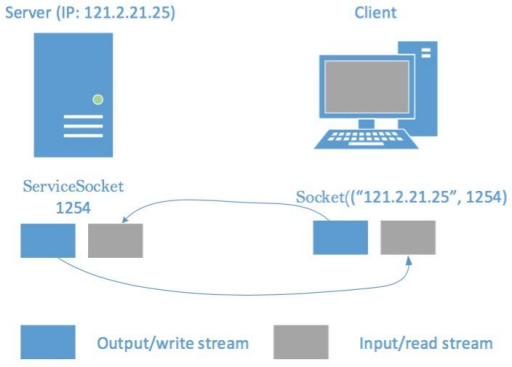
- **Một kết nối**
  - Địa chỉ cục bộ của socket: Địa chỉ IP local và số cổng dịch vụ
  - Địa chỉ từ xa của socket: dành cho các socket thiết lập TCP
  - Giao thức: Giao thức tầng giao vận, ví dụ TCP hoặc UDP.
- **Một socket:**
  - Giao thức
  - Địa chỉ IP
  - Cổng dịch vụ
- **Thiết lập kết nối**
  - Cổng dịch vụ (Server)
  - Cổng dịch vụ kết nối (Client)
  - #



3. Lập trình ứng dụng mạng với TCP - Socket

Các thư viện hỗ trợ trong Java

- **Gói Java.net cung cấp các lớp thư viện cho phát triển ứng dụng mạng**
- **2 lớp quan trọng ứng dụng TCP**
  - ServerSocket (Server)
  - Socket (Client)
- **Sử dụng các luồng vào ra để trao đổi dữ liệu**
  - Luồng kiểu byte
    - InputStream
    - OutputStream
  - Luồng kiểu ký tự
    - BufferedReader
    - PrintWriter



Lớp Socket

- **Lớp Socket dùng để tạo đối tượng socket cho phép truyền thông với giao thức TCP hoặc UDP**
  - Với giao thức UDP thường sử dụng lớp DatagramSocket
- **Một số hàm khởi tạo**
  - *public Socket(String host, int port) throws UnknownHostException, IOException*
  - *public Socket(String host, int port, InetAddress interface, int localPort) throws IOException, UnknownHostException*
  - *public Socket(InetAddress host, int port, InetAddress interface, int localPort) throws IOException*

```
try {
        Socket objSocket = new
        Socket("www.gooogle.com", 80);
        System.out.println(objSocket);
} catch (UnknownHostException e) {
        System.err.println(e.getMessage());
} catch (IOException e) {
        System.err.println(e.getMessage());
}
```

Socket[addr=www.gooogle.com/216.58.199.99,port=80,localport=11328]

Lớp Socket (cont.)

- **Một số phương thức quan trọng**

Phương thức	Mô tả
public InetAddress getInetAddress()	trả về địa chỉ của máy trạm từ xa hiện đang kết nối với socket
public int getPort( )	Trả về số cổng trên máy trạm từ xa mà hiện đang kết nối với socket
public int getLocalPort( )	Trả về số cổng trên máy cục bộ
public InputStream getInputStream( ) throws IOException	Trả về luồng nhập của socket là đối tượng InputStream
public OutputStream getOutputStream( ) throws IOException	Trả về luồng xuất của socket là đối tượng OutputStream
public void close( ) throws IOException	Đóng socket

Lớp Socket (Cont.)

Một số thiết lập cho Socket

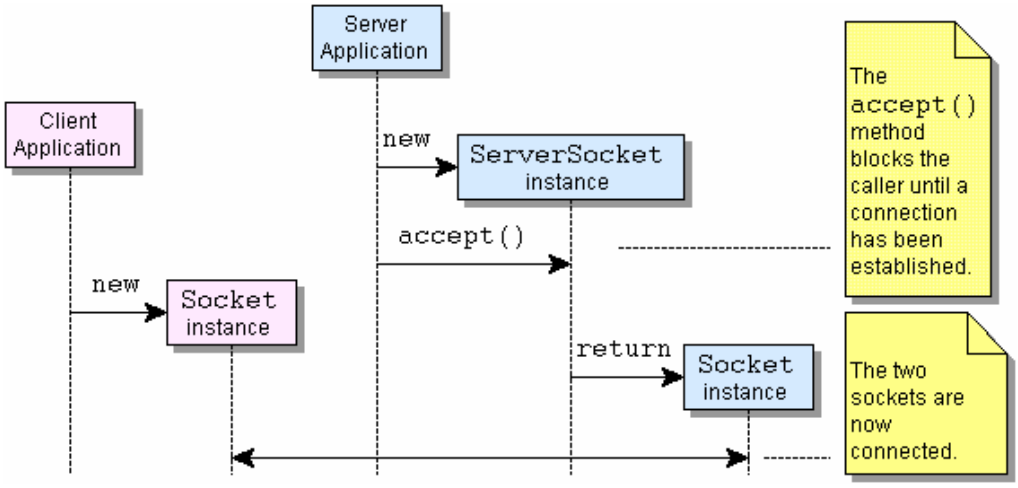
- TCP_NODELAY
- SO_BINDADDR
- SO_TIMEOUT
- SO_LINGER
- SO_SNDBUF (Java 1.2 and later)
- SO_RCVBUF (Java 1.2 and later)
- SO_KEEPALIVE (Java 1.3 and later)
- OOBINLINE (Java 1.4 and later)

Lớp ServerSocket

- **Lớp ServerSocket cho phép tạo đối tượng socket phía server và truyền thông với giao thức TCP**
- **Hàm khởi tạo**
  - *public ServerSocket(int port) throws BindException, IOException*
  - *public ServerSocket(int port, int queueLength) throws IOException, BindException*
  - *public ServerSocket( ) throws IOException*
- **Phương thức**
  - *accept()*
  - *close()*

```
try {
        ServerSocket server = new ServerSocket(1107);
        while(true)
        {
                Socket connection = server.accept();
                OutputStreamWriter out = new
OutputStreamWriter(connection.getOutputStream());
                out.write("connected to server!");
                connection.close();
        }
} catch (IOException e) {
        System.err.println(e.getMessage());
}
```

Kỹ thuật cơ bản lập trình truyền thông với TCP



Kỹ thuật ... TCP (Cont.)

Chương trình phía Server

1. Tạo đối tượng ServerSocket với một số hiệu cổng xác định
2. Đặt đối tượng ServerSocket ở trạng thái nghe tín hiệu đến kết nối bằng phương thức accept(). Nếu có tín hiệu đến kết nối phương thức accept() tạo ra đối tượng Socket mới để phục vụ kết nối đó
3. Khai báo luồng nhập/xuất cho đối tượng Socket mới(tạo ra ở bước trên). Luồng nhập/xuất có thể là luồng kiểu byte hoặc luồng ký tự
4. Thực hiện truyền dữ liệu với client thông qua luồng nhập/xuất
5. Server hoặc client hoặc cả 2 đóng kết nối
6. Server trở về bước 2 và đợi kết nối tiếp theo

Chương trình phía Client

1. Tạo đối tượng Socket và thiết lập kết nối tới server bằng cách chỉ ra các tham số của server.
2. Khai báo luồng nhập/xuất cho Socket. Luồng nhập/xuất có thể là luồng kiểu byte hoặc kiểu char.
3. Thực hiện truyền dữ liệu qua mạng thông qua luồng nhập/xuất
4. Đóng Socket, giải phóng các tài nguyên khác, kết thúc chương trình nếu cần

Demo

- **ServerSocket**
  - Hỗ trợ đa luồng?
  - ...
- **Socket (client)**
- **InetAddress**
- **....**
- **...**
- **..**
- **.**

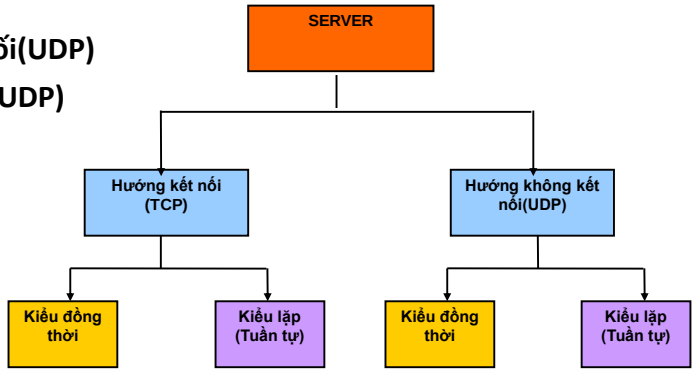
Exercise

- **Socket**
  - Quét cổng sử dụng socket
  - Quét cổng cục bộ (localport) sử dụng lớp ServerSocket
  - Finger client
  - Giao tiếp lấy thời gian server về client
- **URL**
  - Kiểm tra các giao thức hỗ trợ trên máy trạm
  - Kiểm tra khả dụng (URL) tài nguyên trên mạng (thư viện)
  - HTTP GET
  - Truy cập website có bảo mật mật khẩu
- **Cài đặt máy tính cá nhân đơn giản theo mô hình client/server với giao thức TCP/IP**
  - Client gửi yêu cầu về phép toán cần thực hiện add/sub/mul/div ...
  - Server tiếp nhận và xử lý trả về kết quả
  - Sử dụng luồng đối tượng
- **Case study**
  - Cài đặt và mở rộng ứng dụng Login từ xa sử dụng TCP/IP

Giới thiệu một vài kiểu server

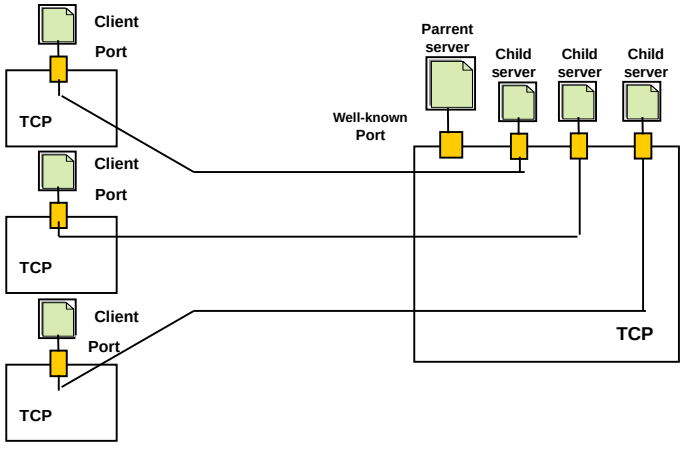
Giới thiệu các kiểu server

- **Đồng thời hướng kết nối(TCP)**
- **Tuần tự hướng kết nối(TCP)**
- **Đồng thời hướng không kết nối(UDP)**
- **Tuần tự hướng không kết nối(UDP)**



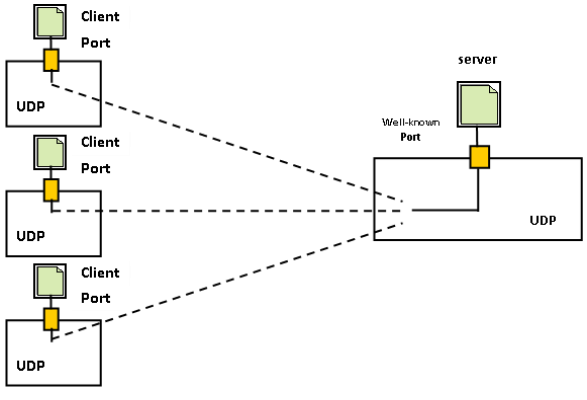
Đồng thời hướng kết nối

- **Sử dụng giao thức truyền thông TCP**
- **Có thể phục vụ nhiều client đồng thời**



Tuần tự hướng không kết nối

- **Sử dụng giao thức UDP**
- **Tại mỗi thời điểm nó chỉ xử lý một yêu cầu**



Nhiều client hướng kết nối

- **Xây dựng chương trình đa tiến trình**
  - Khi có yêu cầu tiến trình cha sẽ sinh tiến trình con
  - Nhược điểm của kỹ thuật lập trình này là không tận dụng được CPU
  - C/C++
- **Xây dựng chương trình đa luồng**
  - Khi có yêu cầu tiến trình sẽ tạo các luồng mới
  - Ưu điểm: tận dụng hiệu quả CPU do không phải chuyển đổi tiến trình
  - VC++, Java , .NET

Exercise

- **Cài đặt các ví dụ ở trên hỗ trợ đa luồng ở mức đơn giản**
- **Cài đặt máy tính cá nhân đơn giản theo mô hình client/server với giao thức TCP/IP**
  - Client gửi yêu cầu về phép toán cần thực hiện add/sub/mul/div ...
  - Server tiếp nhận và xử lý trả về kết quả
  - Sử dụng luồng đối tượng
- **Case study**
  - Cài đặt và mở rộng ứng dụng Login từ xa hỗ trợ đa luồng song song

Tổng kết

- **InetAddress**
- **TCP Socket**
  - ServerSocket
  - Socket
- **Chủ đề**
  - Các bài toán **nghiệp vụ/ game kinh điển** cài đặt theo mô hình TCP/IP
  - Hỗ trợ đa luồng
  - Hỗ trợ chống spam (InetAddress)
- **Mô phỏng gửi các yêu cầu HTTP và xử lý kết quả trả về**
- **Mô phỏng các giao thức tầng ứng dụng (FTP, SMTP, HTTP, IMAP ...)**
- **Nhóm chủ đề truyền thông với máy chủ web thông qua giao thức GET/POST**

Case study: Login sử dụng TCP Socket

Model

View

Controller

Q & A