CHƯƠNG V
LẬP TRÌNH SOCKET VỚI TCP

Chương 5: Lập trình Socket với TCP

1. Mô hình Client/Server
2. Các kiến trúc Client/Server
3. Mô hình truyền tin Socket
4. Socket cho Client
5. Lớp ServerSocket
6. Cài đặt chương trình Client
7. Cài đặt chương trình Server
8. Đa tuyến trong lập trình Java

Mô hình Client/Server

- Thuật ngữ Client/Server xuất hiện vào đầu thập niên 80.
- Dạng phổ biến của mô hình ứng dụng phân tán.
- Mô hình mạng cơ bản nhất hiện nay.
- Đa số các ứng dụng mạng được xây dựng dựa theo mô hình này.
- Một số ứng dụng Client/Server phổ biến: Email, FTP, Web,…

Mô hình Client/Server(tt)

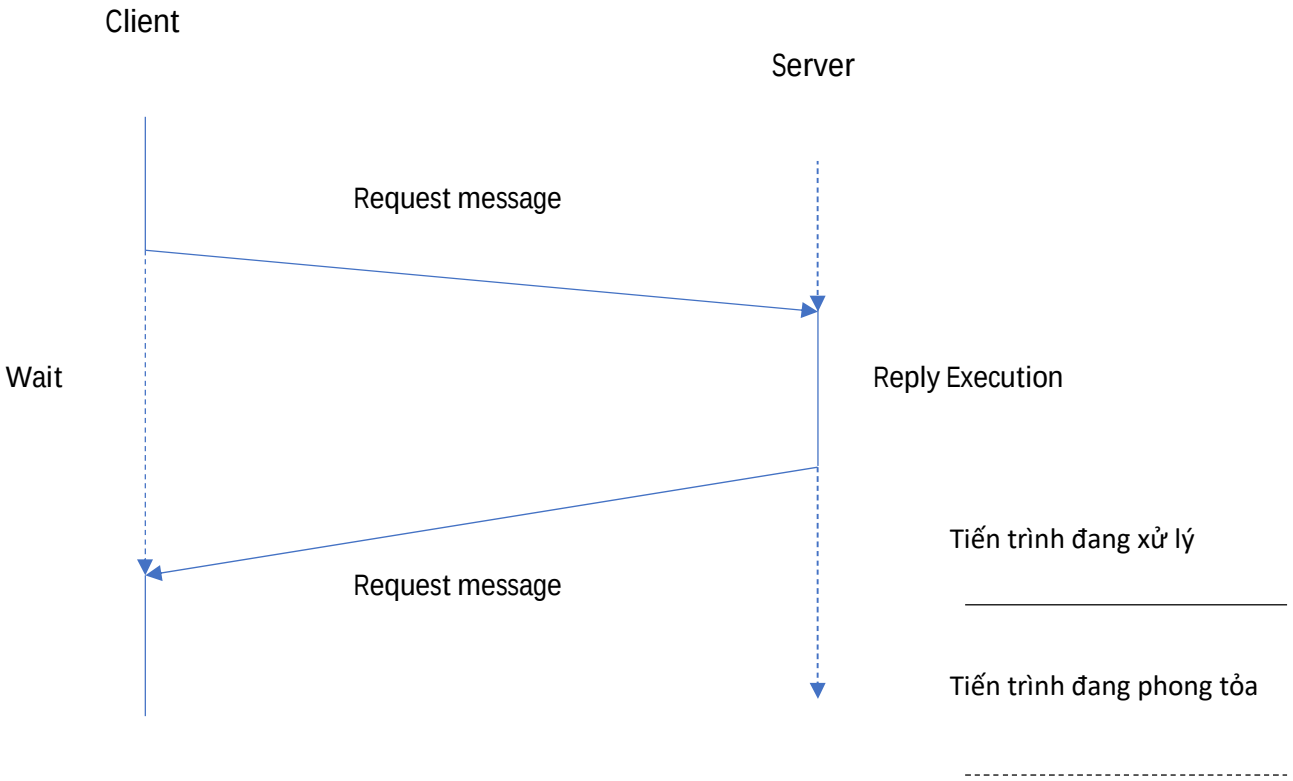
- Mô hình client/server cung cấp một cách tổng quát để chia sẻ tài nguyên trong các hệ thống phân tán.
- Tiến trình client và tiến trình server đều có thể chạy trên cùng một máy tính.
- Một tiến trình server có thể sử dụng dịch vụ của một server khác.
- Mô hình truyền tin client/server hướng tới việc cung cấp.

Mô hình Client/Server(tt)

- Mô hình truyền tin này liên quan đến việc truyền hai thông điệp và được đồng bộ giữa client và server.
- Ở bước 1, tiến trình server phải nhận thức được thông điệp được yêu cầu ngay khi nó đến và hành động phát ra yêu cầu trong client phải được tạm dừng.
- Ở bước 3, tiến trình client ở trạng thái chờ cho đến khi nó nhận được đáp ứng do server gửi về.

Mô hình Client/Server(tt)

- Mô hình client/server, các thao tác cơ bản: gửi (send) và nhận (receive).



Chương 5: Lập trình Socket với TCP

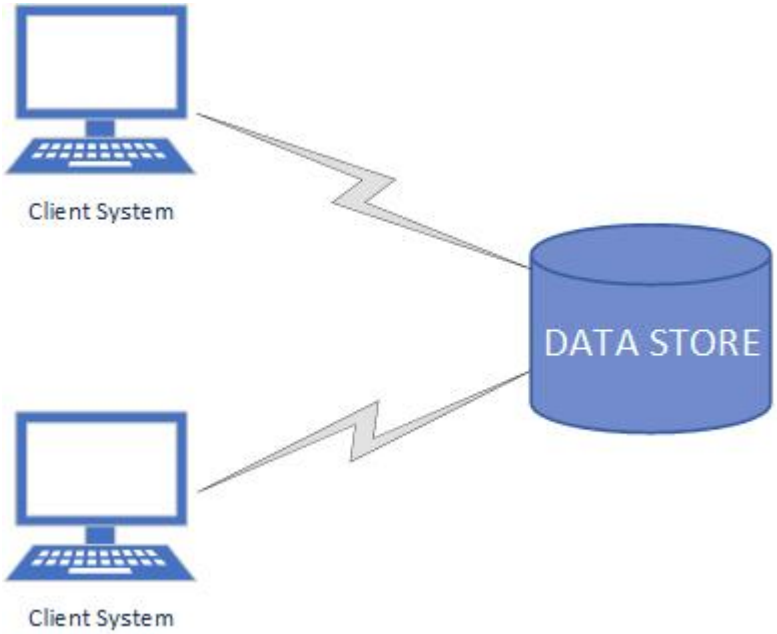
1. Mô hình Client/Server
2. Các kiến trúc Client/Server
3. Mô hình truyền tin Socket
4. Socket cho Client
5. Lớp ServerSocket
6. Cài đặt chương trình Client
7. Cài đặt chương trình Server
8. Đa tuyến trong lập trình Java

Các kiến trúc Client/Server

- Client/Server hai tầng
- Client/Server ba tầng
- Kiến trúc n-tầng

Các kiến trúc Client/Server

- Client/Server hai tầng



Các kiến trúc Client/Server(tt)

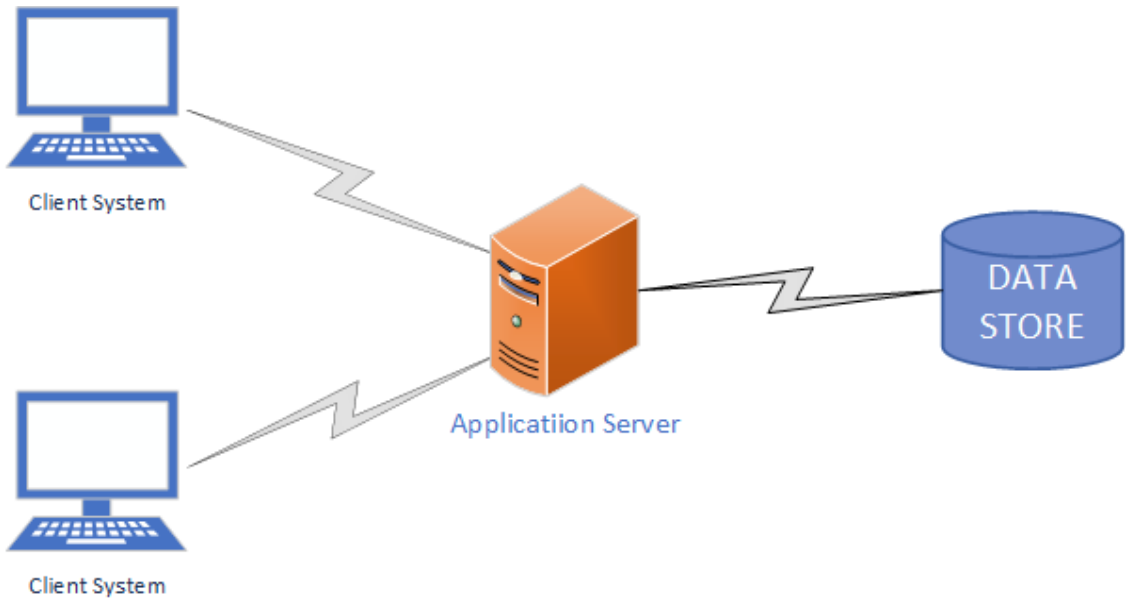
- Client/Server hai tầng:

  - Khối lượng công việc xử lý được dành cho phía client.
  - Khi đó server chỉ đóng vai trò như là chương trình kiểm soát luồng vào ra giữa ứng dụng và dữ liệu.
  - Hiệu năng của ứng dụng bị giảm do tài nguyên hạn chế của PC và khối lượng dữ liệu truyền đi trên mạng cũng tăng theo.

Các kiến trúc Client/Server(tt)

- Client/Server ba tầng

  - Kiến trúc 3 tầng có thêm một tầng mới tách biệt việc xử lý dữ liệu ở vị trí trung tâm..



Các kiến trúc Client/Server(tt)

- Client/Server ba tầng
  - Theo kiến trúc ba tầng, một ứng dụng được chia thành ba tầng tách biệt nhau:
    - Tầng đầu tiên: là tầng trình diễn thường bao gồm các giao diện đồ họa. Tầng trình diễn nhận dữ liệu và định dạng nó để hiển thị.
    - Tầng thứ hai: là tầng trung gian hay tầng tác nghiệp.
    - Tầng thứ ba: chứa dữ liệu cần thiết cho ứng dụng. Tầng thứ ba cơ bản là chương trình thực hiện các lời gọi hàm để tìm kiếm dữ liệu cần thiết.

Các kiến trúc Client/Server(tt)

- Kiến trúc n-tầng, được chia thành các tầng:
  - Tầng giao diện người dùng: quản lý tương tác của người dùng với ứng dụng.
  - Tầng logic trình diễn: Xác định cách thức hiển thị giao diện người dùng và các yêu cầu của người dùng được quản lý như thế nào.
  - Tầng logic tác nghiệp: Mô hình hóa các quy tắc tác nghiệp
  - Tầng các dịch vụ hạ tầng: Cung cấp các chức năng bổ trợ cần thiết cho ứng dụng.

Chương 5: Lập trình Socket với TCP

1. Mô hình Client/Server
2. Các kiến trúc Client/Server
3. Mô hình truyền tin Socket
4. Socket cho Client
5. Lớp ServerSocket
6. Cài đặt chương trình Client
7. Cài đặt chương trình Server
8. Đa tuyến trong lập trình Java

Mô hình truyền tin Socket

- Một socket có thể thực hiện 7 thao tác cơ bản:
  1. Kết nối với một máy ở xa (để gửi và nhận dữ liệu)
  2. Gửi dữ liệu
  3. Nhận dữ liệu
  4. Ngắt liên kết
  5. Gán cổng
  6. Nghe dữ liệu đến
  7. Chấp nhận liên kết từ các máy ở xa trên cổng được gán

Mô hình truyền tin Socket(tt)

- Lớp Socket của Java được sử dụng cả client và server.
- Có các phương thức tương ứng với bốn thao tác đầu tiên.
- Ba thao tác cuối chỉ cần cho server để chờ các client liên kết với chúng.
- Các thao tác này được cài đặt bởi lớp ServerSocket.

Mô hình truyền tin Socket(tt)

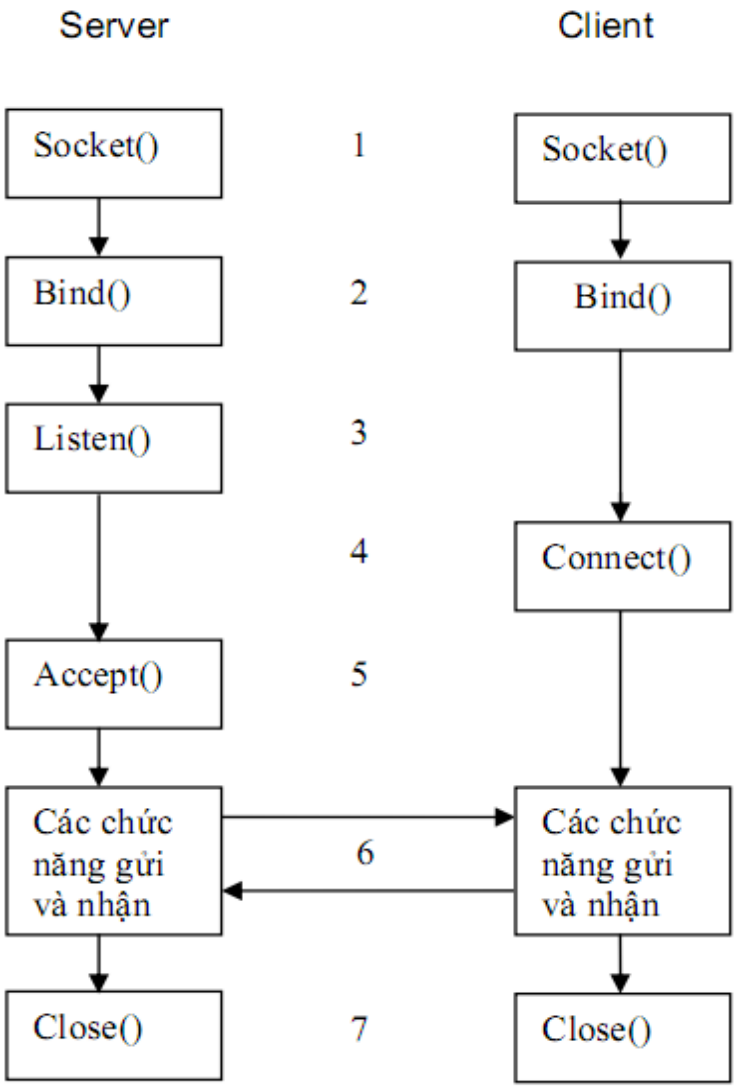
- Khái niệm cổng (port)
  - Thông tin giữa client và server phải sử dụng cổng tương ứng nhau thì mới làm việc được với nhau.
  - Xây dựng ứng dụng trên server, phải chọn cổng có giá trị khác với giá trị cổng thông dụng đã sử dụng

Dịch vụ	Cổng (port)
FTP	21
HTTP	80
Telnet	23
SMTP	25
MySQL	330
DNS	53

Mô hình truyền tin Socket(tt)

- Các lớp thông dụng trong Java.net
  - Lớp InetAddress: quản lý địa chỉ theo tên và theo số. Các phương thức static hay sử dụng.
  - Lớp Socket: tạo kết nối từ phía client với server
  - Lớp ServerSocket: tạo kết nối từ server với client
  - Lớp DatagramSocket: chuyển một gói dữ liệu theo giao thức UDP.
  - Lớp URL(Uniform Resource Locator): địa chỉ tài nguyên trên mạng.

Mô hình truyền tin Socket(tt)



Mô hình truyền tin Socket(tt)

- Các socket cho client thường được sử dụng theo mô hình sau:
  - Sử dụng hàm Socket() để tạo một socket mới.
  - Socket cố gắng liên kết với một host ở xa.
  - Mỗi khi liên kết được thiết lập, các host ở xa nhận các luồng vào và ra từ socket, và sử dụng các luồng này để gửi dữ liệu cho nhau.
  - Khi việc truyền dữ liệu hoàn thành, một hoặc cả hai phía ngắt liên kết.
    Ví dụ : HTTP đòi hỏi mỗi liên kết phải bị đóng sau khi yêu cầu được phục vụ.

Chương 5: Lập trình Socket với TCP

1. Mô hình Client/Server
2. Các kiến trúc Client/Server
3. Mô hình truyền tin Socket
4. Socket cho Client
5. Lớp ServerSocket
6. Cài đặt chương trình Client
7. Cài đặt chương trình Server
8. Đa tuyến trong lập trình Java

Socket cho Client

- Các constructor
- Nhận các thông tin về Socket
- Đóng Socket
- Thiết lập các tùy chọn cho Socket
- Các phương thức của lớp Object
- Các ngoại lệ Socket
- Các lớp SocketAddress

Socket cho Client

- Các Constructor

public Socket(String host, int port) throws UnknownHostException, IOException

Tạo một socket TCP với host và cổng xác định và thực hiện liên kết với host.

Ví dụ:

```
try{
        Socket sk = new Socket( “www.ud.edu.vn”,80);
    }   catch(UnknownHostException e) {System.err.println(e);}
catch(IOException e) {System.err.println(e);}
```

Socket cho Client

- Các Constructor(tt)

public Socket(InetAddress host, int port) throws IOException

Tạo một socket TCP với thông tin là địa chỉ của một host được xác định bởi một đối tượng InetAddres và số hiệu cổng port, sau đó nó thực hiện kết nối tới host.

.

Socket cho Client

- Nhận các thông tin về Socket
  - *public InetAddress getInetAddress():* Trả về địa chỉ của máy tính khác mà socket kết nối tới
  - *public int getPort():* Cho biết số hiệu cổng mà Socket kết nối tới host
  - *public int getLocalPort*(): cho biết số hiệu cổng ở host local

Socket cho Client

- Nhận các thông tin về Socket(tt)

```
public InputStream getInputStream() throws
IOException
```

  - Phương thức getinputStream(): Trả về một luồng nhập để đọc dữ liệu từ một socket vào chương trình.
  - Để nâng cao hiệu năng, có thể đệm dữ liệu bằng cách gắn kết nó với luồng lọc BufferedInputStream hoặc BufferedReader

Socket cho Client

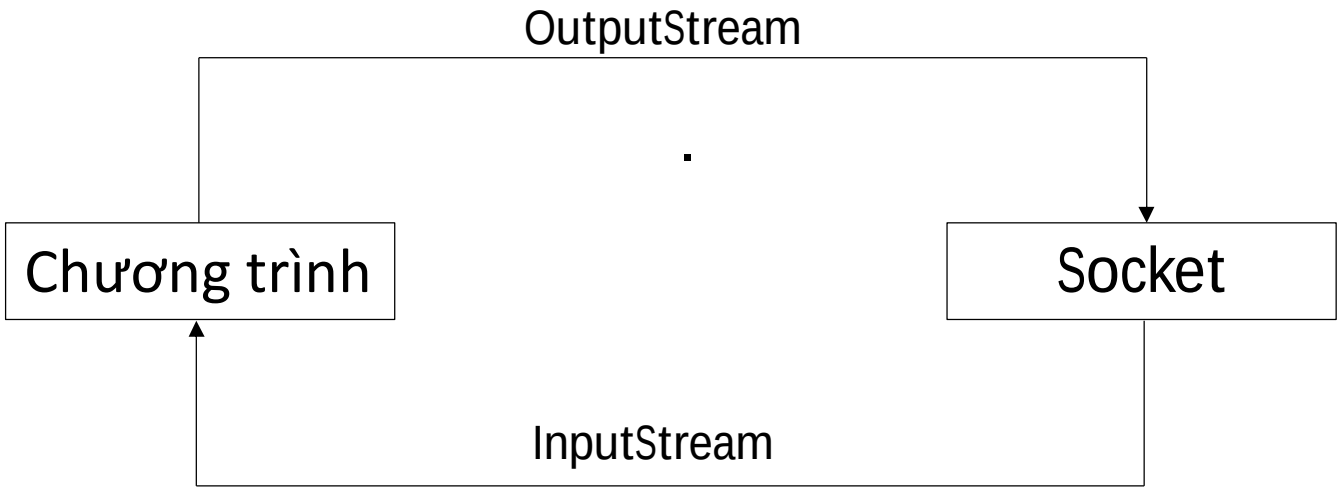
- Nhận các thông tin về Socket(tt)

```
public OutputStream getOutputStream() throws
IOException
```

  - Phương thức getOutputStream(): Trả về một luồng xuất để ghi dữ liệu từ ứng dụng ra đầu cuối của một socket.
  - Gắn kết luồng này với một luồng tiện lợi như lớp DataOutputStream hoặc OutputStreamWriter trước khi sử dụng nó.

Socket cho Client

- Nhận các thông tin về Socket(tt)
  - Hai phương thức getInputStream() và getOutputStream() là các phương thức lấy về các luồng dữ liệu nhập và xuất



Socket cho Client

- Nhận các thông tin về Socket(tt)

```
public OutputStream getOutputStream() throws
IOException
```

  - Phương thức getOutputStream(): Trả về một luồng xuất để ghi dữ liệu từ ứng dụng ra đầu cuối của một socket.
  - Gắn kết luồng này với một luồng tiện lợi như lớp DataOutputStream hoặc OutputStreamWriter trước khi sử dụng nó.