

Môn học

Lập Trình Mạng

Giảng Viên: Phạm Minh Tuấn



Giới thiệu

- ❖ Phạm Minh Tuấn
- ❖ E-mail: pmtuan@dut.udn.vn
- ❖ Tel: 0913230910
- ❖ Khoa Công nghệ thông tin –Trường ĐHBK – ĐHĐN

Thỏa thuận

❖ Đối với giáo viên:

- Dạy đủ tất cả nội dung của môn học.
- Trả lời các câu hỏi của học sinh trong và ngoài giờ giảng liên quan tới môn học.
- Ra bài tập cho học sinh
- Lên lớp đúng giờ

Thỏa thuận (tiếp)

❖ Đối với học sinh:

- Tham gia trên 80% số tiết học.
- Tham gia đóng góp tiết học như phát biểu, trả lời hay đặt câu hỏi cho giáo viên (không nói chuyên riêng)
- Làm bài tập đầy đủ.
- Lên lớp đúng giờ (không được đi trễ hơn giáo viên quá 5 phút)

Mục tiêu môn học

- ❖ Hiểu được các giao thức mạng.
- ❖ Lập trình giao thức mạng
- ❖ Lập trình đa luồng
- ❖ Lập trình với cơ sở dữ liệu

Nội dung môn học

❖ Khái niệm chung

- Kiến Trúc Mạng
- Lập trình mạng
 - Đối tượng lập trình mạng
 - Phạm vi
- Các loại mạng
- Hệ điều hành
 - Unix, Linux, Windows

Nội dung môn học (tt)

❖ Các mô hình mạng

- Nguyên tắc truyền thông
- Mô hình truyền thông
 - Phương pháp phân tầng
 - Nguyên tắc
- Mô hình 7 tầng OSI
- Mô hình 4 tầng TCP/IP
- Mô hình thu gọn 3 tầng

Nội dung môn học (tt)

- ❖ Mô hình ứng dụng client/server
 - Thành phần và chức năng
 - Cách hoạt động
 - Đặc trưng mô hình ứng dụng client/server
 - Ưu nhược điểm
 - Client/server 2 lớp
 - Client/server 3 lớp
 - Giao thức cho ứng dụng

Nội dung môn học (tt)

- ❖ Lập trình với TCP
- ❖ Lập trình với UDP
- ❖ Lập trình đa tuyến(luồng)
- ❖ Lập trình với CSDL



Bài 1:

Khái niệm chung

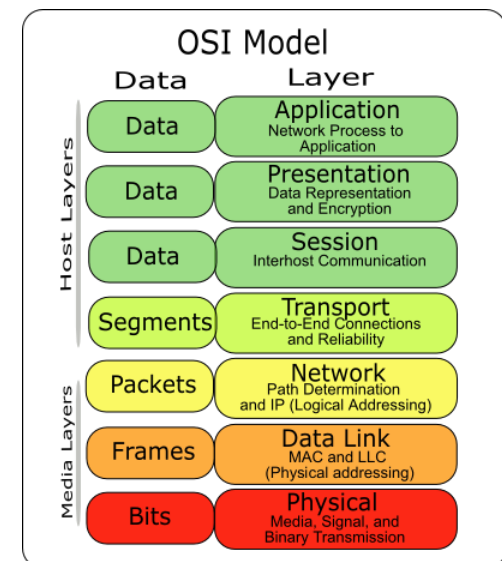
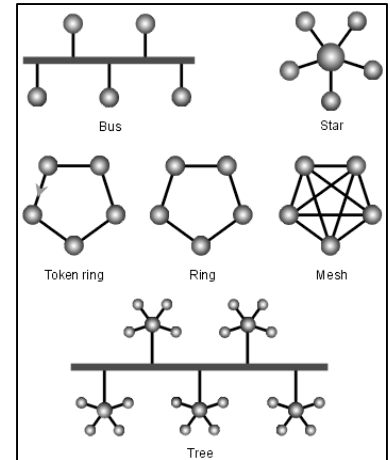
Kiến Trúc Mạng

❖ Các topology

- Là cấu hình kết nối vật lý

❖ Kiến trúc phân tầng

- Hệ thống các tầng giao thức mạng gồm
 - Các thực thể phần cứng
 - Phần mềm
 - Đảm bảo hoạt động của hệ thống
 - Vd: OSI hay TCP/IP










Lập trình mạng

❖ Đối tượng lập trình mạng

- Các thực thể phần mềm thực thi giao thức trong hệ thống mạng
 - Được xây dựng trên nền tảng hệ thống máy tính
 - Phần cứng và hệ điều hành, kiến trúc phân tầng mạng

From Computer Desktop Encyclopedia
© 2003 The Computer Language Co., Inc.



OSI MODEL			TCP / IP
7		Application Layer Type of communication: E-mail, file transfer, client/server.	FTP, Telnet, HTTP, SNMP,
6		Presentation Layer Encryption, data conversion: ASCII to EBCDIC, BCD to binary, etc.	DNS, OSPF, RIP,
5		Session Layer Starts, stops session. Maintains order.	Ping, Traceroute
4		Transport Layer Ensures delivery of entire file or message.	TCP (delivery ensured) UDP (delivery NOT ensured)
3		Network Layer Routes data to different LANs and WANs based on network address.	IP (ICMP, IGMP, ARP, RARP)
2		Data Link (MAC) Layer Transmits packets from node to node based on station address.	
1		Physical Layer Electrical signals and cabling.	

Lập trình mạng (tt)

❖ Vậy, LT mạng?

- Tạo ra các thực thể phần mềm hoạt động trên một tầng
 - Sử dụng các thực thể ở tầng kề dưới
 - Cung cấp dịch vụ cho các thực thể tầng kề trên
- Chủ yếu, tạo các thực thể phần mềm ở tầng ứng dụng
 - Cung cấp dịch vụ cho người dùng



Phạm vi môn học

- ❖ Tập trung vào các kỹ thuật lập trình sử dụng dịch vụ tại tầng transport để xây dựng các ứng dụng mạng
- ❖ Lập trình đa luồng
- ❖ Lập trình truyền tải thông tin với CSDL



Hạ tầng truyền thông

- ❖ Một ứng dụng hay một dịch vụ mạng cần có hạ tầng mạng bên dưới khi hoạt động
- ❖ Tùy theo yếu tố kỹ thuật hay yêu cầu đối với từng ứng dụng mà ta cần phải lựa chọn loại mạng ứng dụng và dịch vụ

Các loại mạng

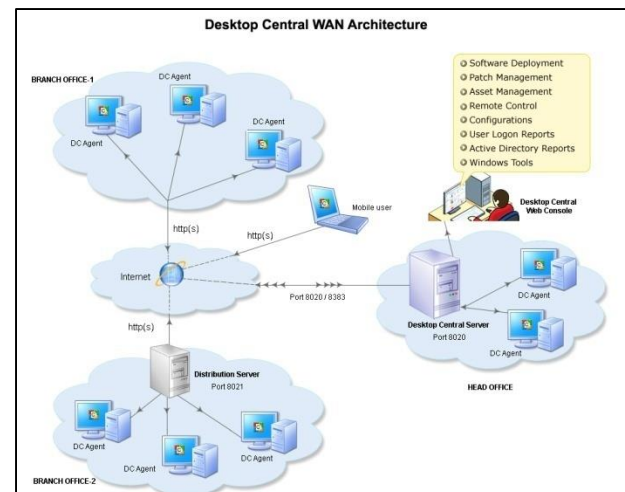
❖ Mạng cục bộ LAN

- Băng thông (bandwidth) rộng
- Tỷ lệ lỗi thấp
- Thích hợp với các ứng dụng
 - Email
 - Truyền file
 - ứng dụng CSDL có độ truy xuất cao
 - ứng dụng truyền đa phương tiện

Các loại mạng(tt)

❖ Mạng diện rộng WAN

- Có nhiều kỹ thuật để lắp đặt mạng WAN
 - VD: Lease-line, Frame-relay, ISDN, ATM...
 - Mỗi kỹ thuật có băng thông khác nhau
- WAN thường kết nối các mạng LAN ở xa nhau
- Đường WAN sử dụng với mục đích truyền số liệu, kết nối từ xa, VoIP,...





Câu hỏi???

- ❖ Băng thông là gì?
- ❖ Tại sao phải chú ý đến băng thông?

Bài tập

❖ BT1:

- Để tải một đoạn phim có dung lượng 5Mbyte bằng đường dây có băng thông 1Mbps thì ta phải mất bao nhiêu thời gian?

❖ BT2:

- Giả sử bộ phim 5Mbyte có thể xem được 5 phút. Vậy tối đa có bao nhiêu người có thể xem phim cùng một lúc mà không bị giật?

❖ BT3:

- Một ngày trung bình có bao nhiêu lượt xem phim thì không bị giật?

Các loại mạng(tt)

❖ Mạng Internet

- Là môi trường kém ổn định và không an toàn so với LAN và WAN
- Các dịch vụ mạng trên internet:
 - Email, Web, thương mại điện tử, Game online...
- **Vấn đề về an ninh mạng**





Hệ điều hành

- ❖ Một ứng dụng mạng hoạt động trên một hoặc nhiều hệ thống máy tính.
 - Để hoạt động được thì ứng dụng cần môi trường hoạt động.
 - Môi trường quan trọng nhất đó là:
Hệ Điều Hành

Hệ điều hành (tt)

❖ Unix

- Bắt đầu được xây dựng tại phòng thí nghiệm Bell Lab.
- Là hệ điều hành đa nhiệm, đa người sử dụng và phục vụ truyền thông rất tốt
- Hạn chế:
 - Có quá nhiều phiên bản
 - Phức tạp trong quản trị
 - Đòi hỏi cấu hình mạnh

Hệ điều hành (tt)

❖ LINUX

- Là một phiên bản thu nhỏ của UNIX
- Có nhiều phiên bản
 - Redhat Linux, Mandrake Linux...
- Dùng cho máy trạm, máy chủ và siêu máy tính
- Là hệ điều hành đa nhiệm, đa người dùng
- Tính ổn định cao
- Hỗ trợ truyền thông cao
- **Miễn phí**

Hệ điều hành (tt)

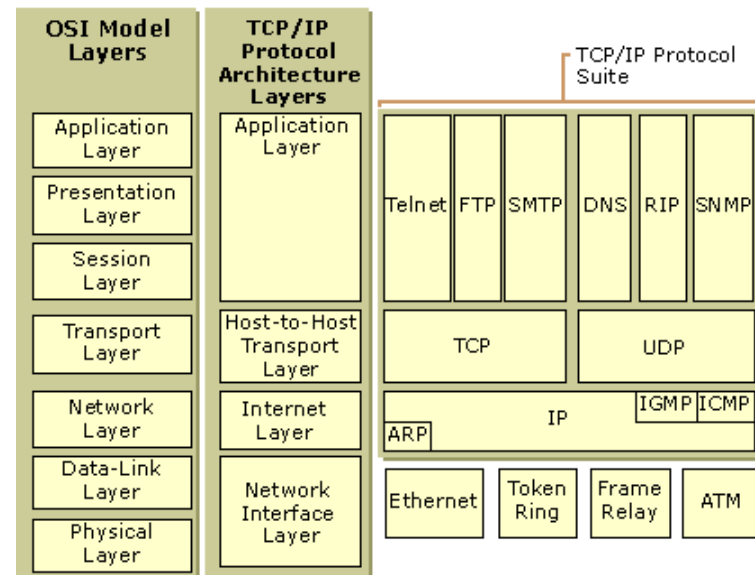
❖ Windows

- Là hệ điều hành đa nhiệm, đa người sử dụng
- Tính năng hỗ trợ mạng
- Dễ sử dụng
- Có các phiên bản cho máy trạm, máy chủ
- Hỗ trợ nhiều loại dịch vụ
- Hạn chế:
 - Bảo mật kém
 - Ít ổn định
 - Không miễn phí

Tập giao thức

❖ Trong phạm vi môn học này, trọng tâm sử dụng bộ giao thức TCP/IP với lý do:

- Là bộ giao thức phổ biến, có thể dùng:
 - Mọi loại mạng
 - LAN, WAN, Internet
 - Mọi hệ điều hành
 - Các thiết bị phần cứng



Ngôn ngữ lập trình và công cụ

❖ Ngôn ngữ lập trình

- C/C++
- Java
- .NET
- BASIC

❖ Công cụ phát triển

- DevC
- Eclipse
- MVS



Một số chú ý về kỹ thuật LT mạng

❖ LT thủ tục

- Chia chương trình thành các chương trình con
- Hàm thủ tục

❖ LT hướng đối tượng

- Thiết kế chương trình theo hướng đối tượng,
 - Tạo thư viện phục vụ LT mạng thành các gói, lớp đối tượng
 - Sử dụng một số thư viện đối tượng có sẵn

❖ LT đa tuyến

- Tận dụng tối đa khả năng của bộ vi xử lý
 - Thực hiện đồng thời nhiều tác vụ

❖ LT với CSDL



Bài 2:

Các mô hình mạng



Nguyên tắc truyền thông

- ❖ Một máy tính trở thành một môi trường truyền dữ liệu cần có các yếu tố sau:
 - Các máy tính phải được kết nối nhau theo một cấu trúc topology nào đó.
 - Việc chuyển dữ liệu thực hiện dưới những quy định thống nhất gọi là giao thức mạng (protocol).
 - Phân chia hoạt động truyền thông của hệ thống thành nhiều lớp theo các nguyên tắc nhất định.



Mô hình truyền thông trong kiến trúc mạng

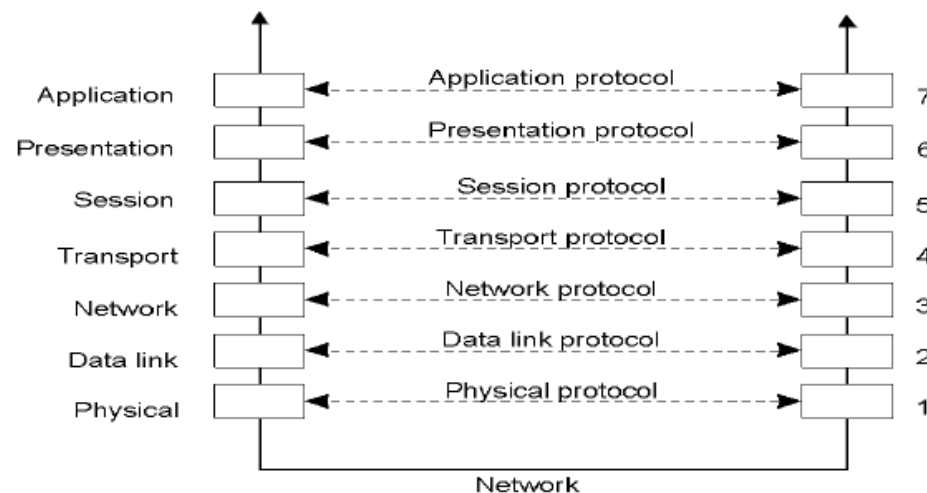
❖ Phương pháp phân tầng mạng

- Tách và xét mô hình mạng thành các môđun độc lập:
 - Giảm độ phức tạp cho việc thiết kế và cài đặt.

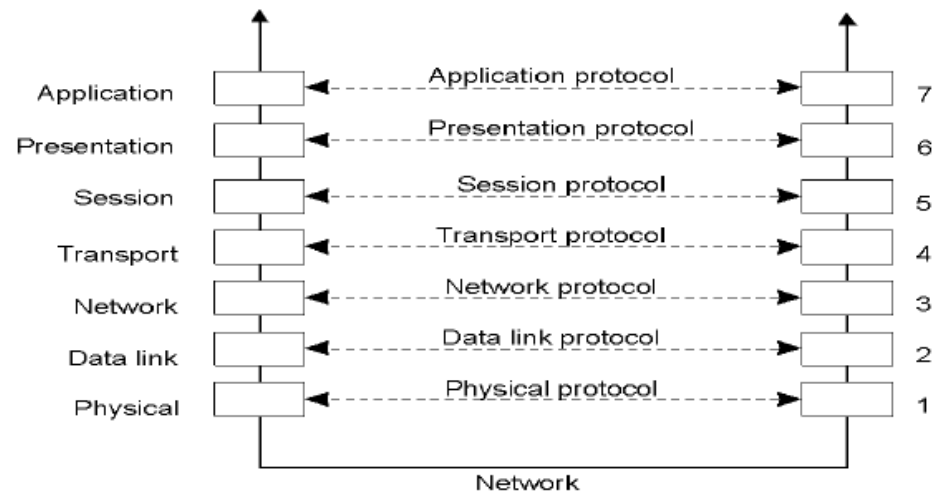
❖ Nguyên tắc

- Mỗi hệ thống xây dựng như một cấu trúc nhiều tầng và có cấu trúc giống nhau:
 - Số lượng tầng và chức năng của các tầng.

- Dữ liệu chỉ được truyền giữa 2 tầng kề nhau
- Bên gửi: Dữ liệu từ tầng cao nhất lần lượt đến tầng thấp nhất.
- Bên nhận: Dữ liệu từ tầng thấp nhất ngược lên đến tầng cao nhất



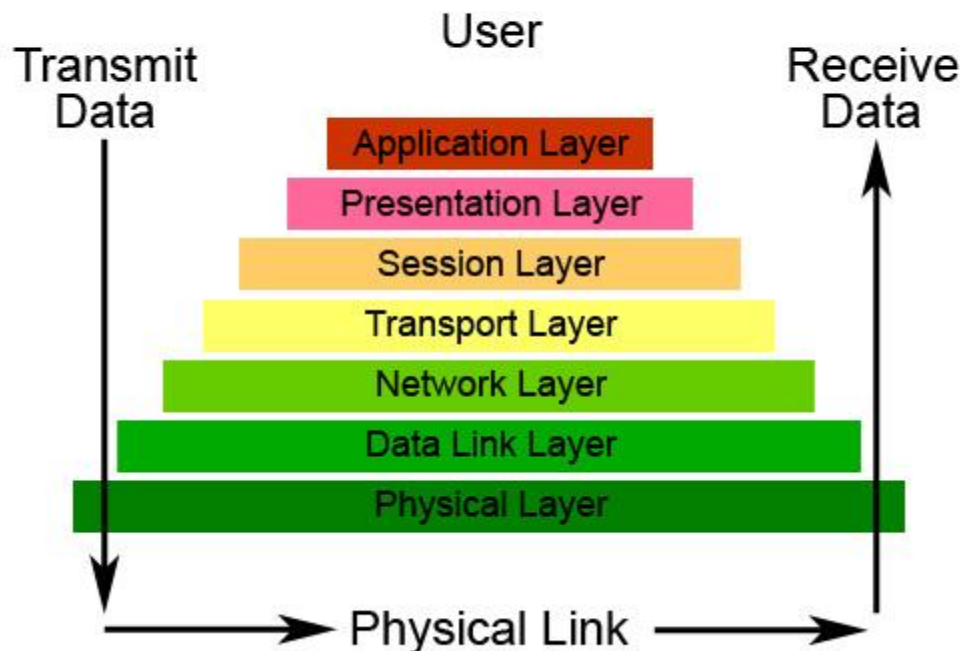
- Chỉ có 2 tầng thấp nhất mới có liên kết vật lý với nhau, còn các tầng trên cùng thứ tự chỉ có liên kết logic với nhau.



Mô hình OSI (Open Systems Interconnection)

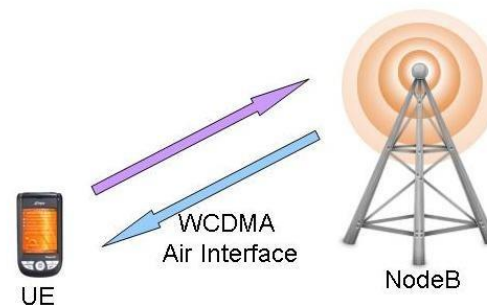
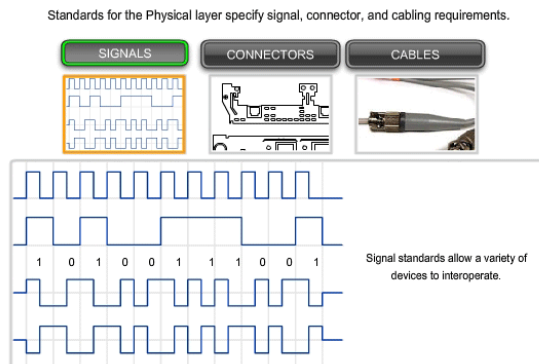
❖ Là mô hình gồm 7 tầng

The Seven Layers of OSI



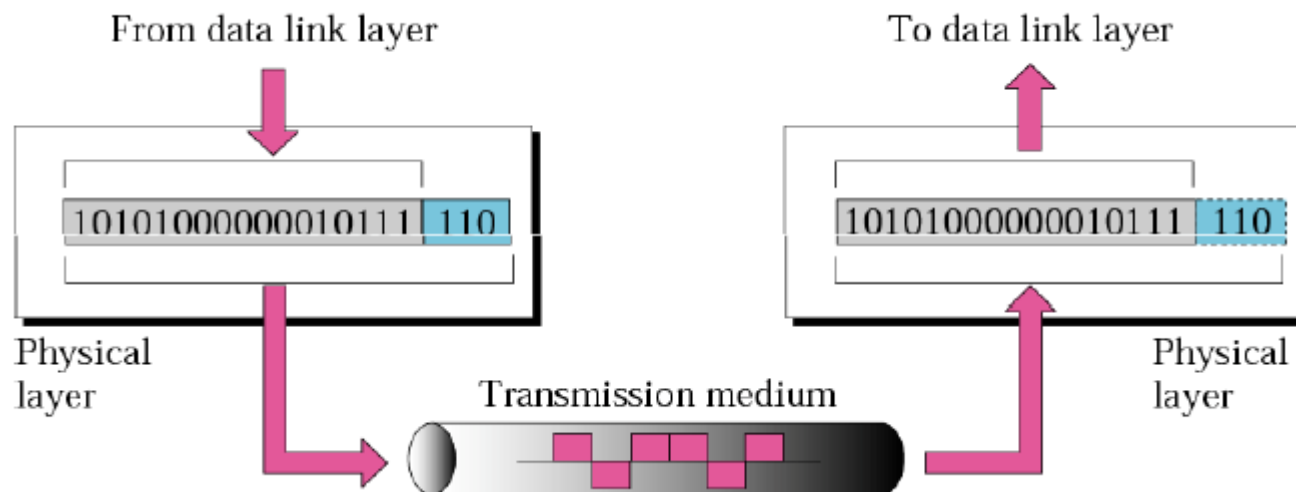
Tầng vật lý (physical layer)

- ❖ Kiểm soát mức thấp nhất việc truyền thông giữa 2 nút mạng.
- ❖ Card mạng và cáp mạng. Truyền dãy các bit giữa 2 nút.
- ❖ Các nhà lập trình không làm việc ở tầng này.
 - Trách nhiệm của các nhà phát triển driver phần cứng và các kỹ sư điện, điện tử.
- ❖ Lỗi có thể xảy ra trong quá trình truyền dữ liệu ở tầng này do điện áp, hay nhiễu đường truyền trên mạng.



Tầng liên kết dữ liệu(Data link layer)

- ❖ Chịu trách nhiệm truyền dữ liệu tin cậy hơn
- ❖ Nhóm dữ liệu thành các frames.
 - Frames tương tự như các packet dữ liệu, nhưng chúng là các khối dữ liệu, được đặc tả theo kiến trúc phần cứng.
- ❖ Frames có trường kiểm tra lỗi truyền
- ❖ Đảm bảo dữ liệu bị méo không được truyền lên tầng trên.



Ví dụ về Kiểm tra lỗi (error detection and correction)

❖ Bít chẵn lẻ (*parity bit*)

7 bit dữ liệu	byte có bit chẵn lẻ	
	Quy luật số chẵn	Quy luật số lẻ
0000000	00000000	00000001
1010001	10100011	10100010
1101001	11010010	11010011
1111111	11111111	11111110

❖ Khối chẵn lẻ (*parity block*)

Bài tập

- ❖ Bt1: Hãy tính khả năng(xác suất) phát hiện lỗi khi sử dụng bit chẵn lẻ với dữ liệu 7 bit?



Lời giải

❖ Giả sử dữ liệu không bị lỗi

- 1 trường hợp → Kiểm tra không thành công

❖ Giả sử dữ liệu bị 1 bit lỗi

- 8 trường hợp → Kiểm tra thành công

❖ Giả sử dữ liệu 2 bit lỗi

- 28 trường hợp → Kiểm tra lỗi không thành công

❖

❖ Giả sử dữ liệu có n bit lỗi ?

Bài tập

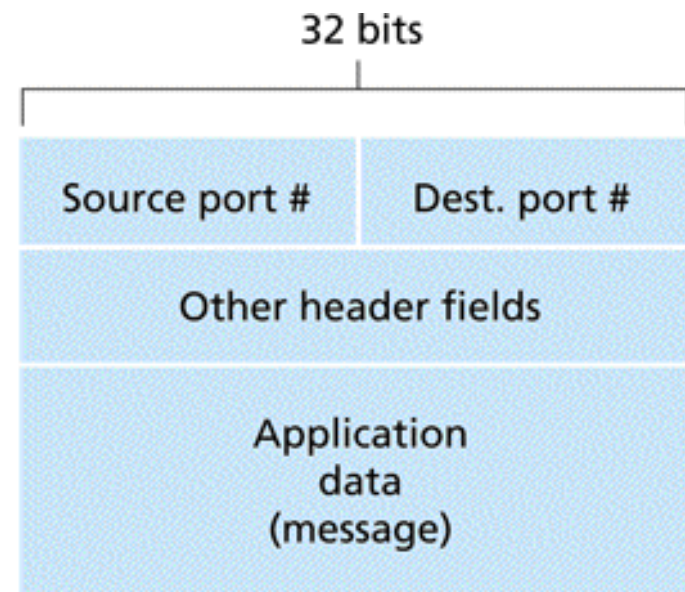
- ❖ Bt1: Hãy tính khả năng(xác suất) phát hiện lỗi khi sử dụng khối chẵn lẻ với khối dữ liệu 3×7 bit?

Tầng mạng (Network layer)

- ❖ Các frames từ tầng Datalink lên hoặc các segments từ tầng transport xuống.
- ❖ Dữ liệu ở dạng packets
- ❖ Phần header chứa các thông tin quan trọng:
 - Địa chỉ mạng (network address)
 - Định tuyến mạng (routing)
- ❖ Packets được gửi qua lại giữa các mạng
- ❖ Các packets thường được định tuyến khác nhau
 - Việc định tuyến do các routers thực hiện
- ❖ **Các lập trình viên rất ít khi làm dịch vụ cho tầng này.**

Tầng vận chuyển (transport layer)

- ❖ Liên quan đến việc dữ liệu được truyền như thế nào
- ❖ Dữ liệu dạng segments
- ❖ Chịu trách nhiệm
 - Xử lý việc kết nối
 - Phát hiện lỗi tự động
 - Điều khiển luồng dữ liệu





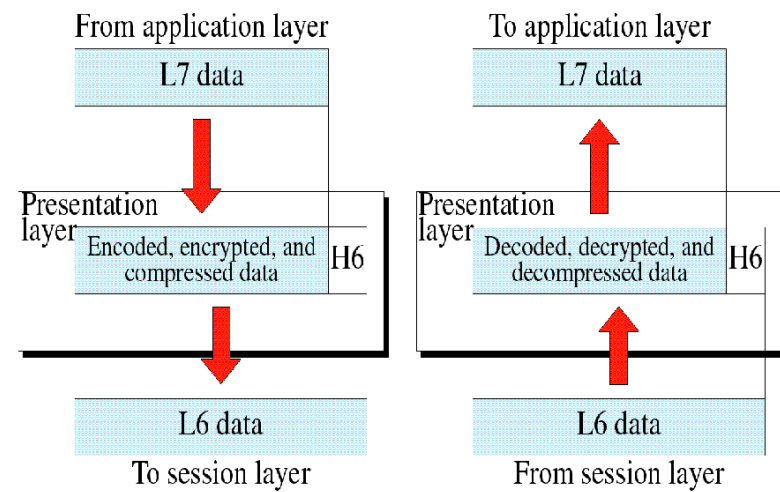
Tầng phiên (session layer)

- ❖ Làm cho dễ dàng việc trao đổi dữ liệu
- ❖ Quản lý phiên truyền thông giữa các ứng dụng
 - Thiết lập một phiên
 - Đồng bộ một phiên
 - Thiết lập lại phiên nếu một phiên bị kết thúc đột ngột
- ❖ Không phải tất cả các ứng dụng đều sử dụng giao thức có kết nối
 - Do vậy việc quản lý phiên không phải lúc nào cũng được yêu cầu.

Tầng trình bày (Presentation layer)

❖ Nhiệm vụ đảm bảo hiển thị và chuyển đổi dữ liệu

- Các máy tính khác nhau có thể sử dụng các kiểu biểu diễn dữ liệu khác nhau
- Nén giải nén dữ liệu
- Mã hóa giải mã dữ liệu



Tầng ứng dụng (Application layer)

- ❖ Tầng cao nhất trong mô hình mạng
- ❖ Hầu hết các ứng dụng mạng được viết ở tầng này



Các giao thức

❖ Application

- HTTP, FTP, SMTP, NSF, Telnet, SSH, ECHO, ...

❖ Presentation

- SMB, NCP, ...

❖ Session

- SSH, NetBIOS, RPC, ...

Các giao thức (tt)

❖ Transport

- TCP, UDP, ...

❖ Network

- IP, ICMP, IPX

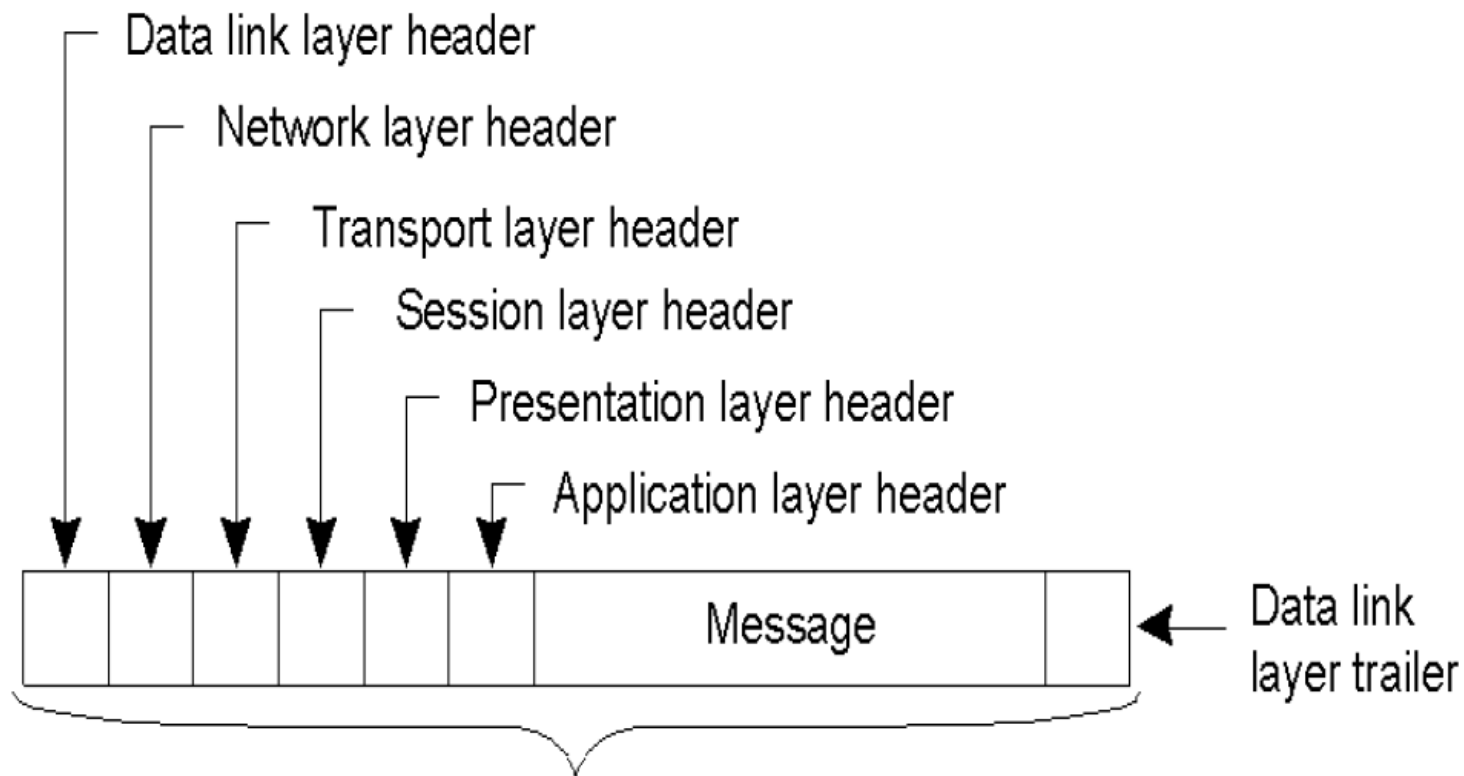
❖ Data link

- Ethernet, Token Ring, ISDN, ...

❖ Physical

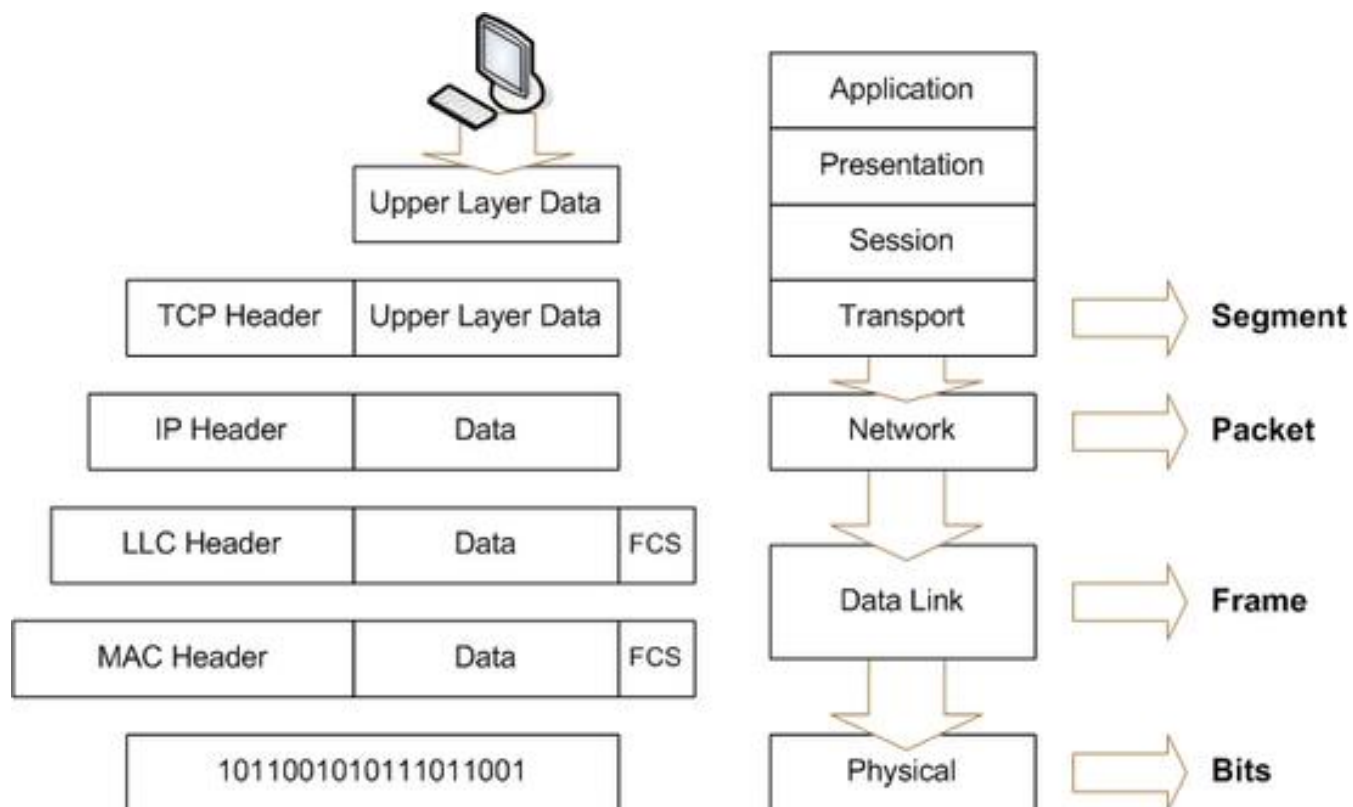
- 100BASE-T, 1000BASE-T, 802.11

Metadata trong một thông điệp

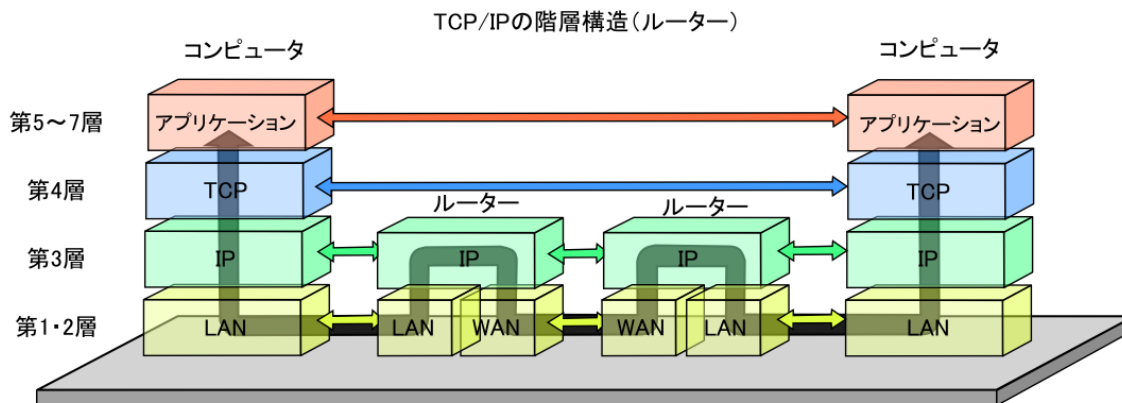
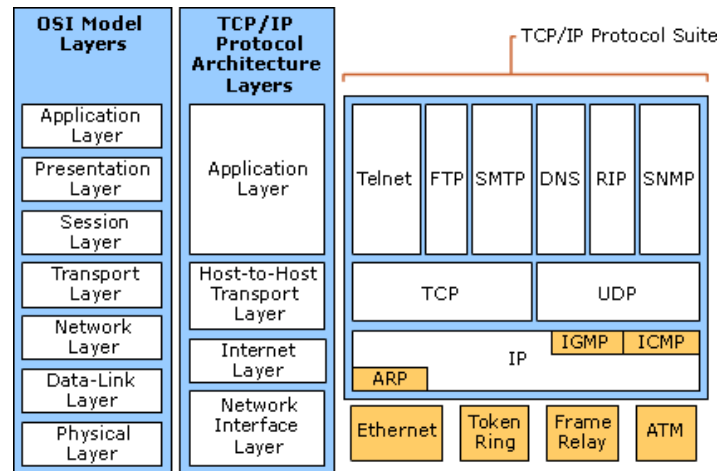


Thông điệp chuyển thành dạng bits để truyền đi trên mạng

Đóng gói dữ liệu



Mô hình 4 tầng TCP/IP



Mô hình phân tầng thu gọn 3 tầng

- ❖ Một số mô hình được phát triển
 - Mô hình 7 tầng OSI
 - Mô hình 4 tầng TCP/IP
- ❖ Xét trên phương diện lập trình
 - Mô hình truyền thông đơn giản 3 tầng.
 - Tầng ứng dụng
 - Tầng giao vận
 - Tầng mạng

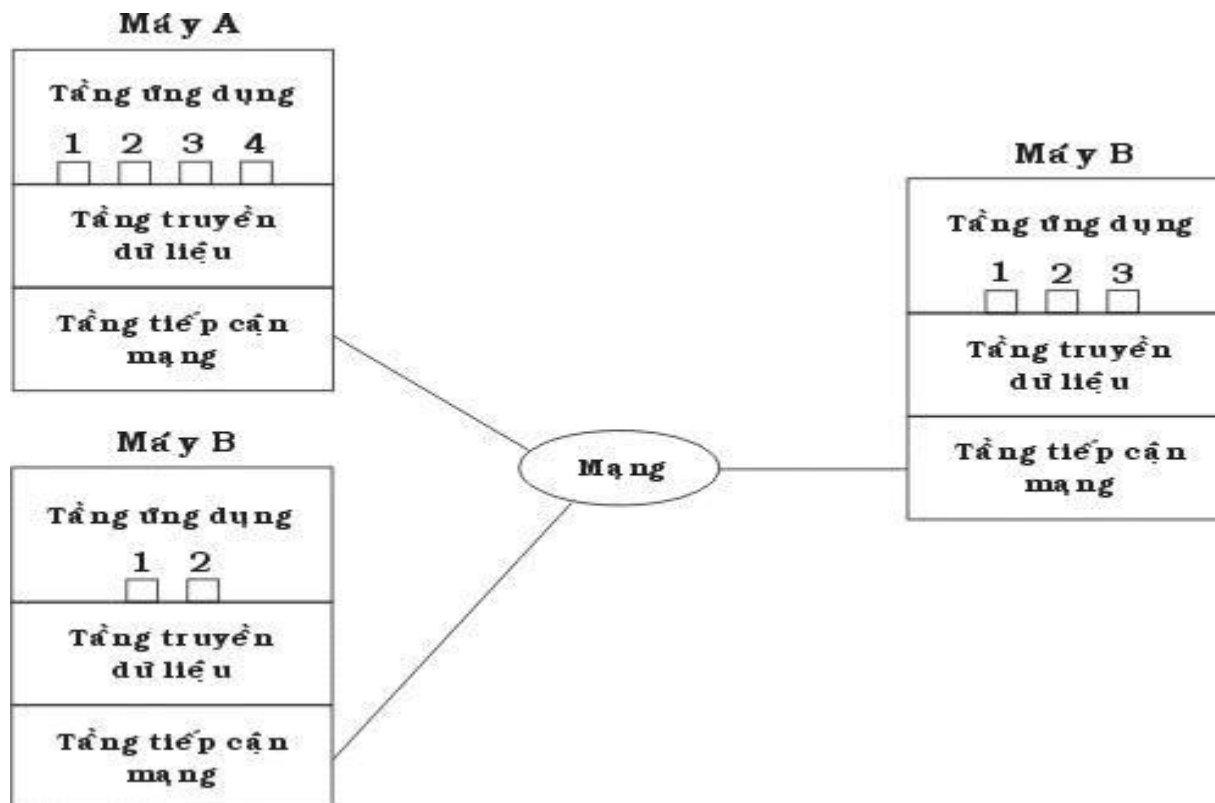




Mô hình phân tầng thu gọn 3 tầng

- ❖ Các thành phần tham gia trong quá trình truyền thông
 - Các chương trình ứng dụng
 - Các chương trình truyền thông
 - Các máy tính và các mạng
- ❖ Gửi dữ liệu các ứng dụng
 - Máy tính gửi
 - Ứng dụng gửi chuyển dữ liệu cho chương trình truyền thông
 - Chương trình truyền thông sẽ gửi dữ liệu cho máy tính nhận
 - Máy tính nhận
 - Chương trình truyền thông sẽ tiếp nhận và kiểm tra dữ liệu
 - Chuyển cho ứng dụng đang chờ dữ liệu

Ví dụ mô hình truyền thông đơn giản





Ví dụ mô hình truyền thông đơn giản

❖ Máy A:

- Ứng dụng 1 cần gửi một khối dữ liệu
- Dữ liệu được chuyển cho tầng giao vận
 - Chia dl thành nhiều đoạn và đóng gói thành các gói tin (packets)
 - Bổ sung thêm các thông tin điều khiển (header) vào mỗi gói tin
- Dữ liệu tiếp tục được chuyển cho tầng tiếp cận mạng và chuyển cho máy B.

❖ Máy B:

- Tầng tiếp cận mạng sẽ tập hợp dữ liệu và chuyển cho tầng giao vận
 - Kiểm tra và ghép dl lại thành khối (nhờ tt header)
 - Khối dữ liệu sẽ được chuyển lên cho tầng ứng dụng



Bài 3:

Mô hình ứng dụng client/server

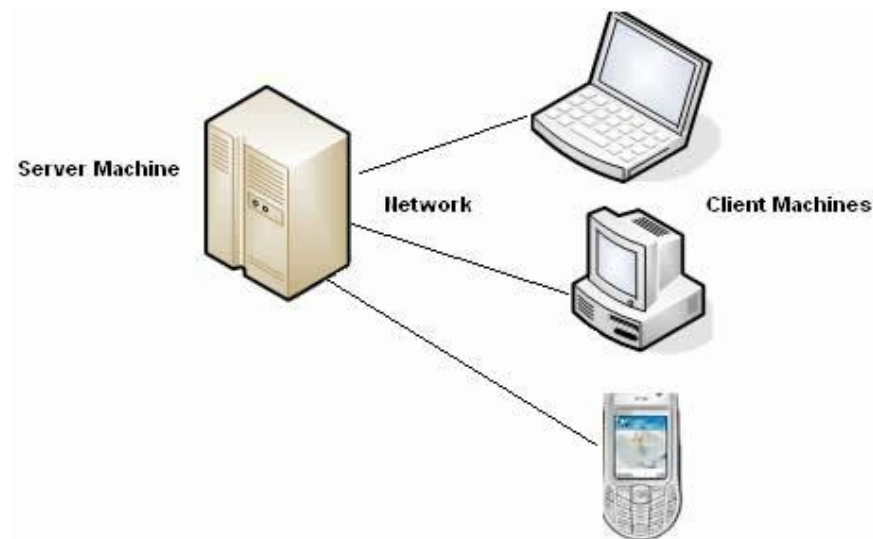


Tổng quan

- ❖ Mô hình mạng cơ bản nhất hiện nay
- ❖ Dạng phổ biến của mô hình ứng dụng phân tán
- ❖ Đa số các ứng dụng mạng dựa theo mô hình này
- ❖ Thuật ngữ client/server xuất hiện từ đầu thập niên 80
- ❖ Ứng dụng client/server phổ biến
 - Email, FTP, Web

Thành phần

- ❖ Một tiến trình Server
- ❖ Một hoặc nhiều tiến trình client
 - Các tiến trình clients và servers có thể chạy trên cùng trạm (host) hoặc khác trạm.
 - Là các đối tượng logic tách biệt và liên lạc với nhau qua mạng cùng thực hiện một công việc chung



Chức năng từng thành phần

❖ Server

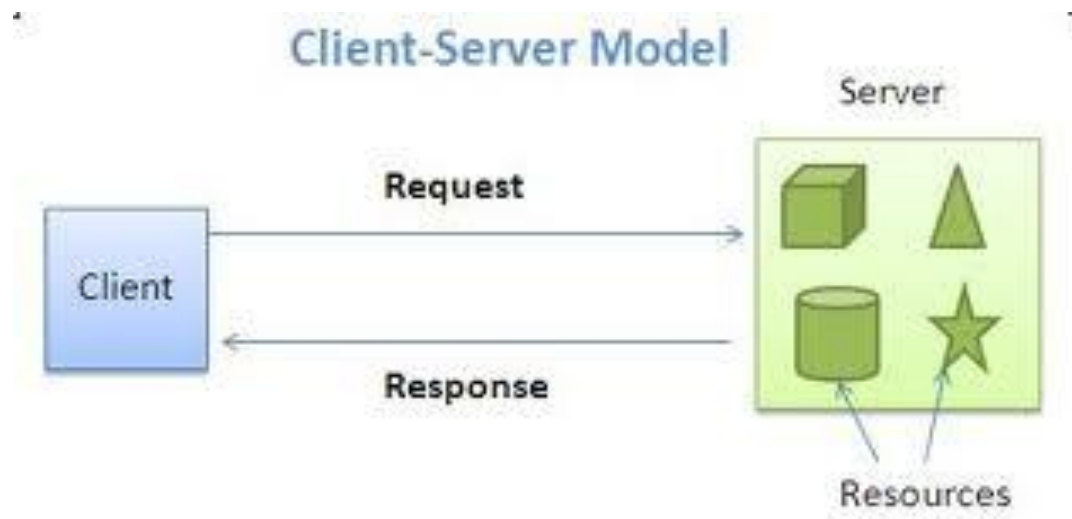
- Quản lý nguồn tài nguyên nào đó
- Cung cấp dịch vụ và phân phối tài nguyên.

❖ Client

- Chương trình giao tiếp với người sử dụng
- Cần yêu cầu về tài nguyên

❖ Một tiến trình có thể vừa là server vừa là client

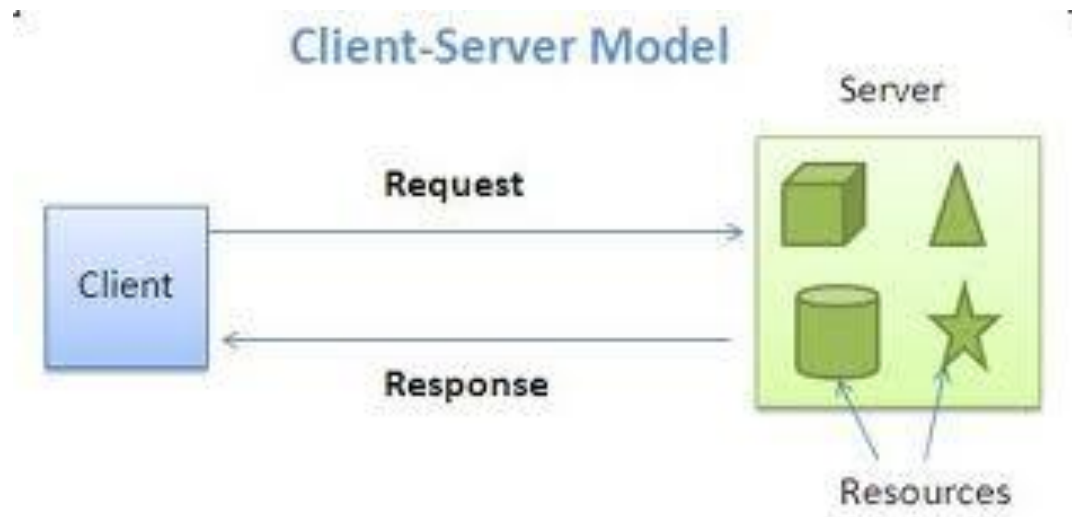
Cách hoạt động



❖ Client

- Khởi tạo liên lạc với server (speaks first)
- Yêu cầu dịch vụ nào đó từ server
- Đối với Web, client được hiện thực trong browser

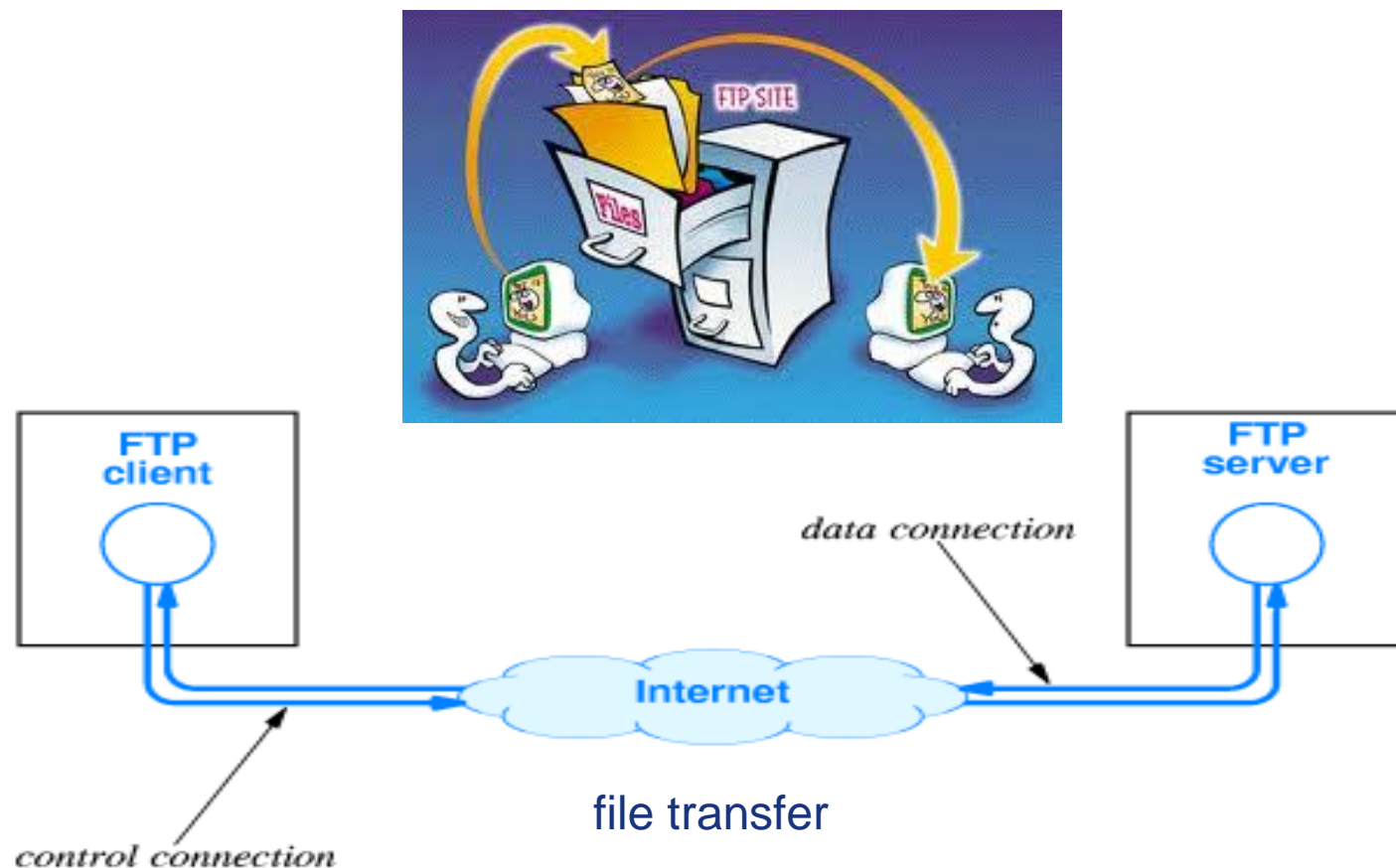
Cách hoạt động



❖ Server

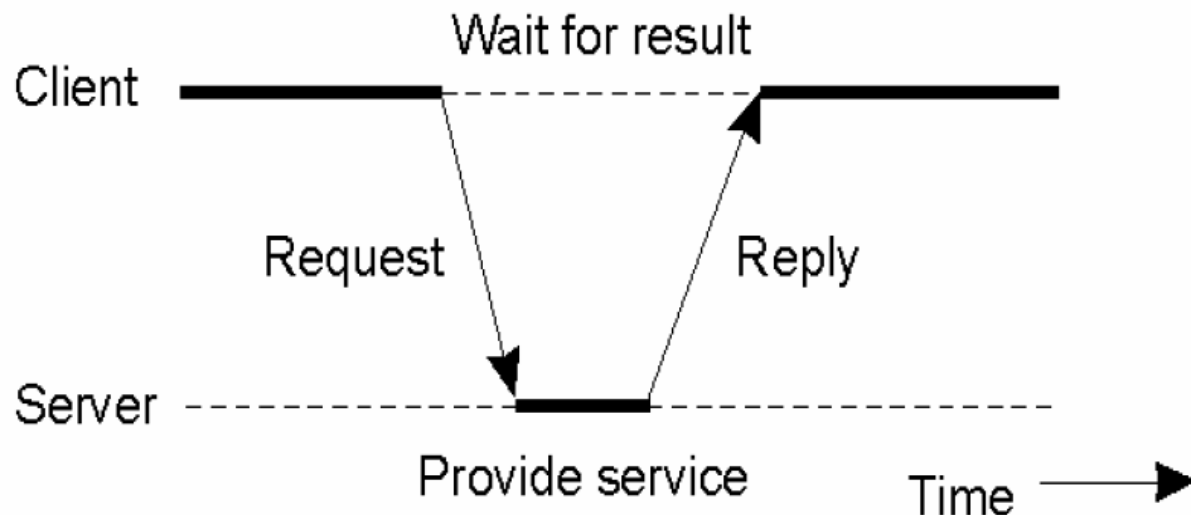
- Cung cấp dịch vụ yêu cầu cho client
- Chẳng hạn, Web server gửi Web yêu cầu hay mail server phân phát email

Ví dụ FTP: The File Transfer Protocol



Đặc trưng của mô hình ứng dụng C/S

- ❖ Hoạt động theo kiểu giao thức bất đối xứng
 - Thể hiện quan hệ một chiều giữa client và một server
 - Client bắt đầu phiên hội thoại bằng cách yêu cầu dịch vụ
 - Server sẵn sàng chờ các yêu cầu từ client





Đặc trưng của mô hình ứng dụng C/S

❖ Đóng gói dịch vụ

- Server như một chuyên gia
 - Hoàn thành tác vụ đáp ứng lại các yêu cầu từ client
- Server có thể được nâng cấp mà không ảnh hưởng đến client

❖ Tính toàn vẹn

- Mã và dữ liệu đối với server được bảo trì tập trung
 - Giảm chi phí và bảo vệ sự toàn vẹn của dữ liệu chung
- Client duy trì tính cá nhân và độc lập

Ưu điểm của mô hình UD C/S

❖ Tính tập trung (Centralization)

- Truy cập tài nguyên và bảo mật dữ liệu
 - Tập trung thông qua server

❖ Tính co giãn (Scalability)

- Nâng cấp bất cứ thành phần nào khi cần thiết

❖ Tính mềm dẻo (Flexibility)

- Công nghệ mới có thể dễ dàng tích hợp vào hệ thống

❖ Tính trao đổi tương tác (Interoperability)

- Tất cả các thành phần (clients, mạng, servers) cùng nhau làm việc

Nhược điểm của mô hình UD C/S

- ❖ Quản trị hệ thống khó khăn
 - Duy trì thông tin cấu hình luôn cập nhật và nhất quán giữa tất cả các thiết bị
- ❖ Nâng cấp phiên bản mới khó đồng bộ
- ❖ Phụ thuộc độ tin cậy của mạng
- ❖ Chi phí thiết kế, cài đặt, quản trị và bảo trì cao
- ❖ Phải giải quyết
 - Sự xung đột trong hệ thống
 - Tính tương thích của các thành phần
 - Việc cấu hình hệ thống



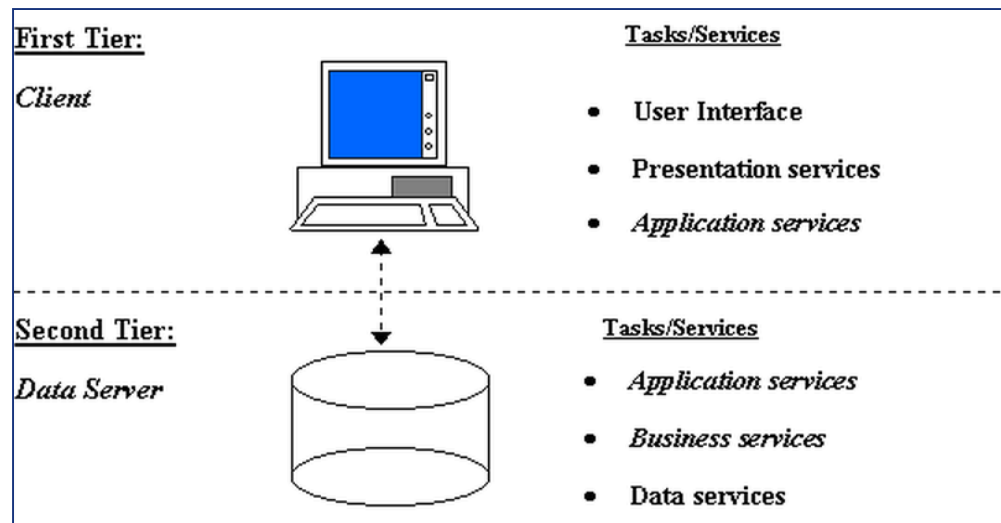
Sự phân lớp trong mô hình UD C/S

- ❖ Mọi ứng dụng mạng client/server đều có 3 khối chức năng
 - Khối biểu diễn hay giao diện người dùng
 - Khối nghiệp vụ: thuật toán điều khiển thông tin giữa khối biểu diễn và khối dữ liệu
 - Khối dữ liệu

Kiến trúc UD C/S 2 lớp (2-tier)

❖ Khối nghiệp vụ (business logic)

- Được đặt bên trong lớp giao diện người dùng tại client, **hoặc**
- Được đặt bên trong CSDL

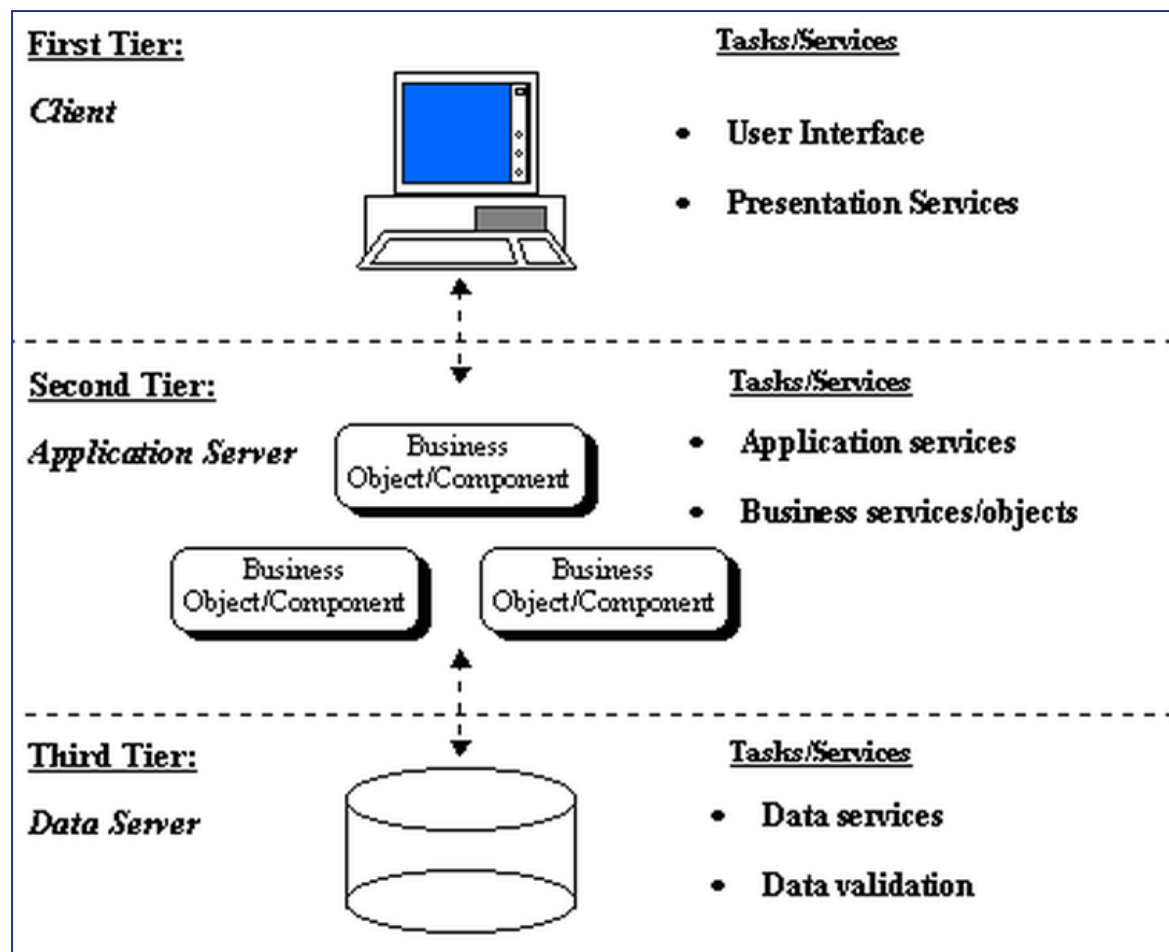




Kiến trúc UD C/S 3 lớp (3-tier)

- ❖ Phát triển vào thập niên 90'
 - Hệ thống lớn và ổn định
- ❖ Mở rộng mô hình 2 lớp: Tách biệt khối nghiệp vụ
 - Nâng cao hiệu năng (performance),
 - Tính linh hoạt (flexibility),
 - Khả năng bảo trì (maintainability),
 - Khả năng dùng lại (reusability)

Kiến trúc UD C/S 3 lớp (3-tier)





Kiến trúc UD C/S 3 lớp (3-tier)

❖ Lớp trên cùng

- Chứa giao diện dịch vụ cho người dùng

❖ Lớp dưới cùng

- Chứa chức năng quản trị CSDL

❖ Lớp trung gian

- Chứa khối nghiệp vụ
- Các tiến trình xử lý
- Điều khiển các giao dịch và các hàng đợi
- Gửi các yêu cầu từ client đến csdl
- Được xem như proxy server

Ứng dụng client/server 3 lớp

❖ Ứng dụng Web

- Lớp trên cùng: Web browser
- Lớp trung gian: Web server engine
- Lớp dưới cùng: Hệ CSDL

❖ Ứng dụng Struts hoặc JSF (trong Java)

- Lớp trên cùng: Views
- Lớp trung gian: Controllers
- Lớp dưới cùng: Models

❖ Kiến trúc client/server n lớp (n-tier)

- Lớp trung gian được chia thành nhiều đơn vị nhỏ

Giao thức cho ỨD client/server

❖ Giao thức?

- Là tập các khuôn dạng bản tin, tập các trạng thái, quy tắc, quy ước trong truyền thông giữa client và server.

❖ Khi tạo một ứng dụng client/server

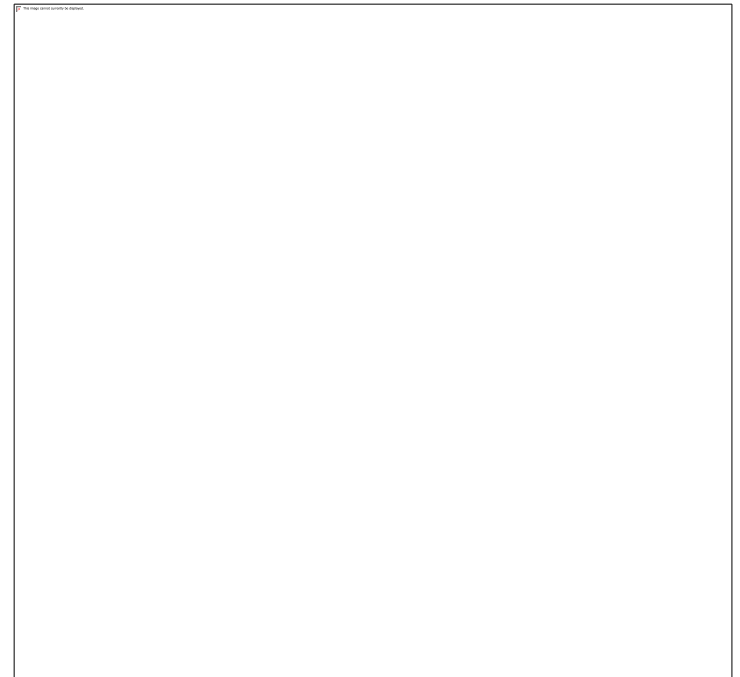
- Phải thiết kế giao thức
- Các giao thức phổ biến FTP, HTTP, ...
 - đã được IETF (Internet Engineering Task Force) chuẩn hóa thành các giao thức chuẩn.

❖ Lập trình CSDL

- Được hỗ trợ và quy định bởi hệ quản lý csdl và các thư viện lập trình.

Phân loại giao thức

- ❖ Giao thức đồng bộ (Synchronous protocol)
 - Truyền thông giữa client và server diễn ra theo 2 chiều nhưng không đồng thời
 - Được thực hiện lần lượt
 - Các giao thức kiểu này là:
 - HTTP SMTP, POP3

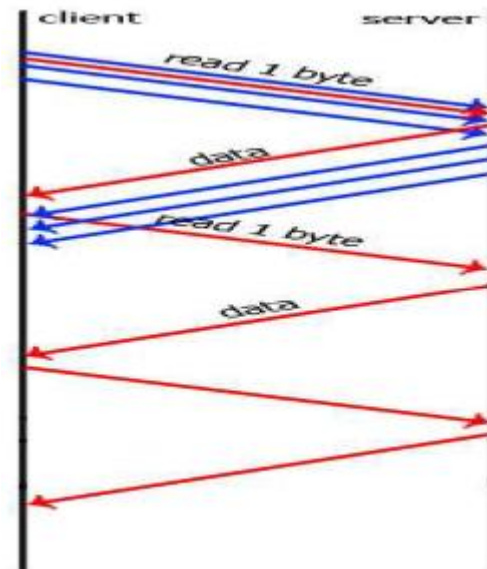


Phân loại giao thức

❖ Giao thức không đồng bộ (Asynchronous protocol)

- Client và server có thể đồng thời gửi thông tin
- Các giao thức này như TELNET, RLOGIN,...

- Ngoài ra, còn có loại giao thức hybrid kết hợp giữa các giao thức trên





Lập trình mạng trên Java

LẬP TRÌNH MẠNG TRÊN JAVA

❖ Gói java.net

- InetAddress
- ServerSocket
- Socket
- URL
- URLConnection
- DatagramSocket

LẬP TRÌNH MẠNG TRÊN JAVA

❖ InetAddress class

- Class mô tả về địa chỉ IP (Internet Protocol)
- Các phương thức `getLocalHost`, `getByName`, hay `getAllByName` để tạo một `InetAddress` instance:
 - *`public static InetAddress InetAddress.getByName(String hostname)`*
 - `public static InetAddress [] InetAddress.getAllByName(String hostname)`
 - `public static InetAddress InetAddress.getLocalHost()`
- Để lấy địa chỉ IP hay tên dùng các phương thức:
 - `getHostAddress()`
 - `getHostName()`

Ví dụ 1

❖ In địa chỉ IP của localhost

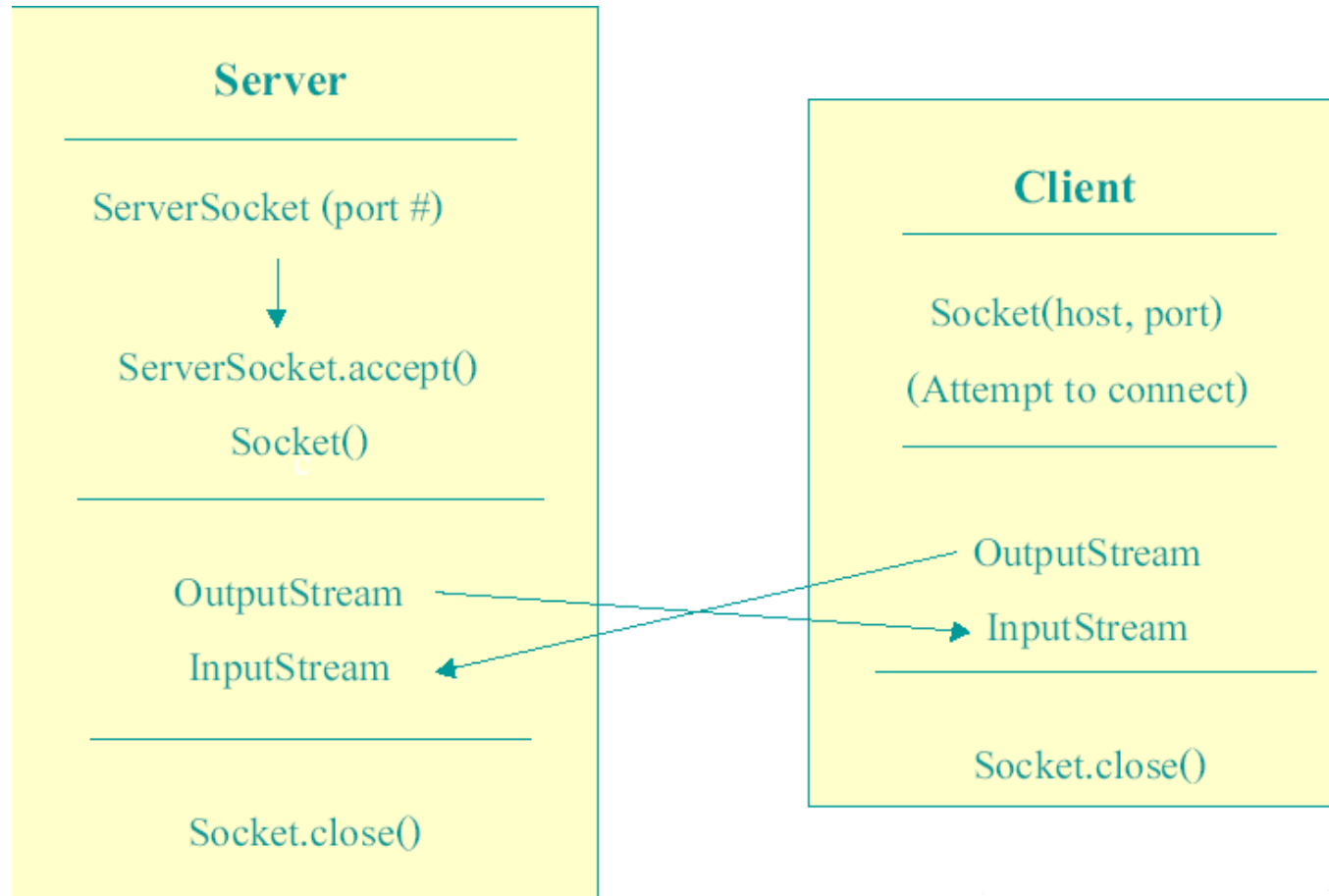
```
import java.net.*;
public class HostInfo {
    public static void main(String args[]) {
        try {
            InetAddress myHost = InetAddress.getLocalHost();
            System.out.println(myHost.getHostAddress());
            System.out.println(myHost.getHostName());
        } catch (UnknownHostException ex) {
            System.err.println("Cannot find local host");
        }
    }
}
```

Ví dụ 2

❖ In địa chỉ IP của một địa chỉ bất kỳ

```
import java.net.*;
public class IPAddresses{
    public static void main (String args[]) {
        try {
            InetAddress[] addresses =
                InetAddress.getAllByName(args[0]);
            for (int i = 0; i < addresses.length; i++)
                System.out.println(addresses[i]);
        }
        catch (UnknownHostException e) {
            System.out.println("Could not find" + args[0]);
        }
    }
}
```

LẬP TRÌNH MẠNG TRÊN JAVA



LẬP TRÌNH MẠNG TRÊN JAVA

❖ **Socket class**

- Class mô tả về *socket*
- Tạo một *socket*
 - **Socket**(InetAddress address, int port)
 - **Socket**(String host, int port)
 - **Socket**(InetAddress address, int port, InetAddress, localAddr, int localPort)
 - **Socket**(String host, int port, InetAddress, localAddr, int localPort)
 - **Socket**()

LẬP TRÌNH MẠNG TRÊN JAVA

❖ **Socket class (tiếp theo)**

■ **Lấy thông tin về một socket**

- InetAddress **getInetAddress()** : trả về địa chỉ mà socket kết nối đến.
- int **getPort()** : trả về địa chỉ mà socket kết nối đến.
- InetAddress **getLocalAddress()** : trả về địa chỉ cục bộ.
- int **getLocalPort()** : trả về địa chỉ cục bộ.

■ **Sử dụng Streams**

- public OutputStream **getOutputStream()** throws IOException
Trả về một output stream cho việc viết các byte đến socket này.
- public InputStream **getInputStream()** throws IOException
Trả về một input stream cho việc đọc các byte từ socket này.

LẬP TRÌNH MẠNG TRÊN JAVA

❖ Kết nối đến 1 số webserver

```
import java.net.*;
import java.io.*;
public class getSocketInfo {
    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++) {
            try {
                Socket theSocket = new Socket(args[i], 80);
                System.out.println("Connected to " +
                    theSocket.getInetAddress() +
                    " on port " + theSocket.getPort() + " from port " +
                    theSocket.getLocalPort() + " of " +
                    theSocket.getLocalAddress());
            }
        }
    }
}
```

Tiếp theo

```
        } catch (UnknownHostException e) {  
            System.err.println("I can't find " + args[i]);  
        } catch (SocketException e) {  
            System.err.println("Could not connect to " + args[i]);  
        } catch (IOException e) {  
            System.err.println(e);  
        }  
    } // end for  
} // end main  
} // end getSocketInfos
```

ServerSocket class

- ❖ Class mô tả về *ServerSocket*
- ❖ Tạo một *ServerSocket*
 - **ServerSocket**(int port) throws IOException
 - **ServerSocket**(int port, int backlog) throws IOException
 - **ServerSocket**(int port, int backlog, InetAddress bindAddr) throws IOException

ServerSocket class (tt)

❖ Các phương thức trong *ServerSocket*

- Socket **accept()** throws IOException: Chấp nhận và lắng nghe một kết nối đến Socket này.
- void **close()** throws IOException: Đóng socket
- InetAddress **getInetAddress()**: Trả về địa chỉ cục bộ của Socket
- int **getLocalPort()**: Trả về Local port
- void **setSoTimeout(int timeout)** throws SocketException

Ví dụ DateTime Server

```
import java.net.*;
import java.io.*;
import java.util.Date;
public class DayTimeServer {
    public final static int daytimePort = 5000;
    public static void main(String[] args) {
        ServerSocket theServer;
        Socket theConnection;
        try {
            theServer = new ServerSocket(daytimePort);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

(tt)

Ví dụ DateTime Server

```
while (true) {  
    theConnection = theServer.accept();  
    DataOutputStream dos = new DataOutputStream(  
        theConnection.getOutputStream());  
    String time = new Date().toString();  
    dos.writeUTF(time);  
    theConnection.close();  
    theServer.close();  
}  
} catch (IOException e) {  
    System.err.println(e);  
}  
}
```



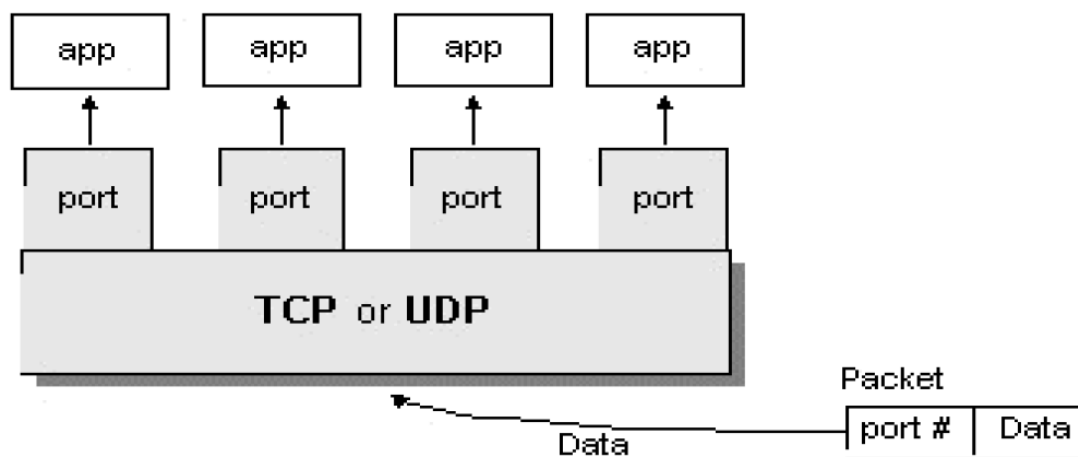
Lập trình với giao thức TCP

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)

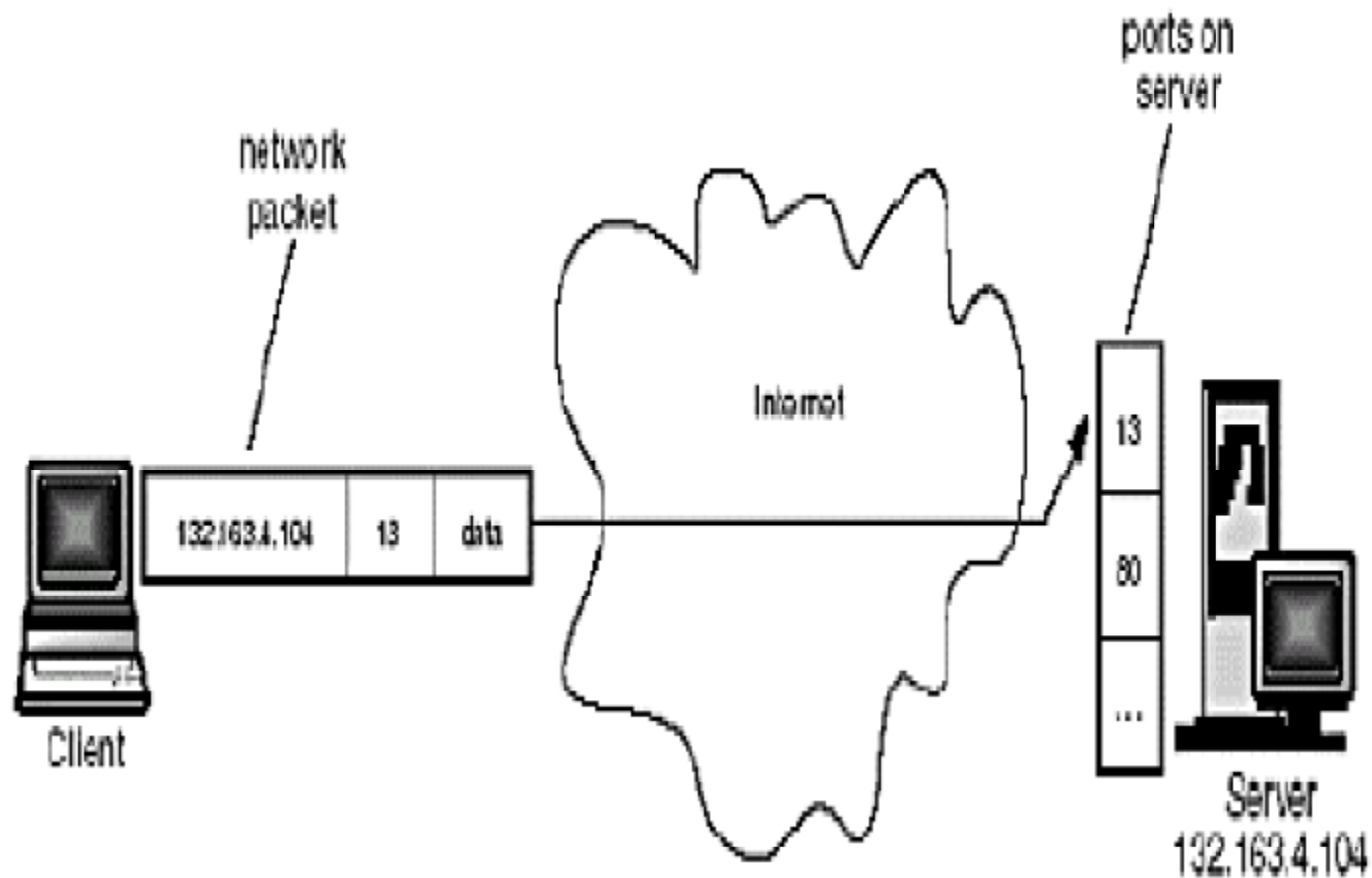
- ❖ Là một số (nhãn) đặc biệt
 - Được gán cho một tiến trình mạng
- ❖ Mỗi tiến trình mạng đều được gán một cổng duy nhất



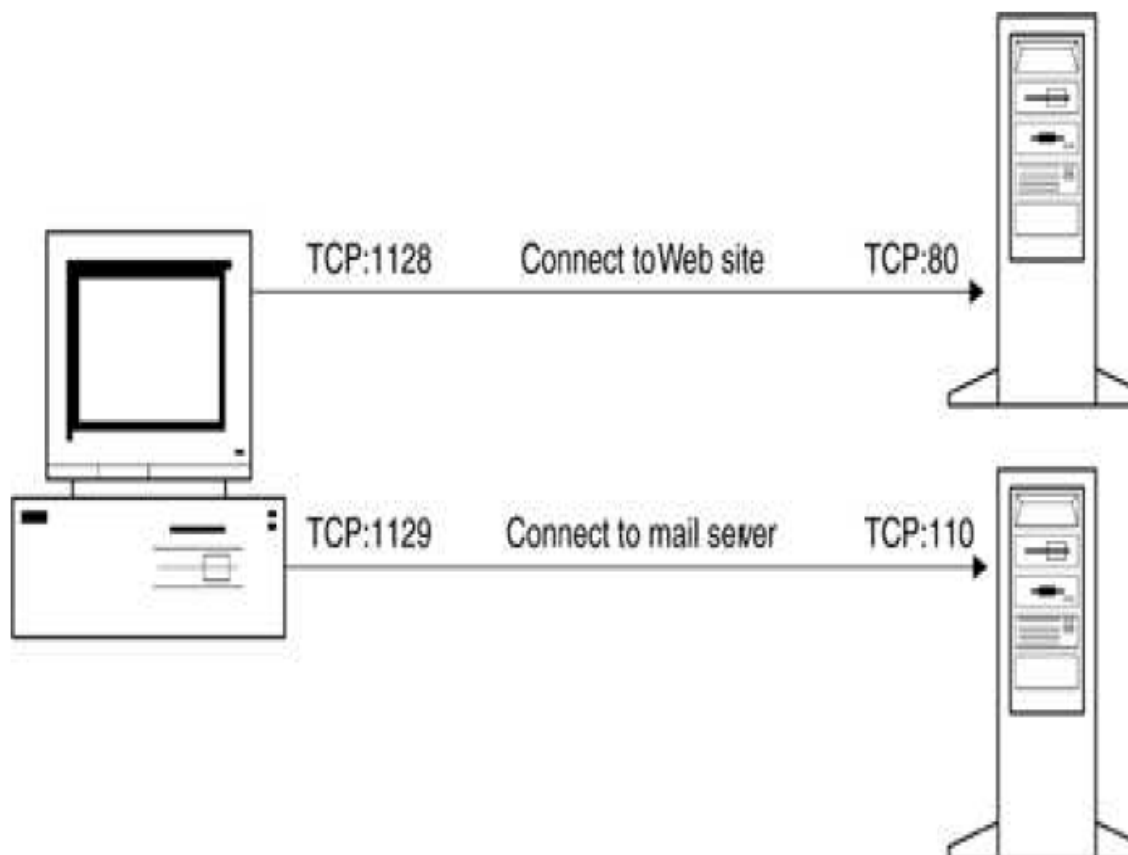
Cổng (Port)

- ❖ 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)



1 ỨD sử dụng nhiều cổng khác nhau



Một số cổng 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 chỉ có một địa chỉ IP duy nhất
 - IPv4: 32 bits, tạo thành từ 4 octets. Vd: 192.169.12.1
 - 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				

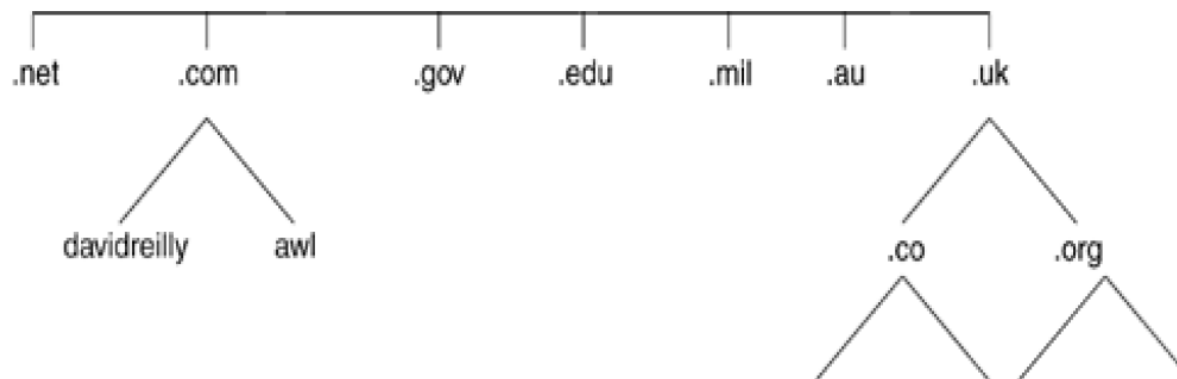
Dãy địa chỉ IP theo lớp

- ❖ 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

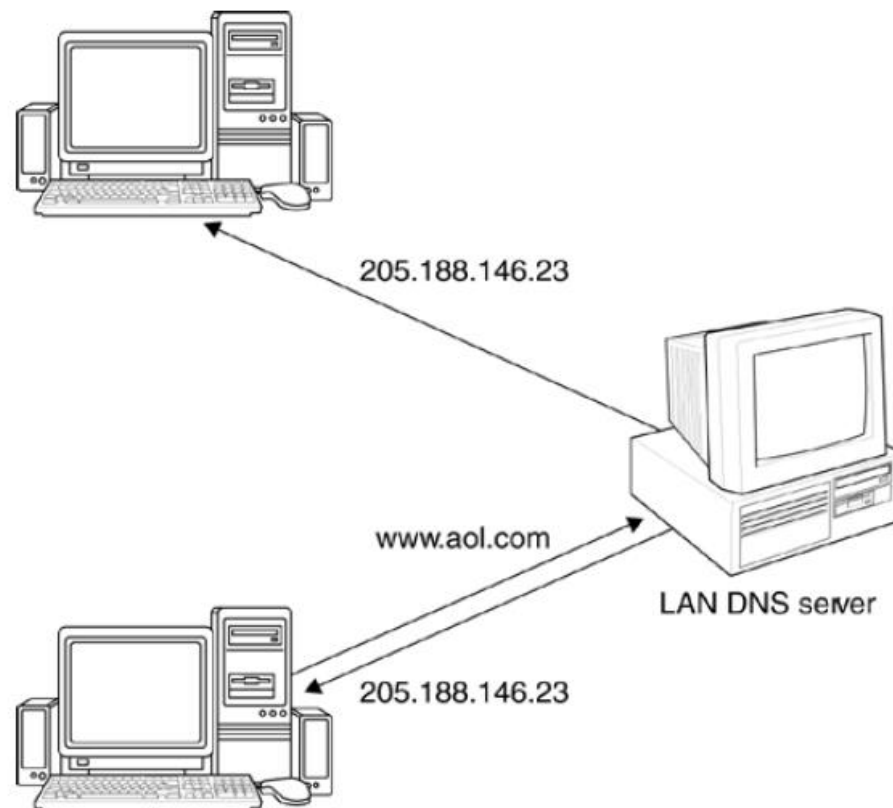
Địa chỉ 127.0.0.1 chỉ địa chỉ IP máy cục bộ

Tên miền (Domain Name)

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



Ảnh 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

Các hoạt động của Socket

- ❖ 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 được gán

Các kiểu socket

❖ Có 3 kiểu socket

- Kiểu 1: Stream sockets, tương tự như điện thoại kết nối đến một tổng đài. Gọi là: kiểu hướng kết nối
 - VD: TCP(Transmission Control Protocol) socket
 - Đảm bảo dữ liệu đến đích an toàn và đầy đủ
- Kiểu 2: Datagram sockets, tương tự như mailbox. Gọi là: kiểu phi kết nối (không giữ kênh kết nối trong quá trình truyền thông)
 - VD: UDP(User Datagram Protocol) socket
- Kiểu 3: Raw sockets

Các bước tạo một TCP

❖ Phía Server

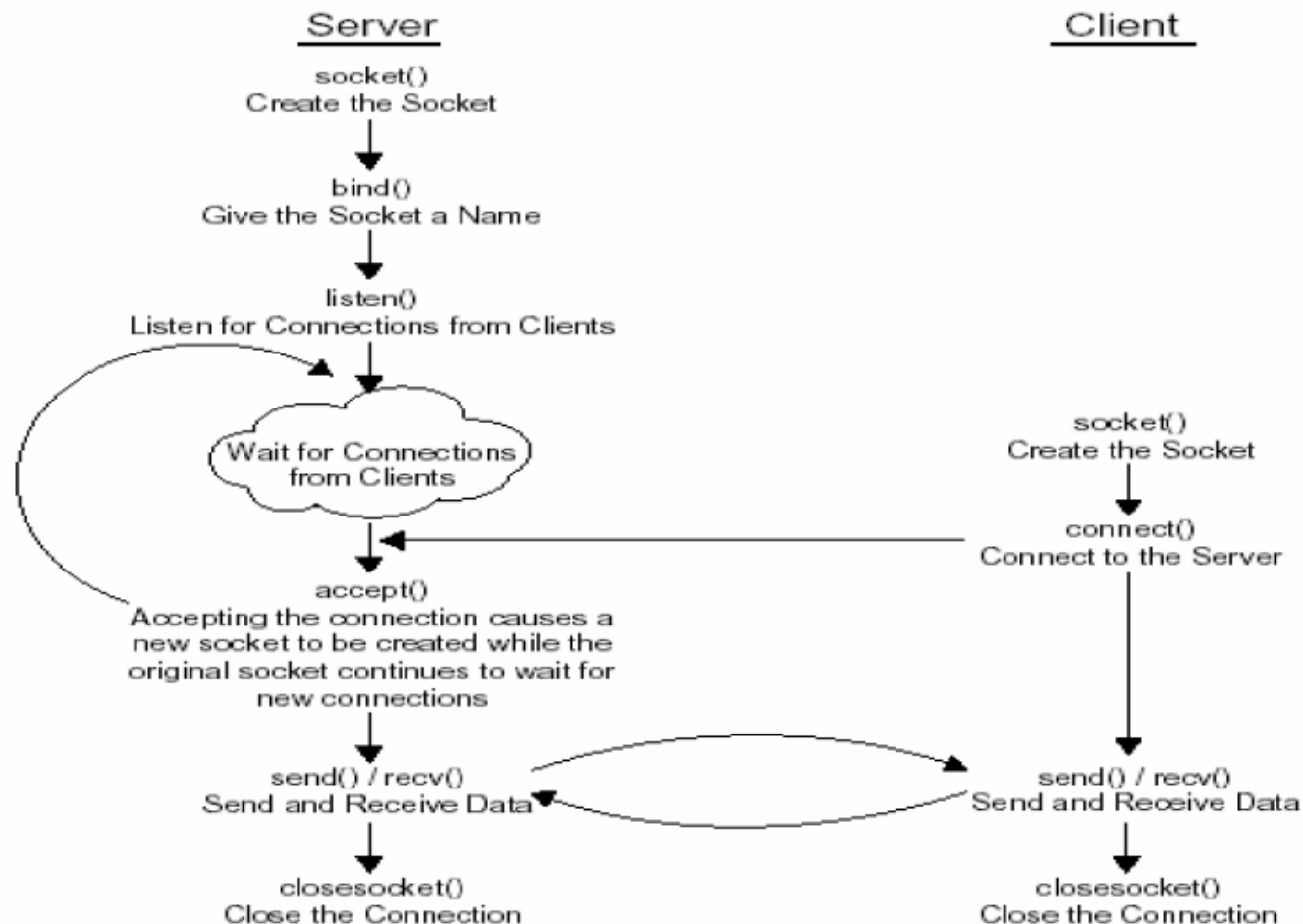
- Gán một cổng với Socket
- Chờ và lắng nghe yêu cầu kết nối từ client
- Chấp nhận kết nối, tạo Socket tương ứng
- Truyền nhận dữ liệu thông qua các streams in/out của đối tượng Socket
- Đóng kết nối

Các bước tạo một TCP

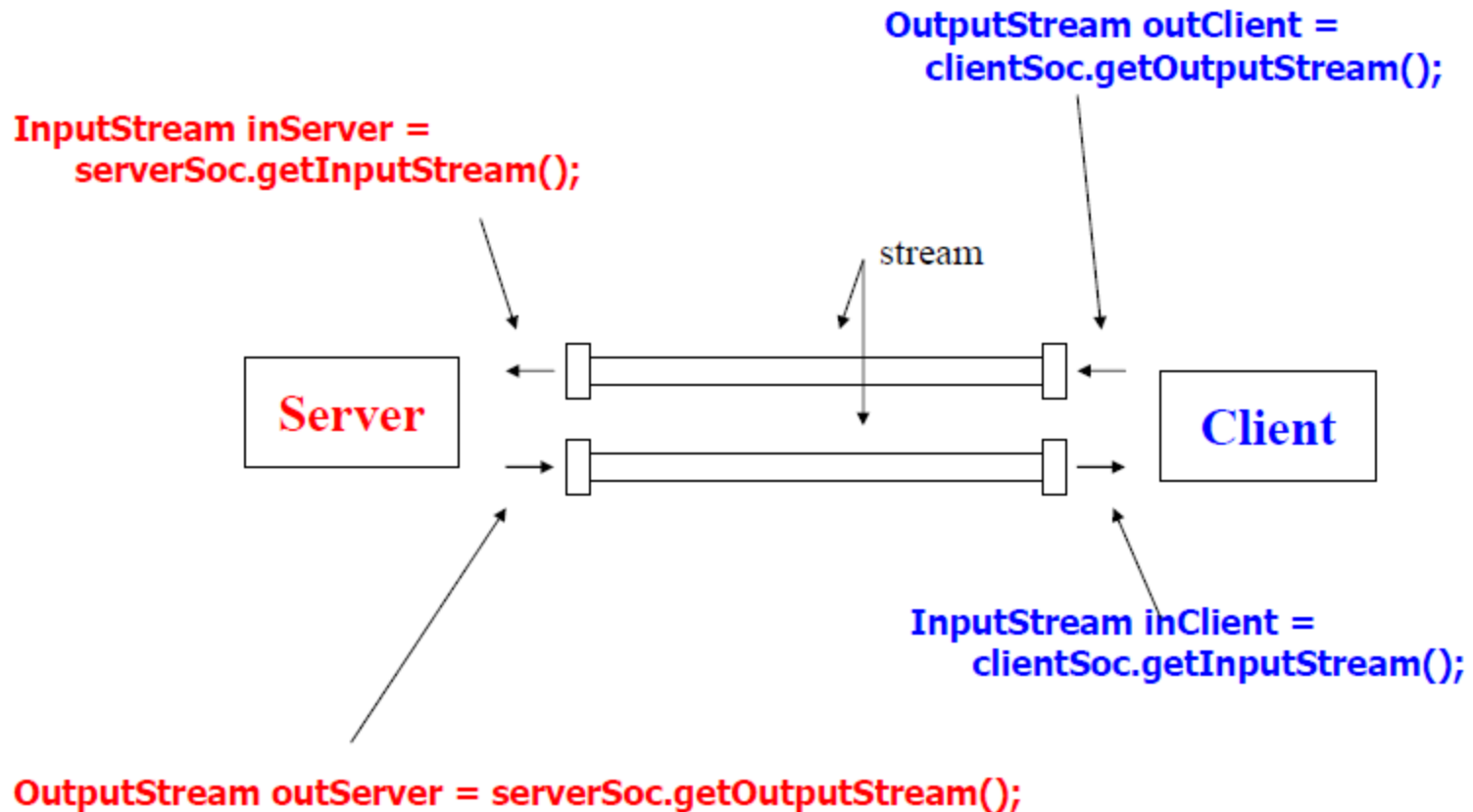
❖ Phía Client

- 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

Các bước tạo một TCP



Mô tả quá trình trao đổi dữ liệu



Ví dụ DateTime Server

```
public class DayTimeServer {  
    public final static int daytimePort = 5000;  
    public static void main(String[] args) {  
        ServerSocket theServer;  
        try {  
            theServer = new ServerSocket(daytimePort);  
            while (true) {  
                Socket theConnection = theServer.accept();  
                DataOutputStream dos = new DataOutputStream(  
                    theConnection.getOutputStream());  
                String time = new Date().toString();  
                dos.writeUTF(time);  
                theConnection.close();  
            }  
        } catch (IOException e) {}  
    }  
}
```

DateTime 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", 5000);
        DataInputStream din = new
        DataInputStream(socket.getInputStream());
        String time = din.readUTF();
        System.out.println(time);
    }
}
```



Threaded Server


- ❖ Ví dụ vừa rồi chỉ đúng với một client kết nối đến server tại một thời điểm
- ❖ Để cho phép nhiều client có thể kết nối đến server cùng lúc thì server phải là chương trình đa tuyến
 - Mỗi tuyến ứng với một kết nối từ client.

Threaded Server

- Đoạn mã chương trình bằng Java

```
while(true){  
    Socket socket = s.accept( );  
    new ThreadedTimeHandler(socket, i).start();  
    i++;  
}
```

```
class ThreadedTimeHandler extends Thread {  
    Socket incoming;  
    int counter;  
    public ThreadedTimeHandler(Socket i, int c)  
    {  
        incoming = i; counter = c;  
    }  
    public void run() { ....  
        // đoạn mã truyền nhận dữ liệu ở đây  
    }  
}
```



Bài tập

■ 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!.



Lập trình với UDP Socket

Giao thức UDP

- ❖ UDP (User Datagram Protocol) là giao thức mà cho phép gửi các gói(packets) dữ liệu độc lập, gọi là datagrams,
 - từ một máy tính đến một máy tính khác
 - Không đảm bảo toàn vẹn dữ liệu.
- ❖ UDP là giao thức phi kết nối(tức không giữ kênh trong quá trình truyền)



Giao thức UDP

- ❖ UDP không cung cấp cơ chế báo nhận
- ❖ UDP không sắp xếp tuần tự các gói tin (datagram) đến và có thể dẫn đến tình trạng mất hoặc trùng dữ liệu mà không có cơ chế báo lỗi cho người gửi
 - Gửi datagrams tương tự như gửi 1 bưu kiện thông qua dịch vụ bưu điện: thứ tự phát gói tin không quan trọng và không bảo đảm
 - Mỗi thông điệp độc lập với các thông điệp khác.

Ứng dụng phổ biến của UDP

- ❖ DNS (Domain Name System),
- ❖ Ứng dụng streaming media
- ❖ Voice over IP
- ❖ Trivial File Transfer Protocol (TFTP)
- ❖ Game trực tuyến

Khuôn dạng UDP datagrams

- ❖ UDP datagrams có tham số đơn giản hơn nhiều so với TCP

+	Bits 0 – 15	16 – 31
0	Source Port	Destination Port
32	Length	Checksum
64	Data	

Header của IP trong TCP

+	Bít 0 – 3	4 – 7	8 – 9	10 – 15	16 – 31
0	Source address				
32	Destination address				
64	Zeros		Protocol		TCP length
96	Source Port				Destination Port
128	Sequence Number				
160	Acknowledgement Number				
192	Data Offset	Reserved	Flags	Window	
225	Checksum				Urgent Pointer
257	Options (optional)				
257/289+	Data				

Khuôn dạng UDP datagrams

❖ Source port

- Trường này xác định cổng của người gửi thông tin và có ý nghĩa nếu muốn nhận thông tin phản hồi từ người nhận.

❖ Destination port: Trường xác định cổng nhận thông tin.

❖ Length: Chiều dài của toàn bộ datagram

❖ Checksum

- Trường checksum 16 bit dùng cho việc kiểm tra lỗi của phần header và dữ liệu.

+	Bits 0 – 15	16 – 31
0	Source Port	Destination Port
32	Length	Checksum
64	Data	



Tính năng của UDP

- ❖ UDP cũng cung cấp cơ chế gán và quản lý các số hiệu cổng.
- ❖ Do ít chức năng phức tạp nên UDP thường có xu thế hoạt động nhanh hơn so với TCP
- ❖ UDP thường dùng cho các ứng dụng không đòi hỏi độ tin cậy cao trong khi truyền

Ưu nhược điểm của UDP

Các đặc trưng	UDP	TCP
Hướng liên kết	Không	Có
Sử dụng phiên	Không	Có
Độ tin cậy	Không	Có
Xác thực	Không	Có
Đánh thứ tự	Không	Có
Điều khiển luồng	Không	Có
Bảo mật	Ít	Nhiều hơn

Khi nào sử dụng UDP

- ❖ Rất nhiều ứng dụng trên Internet sử dụng UDP. Dựa trên các ưu và nhược điểm của UDP chúng ta có thể kết luận UDP có ích khi:
 - Sử dụng cho các phương thức truyền broadcasting và multicasting khi chúng ta muốn truyền tin với nhiều host.
 - Kích thước datagram nhỏ
 - Không cần thiết lập liên kết
 - Không cần truyền lại các gói tin
 - Ứng dụng không gửi các dữ liệu quan trọng
 - Bảng thông của mạng đóng vai trò quan trọng

Các bước tạo ứng dụng UDP socket

❖ Ứng dụng UDP socket không thiết lập kết nối như TCP

■ Client

- Socket, tạo một điểm cuối truyền thông phía client
- Truyền và nhận dữ liệu

■ Server

- Socket, tạo một điểm cuối truyền thông phía server
- Gắn một cổng đến kết nối
- Truyền nhận dữ liệu

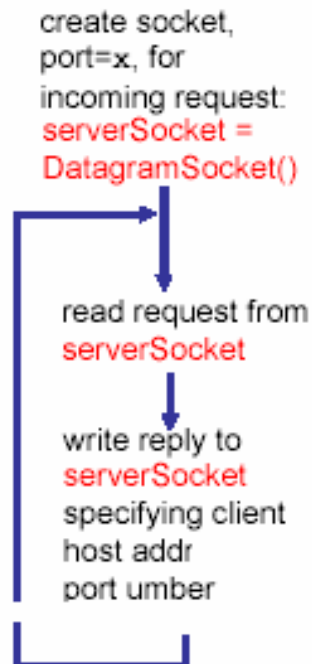


Lưu ý

- ❖ Client không thiết lập kết nối đến server
 - Kết nối không cần thiết
- ❖ Server không chấp nhận kết nối
 - Chờ và lắng nghe không cần thiết
 - Không tồn tại chấp nhận kết nối

Lập trình Socket với UDP

Server (running on `hostid`)



Client



Ví dụ chương trình Java: Client

```
import java.util.*;
import java.net.*;
public class UDPClient{
    public static void main(String args[]) throws Exception{
        Scanner input=new Scanner(System.in);
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getByName("localhost");
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];

        String sentence = input.nextLine();
        sendData = sentence.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
                                                         sendData.length,IPAddress, 8876);
        clientSocket.send(sendPacket);
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
                                                           sentence.length());
        clientSocket.receive(receivePacket);
        String modifiedsentence = new String(receivePacket.getData());
        System.out.println("From Server:"+modifiedsentence);
        clientSocket.close();
    }
}
```

Ví dụ chương trình Java: Server

```
import java.net.*;
public class UDPServer
{
    public static void main(String args[]) throws Exception{
        DatagramSocket serverSocket = new DatagramSocket(8876);
        byte[] receiveData = new byte[1024];
        byte[] sendData = new byte[1024];
        while(true)
        {
            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
            serverSocket.receive(receivePacket);
            String sentence = new String(receivePacket.getData());
            InetAddress IPAddress = receivePacket.getAddress();
            int port = receivePacket.getPort();
            System.out.println(""+IPAddress+", "+port+
            ":"+sentence);
            String capsent=sentence.toUpperCase();
            sendData = capsent.getBytes();
            DatagramPacket sendPacket =
            new DatagramPacket(sendData, capsent.length(), IPAddress, port);
            serverSocket.send(sendPacket);
        }
    }
}
```

Bài tập lập trình UDP Socket

❖ Xây dựng chương trình Exchange Rate

■ ExchangeRateServer

- Khi nhận một chuỗi thì xác định chuỗi đó có cấu trúc sau hay không:
 - “ExchangeRate”+“Loại tiền”+”to”+“Loại tiền”,
 - “Loại tiền”: Yen, VND, USD, ...
- Nếu đúng thì trả về cho máy khách tỉ giá(random) theo ngày giờ của hệ thống

■ ExchangeRateTable

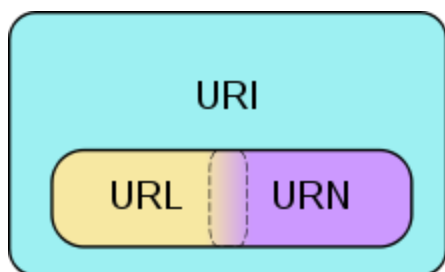
- Cập nhật thông tin về tỉ giá sau mỗi giây.



URI, URL và URLConnection

URI

- ❖ URI (Uniform Resource Identifier) là một chuỗi ký tự được sử dụng để xác định một tên hoặc một tài nguyên.
- ❖ URI nhằm cho phép tương tác với các thể hiện tài nguyên trên mạng (World Wide Web) sử dụng các giao thức cụ thể.



Uniform Resource Locator(URL)

Uniform Resource Name(URN)



URI (tt)

- ❖ URI được biểu hiện bởi lớp `java.net.URI`
- ❖ `java.net.URI` có khả năng:
 - Biểu hiện
 - Phân tích
 - Chuẩn hóa

URI (Ví dụ)

```
import java.net.*;
public class URITest {
    public static void main(String[] argv) throws Exception {
        URI uri1 = new URI("http://www.example.com/hoge/./moge/../../");
        URI uri2 = new URI("http://www.example.com/");
        uri1 = uri1.normalize();

        System.out.println("uri1: " + uri1);
        System.out.println("uri2: " + uri2);
        System.out.println(uri1.compareTo(uri2));
    }
}
```

Kết quả

```
uri1: http://www.example.com/
uri2: http://www.example.com/
0
```

URL

- ❖ URL (Uniform Resource Locator) được dùng để tham chiếu tới tài nguyên trên Internet.
- ❖ URL mang lại khả năng siêu liên kết cho các trang mạng.
- ❖ Các tài nguyên khác nhau được tham chiếu tới bằng địa chỉ, chính là URL.

URL

❖ Một URL gồm có nhiều phần :

- URL scheme, thường là Tên giao thức (ví dụ: http, ftp) nhưng cũng có thể là một cái tên khác (ví dụ: news, mailto).
- Tên miền (ví dụ: http://vi.wikipedia.org)
- Chỉ định thêm cổng (có thể không cần)
- Đường dẫn tuyệt đối trên máy phục vụ của tài nguyên (ví dụ: thumuc/trang)
- Các truy vấn (có thể không cần)
- Chỉ định mục con (có thể không cần)

URL(Ví dụ)

`http://vi.wikipedia.org:80/thumuc/trang?timkiem=cauhoi#dautien`

URL scheme

tên miền

cổng

đường dẫn

truy vấn

mục con

Ví dụ

```
import java.net.*;
public class URLTest
{
    public static void main(String[] argv)
        throws Exception
    {
        URI uri = new
            URI("http://www.example.com/hoge/./moge/../../");
        uri = uri.normalize();

        URL url = uri.toURL();

        System.out.println(url);
    }
}
```

Phân biệt URI và URL

❖ URI

- Dùng để xác định một resource nào đó trên web, về mặt tên (cách gọi nó như thế nào) hoặc địa chỉ (nó nằm ở đâu, làm sao đưa được nó về máy trạm!?)

❖ URL

- Là địa chỉ tới một resource nào đó.

Phân biệt URL và tên miền

❖ Đôi khi có thể hiểu nhầm URL là tên miền.

Ví dụ URL:

- URL:

- <http://www.viphanoi.com/index.php> <- Đây là URL

- Tên miền:

- www.viphanoi.com <- Đây là tên miền.

❖ URL là địa chỉ của một đối tượng thường được gõ vào vùng Address của các Web Browser. Về cơ bản, URL là con trỏ chỉ tới vị trí của một đối tượng.

Khởi tạo URL trong Java

- ❖ `public URL(String url)` throws `MalformedURLException`
 - `URL u = new URL("http://www.sun.com/index.html");`
- ❖ `public URL(String protocol, String host, String file)` throws `MalformedURLException`
 - `URL u = new URL("http", "www.sun.com", "index.html");`
- ❖ `public URL(String protocol, String host, int port, String file)` throws `MalformedURLException`
 - `URL u = new URL("http", "www.sun.com", 80, "index.html");`
- ❖ `public URL(URL u, String s)` throws `MalformedURLException`

Các thành phần

- ❖ `public String getProtocol()`
- ❖ `public String getHost()`
- ❖ `public int getPort()`
- ❖ `public int getDefaultPort()`
- ❖ `public String getFile()`
- ❖ `public String getRef()`

Ví dụ

- ❖ Viết chương trình nhập vào một URL từ đối dòng lệnh và hiển thị từng thành phần tạo nên URL lên màn hình.

```
try{  
    URL u = new URL(args[0]);  
    System.out.println("URL is "+u);  
    System.out.println("The protocol part is "+u.getProtocol());  
    System.out.println("The host part is "+u.getHost());  
    System.out.println("The file part is "+u.getFile());  
    System.out.println("The reference part is "+u.getRef());  
}  
catch(MalformedURLException e){}
```

URLConnection

- ❖ URLConnection là một phương thức thể hiện sự truyền tải tín hiệu của URL

URI → URL → URLConnection

Ví dụ về URL Connection

```
import java.io.*;
import java.net.*;
import java.util.*;
public class NetCat {
    public static void main(String[] argv) throws Exception {
        URI uri = new URI(argv[0]);
        URLConnection connection = uri.toURL().openConnection();
        Map headers = connection.getHeaderFields();
        for (Object key : headers.keySet()) {
            System.out.println(key + ": " + headers.get(key));
        }
        BufferedReader reader =
            new BufferedReader(new InputStreamReader
                (connection.getInputStream(), "JISAutoDetect"));
        String buffer = reader.readLine();
        System.out.println();
        while (null != buffer) {
            System.out.println(buffer);
            buffer = reader.readLine();
        }
    }
}
```

Kết quả

```
% java NetCat http://www.example.com/
```

```
Set-Cookie:
```

```
[PREF=ID=xxxxxxxxxxxxxxxx:TM=xxxxxxxx:LM=xxxxxxxx:S=xxxxxxxxxxxxxxxx;  
expires=Sun, 17-Jan-2038 19:14:07 GMT; path=/; domain=.example.com]
```

```
null: [HTTP/1.1 200 OK]
```

```
Date: [Mon, 03 Jul 2006 09:22:03 GMT]
```

```
Content-Type: [text/html]
```

```
Server: [XXX/YYY]
```

```
Transfer-Encoding: [chunked]
```

```
Cache-Control: [private]
```

```
<html>
```

```
<head>
```

```
<meta http-equiv="content-type" content="text/html; charset=UTF-  
8"><title>Example</title><script>
```

```
</head>
```

```
<body>
```

```
It is an example.
```

```
</body>
```

```
%
```

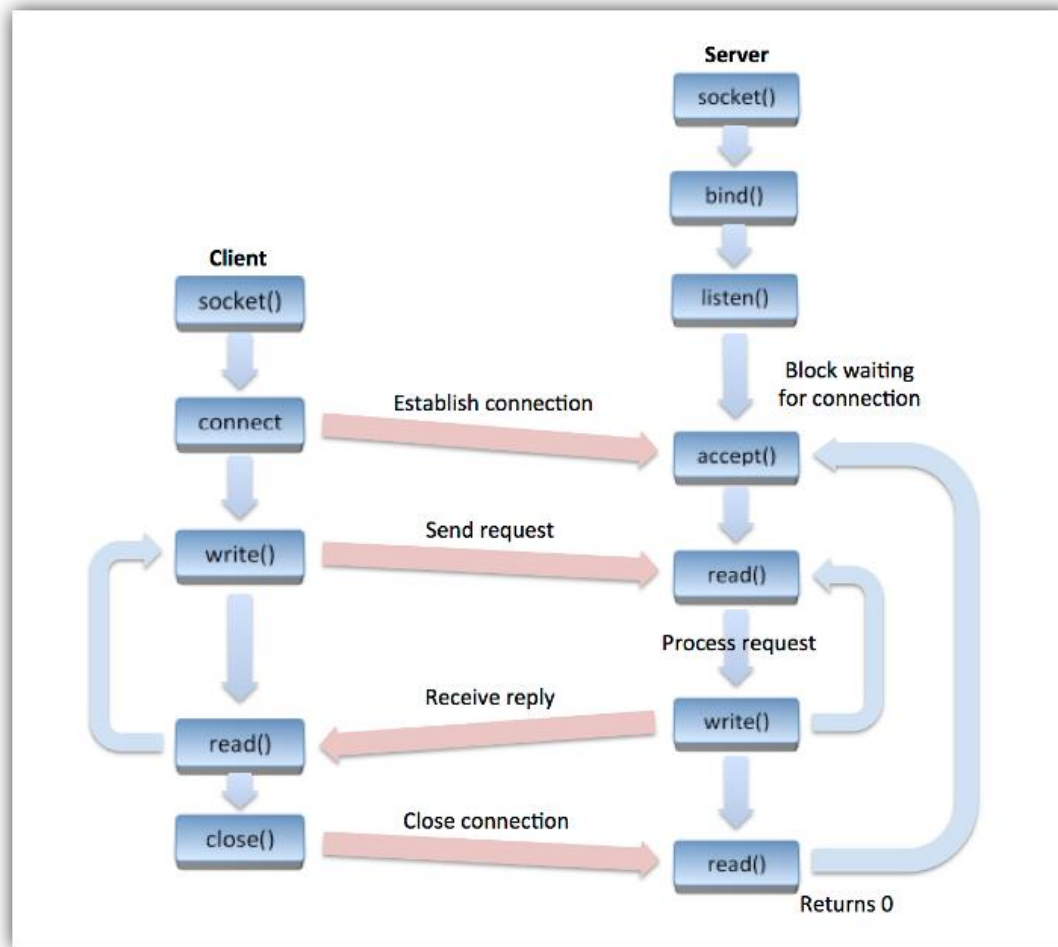
Ưu điểm của URLConnection

- ❖ Có thể lấy thông tin của các trang Web một cách nhanh chóng mà không cần phải lập trình tương tác sử dụng Socket
- ❖ Không cần điều khiển các luồng thông tin như khi lập trình Socket
- ❖ Nâng cao hiệu quả lập trình.



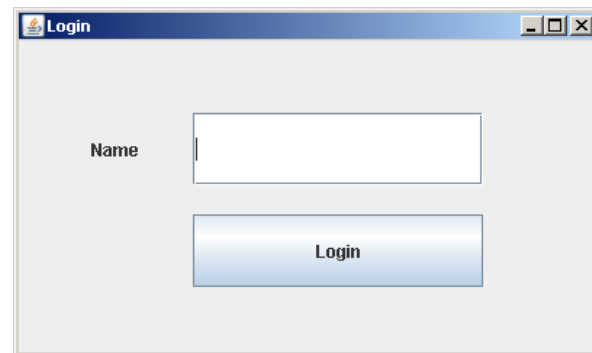
Hướng dẫn tạo Chat Room

Ôn lại lập trình giao thức TCP

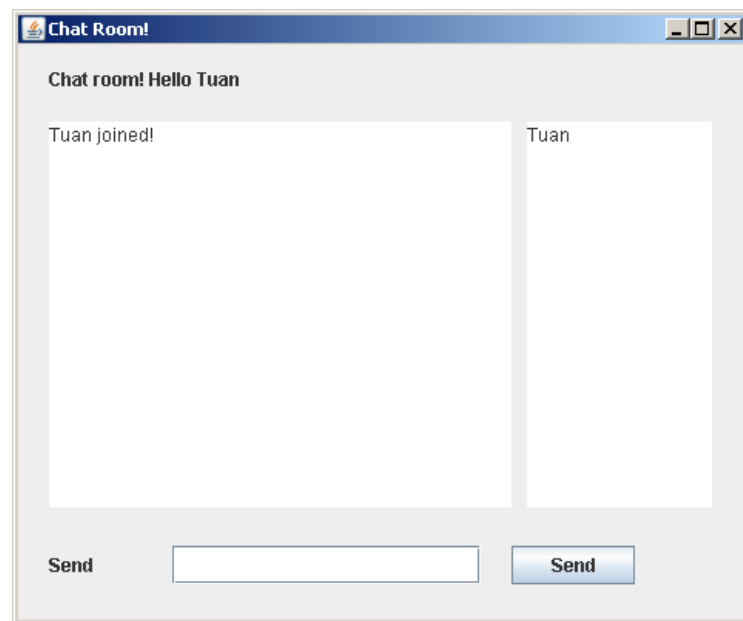


Chuẩn bị giao diện

❖ Giao diện Login



❖ Giao diện Chat room



Giao diện Login

```
import javax.swing.*;
public class LoginFrame {
public JFrame frame;
    public LoginFrame(String ms){
        frame = new JFrame("Login");
        frame.setSize(400 ,300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(null);

        JLabel lname=new JLabel("Name");
        lname.setBounds(50, 50, 50, 50);
        frame.add(lname);

        final JTextField Name=new JTextField("");
        Name.setBounds(120, 50, 200, 50);
        frame.add(Name);
    }
}
```

Giao diện Login (tt)

```
final JLabel msg=new JLabel("Msg:"+ms);  
msg.setBounds(120, 200, 200, 50);  
frame.add(msg);  
  
JButton OK=new JButton("Login");  
OK.setBounds(120, 120, 200, 50);  
frame.add(OK);  
  
frame.setVisible(true);  
}  
public static void main(String[] args){  
    new LoginFrame("");  
}  
}
```

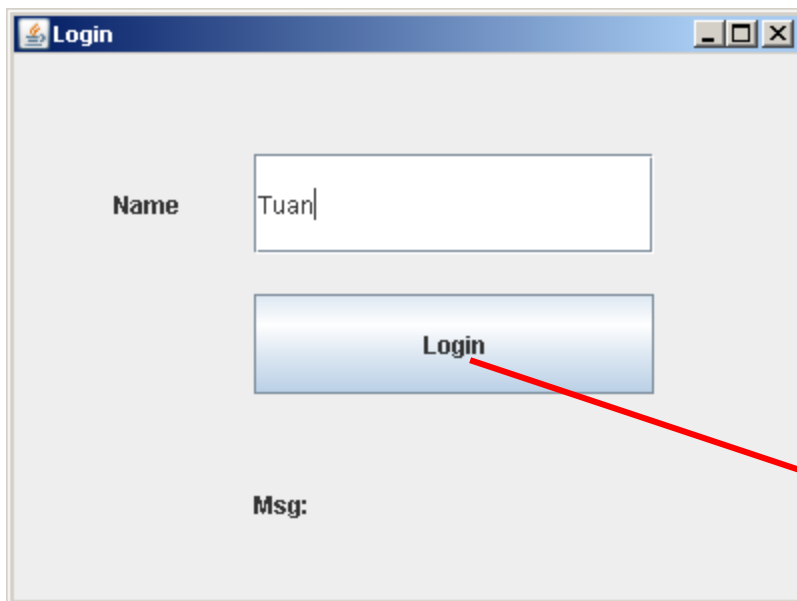
Giao diện Chat room

```
public class ChatRoom {  
    public JFrame frame;  
    public JTextArea Room;  
    public JTextField msg;  
    public JTextArea Joiners;  
    public String NickName;  
    public ChatRoom(String NickName){  
        this.NickName = NickName;  
        this.frame = new JFrame("Chat Room!");  
        this.frame.setSize(480 ,400);  
        this.frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        this.frame.setLayout(null);  
        JLabel lr=new JLabel("Chat room! Hello "+this.NickName);  
        lr.setBounds(20, 10, 300, 25);  
        this.frame.add(lr);  
        this.Room=new JTextArea("");  
        this.Room.setBounds(20, 50, 300, 250);  
        this.Room.setEditable(false);  
        this.frame.add(Room);  
    }  
}
```

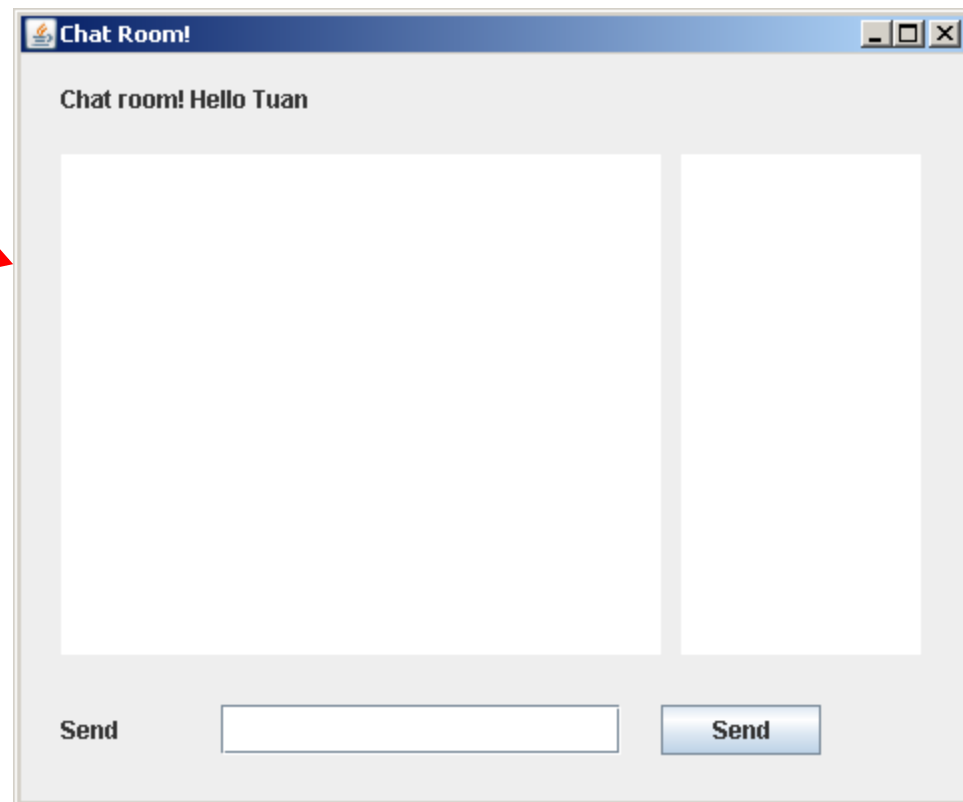
Giao diện Chat room(tt)

```
JLabel lsd=new JLabel("Send");
lsd.setBounds(20, 325, 50, 25);
this.frame.add(lsd);
this.msg=new JTextField("");
this.msg.setBounds(100, 325, 200, 25);
this.frame.add(msg);
JButton OK=new JButton("Send");
OK.setBounds(320, 325, 80, 25);
this.frame.add(OK);
JLabel lj=new JLabel("Joiners");
lj.setBounds(620, 10, 50, 50);
this.frame.add(lj);
this.Joiners = new JTextArea("");
this.Joiners.setBounds(330, 50, 120, 250);
this.Joiners.setEditable(false);
this.frame.add(Joiners);
frame.setVisible(true);
}
}
```

Mở Chat Room trong Login



A screenshot of a Windows-style window titled "Login". It contains a text input field with the name "Tuan" entered. Below the input field is a blue button labeled "Login". At the bottom left, there is a label "Msg:" followed by a large empty text area. A red arrow points from the "Login" button to the "Chat Room!" window on the right.



A screenshot of a Windows-style window titled "Chat Room!". The title bar includes standard minimize, maximize, and close buttons. The main content area displays the text "Chat room! Hello Tuan". Below this text is a large empty rectangular area for chat messages. At the bottom, there is a "Send" label, a text input field, and a blue "Send" button.

Mở Chat Room trong Login

LoginFrame.java

Thêm vào

```
//Lược
OK.setBounds(120, 120, 200, 50);
OK.addActionListener(new ActionListener(){
    @Override
    public void actionPerformed(ActionEvent arg0) {
        if (!Name.getText().equals("")){
            new ChatRoom(Name.getText());
            frame.dispose();
        }
        else msg.setText("Msg: Please input your name!");
    }
});
frame.add(OK);
//Lược
```

Tạo TCP Server

```
public class ChatRoomServer {  
    public final static int daytimePort = 5000;  
    public ChatRoomServer(){  
        ServerSocket theServer;  
        Socket theConnection;  
        try {  
            theServer = new ServerSocket(daytimePort);  
            while (true) {  
                theConnection = theServer.accept();  
                System.out.println("Have Connection!");  
                new ThreadedHandler(this,theConnection).start();  
            }  
        } catch (IOException e) {  
            System.err.println(e);  
        }  
    }  
    public static void main(String[] args) {new ChatRoomServer();}
```

Tạo TCP Server (tt)

```
public class ThreadedHandler extends Thread{
    ChatRoomServer crsv;
    public Socket incoming;
    public DataInputStream dis;
    public DataOutputStream dos;
    public ThreadedHandler(ChatRoomServer crsv, Socket i)
    {
        this.crsv=crsv;
        this.incoming=i;
        try{
            this.dis = new DataInputStream(
                                incoming.getInputStream());
            this.dos = new DataOutputStream(
                                incoming.getOutputStream());
        }catch(IOException e){}
    }
    public void run(){} //Chưa thực thi
}
```

Tạo Socket kết nối với Server

Khai báo trong lớp ChatRoom

```
public Socket soc;  
public DataInputStream dis;  
public DataOutputStream dos;
```

Tạo Socket kết nối với Server vào cuối

```
public ChatRoom(String NickName){  
    //lược
```

```
    try{  
        soc = new Socket("localhost", 5000);  
        this.dis = new DataInputStream(soc.getInputStream());  
        this.dos = new DataOutputStream(soc.getOutputStream());  
    }catch(IOException e){this.frame.dispose();}  
}
```

Cấu trúc thông điệp của Chat Room

Command , Message

Mục Đích: Phân biệt các thông điệp của client và server

Chuyển các thông tin cần thiết ứng với các thông điệp.

Cấu trúc thông điệp

❖ Phía Client

■ Nhận:

- “Msg,”+[Tin nhắn] : Nhận tin nhắn từ Server
- “Jnr,”+ [Tất cả tên User] : Nhận tất cả tên user có trong room từ Server

■ Gửi

- “Join”+[Tên]: Gửi cho Server tên đăng nhập
- “Msg,”+[Tin nhắn] : Gửi cho Server tin nhắn

Cấu trúc thông điệp

❖ Phía Server

■ Gửi:

- “Msg,”+[Tin nhắn] : Gửi cho Client tin nhắn
- “Jnr,”+ [Tất cả tên User] : Gửi tất cả tên user có trong room đến Client

■ Nhận:

- “Join”+[Tên]: Nhận tên đăng nhập từ Client
- “Msg,”+[Tin nhắn] : Nhận tin nhắn từ Client

Hoàn thiện chương trình

❖ Phía Server

- Tạo một Vector trong ChatRoomServer quản lý các luồng kết nối.

```
public Vector<ThreadedHandler> cls=new Vector<ThreadedHandler>();
```

- Trong luồng kết nối (lớp ThreadedHandler)
 - Quản lý tên User bằng cách Khai báo tên

```
public String name;
```
 - Xử lý đăng nhập trong run(){}

Xử lý đăng nhập trong run(){}

```
String ch="";  
try{  
    ch = dis.readUTF();  
    String cmd=ch.substring(0, ch.indexOf(","));  
    String msg=ch.substring(ch.indexOf(",")+1);  
    if (!cmd.equals("Join")) incoming.close();  
        System.out.println("Hello "+msg);  
    this.name=msg;  
    this.crsv.cls.add(this);  
  
    //Nhận và gửi thông điệp  
}catch(IOException e){  
    crsv.cls.remove(this);  
}
```

Nhận và gửi đoạn chat

```
while (true)
{
    ch = dis.readUTF();
    cmd=ch.substring(0, ch.indexOf(","));
    msg=ch.substring(ch.indexOf(",")+1);
    if (cmd.equals("Msg")) {
        for (int i=0;i<this.crsv.cls.size();i++){
            ThreadedHandler temp=this.crsv.cls.get(i);
            if (temp!=this){
                temp.dos.writeUTF("Msg,"+this.name+">>" +msg);
            }
        }
    }
    else{
        incoming.close();
        this.crsv.cls.remove(this);
    }
}
```

Hoàn thiện chương trình (Client)

❖ Phía Client

- Tạo luồng nhận thông điệp từ Server

```
public class ThreadedHandler extends Thread{  
    ChatRoom cr;  
    public ThreadedHandler(ChatRoom cr){  
        this.cr=cr;  
    }  
    public void run(){  
        String ch="";  
        try{
```

```
while (true){
    ch = dis.readUTF();
    String cmd=ch.substring(0, ch.indexOf(","));
    String msg=ch.substring(ch.indexOf(",")+1);
    if (cmd.equals("Msg"))
        this.cr.Room.setText(msg+"\n"+cr.Room.getText());
    else
        this.cr.soc.close();
}
} catch (IOException e){cr.frame.dispose();new LoginFrame();}
}
```

Khởi tạo luồng nhận thông điệp từ Server và Tham gia vào Room

```
public ChatRoom(String NickName){  
    //lược  
    try{  
        soc = new Socket("localhost", 5000);  
        this.dis = new DataInputStream(soc.getInputStream());  
        this.dos = new DataOutputStream(soc.getOutputStream());  
        new ThreadedHandler(this).start();  
        this.dos.writeUTF("Join,"+this.NickName);  
    }catch(IOException e){this.frame.dispose();}  
}
```

Tạo ActionListener cho Client

```
public class SendActionListener implements ActionListener{
    ChatRoom cr;
    public SendActionListener(ChatRoom cr){
        this.cr = cr;
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if (!cr.msg.getText().equals("")){
            cr.Room.setText(cr.NickName+">"+
                cr.msg.getText()+"\n"+cr.Room.getText());
            try{
                this.cr.dos.writeUTF("Msg,"+cr.msg.getText());
            }catch(IOException e1){
                cr.frame.dispose();new LoginFrame();}
            cr.msg.setText("");
        }
    }
}
```



Bài tập

❖ Thêm vào chương trình chat room các chức năng sau:

- Khi có một người đăng nhập hay thoát ra thì:
 - Server thông báo cho tất cả Clients biết và cập nhật tên của tất cả Users
 - Clients hiển thị tên các Users bên phải Frame



Lập trình truy xuất Cơ sở dữ liệu

Lập trình truy xuất cơ sở dữ liệu

❖ Java cung cấp thư viện java.sql gồm:

- Các lớp
- Interface

Cho phép chương trình kết nối với các cơ sở dữ liệu để truy xuất và xử lý dữ liệu.

❖ Các hệ cơ sở dữ liệu như

- MySQL Server, Access, Oracle,...

Được xem như các chương trình server lưu trữ dữ liệu.

Lập trình truy xuất cơ sở dữ liệu

- ❖ Để kết nối và trao đổi dữ liệu với các CSDL, chương trình cần nạp Driver tương ứng. Driver có nhiệm vụ:
 - Chuyển đổi các yêu cầu đến CSDL từ chương trình thành định dạng mà hệ CSDL hiểu được
 - Chuyển đổi dữ liệu lên chương trình

Các bước kết nối và truy xuất

- ❖ Nạp Driver tương ứng với hệ CSDL
- ❖ Thực hiện kết nối đến CSDL
- ❖ Trao đổi dữ liệu với CSDL
 - executeQuery()
 - executeUpdate()
 - execute()
 -

Ví dụ về cài đặt hệ CSDL

❖ Cài đặt hệ CSDL MySQL Server

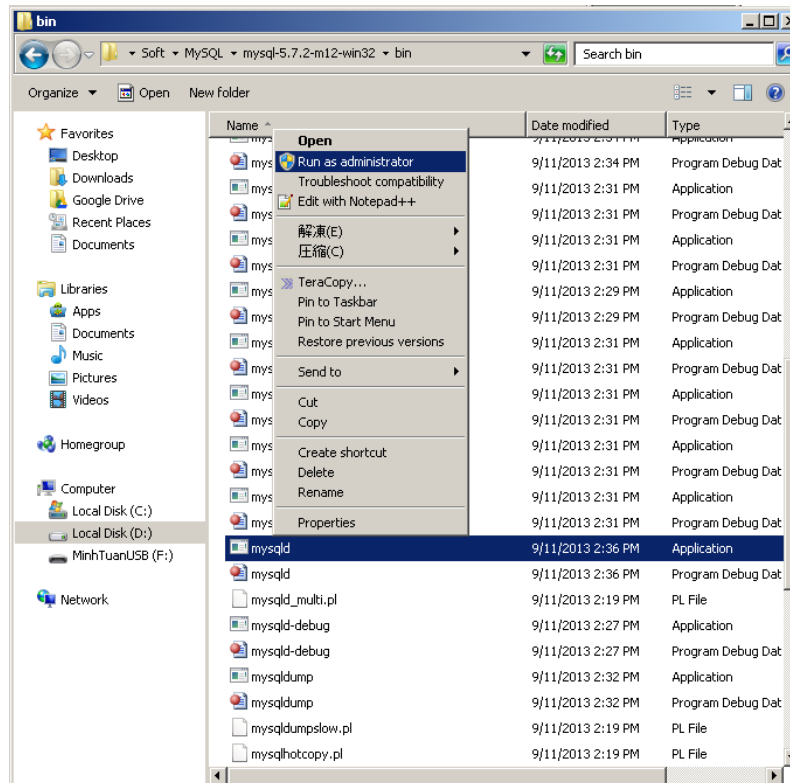
- Tải và cài đặt MySQL Community Server
 - <http://downloads.mysql.com/archives/community/>
- Tải và cài đặt MySQL Query Browser
 - <http://downloads.mysql.com/archives/query/>



Khởi động và cấu hình

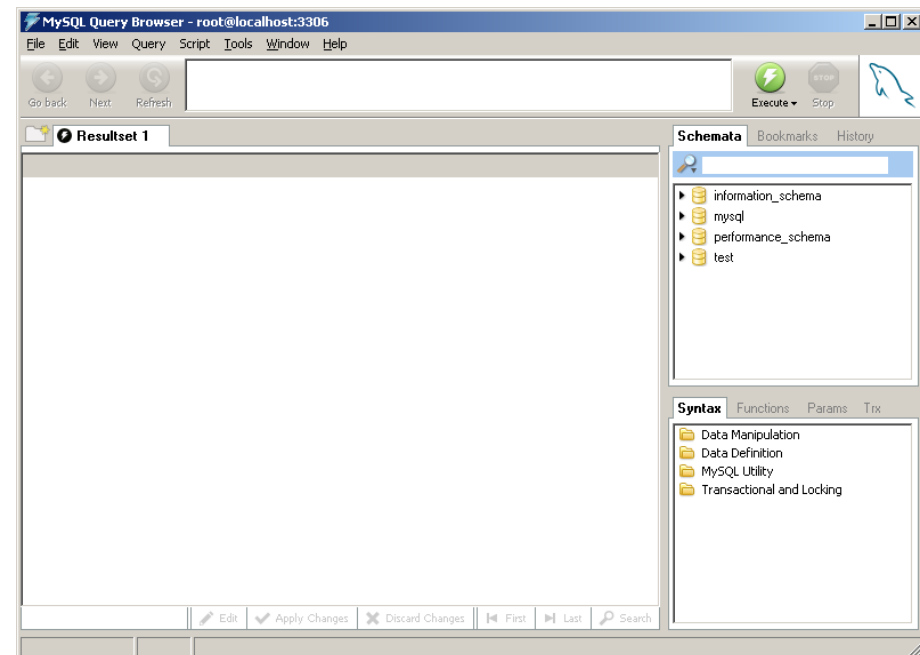
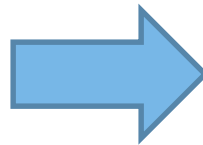
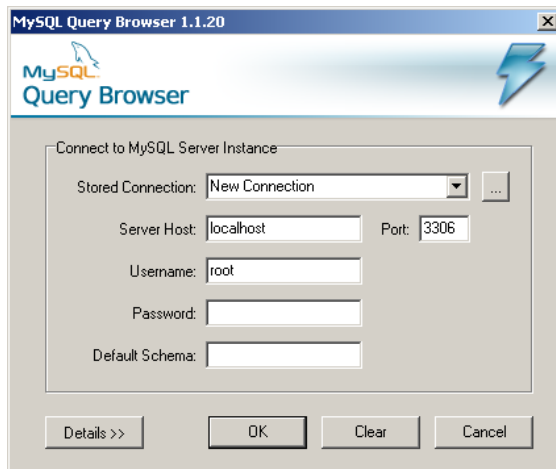
❖ Khởi động MySQL Server

- Chạy file mysqld.exe với chế độ administrator



Khởi động và cấu hình (tt)

❖ Khởi động thiết lập MySQL Query Browser

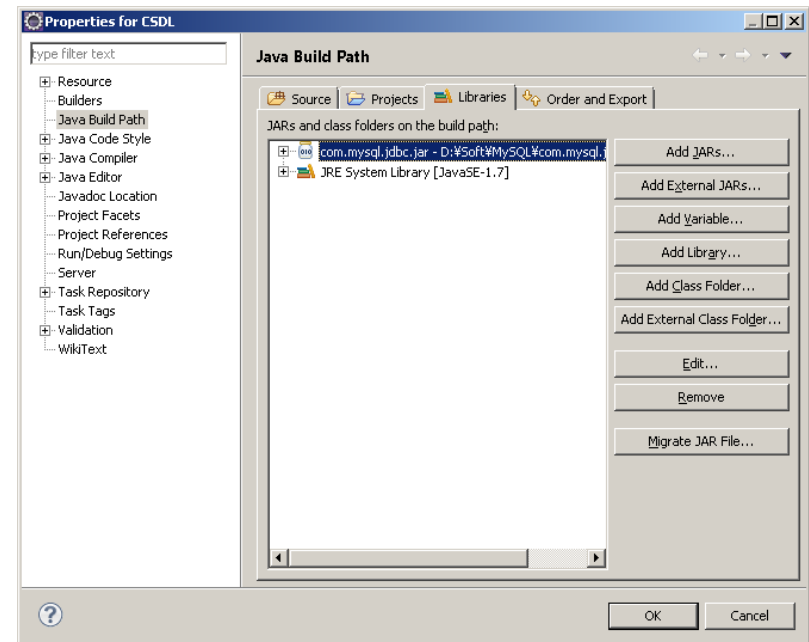


Nạp driver

❖ Tải driver mysql: com.mysql.jdbc.jar

- <http://www.java2s.com/Code/Jar/c/Downloadcommysqljdbc515jar.htm>

❖ Nạp driver vào Eclipse



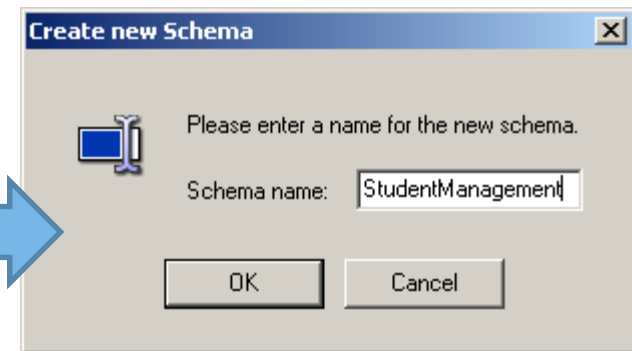
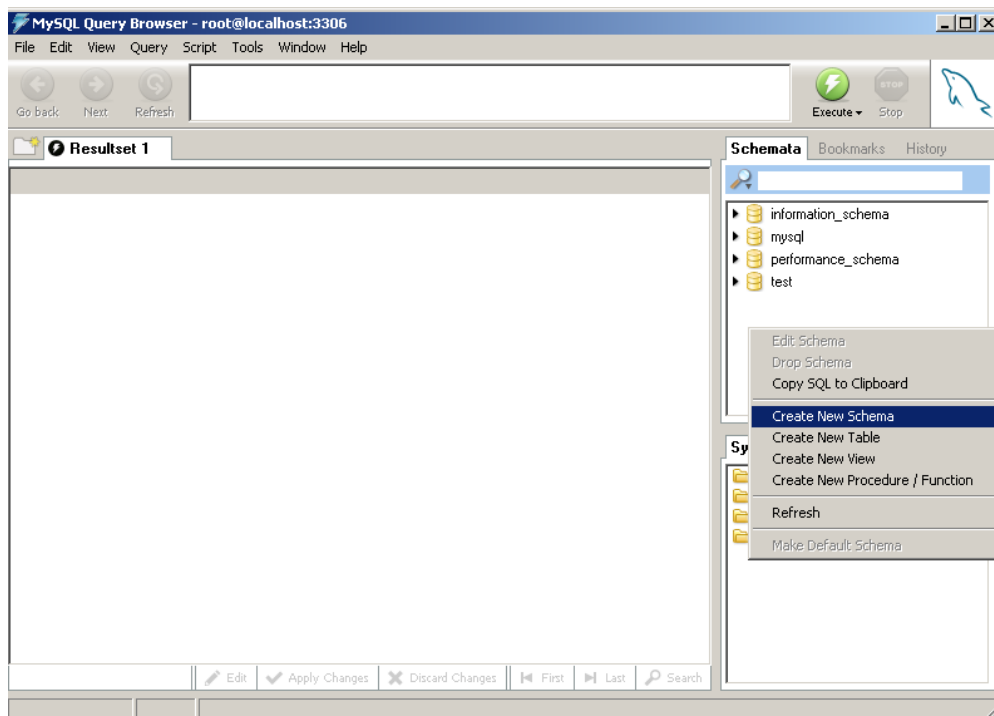
Bài tập CSDL

- ❖ Tạo cơ sở dữ liệu có tên
 - StudentManagement
- ❖ Tạo bảng Students gồm các trường
 - ID, Name, Math, Phys, Chem và Aver
- ❖ Viết chương trình Java kết nối đến CSDL, truy vấn và hiển thị tên cột của bảng Students

Đáp án

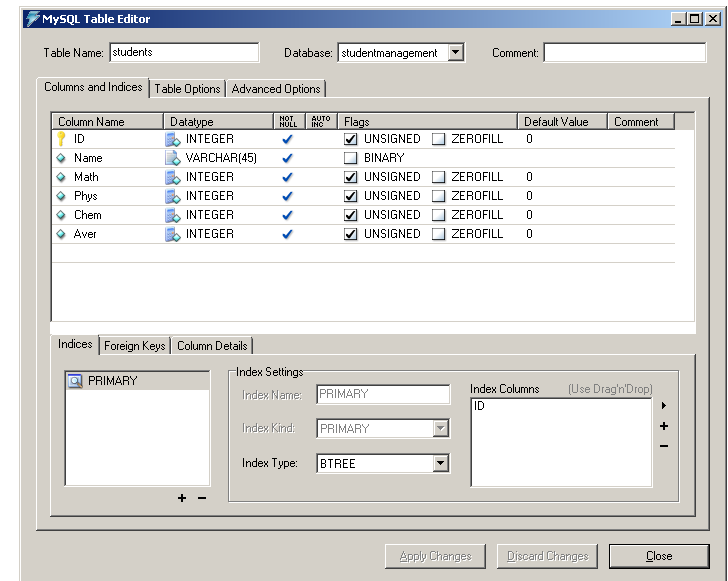
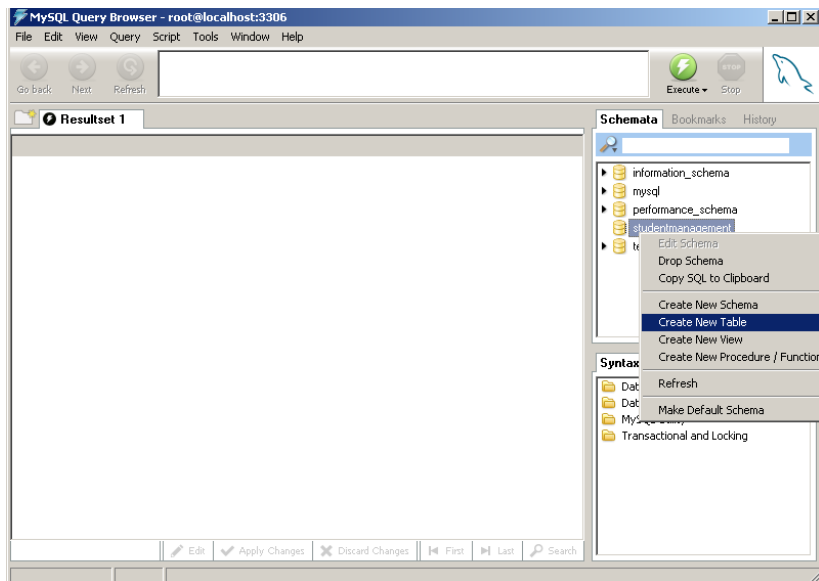
❖ Mở MySQL Query Browser

■ Tạo Schema



Tiếp theo

❖ Tạo Bảng Students



Chương trình Java

```
import java.sql.*;
public class Ketnoi {
    public static void main(String [] args){
        try{
            //Nap Driver
            Class.forName("com.mysql.jdbc.Driver");
            //Ket noi toi CSDL
            Connection conn = DriverManager.getConnection("jdbc:mysql:" +
                "localhost:3306/studentmanagement","root","");
            Statement sm=conn.createStatement();
            //Truy van
            ResultSet rs = sm.executeQuery("Select * from students");
            //Lay thong tin
            ResultSetMetaData rsm=rs.getMetaData();
            int cn=rsm.getColumnCount();
            for (int i=1;i<=cn;i++)
                System.out.print(rsm.getColumnLabel(i)+" ");
        }catch(Exception e){}
    }
}
```

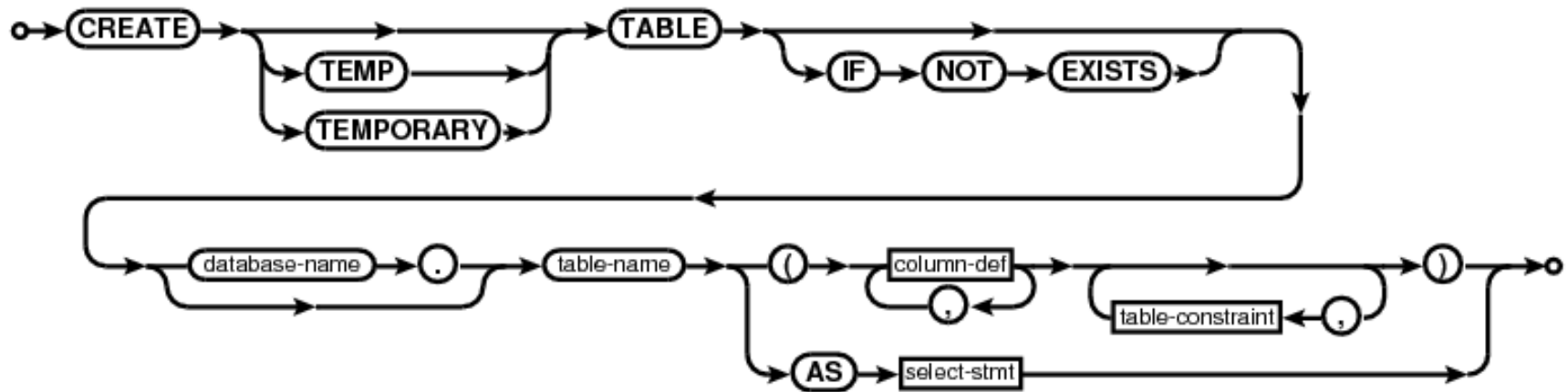


Bài Tập 2

- ❖ Xây dựng chương trình thêm, xóa, sửa thông tin bảng Students và truy vấn thông tin từ bảng để hiển thị ra màn hình.

Các lệnh truy xuất thường gặp

Create

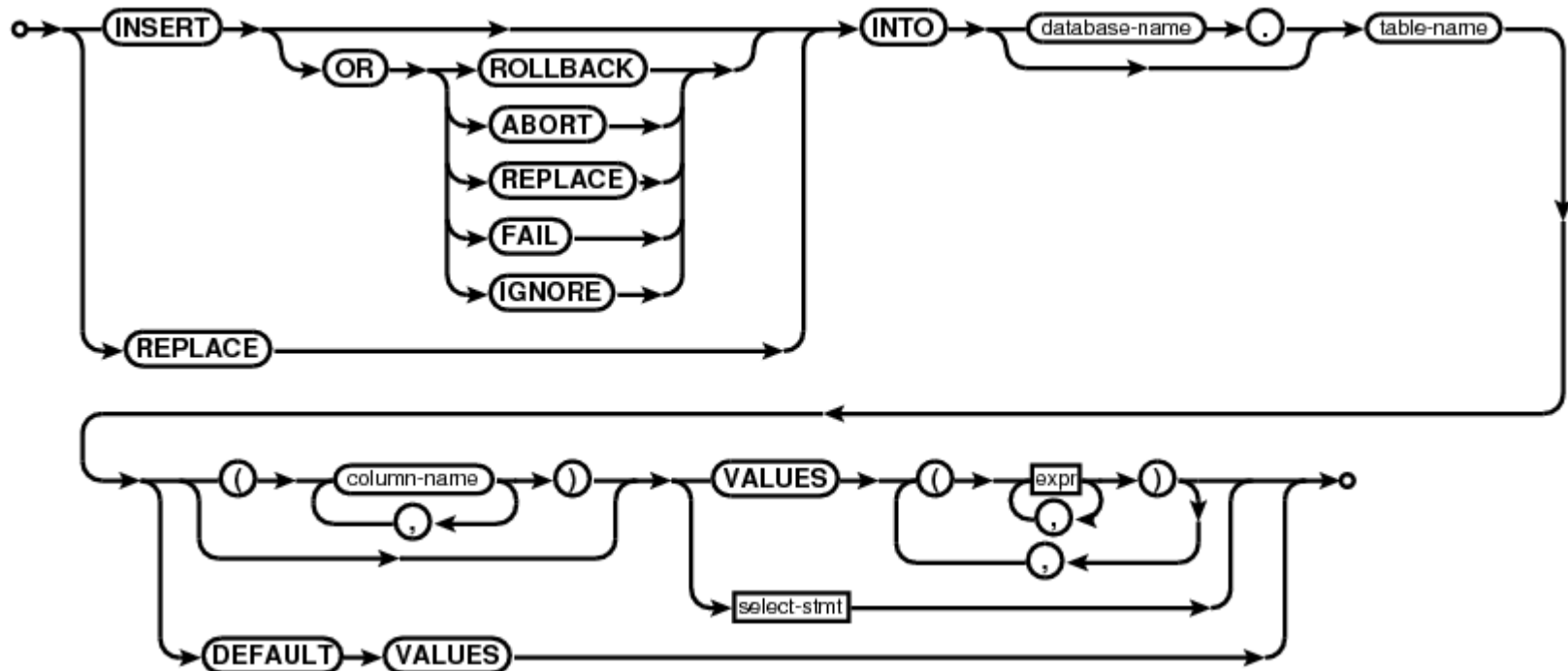


Delete



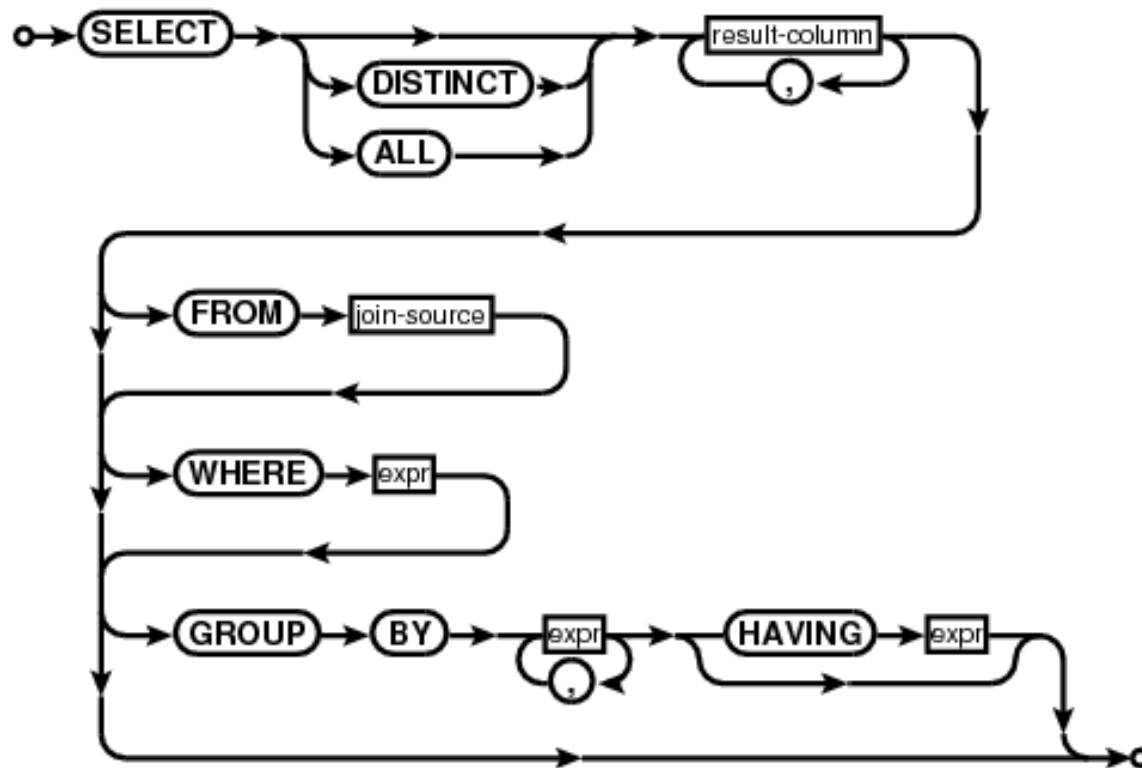
Các lệnh truy xuất thường gặp

Insert



Các lệnh truy xuất thường gặp

Select



Đáp Án

```
import java.sql.*;
import java.util.*;
public class Ketnoi {
public static void main(String [] args){
Scanner sc=new Scanner(System.in);
try{
    //Nap Driver
    Class.forName("com.mysql.jdbc.Driver");
    //Ket noi toi CSDL
    Connection conn = DriverManager.getConnection("jdbc:mysql:" +
        "///localhost:3306/studentmanagement","root","");
    Statement sm=conn.createStatement();
    lap:while(true){
        //Truy van
        ResultSet rs = sm.executeQuery("Select * from students");
        //Lay thong tin
        ResultSetMetaData rsm=rs.getMetaData();
        int cn=rsm.getColumnCount();
        for (int i=1;i<=cn;i++)
            System.out.print(rsm.getColumnLabel(i)+"\t");
        System.out.println("");
    }
}
```



```
while(rs.next()){
    for (int i=1;i<=cn;i++)
        System.out.print(rs.getString(i)+"\t");
        System.out.println("");
}
System.out.println("1. Them....2.Xoa....3.Sua....4.Ketthuc");
int chon=sc.nextInt();
int id;
String name;
int math,phys,chem,avr;
switch(chon){
    case 1:
        System.out.print("Nhap ID:");id=sc.nextInt();
        System.out.print("Nhap ten:");name=sc.next();
        System.out.print("Nhap diem toan:");math=sc.nextInt();
        System.out.print("Nhap diem ly:");phys=sc.nextInt();
        System.out.print("Nhap diem hoa:");chem=sc.nextInt();
        avr=(math+phys+chem)/3;
        try{
            sm.execute("Insert into students values("+id+",\""+
                +name+"\", "+math+", "+phys+", "+chem+", "+avr+"");
        }catch(Exception e){System.out.println("Error!");}
        break;
```

case 2:

```
System.out.print("Nhap ID:");id=sc.nextInt();
try{
    sm.execute("Delete from students where id="+id);
}catch(Exception e){System.out.println("Error!");}
break;
```

case 3:

```
System.out.print("Nhap ID:");id=sc.nextInt();
System.out.print("Nhap ten:");name=sc.next();
System.out.print("Nhap diem toan:");math=sc.nextInt();
System.out.print("Nhap diem ly:");phys=sc.nextInt();
System.out.print("Nhap diem hoa:");chem=sc.nextInt();
avr=(math+phys+chem)/3;
try{
    sm.execute("Update students set Name = \"
        +name+\"\",Math = "+math+",Phys="
        +phys+",Chem="+chem+",Aver="
        +avr+" where ID="+id);
}catch(Exception e){System.out.println("Error!");}
break;
```

case 4:break lap;

default:break;

```
}
}
}
}catch(Exception e){}
}
}
```

Bài Tập 3

❖ Xây dựng chương trình theo mô hình Client/Server như sau:

- Server
 - Chứa CSDL
 - Nhận các câu truy vấn từ Client và trả kết quả cho Client
- Client
 - Truy vấn tới Server
 - Hiển thị kết quả Server trả về



Bài tập 4

- ❖ Mô hình Client/Server ở bài 3 có gì không ổn?
- ❖ Hãy chỉ ra chỗ không ổn và nêu hướng giải quyết.
- ❖ Lập Trình