Cơ sở dữ liệu
(Database)

Giảng viên: ThS **Lê Văn Tấn**
Khoa Công nghệ thông tin - Đại học Vinh

Các nội dung chính

- Chương 1: TỔNG QUAN VỀ CƠ SỞ DỮ LIỆU
- Chương 2: MÔ HÌNH DỮ LIỆU QUAN HỆ
- Chương 3: NGÔN NGỮ ĐỊNH NGHĨA VÀ THAO TÁC DỮ LIỆU
- Chương 4: LÝ THUYẾT THIẾT KẾ CƠ SỞ DỮ LIỆU QUAN HỆ
- Chương 5: TỐI ƯU HOÁ CÂU HỎI
- Chương 6: QUẢN LÝ GIAO DỊCH VÀ ĐIỀU KHIỂN TƯƠNG TRANH
- Chương 7: AN TOÀN VÀ TOÀN VẸN DỮ LIỆU

Tài liệu tham khảo

[1]. Nguyễn Kim Anh. ***Nguyên lý của các hệ cơ sở dữ liệu***, Nhà xuất bản Đại học Quốc gia Hà nội, 2004.

[2]. Lê Văn Tấn, ***Bài giảng cơ sở dữ liệu***

[3]. Phạm Hữu Khang, ***Lập trình ứng dụng chuyên nghiệp - SQL Server 2000*** - tập 1, Nhà xuất bản giáo dục, 2002.

[4]. Lê Tiến Vương, ***Nhập môn cơ sở dữ liệu quan hệ***, Nhà xuất bản Khoa học và Kỹ thuật, 1996.

[5]. Rob-Coronel, ***Database Systems – Design, Implementation & Management***, 2000.

Chương 1: TỔNG QUAN VỀ CƠ SỞ DỮ LIỆU

- **CÁC KHÁI NIỆM CƠ BẢN**
  - Dữ liệu và hệ thống tệp cổ điển
  - Cơ sở dữ liệu (Database)
  - Vai trò của cơ sở dữ liệu
  - Hệ quản trị cơ sở dữ liệu (DataBase Management System - DBMS)
  - Hệ cơ sở dữ liệu (Database System)
- **CÁC MÔ HÌNH DỮ LIỆU**
  - Mô hình hoá trong tin học
  - Các mô hình dữ liệu
    - Sơ đồ thực thể liên kết
    - Mô hình dữ liệu quan hệ (Relational data model)
    - Mô hình dữ liệu mạng (Network data model)
    - Mô hình dữ liệu phân cấp
    - Mô hình dữ liệu hướng đối tượng (Object oriented data model)

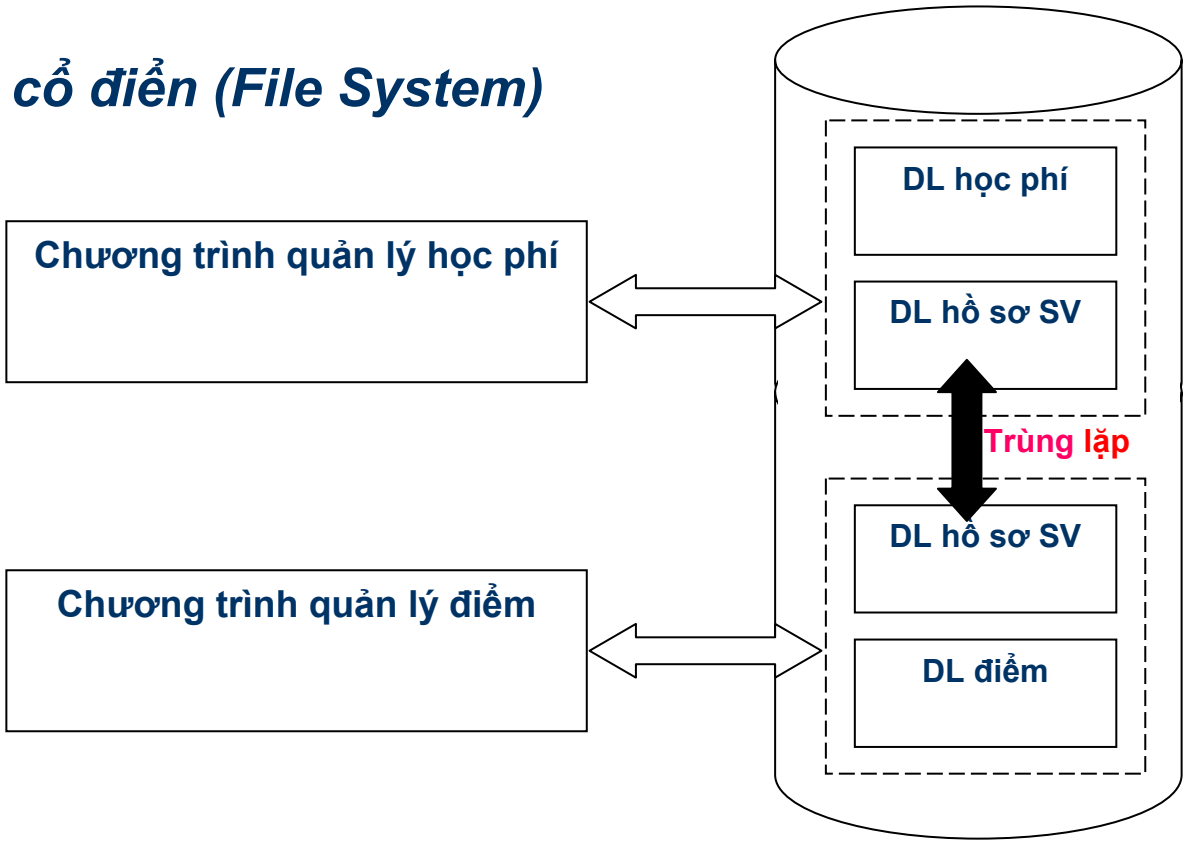
I. CÁC KHÁI NIỆM CƠ BẢN

1.1. Dữ liệu và hệ thống tệp cổ điển

- ***Dữ liệu (Data)***

  Dữ liệu là tất cả những gì máy tính xử lý, nó có thể là các số, chữ cái hay các giá trị logic,..

- ***Hệ thống cổ điển (File System)***



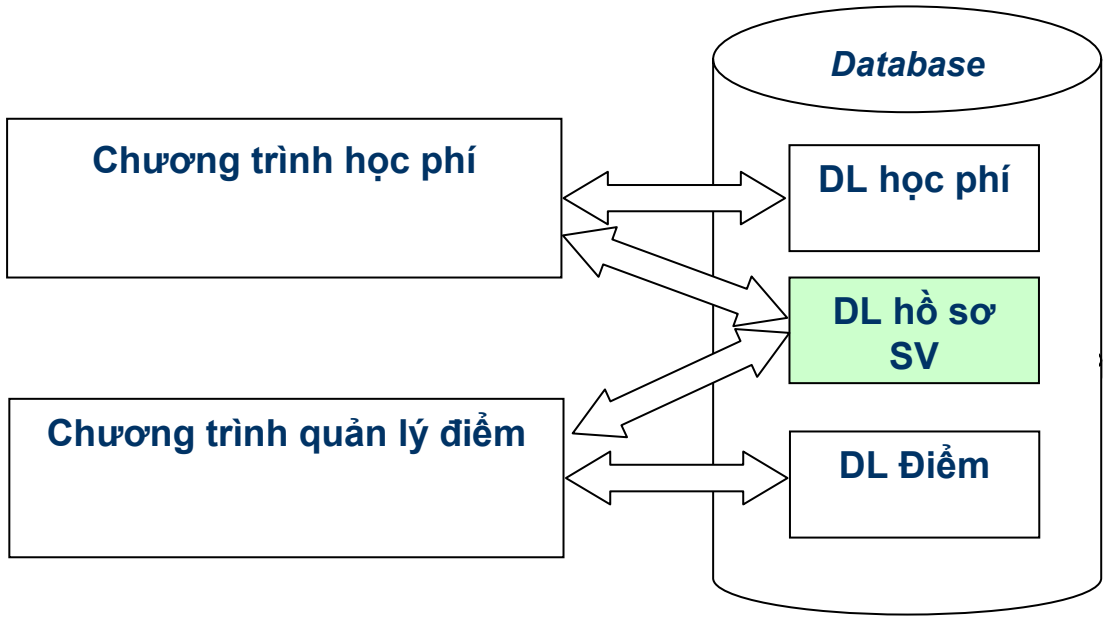
1.1. Dữ liệu và hệ thống tệp cổ điển (Tiếp)

Hạn chế của hệ thống tệp:

- Các tệp được tạo ra cho mỗi một hệ ứng dụng → dữ liệu được lưu trữ ở mỗi hệ, nên bộ nhớ (ngoài) bị lãng phí.
- Khi những dữ liệu được thay đổi độc lập ở một hệ → sự không nhất quán của dữ liệu.
- Định nghĩa tệp nằm trong chương trình → thay đổi dữ liệu thì phải thay đổi chương trình.

1.1. Dữ liệu và hệ thống tệp cổ điển (Tiếp)

➡ Tổ chức dữ liệu thành cơ sở dữ liệu



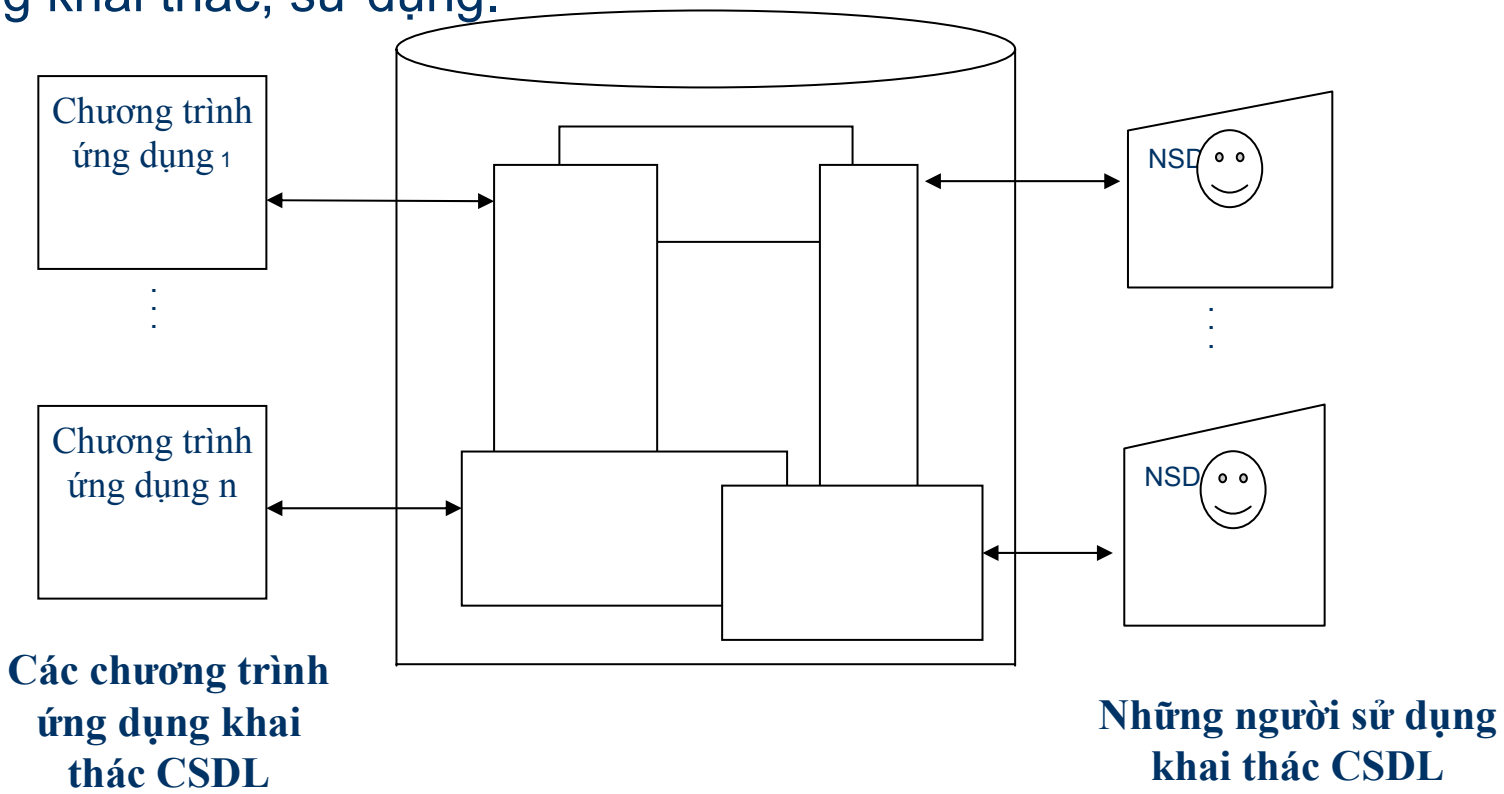
Mục đích:

- Chia sẻ dữ liệu (không dư thừa)
- Dữ liệu được độc lập với chương trình
- Toàn vẹn dữ liệu và khôi phục dữ liệu
- An toàn dữ liệu

I. CÁC KHÁI NIỆM CƠ BẢN (tiếp)

1.2. Cơ sở dữ liệu (DataBase)

Cơ sở dữ liệu là một tập các dữ liệu về các đối tượng cần quản lý, được tổ chức theo một cơ chế thống nhất, được lưu trữ đồng thời trên các vật mang tin của máy tính điện tử, cho phép đồng thời nhiều người cùng khai thác, sử dụng.



1.2. Cơ sở dữ liệu (DataBase)

Các mức của cơ sở dữ liệu:

- Cơ sở dữ liệu mức vật lý

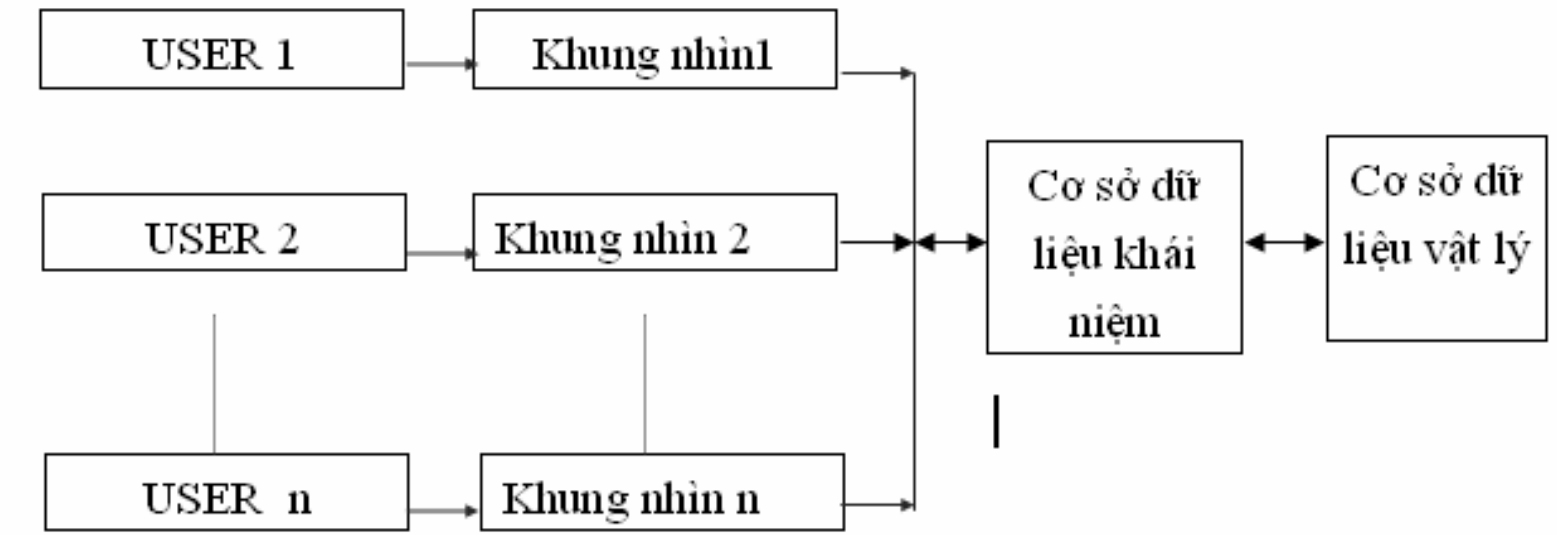
  Bao gồm các tệp dữ liệu trên bộ nhớ thứ cấp.

- Cơ sở dữ liệu mức khái niệm

  Là sự biểu diễn trừu tượng của cơ sở dữ liệu vật lý, nói cách khác cơ sở dữ liệu vật lý là sự cài đặt cụ thể của cơ sở dữ liệu khái niệm.

- Khung nhìn (View)

  Là cách nhìn, quan niệm của từng người sử dụng đối với cơ sở dữ liệu k.niệm.



1.2. Cơ sở dữ liệu (DataBase)

Thể hiện (Instance)

Là dữ liệu mô tả đối tượng hiện có trong cơ sở dữ liệu.

Lược đồ (Scheme)

Lược đồ cơ sở dữ liệu là "bộ khung" của cơ sở dữ liệu. Chẳng hạn, khi xét một cơ sở dữ liệu vật lý được cài đặt trong Hệ quản trị cơ sở dữ liệu trong FOXPRO, lược đồ bao gồm tập các cấu trúc của các tệp DBF cùng với các mối quan hệ giữa chúng. Các mức của lược đồ cơ sở dữ liệu:

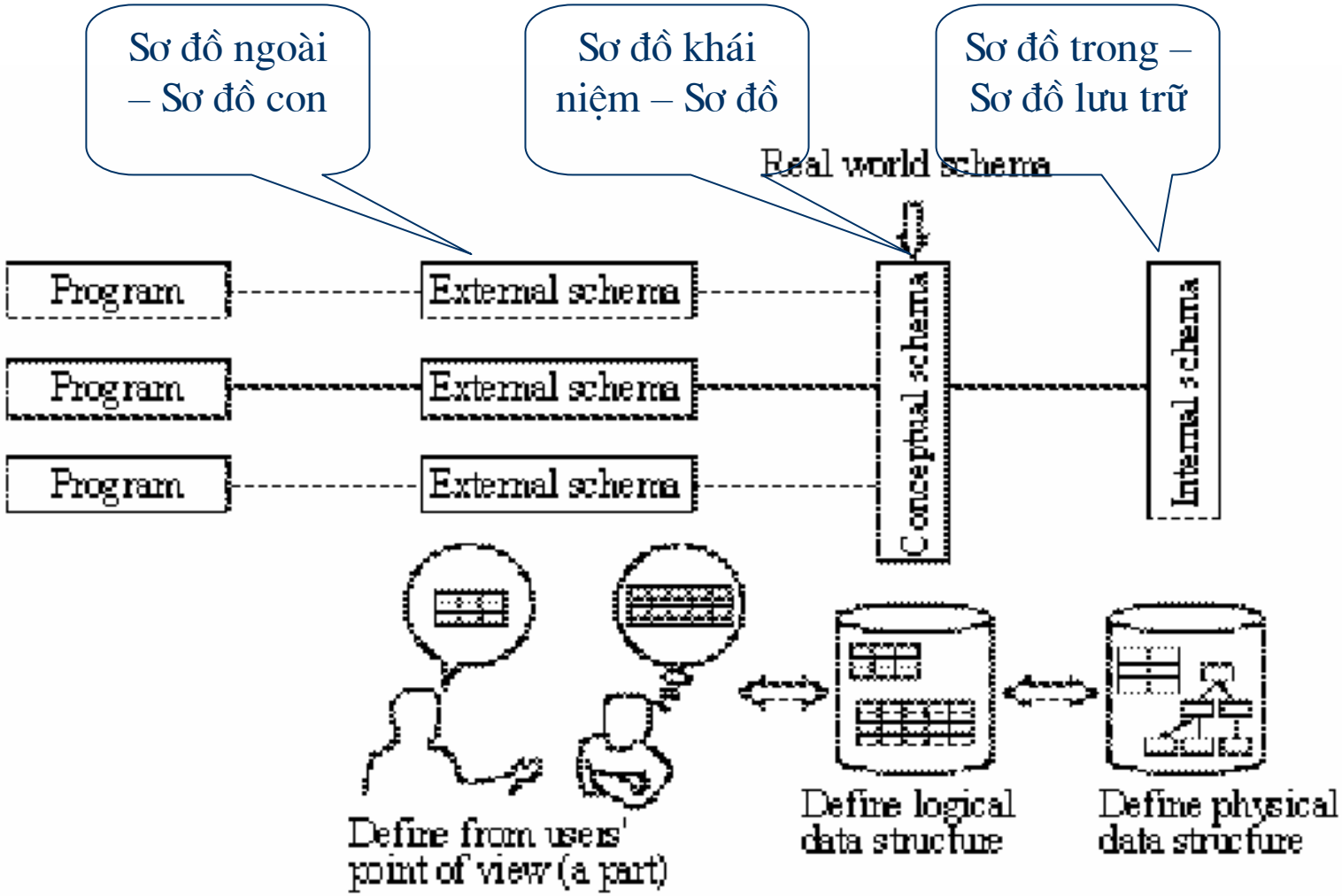
- Lược đồ khái niệm - Sơ đồ (CONCEPTUAL SCHEMA)

  Xác định logic dữ liệu của thế giới thực để hệ thống máy tính xử lý.
- Lược đồ ngoài – Sơ đồ con (EXTERNAL SCHEMA)

  Xác định dữ liệu theo quan điểm của chương trình sử dụng CSDL, như 1 phần của cấu trúc dữ liệu đã được xác định bởi lược đồ **Khái niệm**
- Lược đồ trong – Sơ đồ Lưu trữ (INTERNAL SCHEMA)

  Xác định cách lưu trữ Vật lý của CSDL đã được xác định bởi lược đồ **Khái niệm**

1.2. Cơ sở dữ liệu (DataBase)



I. CÁC KHÁI NIỆM CƠ BẢN (tiếp)

1.3. Vai trò của cơ sở dữ liệu

- Tổ chức thông tin trong các bài toán khoa học kỹ thuật.
- Làm kho dữ liệu trong hệ thống thông tin quản lý.
- Tổ chức dữ liệu có cấu trúc phức tạp như các dữ liệu địa lý, thủy văn, môi trường,... đòi hỏi khả năng lưu trữ lớn và tốc độ cao.
- Ứng dụng trong các hệ thống hỗ trợ công nghiệp, giảng dạy, đặc biệt là trong lĩnh vực hệ chuyên gia, người máy.
- Ứng dụng trong hệ thống đa phương tiện, xử lý tri thức (mối quan hệ giữa các sự kiện, hệ thống câu hỏi).

1.3. Vai trò của cơ sở dữ liệu (tiếp)

Một số vấn đề cần quan tâm khi tổ chức dữ liệu thành CSDL

- Việc đảm bảo sự an toàn của dữ liệu.
- Tính chính xác của dữ liệu.
- Phải có cơ chế bảo mật và phân quyền.
- Vấn đề xử lý tương tranh.
- Khả năng phục hồi dữ liệu khi cần thiết.
- Phải có hệ quản trị cơ sở dữ liệu để tổ chức và khai thác cơ sở dữ liệu.

I. CÁC KHÁI NIỆM CƠ BẢN (tiếp)

1.4. Hệ quản trị cơ sở dữ liệu (DataBase Management System - DBMS)

Hệ quản trị cơ sở dữ liệu là một hệ thống phần mềm giúp người sử dụng khai thác cơ sở dữ liệu một cách có hiệu quả.

Một số hệ quản trị CSDL trên thị trường

Microsoft Office Access 2003

Microsoft SQL Server 2005 Developer Edition

FOXPROW

Oracle

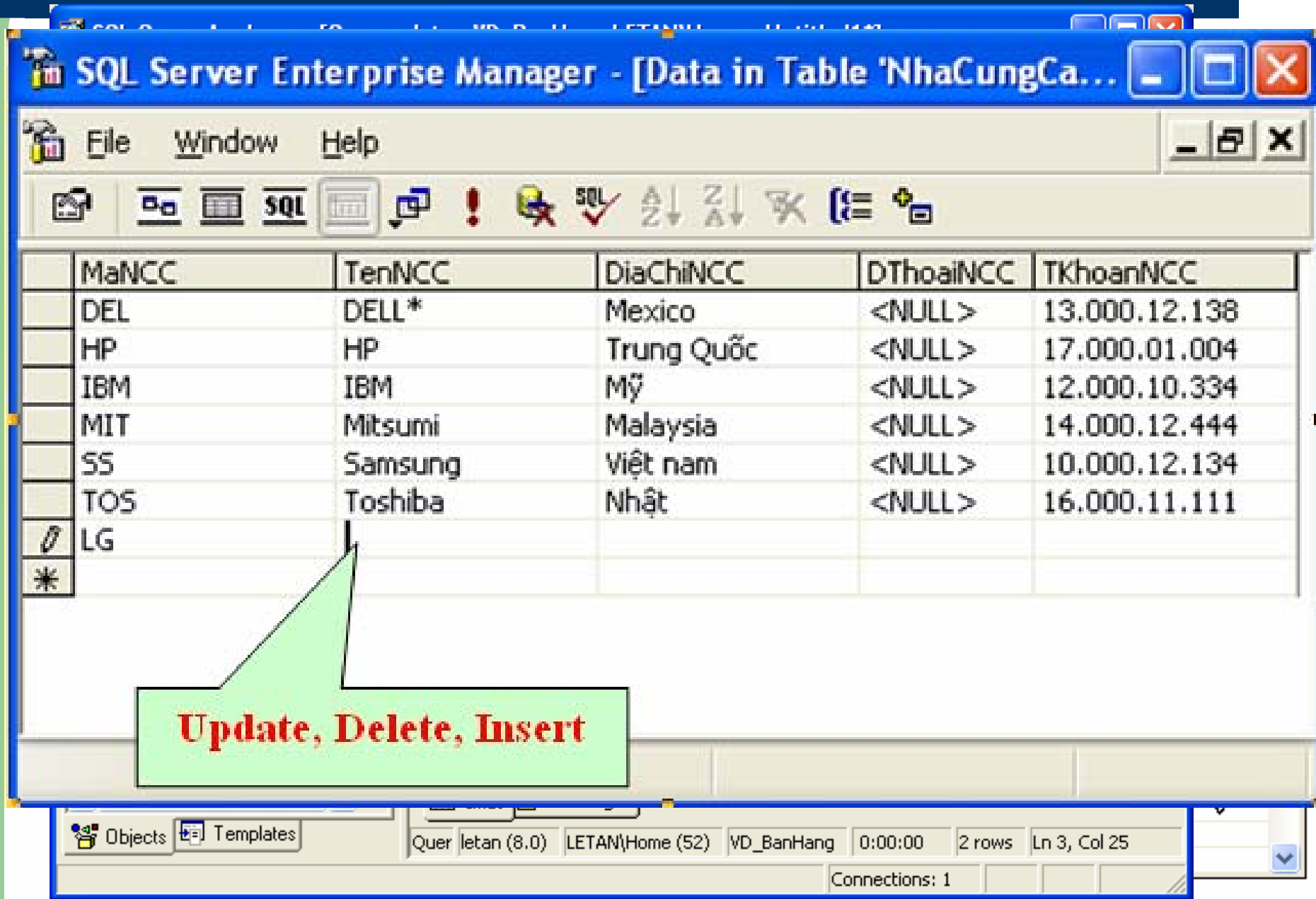
- - -

1.4. Hệ quản trị cơ sở dữ liệu

Chức năng của DBMS:

- Mô tả dữ liệu
- Tìm kiếm dữ liệu
- Cập nhật dữ liệu
- Chuyển hoá dữ liệu
- Điều khiển tính toàn vẹn
- Quản lý các giao dịch và an toàn của dữ liệu

1.4. Hệ quản trị cơ sở dữ liệu



1.4. Hệ quản trị cơ sở dữ liệu

Các thành phần của DBMS:

- Ngôn ngữ mô tả dữ liệu

  Cho phép khai báo cấu trúc dữ liệu, mô tả các mối liên kết, cung cấp các quy tắc áp đặt lên dữ liệu.

- Ngôn ngữ thao tác dữ liệu

  Cho phép cập nhật dữ liệu, khai thác dữ liệu theo nhiều mục đích khác nhau.

- Ngôn ngữ con truy vấn dữ liệu

  Nhằm hỗ trợ cho việc tìm kiếm, truy vấn.

I. CÁC KHÁI NIỆM CƠ BẢN (tiếp)

1.5. Hệ cơ sở dữ liệu (Database System)

- **CSDL hợp nhất**
  - Dư thừa tối thiểu
  - Dùng chung được
- **Người sử dụng**
  - Người dùng cuối (End User): Những người làm việc tại các Teminal
  - Người viết chương trình ứng dụng (Application Programer)
  - Người quản trị CSDL: Đây là lớp người quan trọng, có nhiệm vụ:
    - Quyết định nội dung thông tin của CSDL
    - Quyết định cấu trúc lưu trữ và chiến lược truy cập
    - Kiểm soát thẩm quyền và kiểm tra tính đúng đắn của dữ liệu
    - Xác định chiến lược sao lưu, phục hồi

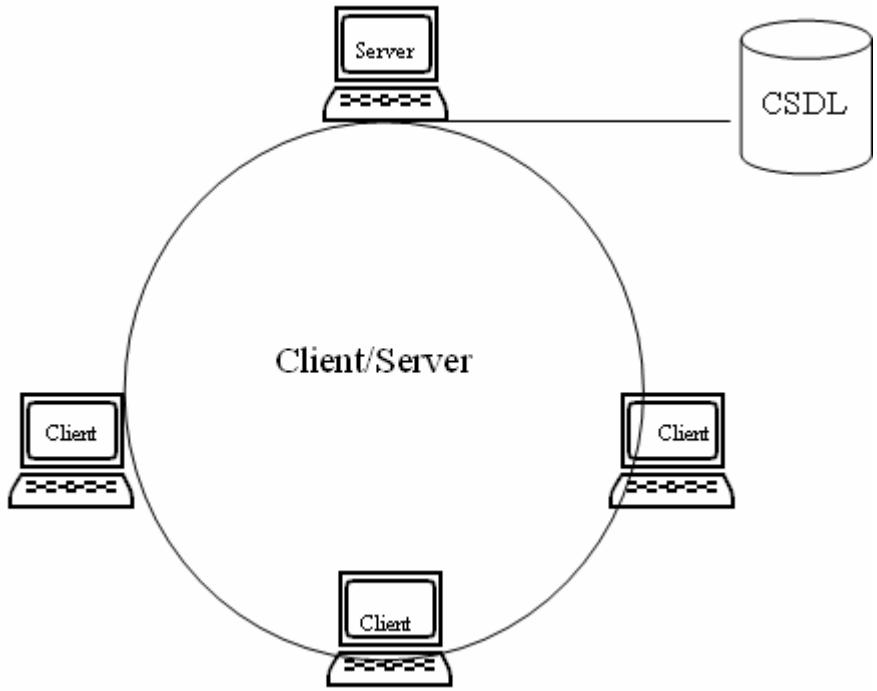
  ☞ *Lớp người này thường được cung cấp một số chương trình hỗ trợ để phục vụ cho công việc quản trị.*
- **Hệ quản trị cơ sở dữ liệu**
- **Phần cứng**

I. CÁC KHÁI NIỆM CƠ BẢN (tiếp)

1.6. Sự phân loại hệ cơ sở dữ liệu

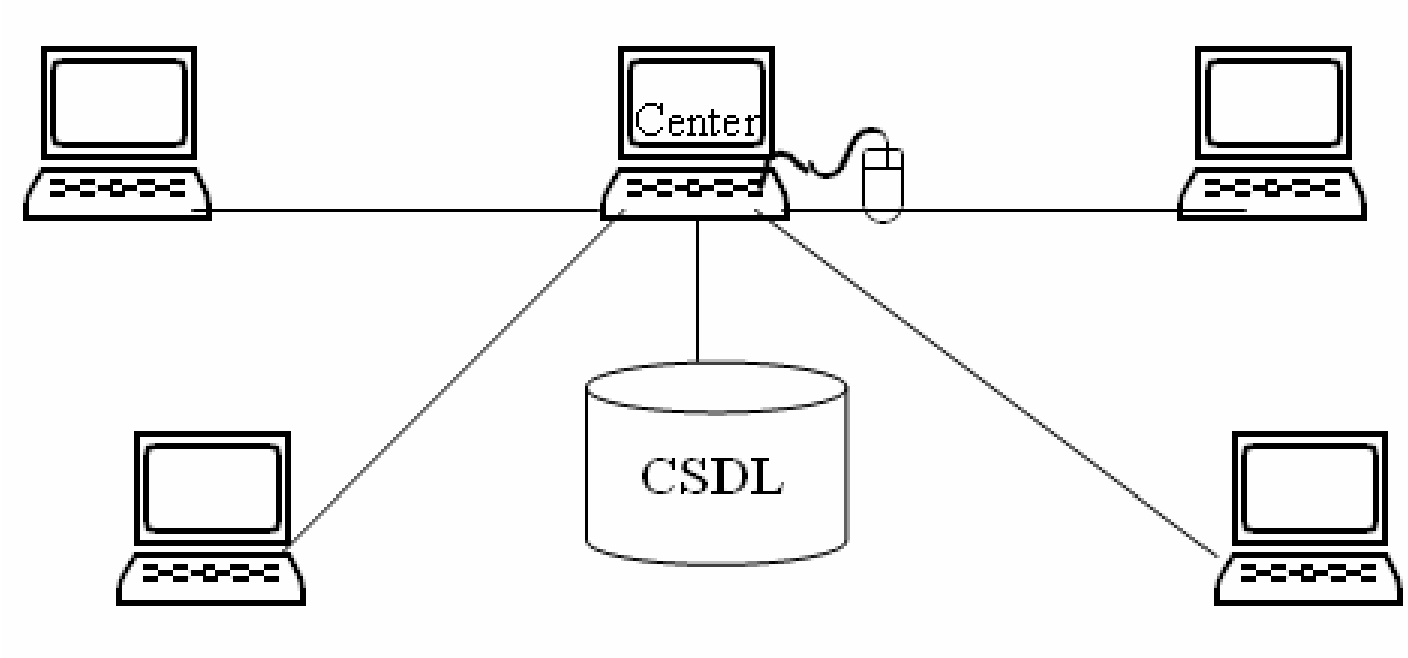
- **Hệ cơ sở dữ liệu tập trung**
  - Hệ cơ sở dữ liệu cá nhân

  - Hệ cơ sở dữ liệu Client/ Server



1.6. Sự phân loại hệ cơ sở dữ liệu (tiếp)

- Hệ cơ sở dữ liệu trung tâm

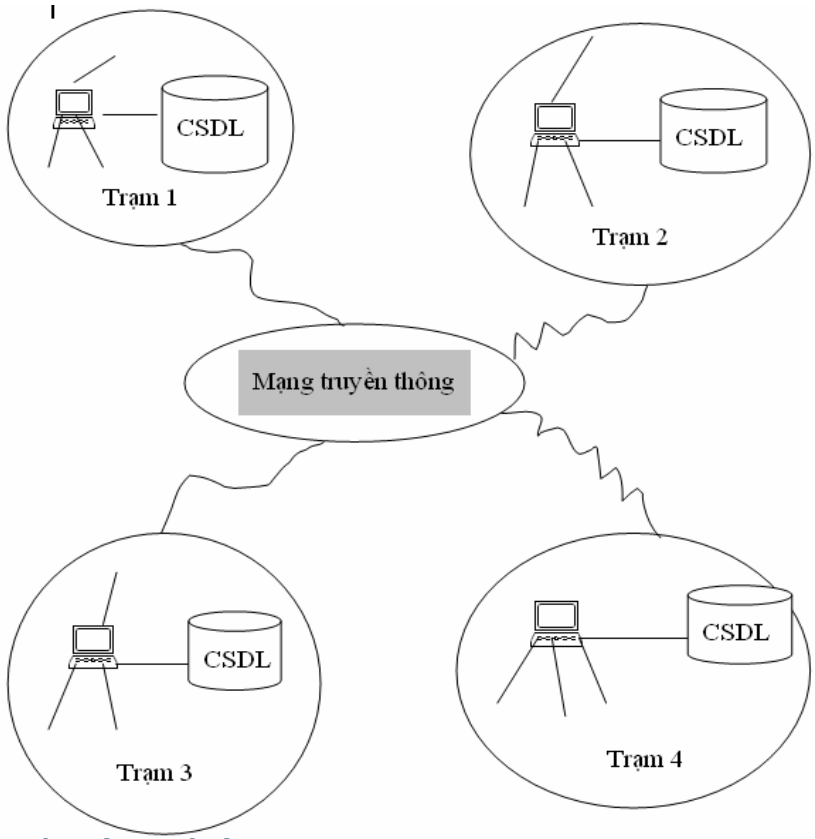


1.6. Sự phân loại hệ cơ sở dữ liệu (tiếp)

- **Hệ cơ sở dữ liệu phân tán**
  - Hệ cơ sở dữ liệu phân tán thuần nhất

Tại các vị trí giống nhau hay có tính tương thích cao về:

- Hệ điều hành
- Mô hình dữ liệu
- Hệ quản trị CSDL
- Định nghĩa và khuôn dạng DL



  - Hệ cơ sở dữ liệu phân tán không thuân nhất

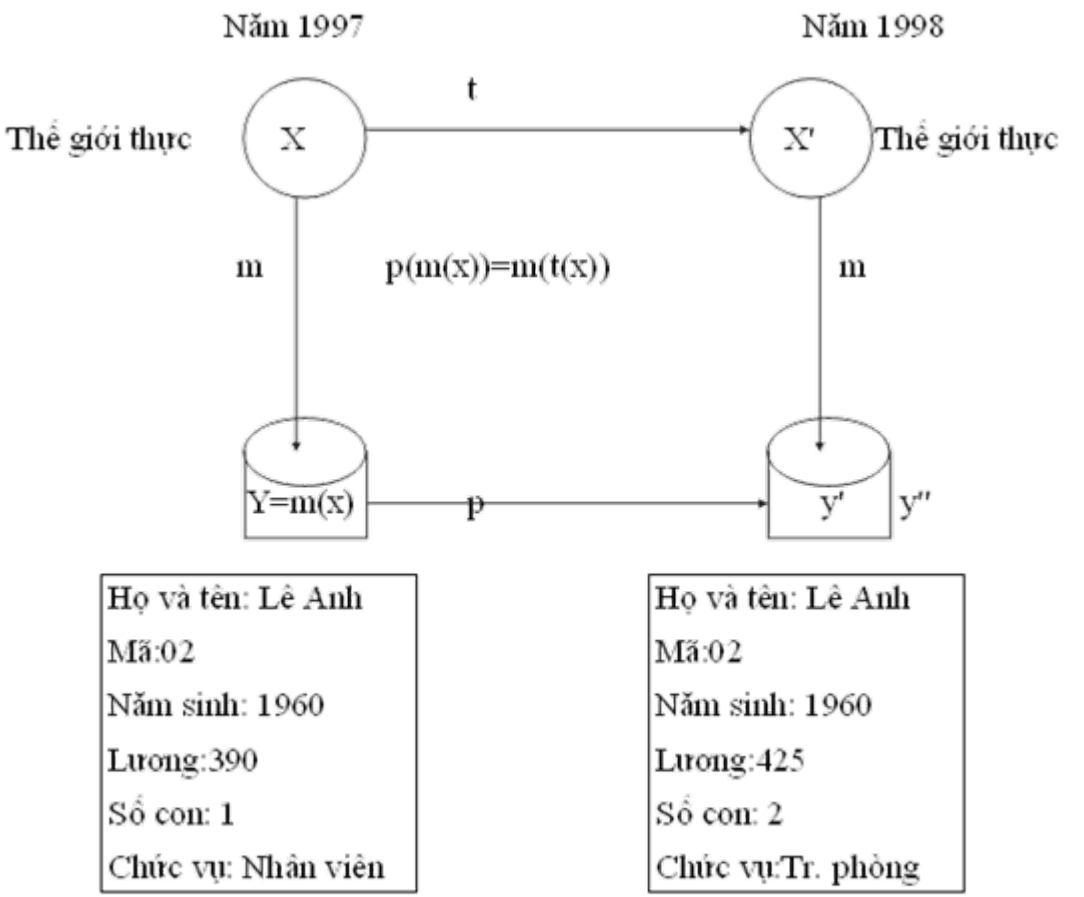
II. CÁC MÔ HÌNH DỮ LIỆU

2.1. Mô hình hoá trong tin học

Mô hình hoá trong tin học là sự xác lập tương quan giữa các đặc trưng (thuộc tính) của đối tượng trong thế giới thực với các phần tử của một tập cho trước sao cho các thông tin của các phần tử này trong quá trình vận động luôn phù hợp với sự vận động của đối tượng được mô hình hoá.

2.1. Mô hình hoá trong tin học (tiếp)

Ví dụ:



Hình 1-4: Mô tả việc mô hình hoá

Trong đó:

- *X là một nhân viên nào đó.*
- *m: phép mã hoá đối tượng X thành bộ các thuộc tính: Họ tên, mã, năm sinh, lương, số con, chức vụ.*
- *Thời điểm mô tả đối tượng X vào năm 1997 được lưu trong CSDL*
- *Trong thế giới thực, đối tượng X được biến đổi theo thời gian t; đến năm 1998 lương tăng lên 425, số con tăng lên 2 và có chức vụ mới là Tr. Phòng. Do đó X biến thành đối tượng mới X'=t(X).*

II. CÁC MÔ HÌNH DỮ LIỆU

2.2. Các mô hình dữ liệu

a, Sơ đồ thực thể liên kết (Entity Relationship Model - ERM)

- **Thực thể (Entity):** Là một đối tượng có thực hay trừu tượng mà ta muốn ghi chép thông tin về nó. Các thực thể là một bộ phận của dữ liệu tác nghiệp.

  Ví dụ. Muốn quản lý nhân sự của một cơ quan, có các thực thể:

  - Nhân viên
  - Phòng

- **Liên kết thực thể (Relationship):** Là liên kết giữa các dữ liệu mô tả các thực thể. Liên kết thực thể là một bộ phận của dữ liệu tác nghiệp.

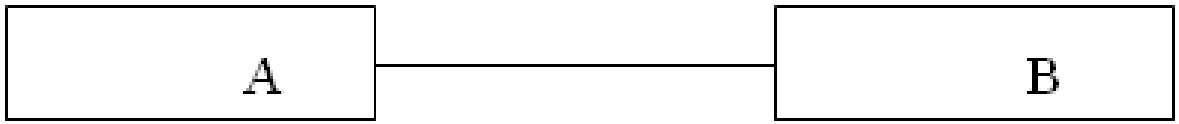
II. CÁC MÔ HÌNH DỮ LIỆU

2.2. Các mô hình dữ liệu

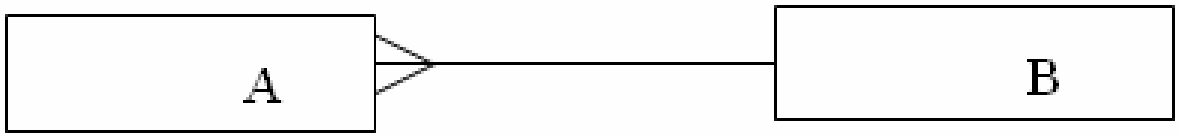
a, Sơ đồ thực thể liên kết (tiếp)

Các kiểu liên kết

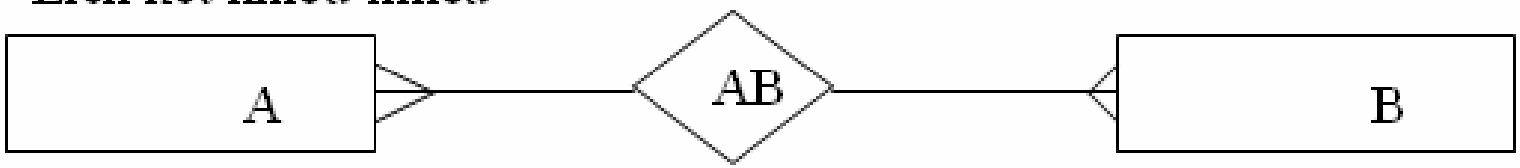
- Liên kết 1-1



- Liên kết nhiều-1



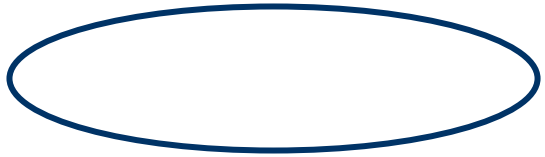
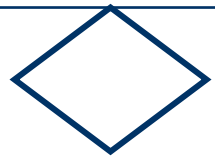
- Liên kết nhiều-nhiều



2.2. Các mô hình dữ liệu (tiếp)

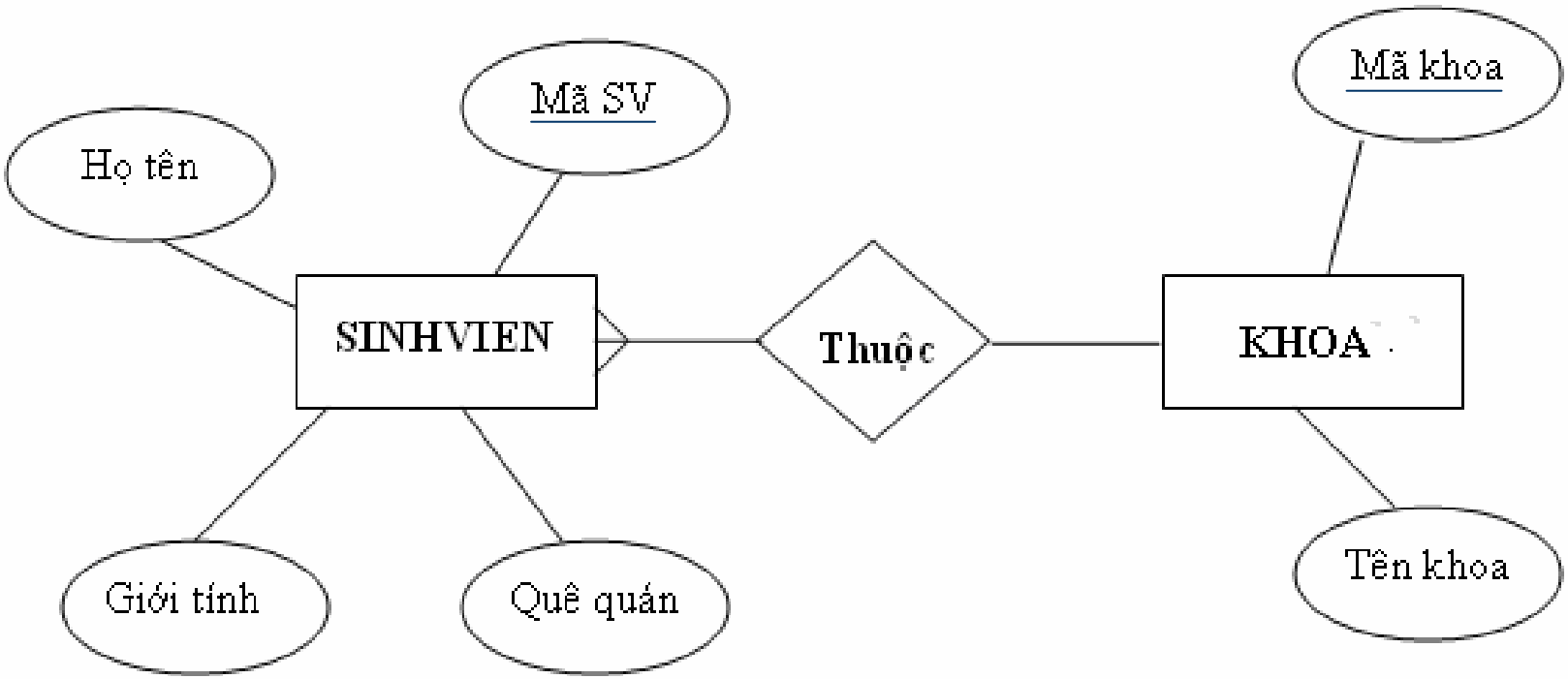
Sơ đồ liên kết thực thể:

Sơ đồ liên kết thực thể gồm 3 phần:

LOẠI	KÝ HIỆU
Thực thể	
Thuộc tính	
Liên kết thực thể	
Đầu một	hoặc
Đầu nhiều	hoặc 
Thuộc tính khoá	**Được gạch chân**

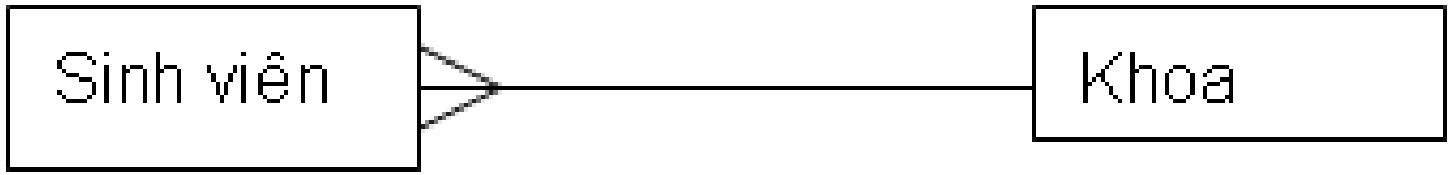
2.2. Các mô hình dữ liệu (tiếp)

Ví dụ:



2.2. Các mô hình dữ liệu (tiếp)

hoặc



.Mã SV
.Họ tên
.Giới tính
.Quê quán

.Mã khoa
.Tên khoa

2.2. Các mô hình dữ liệu (tiếp)

* ***Thuộc tính hay thực thể?*** Với một phần tử dữ liệu, để khẳng định nó là thuộc tính hay thực thể ta nên trả lời các câu hỏi:
  - Có thuộc tính riêng hay không?
  - Có liên kết với các thực thể khác hay không, dạng liên kết như thể nào?

Khi đó:

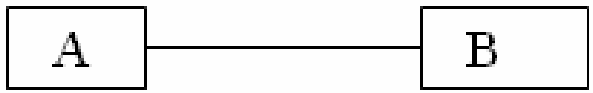
- Nếu không có thuộc tính riêng và liên kết là 1-1 thì nên xem nó là thuộc tính
- Nếu có thuộc tính riêng và liên kết là 1-nhiều hoặc nhiều-nhiều thì xem nó là thực thể.

2.2. Các mô hình dữ liệu (tiếp)

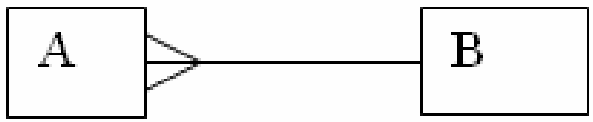
b, Mô hình dữ liệu quan hệ (Relational data model)

Mô hình quan hệ lần đầu tiên được E. F. Codd - một nhân viên của hãng IBM đề xuất vào năm 1970 với các khái niệm chính: Thuộc tính, lược đồ quan hệ, bộ, quan hệ và khoá.

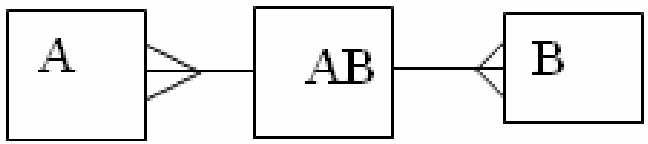
Liên kết thực thể trong mô hình quan hệ:

1/ 

$\begin{cases} A(K_A,\ldots,K_B) \text{ Lấy } K_B \text{ đặt vào A} \\ B(K_B, \ldots) \end{cases}$

2/ 

$\begin{cases} A(K_A,\ldots,K_B) \text{ Lấy khoá đầu 1 đặt vào} \\ B(K_B, \ldots) \text{ đầu nhiều} \end{cases}$

3/ 

$\begin{cases} A(K_A,\ldots) \\ B(K_B, \ldots) \\ AB(K_A\ ,\ K_B\ ,\ \ldots.) \end{cases}$

2.2. Các mô hình dữ liệu (tiếp)

b, Mô hình dữ liệu quan hệ (Relational data model)

MaSV	Hoten	QueQuan	MaKhoa
01	LE VAN AN	Nghe An	K1
02	NGUYEN AN	HT	K1
03	LE BINH	HT	K1
04	PHAM TUAN	Nghe An	K2
05	PHAN ANH	Nghe An	K2

Nhiều

1

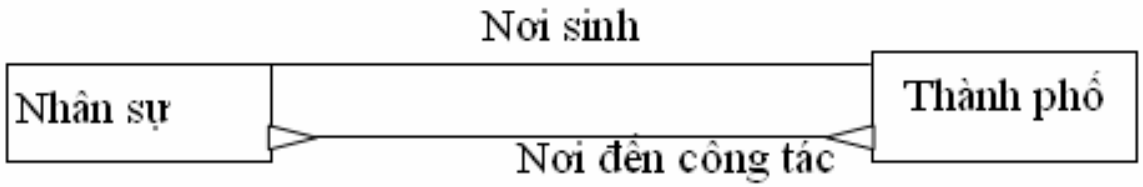
makhoa	tenkhoa
K1	TOAN TIN
K2	LY
K3	NGOAI NGU
K4	VAN

2.2. Các mô hình dữ liệu (tiếp)

c, Mô hình dữ liệu mạng (Network data model)

Mô hình mạng được nhóm công tác DBGT của CODASYL trình bày vào năm 1971. Mô hình mạng được xây trên các tập cơ sở dữ liệu và các mối quan hệ. Tập cơ sở dữ liệu được tạo ra từ các dữ liệu cùng kiểu gọi là bản ghi (record). Mỗi bản ghi được tạo bởi từ các trường.

Ví dụ. Giả sử có 2 tập



Họ và tên:
Mã:
Năm sinh:
Lương:
Số con:
Chức vụ:

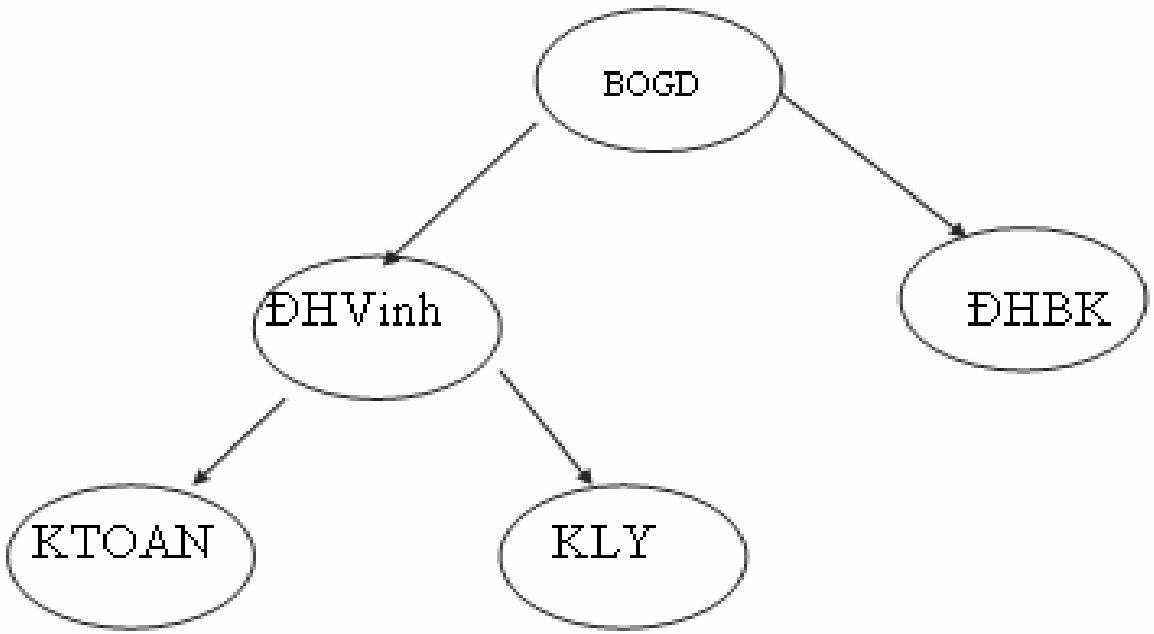
Mã thành phố
Tên thành phố
Diện tích
K/C tới thủ đô

2.2. Các mô hình dữ liệu (tiếp)

d, Mô hình dữ liệu phân cấp (Hierarchical Data Model)

Đây là một trường riêng của mô hình mạng, trong đó khái niệm tập được giữ nguyên, còn khái niệm quan hệ được giới hạn bởi kiểu phân cấp. Giữa 2 tập có không quá một quan hệ và quan hệ này tuân theo trật tự trên dưới.

Ví dụ. Cấu trúc cây thư mục của Dos



2.2. Các mô hình dữ liệu (tiếp)

e, Mô hình dữ liệu hướng đối tượng (Object oriented data model)

Mô hình hướng đối tượng dựa trên cách tiếp cận hướng đối tượng bao gồm các khái niệm: Lớp, kế thừa,... Đặc tính nổi bật của phương pháp pháp tiếp cận này là tính đóng gói các dữ liệu, tính đa hình, tái sử dụng,... Hiện nay, chưa có nhiều hệ quản trị cơ sở dữ liệu cài đặt theo mô hình này nên nó chưa được ứng dụng rộng rãi.

Phần I : Giới thiệu

Mục tiêu:

- Giới thiệu khái quát về Mạng máy tính
- Đi sâu hơn trong các khoá học sau
- Tiếp cận :
  - Mô tả được mạng máy tính
  - Sử dụng mô hình mạng Internet làm trọng tâm nghiên cứu

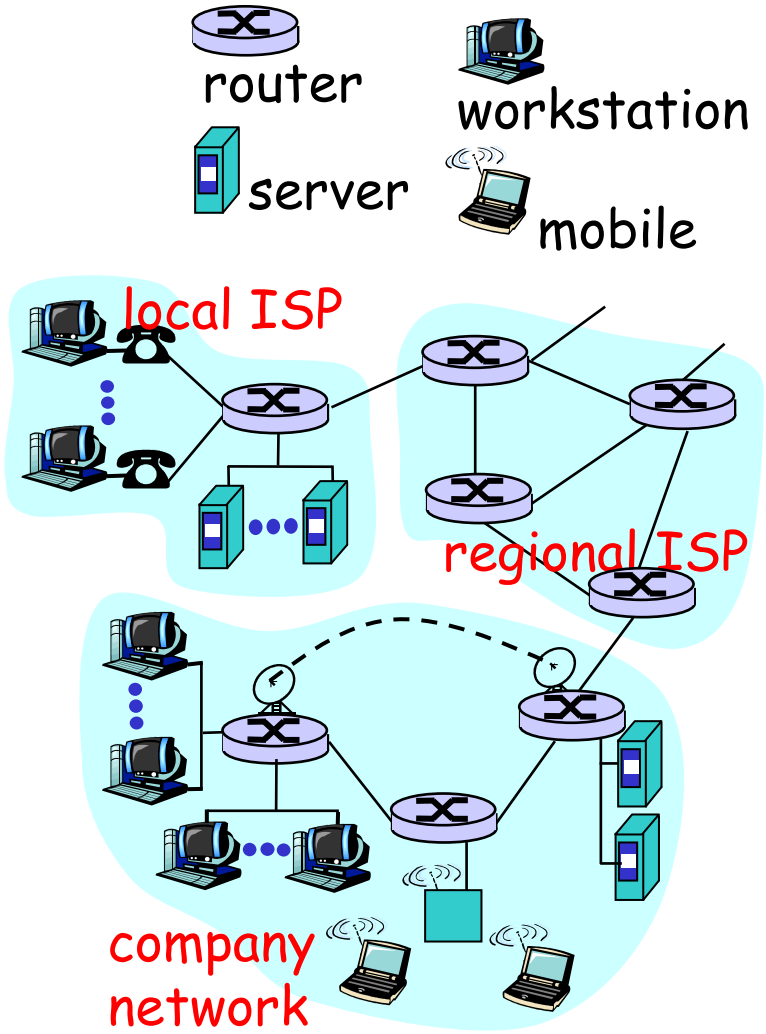
Học cái gì ?

- Internet là gì ?
- Giao thức (Protocol) là gì ?
- Network edge (“Rìa”)
- Network core (“Lõi”)
- Truy nhập mạng, Môi trường truyền
- Hiệu suất : Mất mát và Độ trễ
- Các tầng giao thức và Mô hình dịch vụ
- Trục chính (backbones), NAP, ISP
- Lịch sử phát triển Internet

Đơn giản - Internet là gì ?

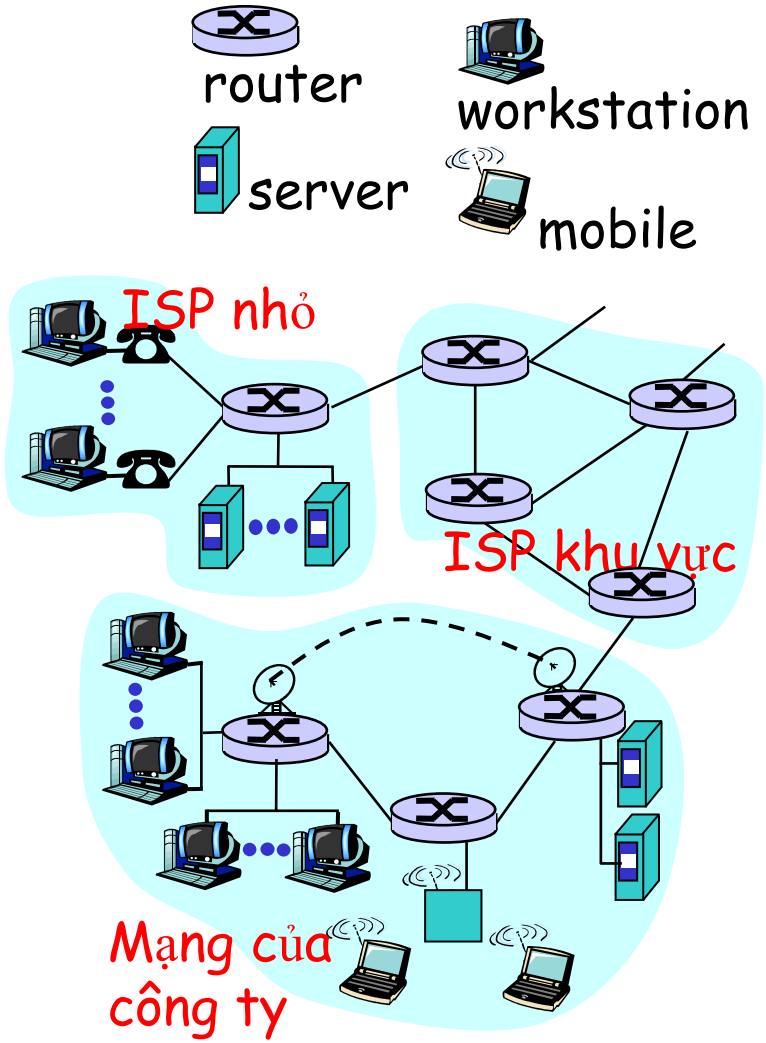
- **Hàng triệu các thiết bị : hệ thống đầu cuối**
  - Trạm làm việc (workstation), trạm phục vụ (server)
  - Điện thoại PDA, Thiết bị sinh hoạt

  Chạy các ứng dụng mạng
- **Đường truyền**
  - Cáp quang, dây đồng, sóng radio, vệ tinh
- **Router (Bộ định tuyến)**:
  - Chuyển tiếp các gói dữ liệu qua mạng



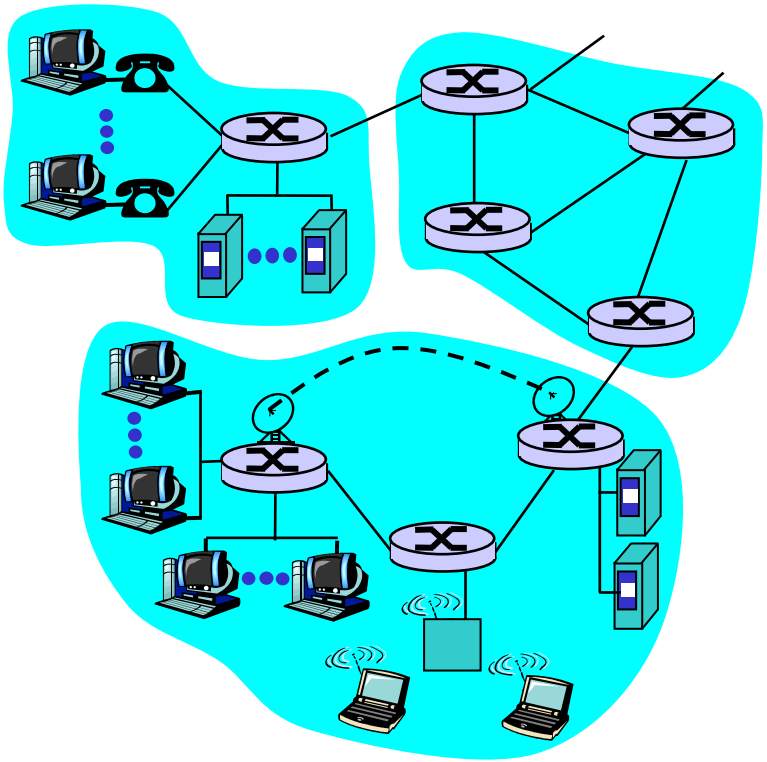
Đơn giản - Internet là gì ?

- **Giao thức :** Kiểm soát việc gửi và nhận thông điệp
  - Ví dụ: TCP, IP, HTTP, FTP, PPP
- **Internet:** "mạng của mạng"
  - Cấu trúc lỏng lẻo
  - Mạng Internet dùng chung đối lập với mạng Intranet dùng riêng.
- Các chuẩn Internet
  - RFC : Khuyến nghị (Request for Comments)
  - IETF: Internet Engineering Task Force



Internet là gì ? Theo quan điểm Dịch vụ

- ***Cơ sở hạ tầng truyền thông cho phép cài đặt ứng dụng phân tán*:**
  - WWW, email, games, thương mại điện tử, tổ chứccơ sở dữ liệu, bầu cử, chia sẻ tệp (MP3)



- Những dịch vụ truyền thông cung cấp:
  - Không hướng nối
  - Hướng nối

- cyberspace [Gibson]:

  "Một ảo giác liên ứng giàu kinh nghiệm hàng ngày bởi hàng triệu hệ điều hành, trên mỗi quốc gia, ..."

Giao thức là gì?

Con người:

- "Mấy giờ rồi?"
- "Tôi muốn hỏi"
- Giới thiệu làm quen

… **Thông điệp gửi đi được viết ntn ?**

… **Khi nhận được thông điệp thì phải làm gì ?**

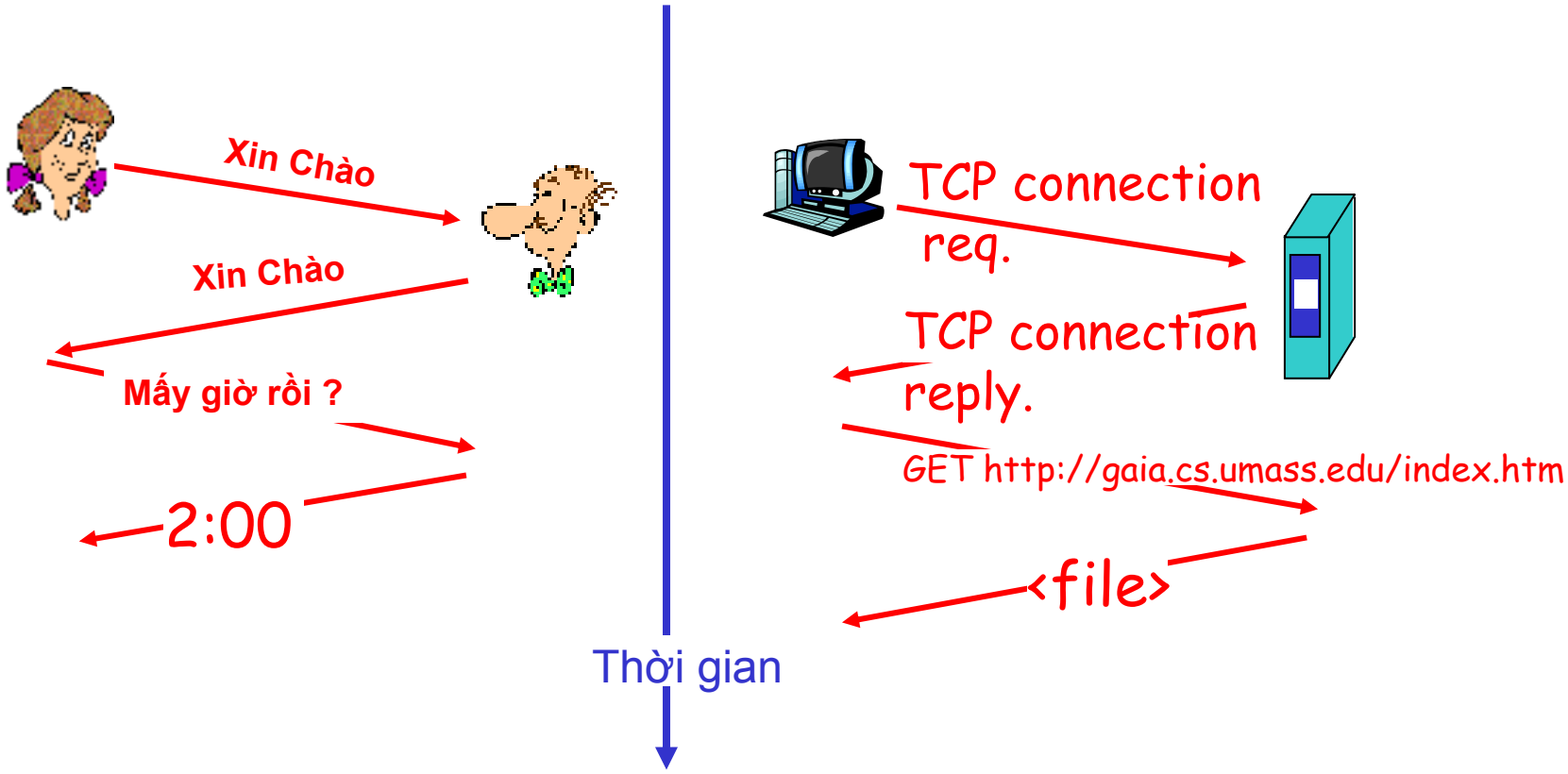
Giao thức mạng:

- Phức tạp hơn nhiều !
- Mọi hoạt động truyền thông trên Internet phải tuân thủ giao thức

Giao thức định nghĩa Khuôn dạng, Trình tự gửi và nhận các thông điệp giữa các thực thể mạng, cũng như các Hành động khi nhận và gửi thông điệp

Giao thức là gì ?

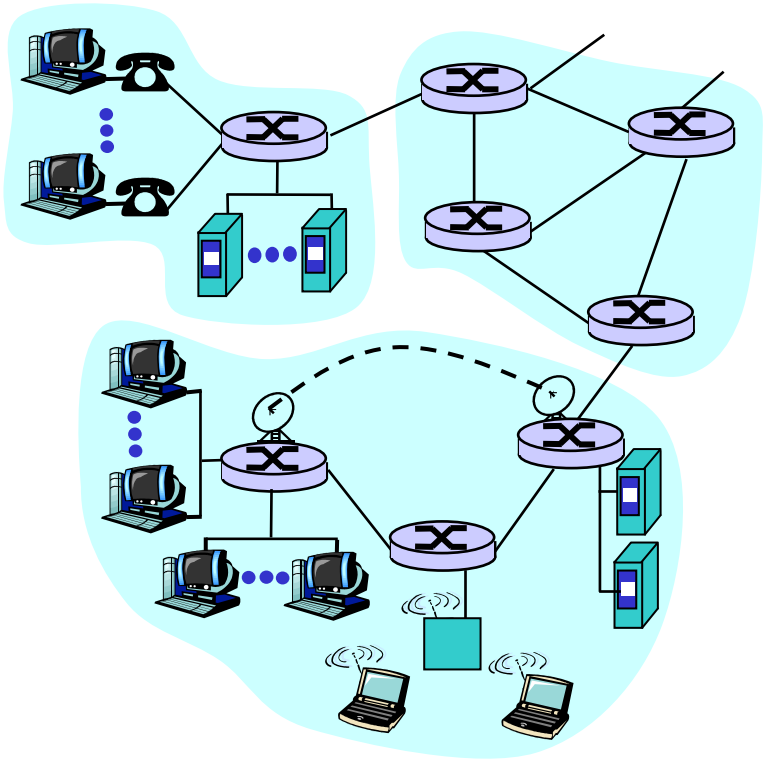
Quy tắc Xử thế của Con người và Giao thức Mạng Máy Tính



Q: Còn quy tắc xử thế nào khác ? ***Rất nhiều !***

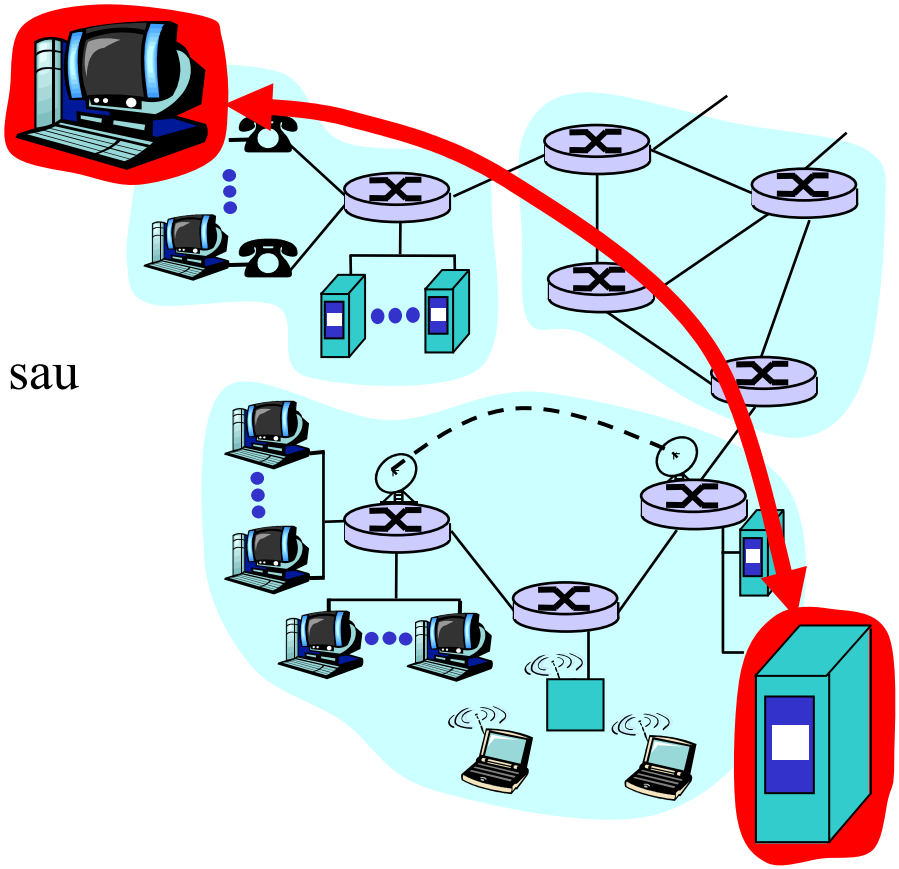
Cấu trúc Mạng – Nhìn gần

- **Rìa của Mạng:**
  - Ứng dụng
  - Thiết bị đầu cuối
- **Lõi của Mạng:**
  - Bộ định tuyến (router)
  - Mạng của Mạng
- **Truy cập mạng và Môi trường truyền:**
  - Các đường kết nối



“Rìa” của Mạng

- **Hệ thống đầu cuối (hosts):**
  - Chạy chương trình ứng dụng
  - Ví dụ: WWW, email
  - Nằm ở “Rìa của Mạng”
- **Mô hình Client / Server**
  - Máy tính client gửi yêu cầu, và sau đó nhận dịch vụ từ server
  - Ví dụ WWW client (browser)/ server; email client/server
- **Mô hình đồng đẳng (peer-peer) :**
  - Các thiết bị đầu cuối có vai trò tương đương
  - Ví dụ : teleconferencing



Lớp Rìa - Dịch vụ Hướng nối

Mục tiêu: Truyền dữ liệu giữa các thực thể đầu cuối.

- ***Bắt tay***: Thiết lập (chuẩn bị trước) cho quá trình truyền dữ liệu
  - Ví dụ : Chào hỏi trước khi trò chuyện
  - *Thiết lập "**trạng thái**"* ở hai phía đầu cuối
- **TCP** - **T**ransmission **C**ontrol **P**rotocol
  - Dịch vụ Hướng nối của Internet

Dịch vụ TCP [RFC 793]

- Chuyển dữ liệu là một luồng byte ***tin cậy***, ***đúng thứ tự***
  - Mất dữ liệu: Biên nhận và Truyền lại
- ***Điều khiển lưu lượng***:
  - Tốc độ Nhận thấp hơn Tốc độ Gửi
- ***Kiểm soát tắc nghẽn***:
  - Phía Gửi sẽ giảm tốc độ khi Mạng bị tắc nghẽn

Lớp Rìa - Dịch vụ Không Hướng nối

Mục tiêu : Truyền dữ liệu giữa các thực thể đầu cuối

- Giống Hướng nối !

- **UDP** - **U**ser **D**atagram **P**rotocol [RFC 768]: Dịch vụ Không Hướng nối của Internet
  - Không tin cậy
  - Không điều khiển lưu lượng
  - Không kiểm soát tắc nghẽn

Ứng dụng sử dụng TCP:

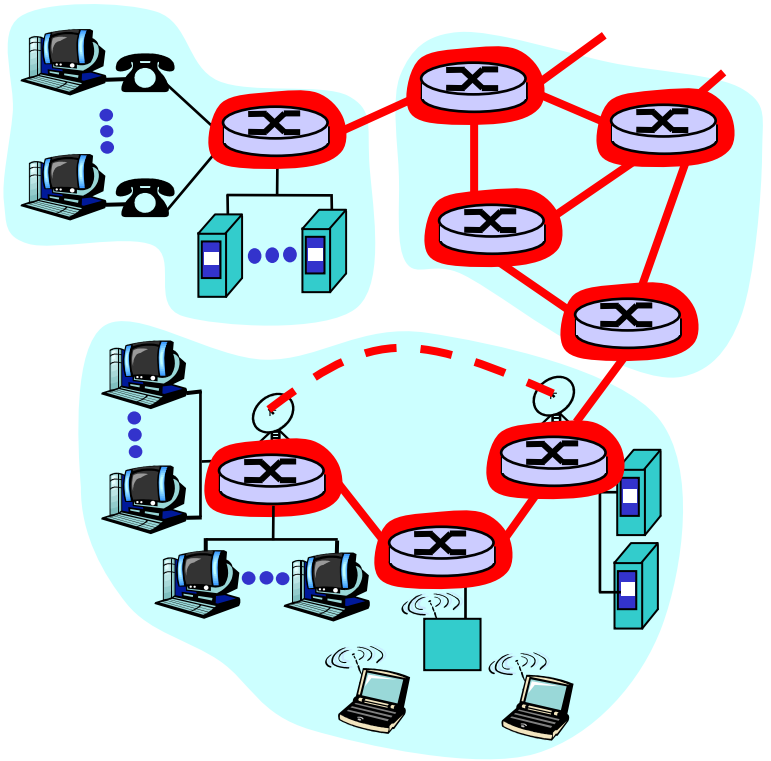
- HTTP (WWW), FTP (file transfer), Telnet (remote login), SMTP (email)

Ứng dụng sử dụng UDP:

- Đa phương tiện theo luồng, Hội thảo từ xa, Điện thoại Internet

“Lõi” của Mạng

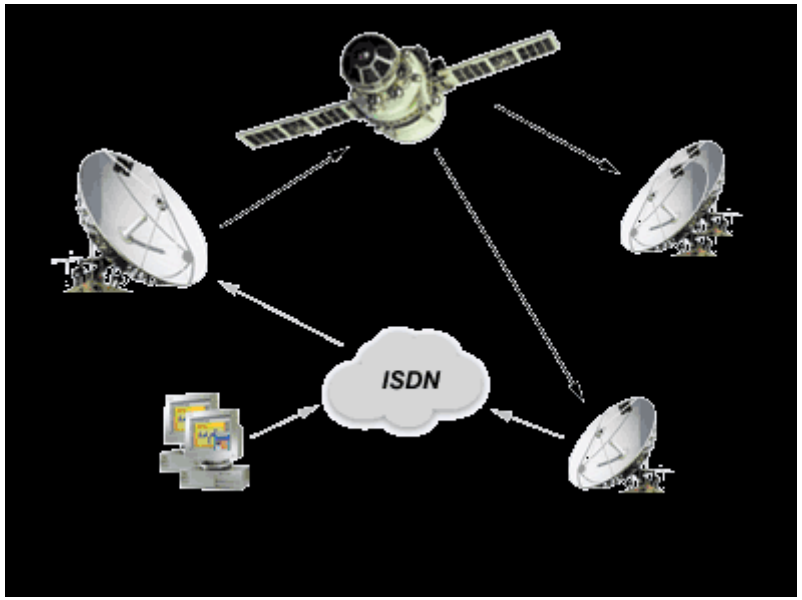
- Hệ thống các Thiết bị Định tuyến kết nối với nhau
- **Vấn đề cơ bản**: Làm thế nào để chuyển dữ liệu qua mạng?
  - Chuyển mạch ảo: Mạch dùng riêng cho mỗi cuộc truyền: Mạng điện thoại
  - Chuyển gói: Dữ liệu được truyền theo từng cụm



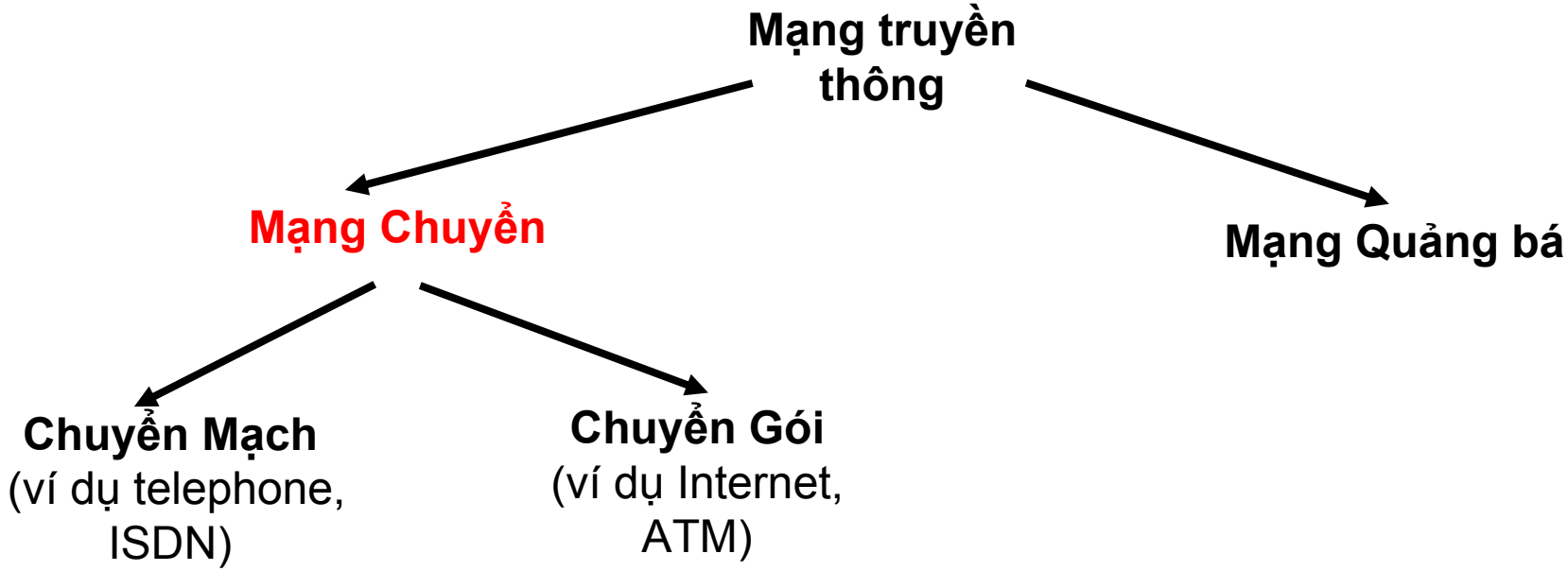
Quảng bá (Broadcast) và Chuyển (Switched)

- **Mạng Quảng bá**
  - Tất cả các nút chia nhau kênh truyền chung; thông tin được một nút truyền sẽ được tất cả các nút khác nhận được
  - Ví dụ: TV, radio

- **Mạng Chuyển**
  - Thông tin phải chuyển qua nhiều chặng mới đến được đích



Phân loại Mạng Chuyển

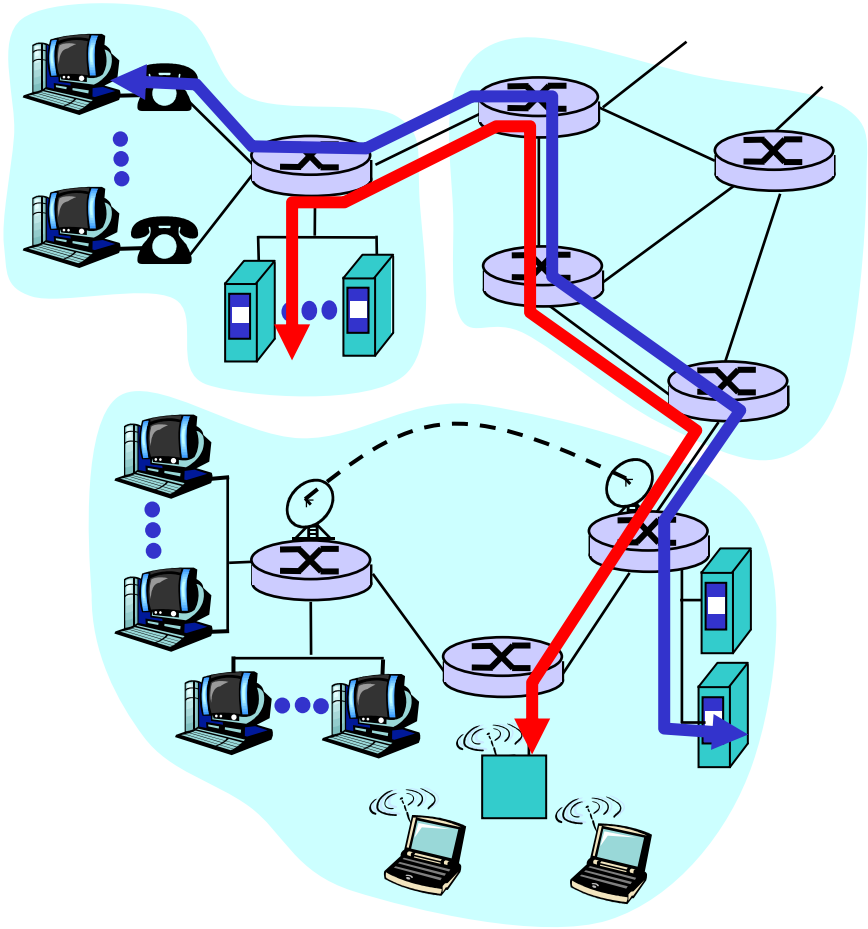


- **Chuyển mạch ảo**: Mạch dùng riêng cho mỗi phiên truyền thông:
  - Ví dụ : Điện thoại, GSM High-Speed Circuit-Switched Data (HSCSD), Integrated Services Digital Networks (ISDN)
- **Chuyển gói**: Dữ liệu được gửi đi theo từng cụm
  - Ví dụ：Internet, General Packet Radio Service (GPRS), Asynchronous Transfer Mode (ATM)

“Lõi”: Chuyển mạch ảo

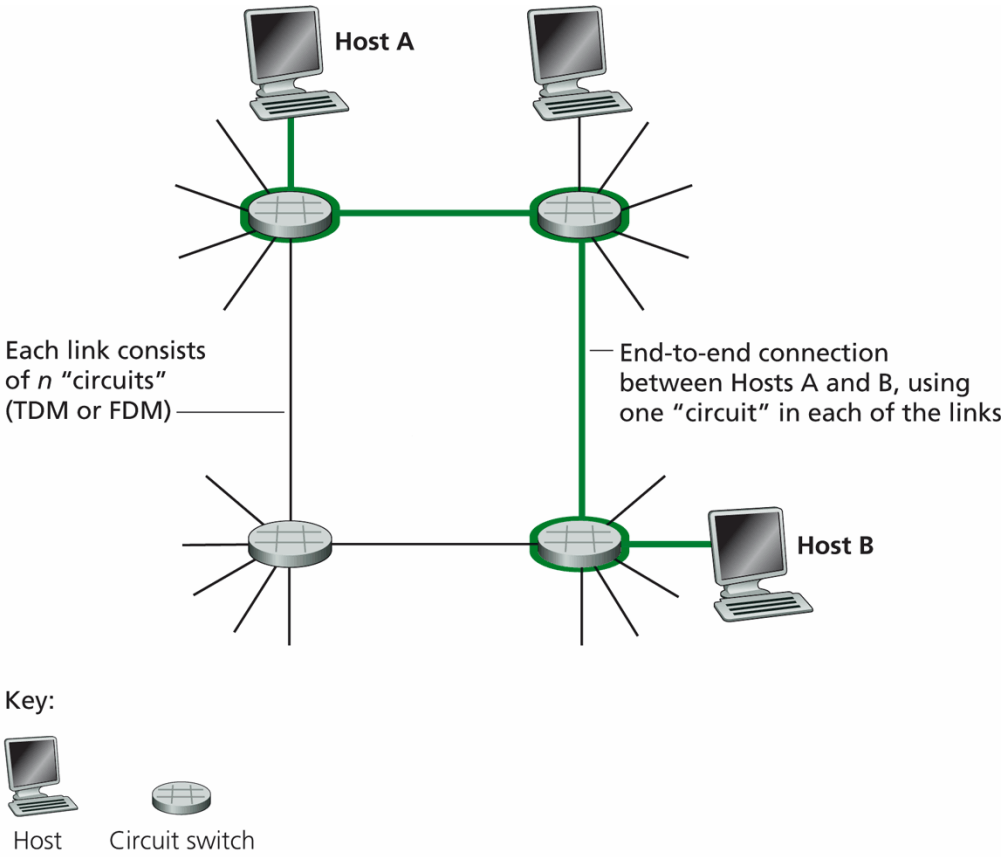
Tất cả tài nguyên trên đường truyền được cấp phát cho kết nối

- Băng thông kênh truyền, Năng lực xử lý của thiết bị trung gian
- Không chia sẻ tài nguyên đã được cấp phát
- Đảm bảo Hiệu suất như trong Mạch điện thoại
- Đòi hỏi thiết lập đường truyền



Chuyển mạch ảo

- Mỗi đường kết nối gồm nhiều "mạch - circuits"
  - "circuit" còn được gọi là channel
- Trong kênh truyền giữa hai đầu cuối phải có riêng một "circuit" tại mỗi đường kết nối

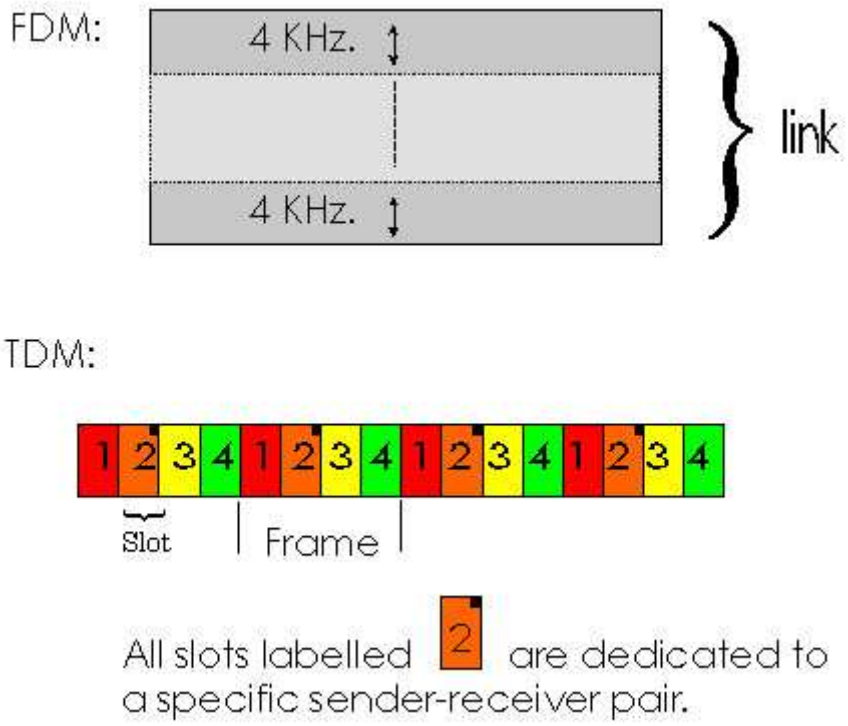


Tổng đài điện thoại thương mại đầu tiên khai trương vào năm 1878 và cho đến nay vẫn tiếp tục được sử dụng ở nhiều nơi

“Lõi”: Chuyển Mạch Ảo

Tài nguyên (băng thông)
được chia thành “phần”

- Mỗi phần được cấp phát cho một kết nối
- “Phần” sẽ rơi vào trạng thái *rỗi* nếu kết nối sở hữu không sử dụng *(không chia sẻ)*
- Băng thông được chia theo:
  - Tần số
  - Thời gian

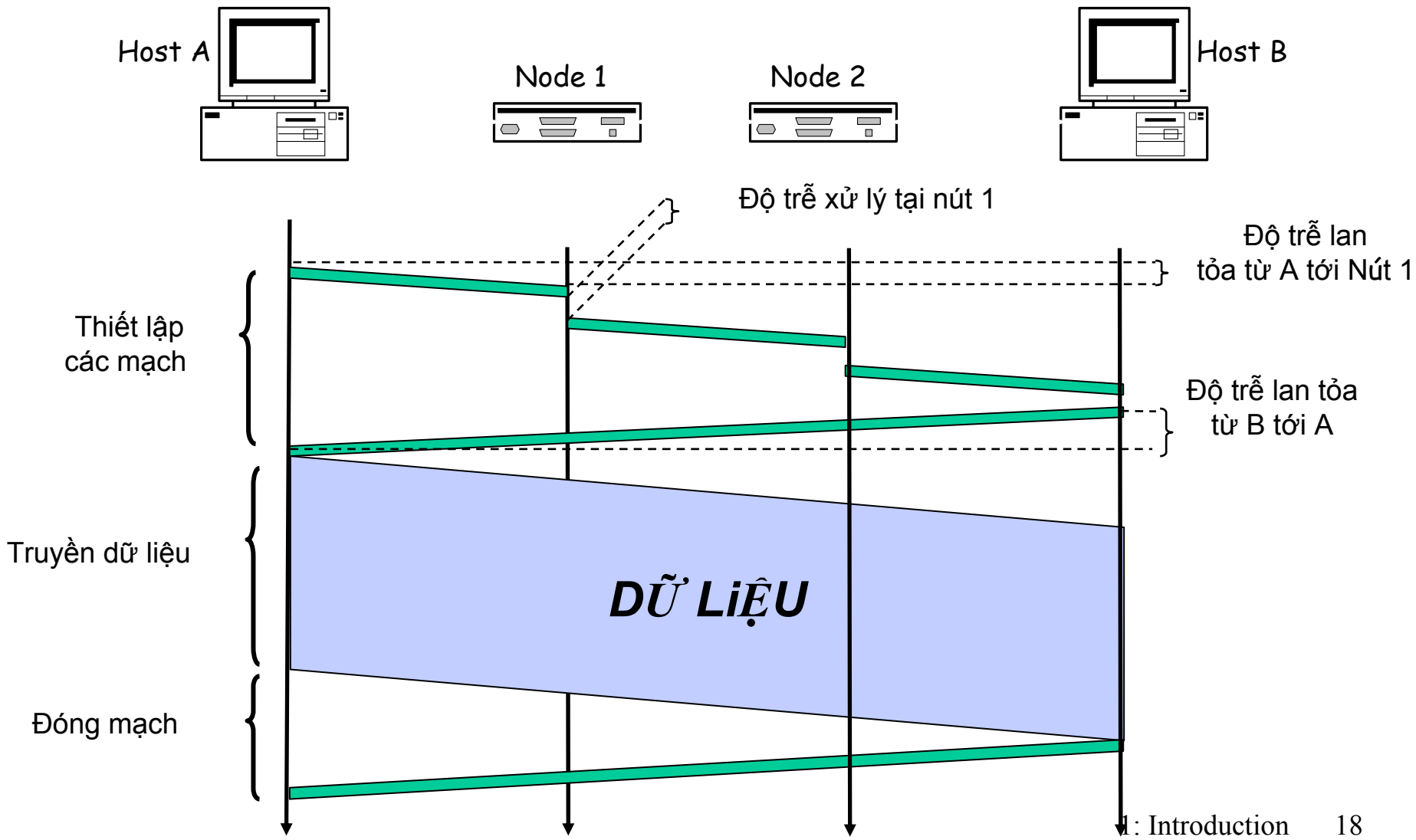


Chuyển Mạch ảo : Quá trình

- Ba giai đoạn
  1. Thiết lập các mạch
  2. Truyền Dữ liệu
  3. Giải phóng tất cả các mạch

- Nếu không thiết lập được một mạch nào đó: “tín hiệu bận”

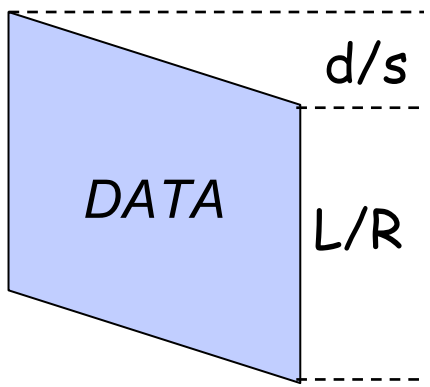
Lược đồ Thời gian Chuyển mạch ảo



Tính toán Độ trễ trong Mạng Chuyển Mạch ảo

- Độ trễ Lan tỏa: Thời gian để bit đầu tiên đi từ nơi Gửi đến nơi Nhận

- Độ trễ Truyền: Thời gian để "đổ" dữ liệu ra kênh truyền với tốc độ định trước



Độ trễ lan tỏa:

- d = Độ dài kênh truyền vật lý
- s = Tốc độ lan tỏa trong môi trường (~$2x10^5$ km/sec)
- Độ trễ lan tỏa = d/s

Độ trễ Truyền:

- R = Băng thông định trước (bps)
- L = Kích thước gói dữ liệu (bits)
- Thời gian để truyền = L/R

Ví dụ

- Độ trễ Lan tỏa
  - Giả sử khoảng cách giữa A và B là 4000 km, Độ trễ lan tỏa một chiều sẽ là:

$$\frac{4000km}{200{,}000km/s} = 20ms$$

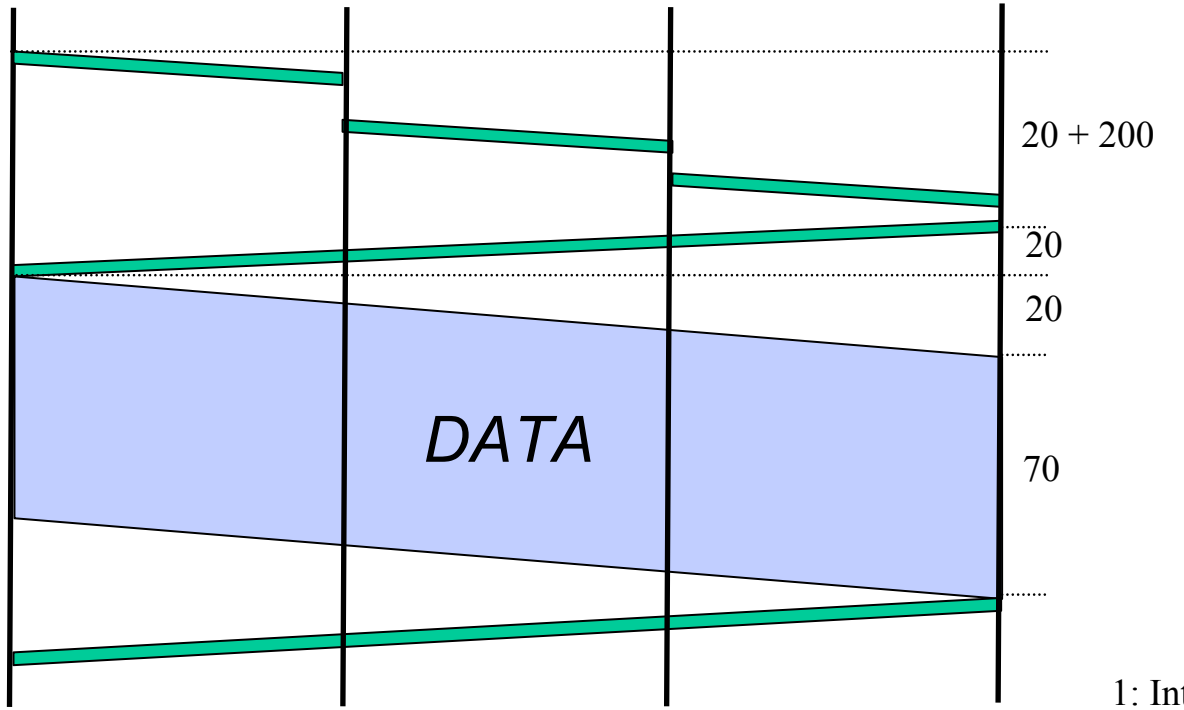
- Độ trễ Truyền
  - Giả sử chúng ta dành một slot HSCSD
    - GSM frame có thể truyền với tốc độ 115 kbps
    - GSM frame được chia thành 8 slots
    - Mỗi slot HSCSD có băng thông khoảng 14 Kbps (=115/8)
  - Độ trễ truyền của gói tin 1 Kbits là

$$\frac{1kbits}{14kbps} \approx 70ms$$

Ví dụ (Tiếp)

- Giả sử Thông điệp thiết lập đường truyền vô cùng nhỏ và Tổng thời gian trễ do xử lý thiết lập là 200 ms
- Độ trễ để chuyển gói dữ liệu 1 Kbits từ A đến B (từ đầu cho đến khi bên nhận nhận được bit cuối cùng của file) là:

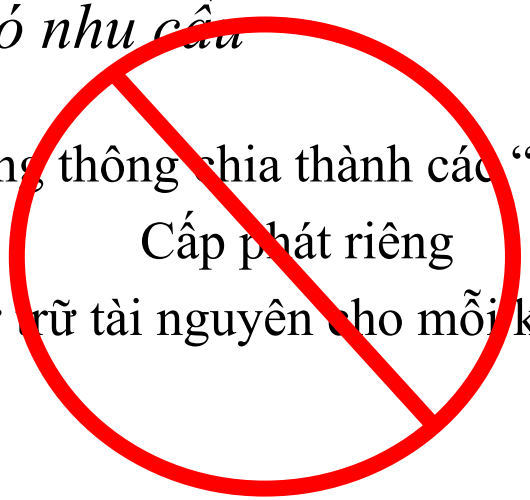
$$20 + 200 + 20 + 20 + 70 = 310\,ms$$



“Lõi”: Chuyển Gói – Tổng quan

Luồng dữ liệu được chia thành các gói tin (*packet)*

- Gói tin của người A và B *chia sẻ* tài nguyên mạng
- Mỗi gói tin có thể sử dụng toàn bộ băng thông của kênh truyền
- Tài nguyên được sử dụng *khi có nhu cầu*



Băng thông chia thành các “phần”

Cấp phát riêng

Dự trữ tài nguyên cho mỗi kết nối

Tranh chấp Tài nguyên

- Tổng lượng tài nguyên yêu cầu có thể lớn hơn tài nguyên của mạng
- Tắc nghẽn: Các gói tin “xếp hàng” đợi đến lượt sử dụng tài nguyên
- Giữ và Chuyển: Mỗi lần gói tin di chuyển qua từng chặng
  - Truyền qua một chặng
  - Đợi ở một chặng kế tiếp

Tiêu đề Gói tin

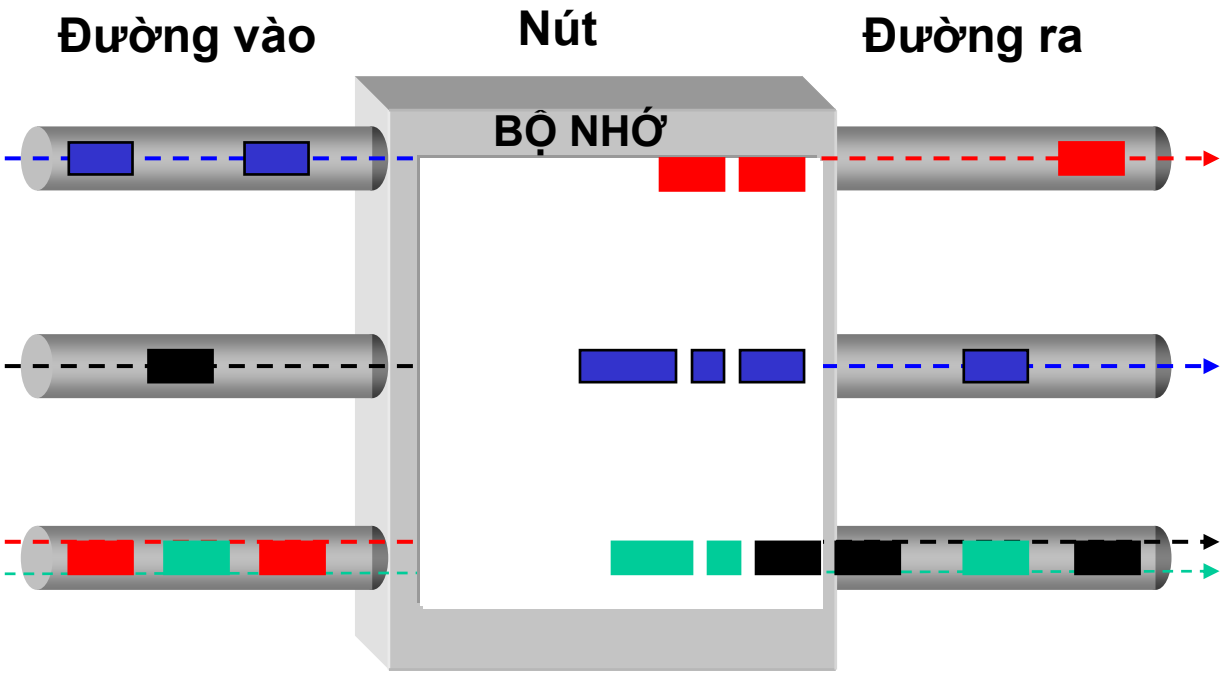
Luồng dữ liệu giữa hai đầu cuối (chẳng hạn tiến trình Gửi-Nhận) được chia thành ***gói tin (packet)*** Packet có khuôn dạng chung sau:

- ***Header and Trailer chứa thông tin kiểm soát (địa chỉ, checksum)***
- ***Thông tin điều khiển cho chuyển mạch đặt ở đâu ?***

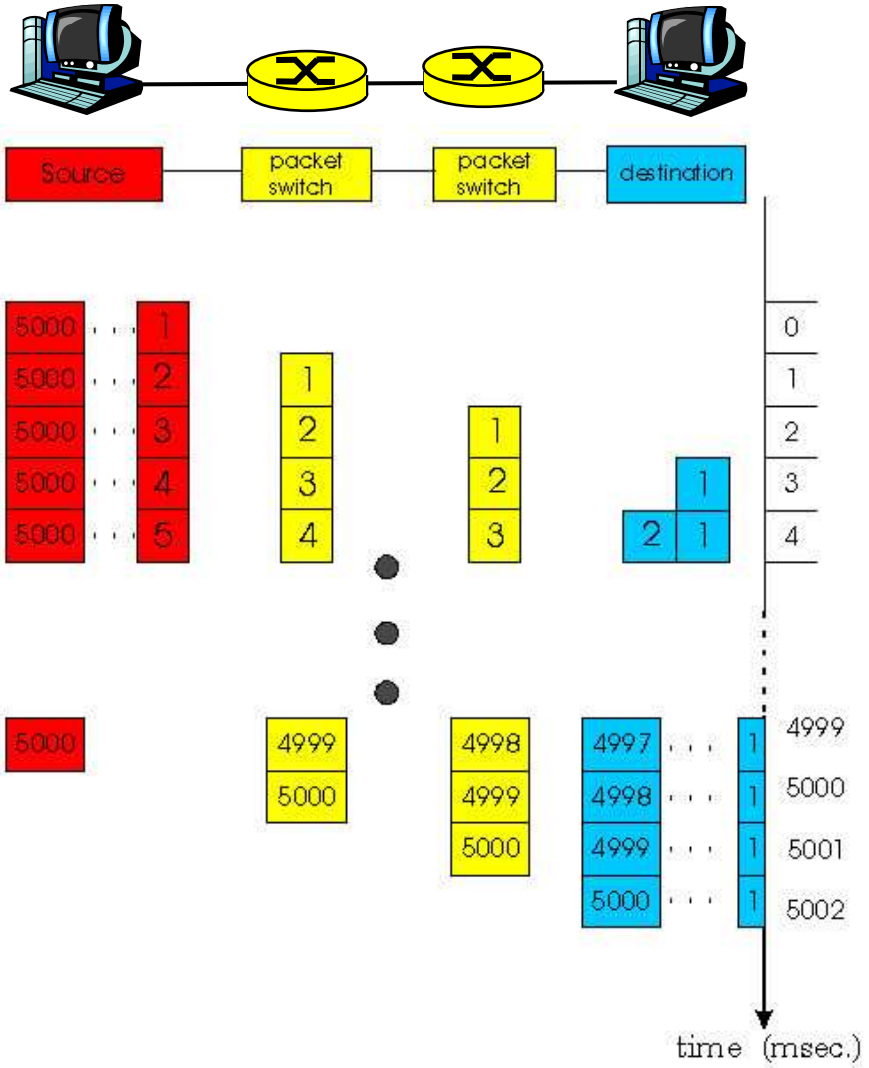
- Mỗi nút trung gian sẽ nhận toàn bộ packet, xử lý (chẳng hạn định tuyến), lưu trữ một thời gian và sau đó chuyển tiếp packet tới nút kế tiếp. Hành vi này còn được gọi là **Giữ và Chuyển**

Bên trong Bộ định tuyến Mạng Gói

Thiết bị trung chuyển



Hành vi Giữ và Chuyển



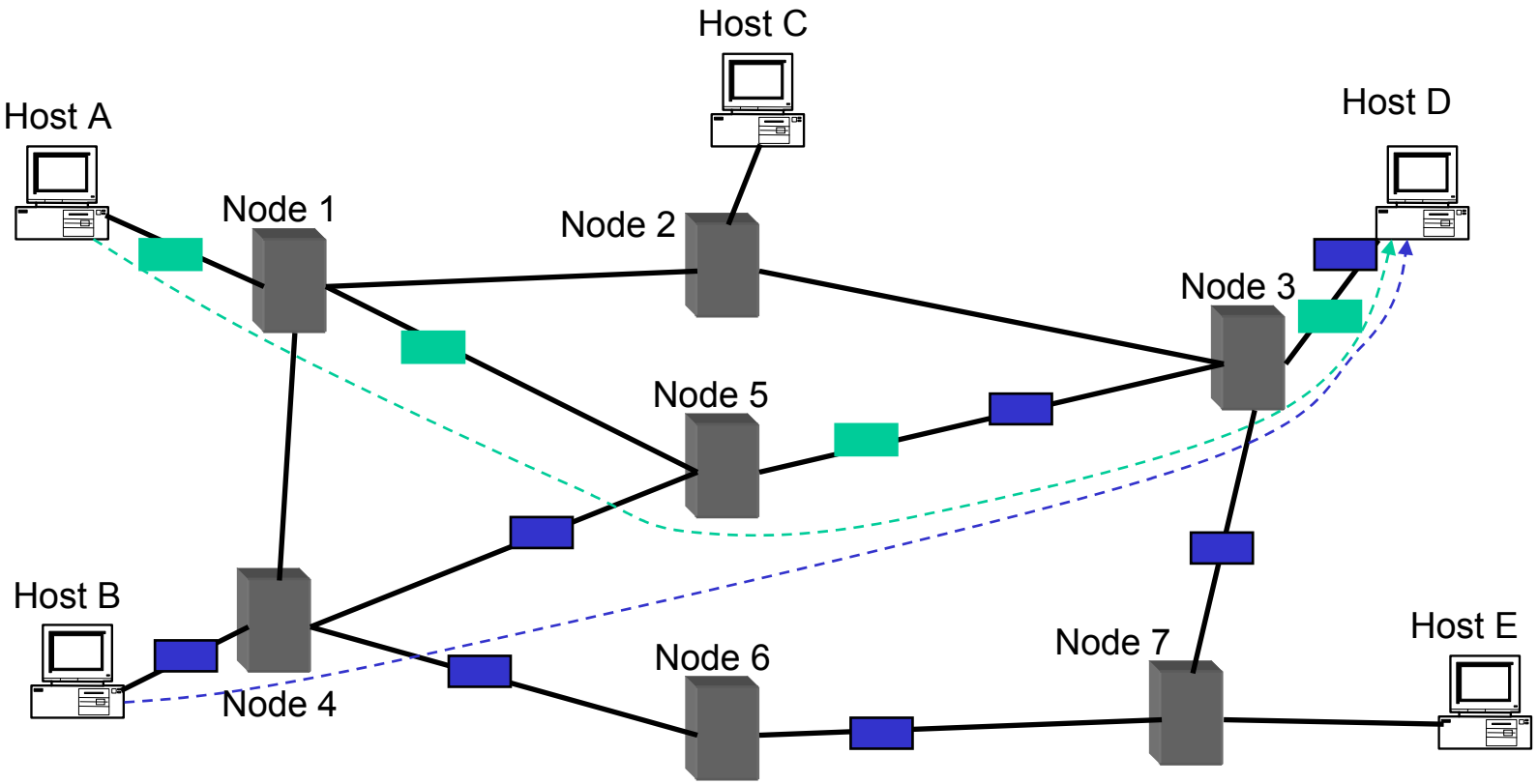
Chuyển mạch Gói:

Hành vi **Giữ và Chuyển**

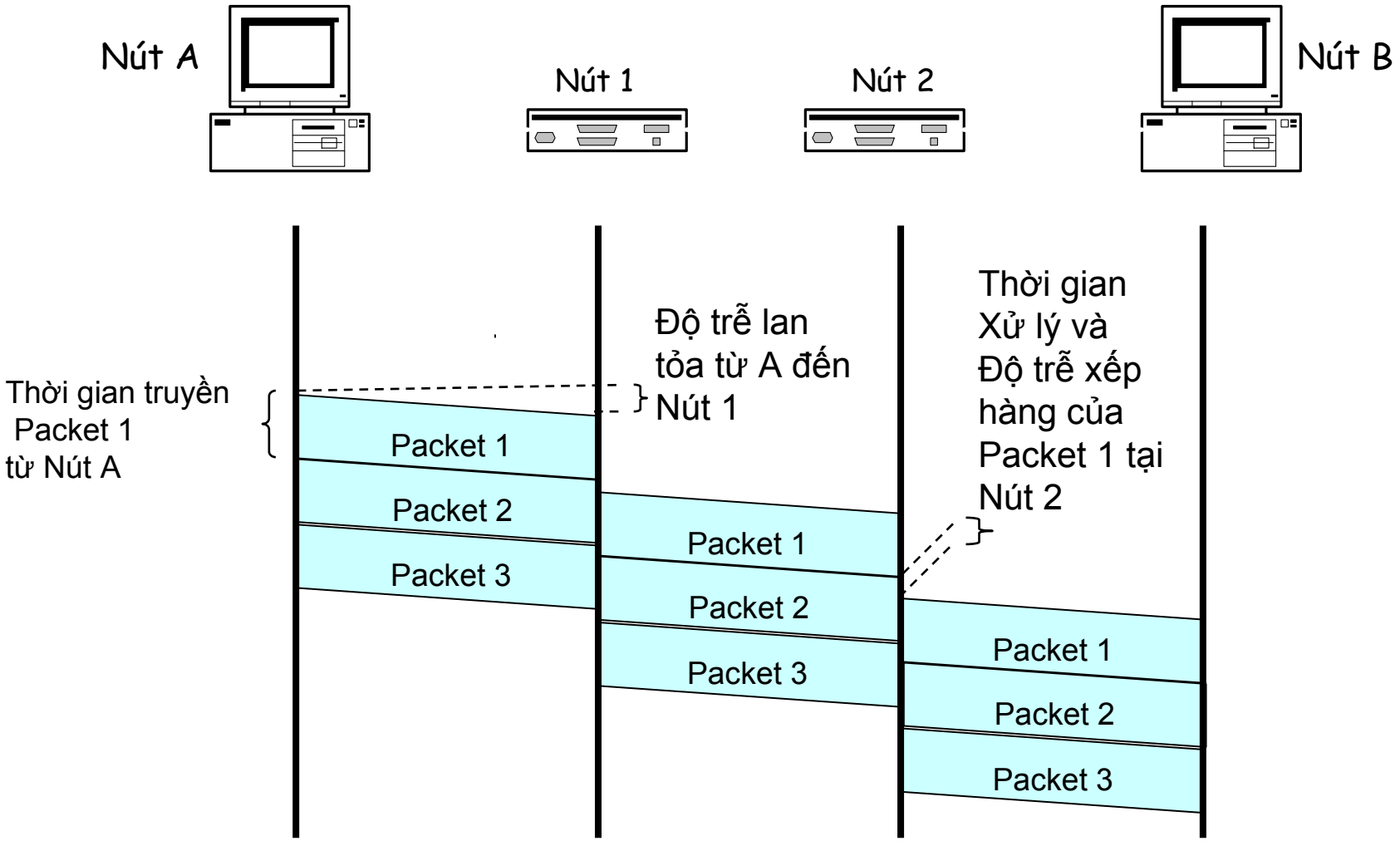
Chuyển Gói : Datagram

- Ví dụ: Mạng IP
- Mỗi gói tin được chuyển độc lập
  - Tiêu đề mỗi gói tin chứa *Địa chỉ đích*
  - Khi nhận được một gói tin, router sẽ nhìn vào địa chỉ đích của gói tin và tìm kiếm trên Bảng định tuyến để xác định nút đến kế tiếp
- Thảo luận
  - Ví dụ nào tương tự trong đời sống con người ?
  - Ưu điểm ?
  - Vấn đề phát sinh ?

Chuyển Gói : Datagram



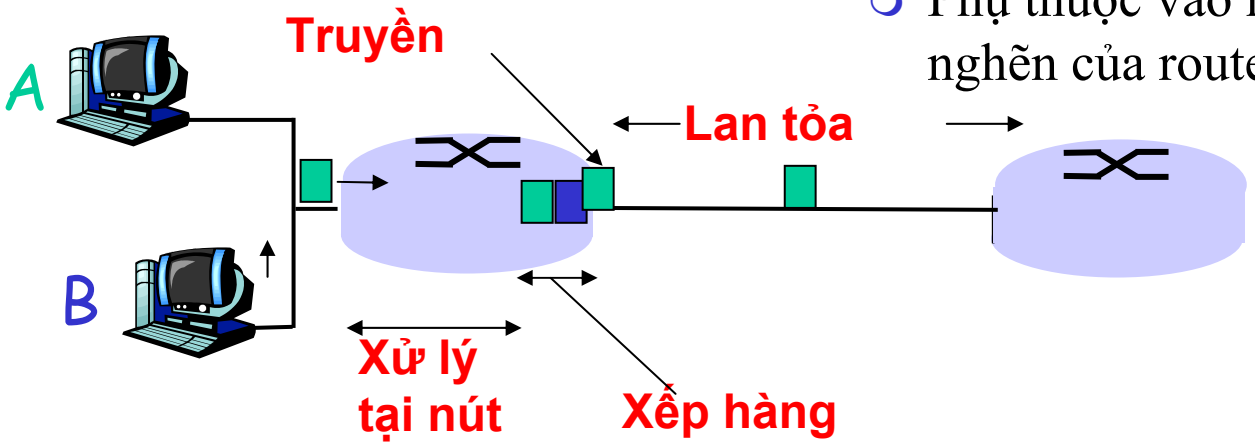
Thời gian trong Chuyển gói Datagram



Độ trễ trong Mạng chuyển mạch Gói

Các gói dữ liệu bị **trễ** trên toàn tuyến đường đến đích

- **4 nguyên nhân** gây trễ tại mỗi chặng
  - Trễ do phải xử lý tại nút:
    - Kiểm tra xem có lỗi bit không?
    - Xác định đường ra
  - Trễ do xếp hàng
    - Đợi tại cổng ra đề truyền đi tiếp
    - Phụ thuộc vào mức độ tắc nghẽn của router



Độ Trễ trong Mạng chuyển mạch Gói

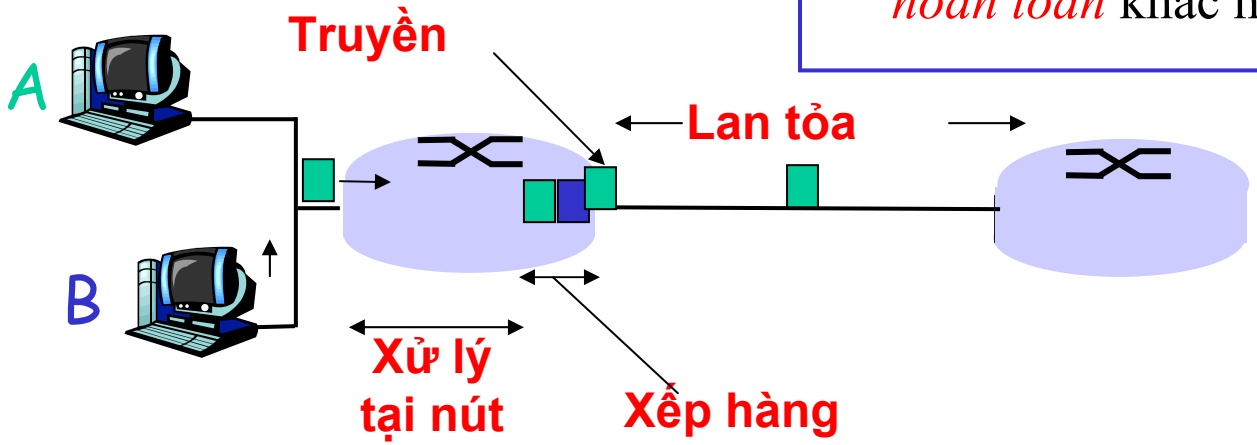
Độ trễ do truyền:

- R=Băng thông kênh truyền (bps)
- L=Kích thước gói dữ liệu (bits)
- Thời gian đề gửi gói tin trên kênh truyền = L/R

Độ trễ lan tỏa

- d = Độ dài kênh truyền
- s = Tốc độ lan tỏa trong môi trường (~2x10^8 m/sec)
- Độ trễ lan tỏa = d/s

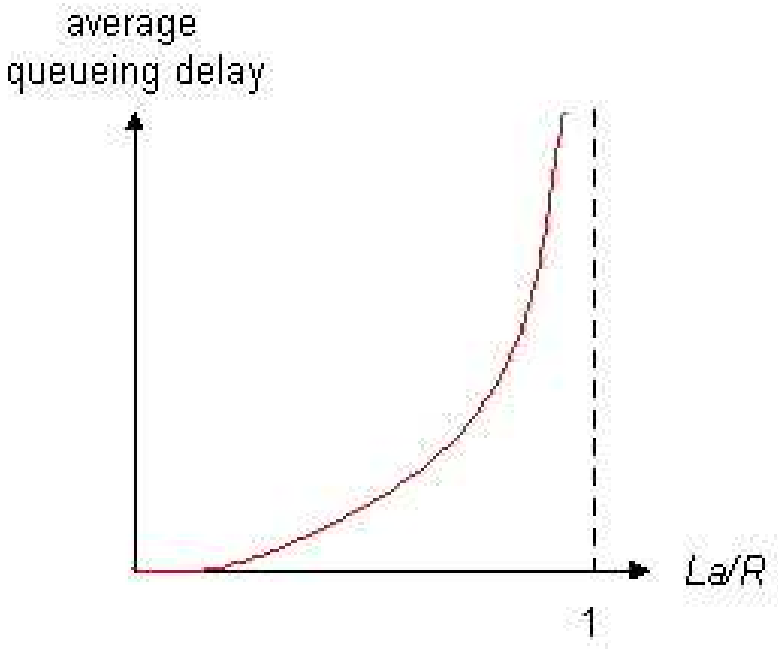
Chú ý: s và R là những đại lượng *hoàn toàn* khác nhau!



Độ trễ Xếp hàng

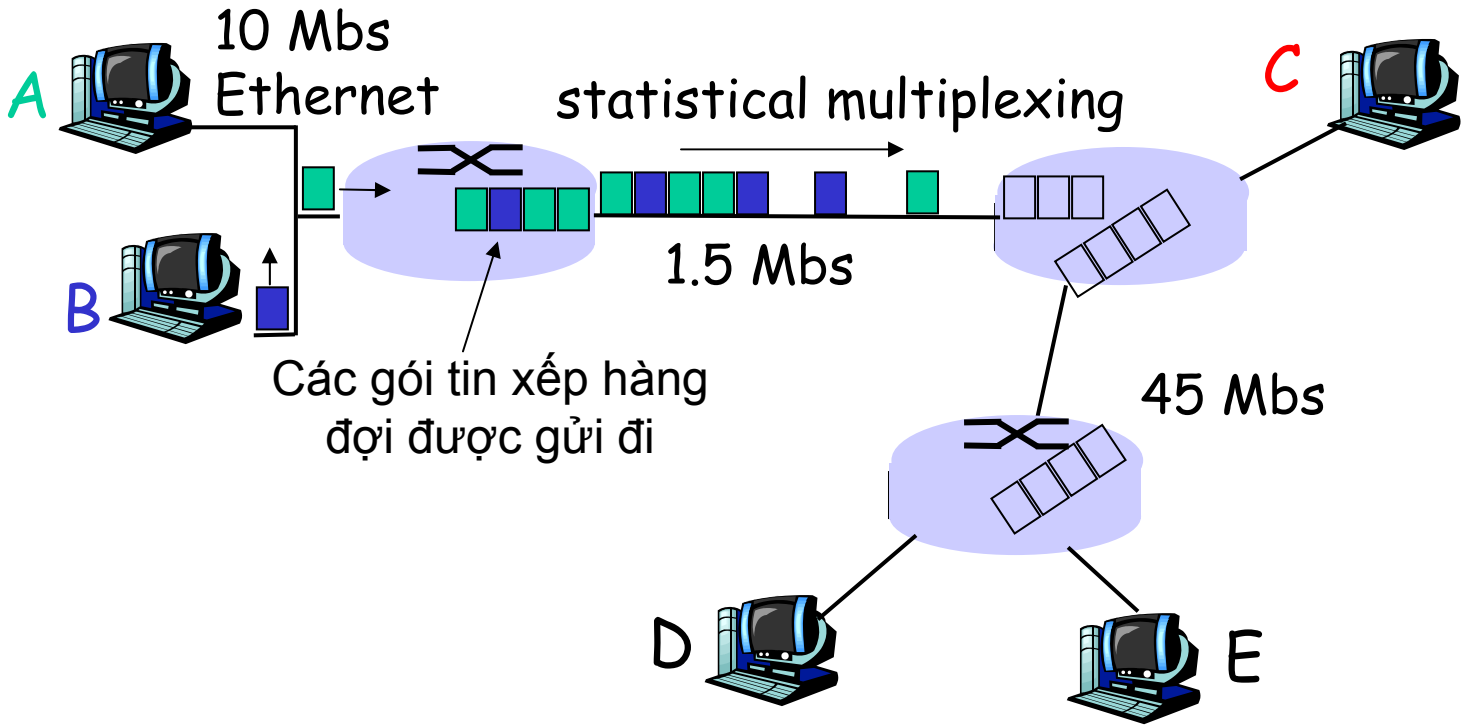
- R=Băng thông kênh truyền (bps)
- L=Kích thước gói dữ liệu (bits)
- a=Tốc độ đến trung bình của các gói tin

Mật độ giao thông = La/R



- La/R ~ 0: Độ trễ trung bình tại hàng đợi nhỏ
- La/R -> 1: Độ trễ lớn dần
- La/R > 1: Lượng dữ liệu đến nhiều hơn khả năng có thể phục vụ. Độ trễ tiến ra vô cùng !

“Lõi”: Chuyển Gói



Chuyển gói khác với **Chuyển mạch**: Đến Khách sạn có đặt chỗ trước ?

- Ví dụ nào khác ?

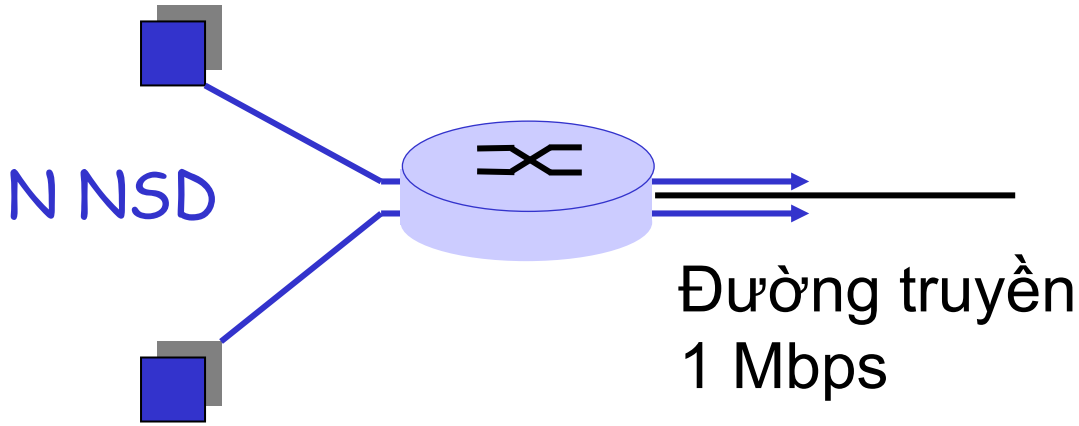
Chuyển Mạch Gói : Định tuyến

- ***Mục tiêu:*** Di chuyển gói tin qua các router từ điểm đầu đến điểm cuối
  - Chương 4 : Các thuật toán lựa chọn đường đi
- **Mạng chuyển mạch gói:**
  - Sử dụng **Địa chỉ đích** để xác định chặng kế tiếp
  - Có thể đi theo nhiều tuyến đường khác nhau
  - Ví dụ : Đi đến ngã tư, Hỏi đường chú Công An
- **Chuyển mạch ảo:**
  - Mỗi gói tin có một thẻ (Định danh mạch ảo), được sử dụng để xác định chặng kế tiếp
  - Tuyến đường cố định được xác định tại thời điểm thiết lập kênh truyền
  - Các routers duy trì trạng thái cho mỗi kênh truyền

So sánh Chuyển Mạch và Chuyển Gói

Chuyển gói cho phép nhiều người sử dụng Mạng !

- 1 Mbit link
- Mỗi người dùng:
  - 100Kbps khi sử dụng
  - Sử dụng trong 10% thời gian
- Chuyển mạch:
  - 10 người sử dụng
- Chuyển gói:
  - Với 35 NSD, xác suất > 10 người đồng thời sử dụng là .004



So sánh Chuyển Mạch và Chuyển Gói

Mạch gói ?

- Ưu điểm khi gửi Khối dữ liệu lớn
  - Chia sẻ tài nguyên
  - Không cần thiết lập kết nối
- **Tắc nghẽn**: Gói tin bị mất hoặc đến trễ
  - Cần giao thức để truyền tin cậy và kiểm soát tắc nghẽn
- **Làm sao để cung cấp hành vi giống chuyển mạch ?**
  - Đảm bảo băng thông cho các ứng dụng truyền thông đa phương tiện vẫn còn là những vấn đề chưa giải quyết.

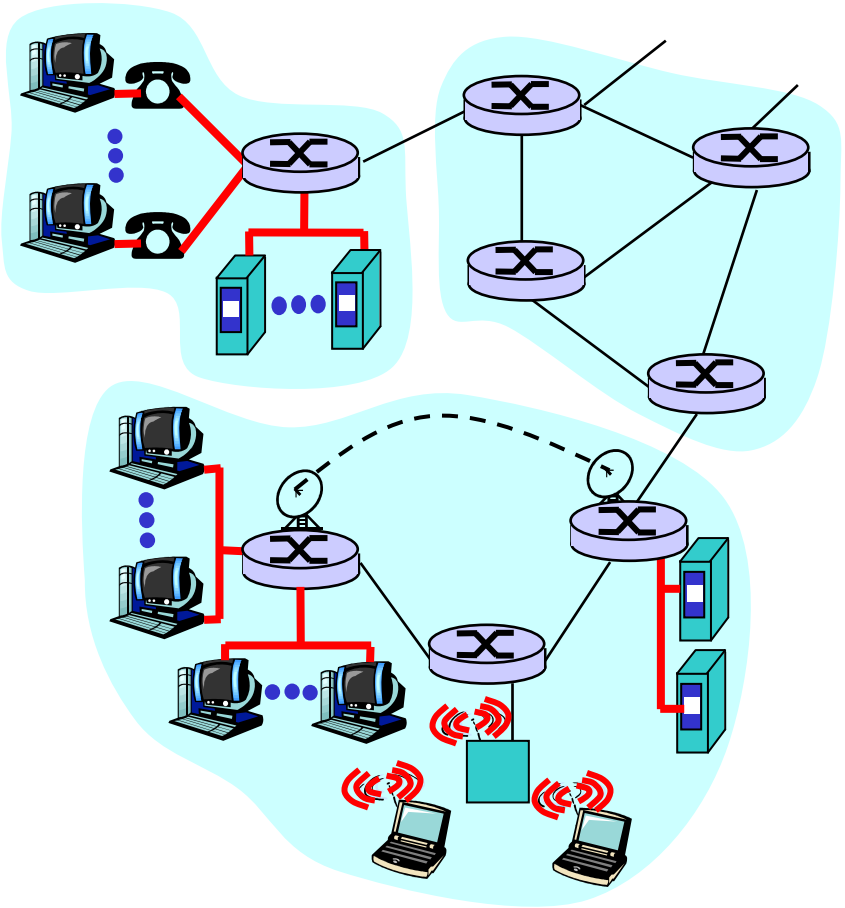
Truy cập vào Mạng và Môi trường Vật lý

Làm thế nào để kết nối thiết bị đầu cuối vào các Router ?

- Truy cập từ nhà riêng
- Truy cập từ cơ quan, trường học
- Truy cập qua mạng di động

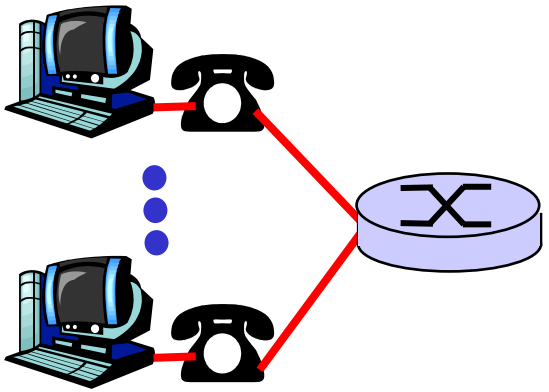
Chú ý:

- Băng thông của đường truyền (bps)
- Chia sẻ hay Dùng riêng?

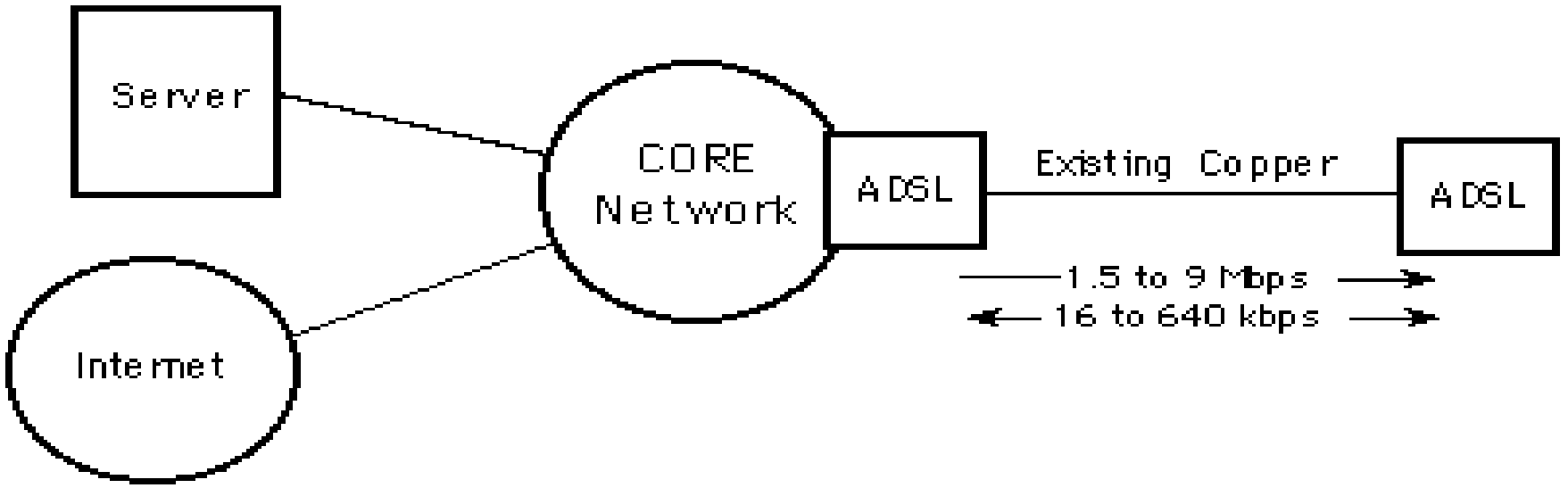


Truy cập từ nhà : Điểm nối Điểm

- Dialup qua modem
  - Tốc độ kết nối trực tiếp đến router là 56kbps
- ISDN: Intergrated Services Digital Network: tốc độ 128Kbps trên tất cả kết nối đến router
- ADSL: Asymmetric Digital Subscriber Line
  - Có thể tới 1 Mbps nhà => router
  - Có thể tới 8 Mbps router => nhà
  - Sử dụng ADSL ?



Local Access: ADSL

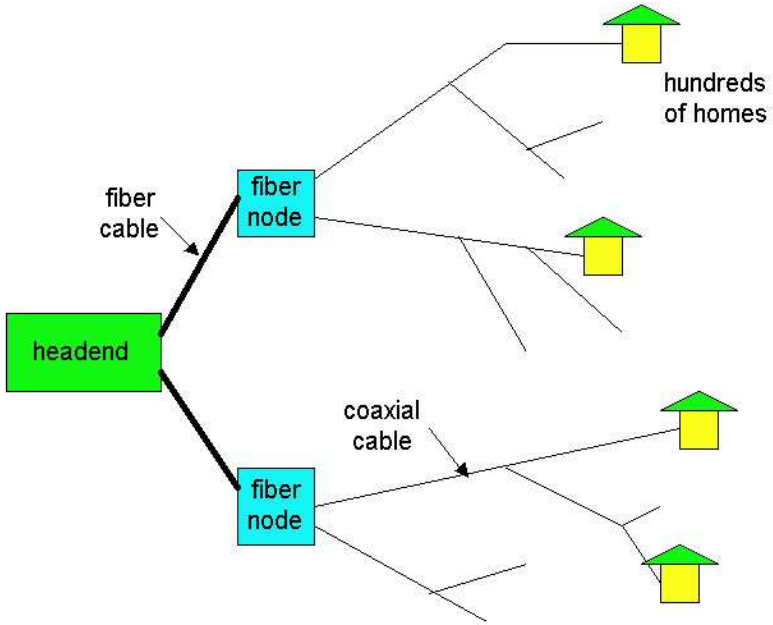


ADSL Connection

- Asymmetrical Digital Subscriber Loop (ADSL)

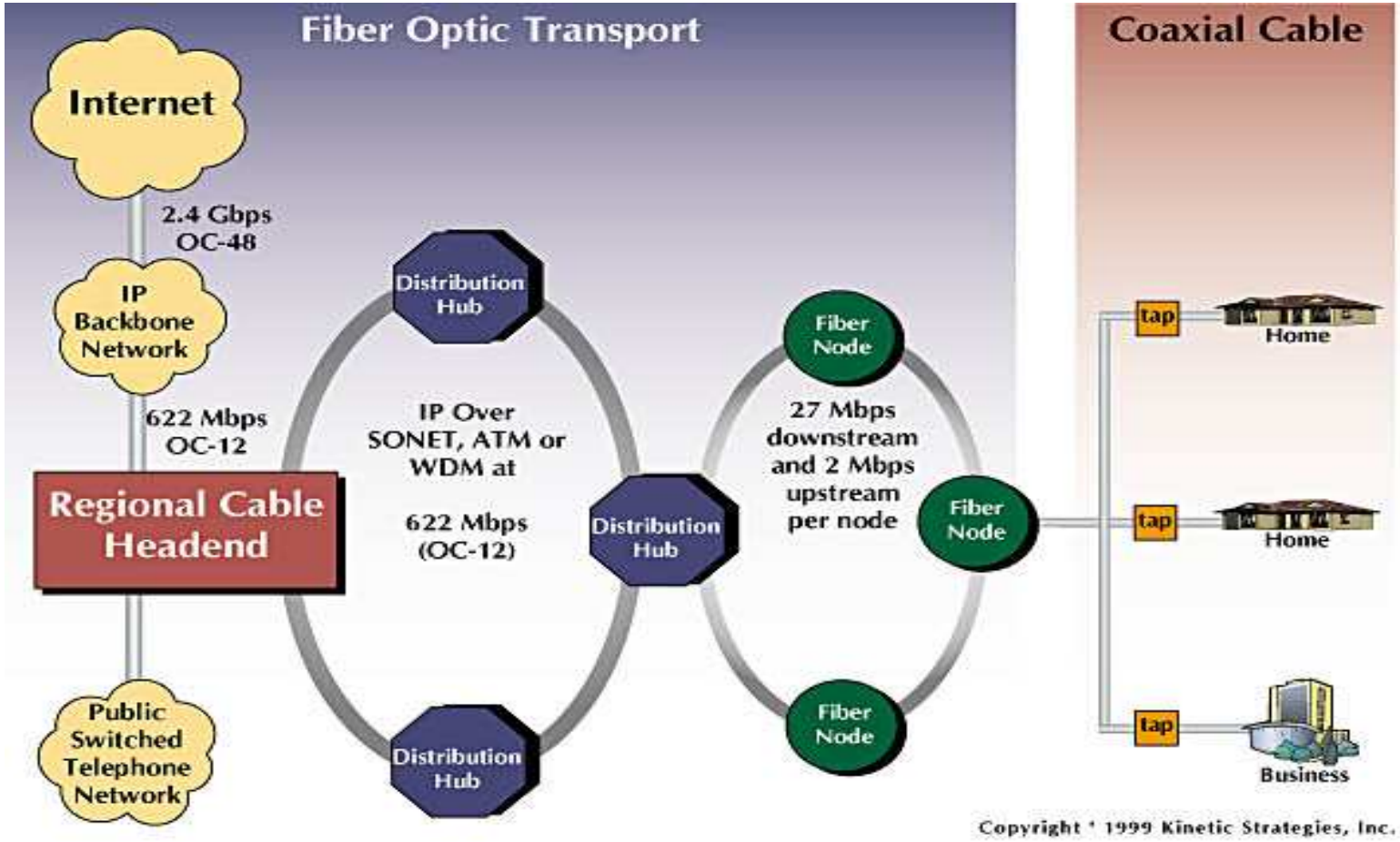
Truy cập từ nhà: cable modem

- HFC: hybrid fiber coax
  - Không đối xứng: 10Mbps download, 1 Mbps upload
- Mạng các cáp và cáp quang nối từ nhà đến router của IS
  - Chia sẻ router giữa nhiều nhà
  - Vấn đề: Tắc nghẽn và Mở rộ
- Cài đặt: Các công ty cáp



Local Access: Cable Modems

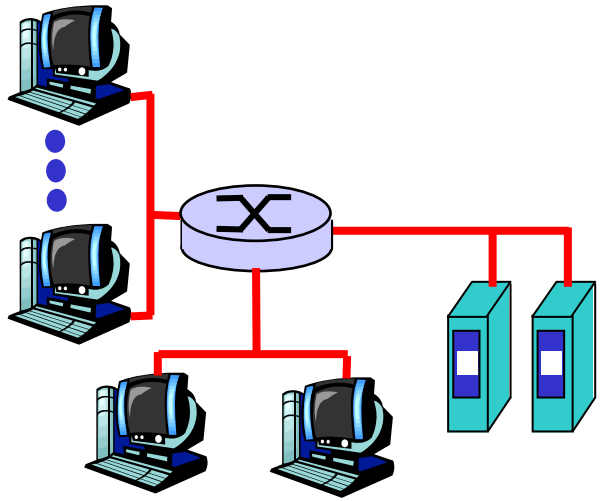
Also called Hybrid Fiber-coaxial Cable (HFC)



- Fiber node: 500 - 1K Ngôi nhà
- Distribution hub: 20K - 40 K Ngôi nhà
- Regional headend: 200 K - 400 K Ngôi nhà

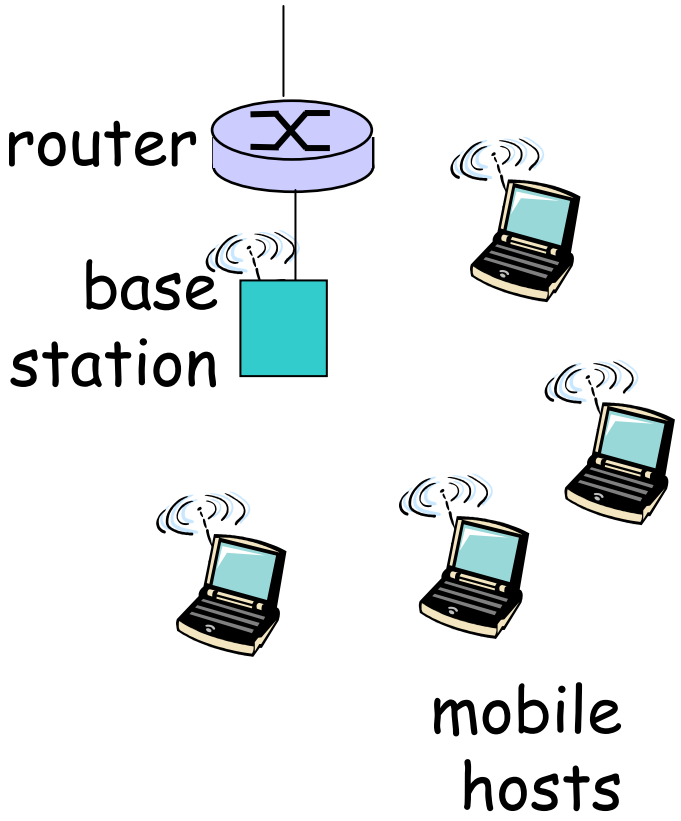
Truy cập từ Cơ quan : Mạng Cục bộ

- **Mạng cục bộ (LAN)** của cơ quan kết nối các máy tính đầu cuối vào router
- Ethernet:
  - Cáp dùng riêng hoặc chia sẻ kết nối máy tính với các router
  - 10 Mbs, 100Mbps, Gigabit Ethernet
- Sử dụng tại: cơ quan, nhà riêng (đã có nhiều)
- LAN: Chương 5



Truy cập qua mạng không dây

- Môi trường không dây dùng chung sẽ kết nối thiết bị đầu cuối với router
- **LAN không dây:**
  - Băng tần radio thay thế dây dẫn
  - Ví dụ : Lucent Wavelan 10 Mbps
- **Không dây trên diện rộng**
  - CDPD: Truy cập không dây tới router của ISP thông qua mạng cellular



Môi trường truyền Vật lý

- Kết nối vật lý: Bit chứa dữ liệu lan tỏa trong môi trường kết nối
- Môi trường định hướng:
  - Lan tỏa trong môi trường Đặc: dây đồng, cáp quang
- Môi trường không định hướng :
  - Lan tỏa trong môi trường Tự do: sóng radio

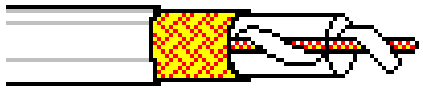
Cáp đồng trục (TP)

- Hai loại
  - Loại 3: Cáp điện thoại, 10 Mbps Ethernet
  - Loại 5: 100Mbps Ehernet

Môi trường truyền Vật lý: Dây đồng, Cáp quang

Cáp đồng

- Dây (môi trường truyền dẫn) nằm trong dây (lớp vỏ)
  - baseband: một kênh trên một cáp
  - broadband: nhiều kênh trên một cáp
- Truyền trên cả 2 hướng
- Chủ yếu sử dụng trong 10Mbs Ethernet



Cáp quang:

- Sợi quang là môi trường truyền dẫn xung ánh sáng
- Tốc độ truyền cao:
  - 100Mbps Ethernet
  - Tốc độ truyền rất cao giữa 2 điểm (5 Gps)
- Tỷ lệ lỗi rất thấp

Môi trường truyền Vật lý: Radio

- Tín hiệu lan truyền trong môi trường sóng điện từ
- Không có "dây" vật lý
- Hai hướng
- Môi trường lan tỏa bị tác động bởi:
  - Phản xạ
  - Các đối tượng cản
  - Giao thoa

Các kiểu sóng Radio:

- microwave
  - Tốc độ truyền 45 Mbps
- LAN (waveLAN)
  - 2Mbps, 11Mbps
- wide-area (cellular)
  - CDPD, 10's Kbps
- Vệ tinh
  - Kênh 50Mbps
  - Độ trễ 270 Msec

“Tầng” các Giao thức

Mạng cực kỳ Phức tạp !

- Quá nhiều thứ :
  - Máy tính
  - Router
  - Các môi trường truyền cực kỳ đa dạng
  - Ứng dụng
  - Giao thức
  - Phần cứng, phần mềm

Vấn đề :

Phải xây dựng mạng một cách có ***Tổ chức***

Trong xã hội loài người có cần ***Tổ chức*** không?

Ví dụ trong ngành Hàng Không

- Các bước tuần tự

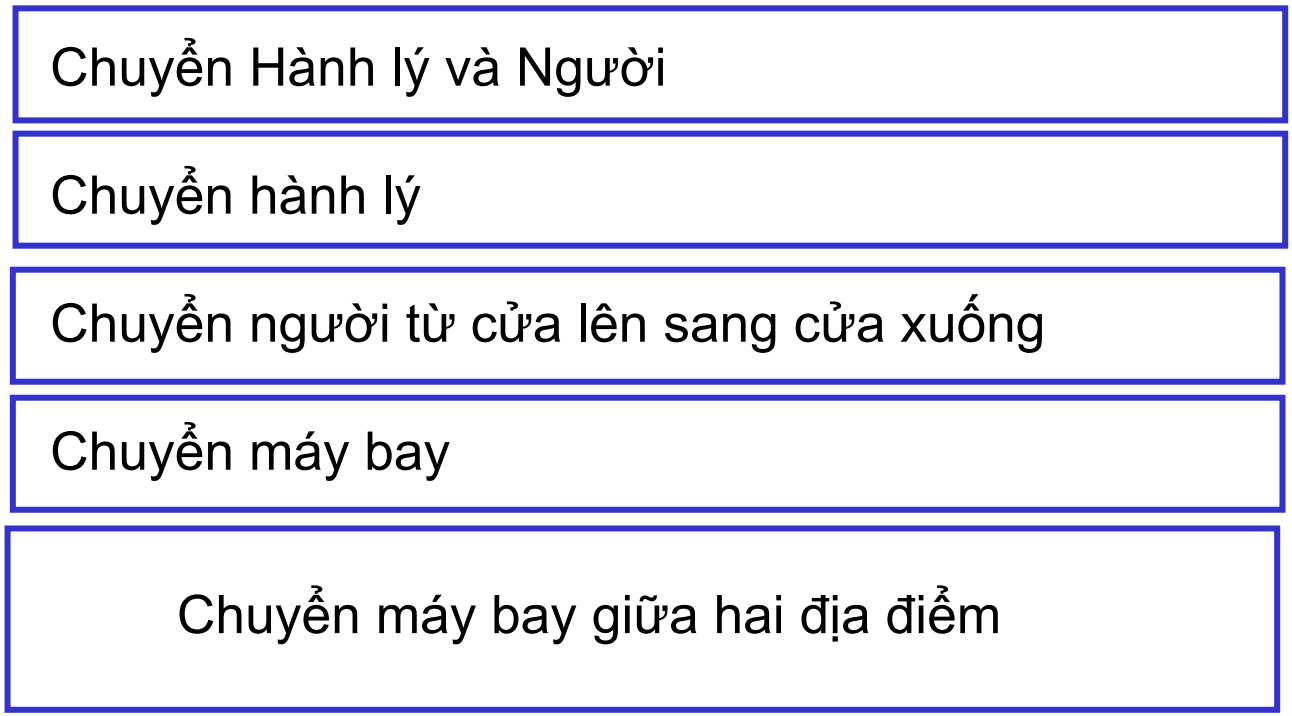
Tổ chức trong Hàng không : cách nhìn khác

Mua vé	Trả vé cho tài vụ
Kiểm tra Hành lý	Lấy Hành lý
Lên Máy bay	Xuống Máy bay
Máy bay cất cánh	Máy bay hạ cánh
Máy bay bay	Máy bay bay
Máy bay bay	

Các tầng: Mỗi tầng cài đặt một dịch vụ

- Thông qua hoạt động nội tại của tầng
- Dựa trên dịch vụ do tầng bên dưới cung cấp

Phân tầng trong Hàng không: Dịch vụ



Chức năng các tầng được cài đặt Phân tán

Gửi thư

From : Tôn Ngộ Không,
To : Doremon, Nhật Bản
"Bánh rán"

From : Linh Sơn, Ấn Độ
To : Tokyo, Nhật Bản

From : Tôn Ngộ Không,
To : Doremon, Nhật Bản
"Bánh rán"

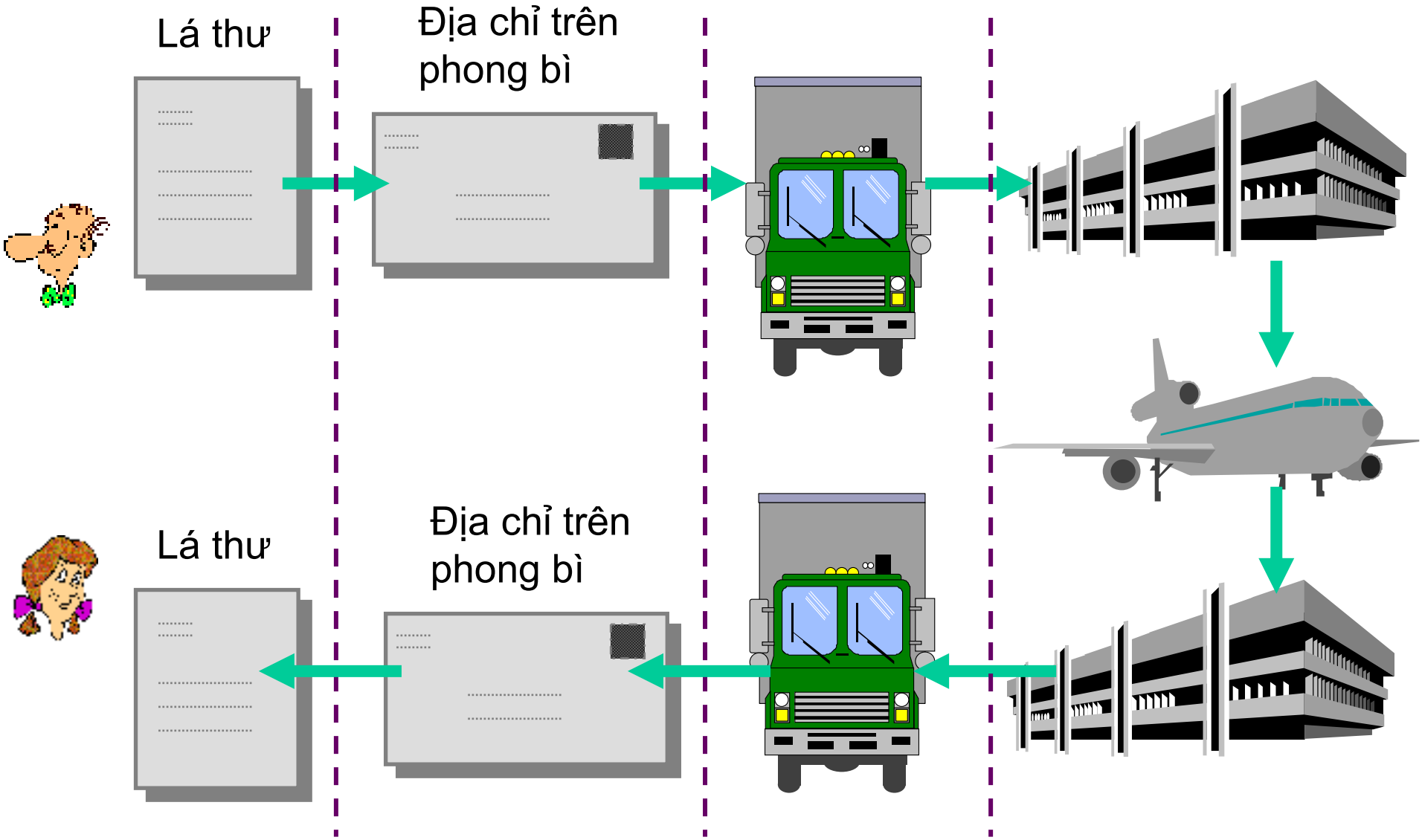
Bưu cục Linh Sơn

Bưu cục Tokyo

Bưu cục Ấn Độ

Bưu cục Nhật Bản

Gửi Thư – Phân tầng



Định nghĩa Phân tầng?

- Kỹ thuật tổ chức Hệ thống Mạng thành các thức thể độc lập về mặt logic nhưng nối tiếp nhau thành một chuỗi sao cho dịch vụ do thực thể này cung cấp hoàn toàn dựa trên dịch vụ do thực thể đằng trước trong chuỗi (tầng thấp hơn) cung cấp.

Khái niệm Mô hình Hệ thống mở

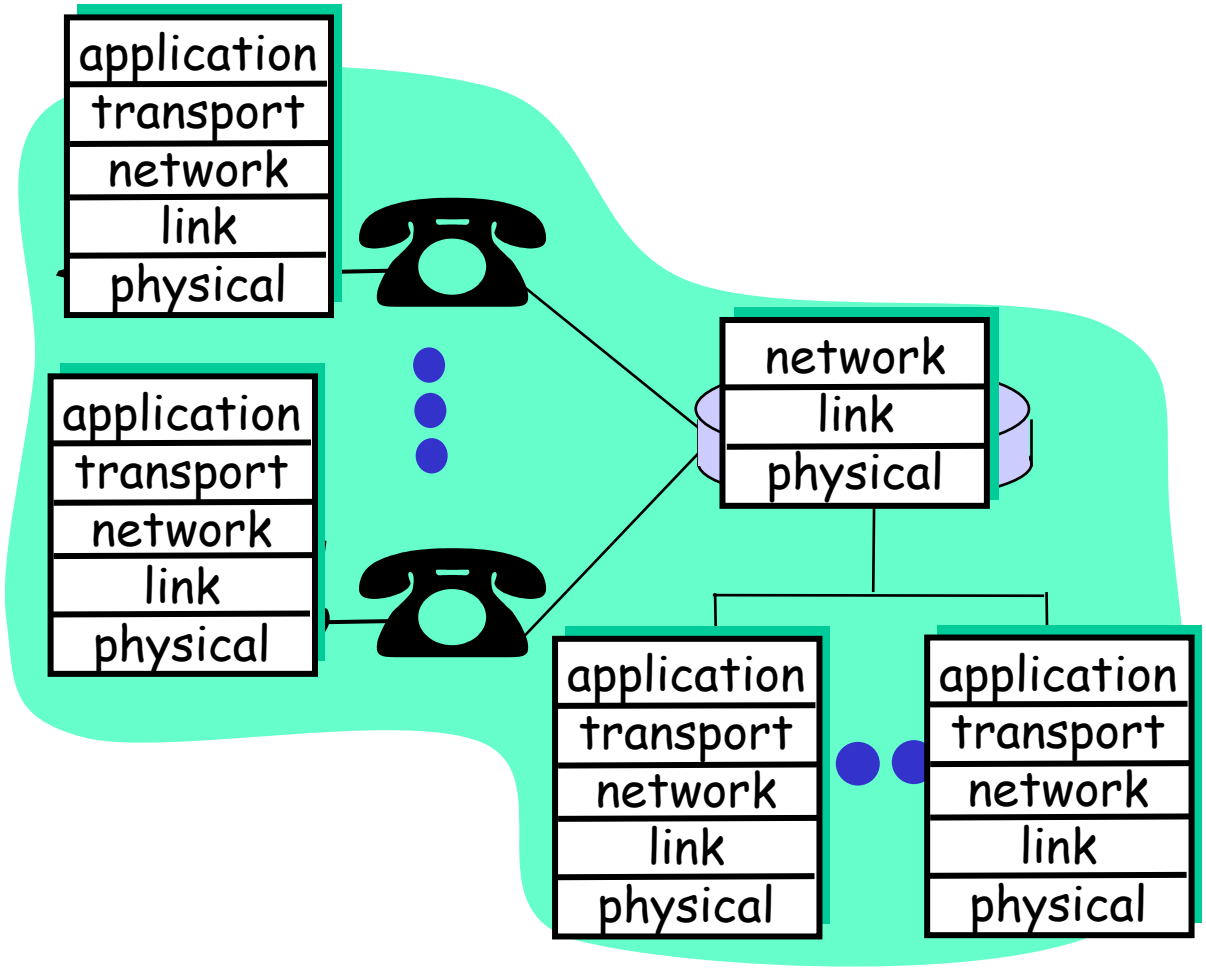
- **ISO** – **I**nternational **S**tandard **O**rganization (Tổ chức Tiêu chuẩn Quốc tế)
- **OSI** – **O**pen **S**ystem **I**nterconnection (Mô hình Kết nối các Hệ thống mở)

- Dịch vụ – Tầng sẽ làm gì ?
- Giao diện – Làm thế nào để sử dụng Dịch vụ ?
- Giao thức – Cài đặt các tầng như thế nào ?
  - Tập hợp các quy tắc và khuôn dạng mà hai bên tham gia truyền thông phải tuân thủ

Phân tầng : Kênh truyền Logic

Các tầng:

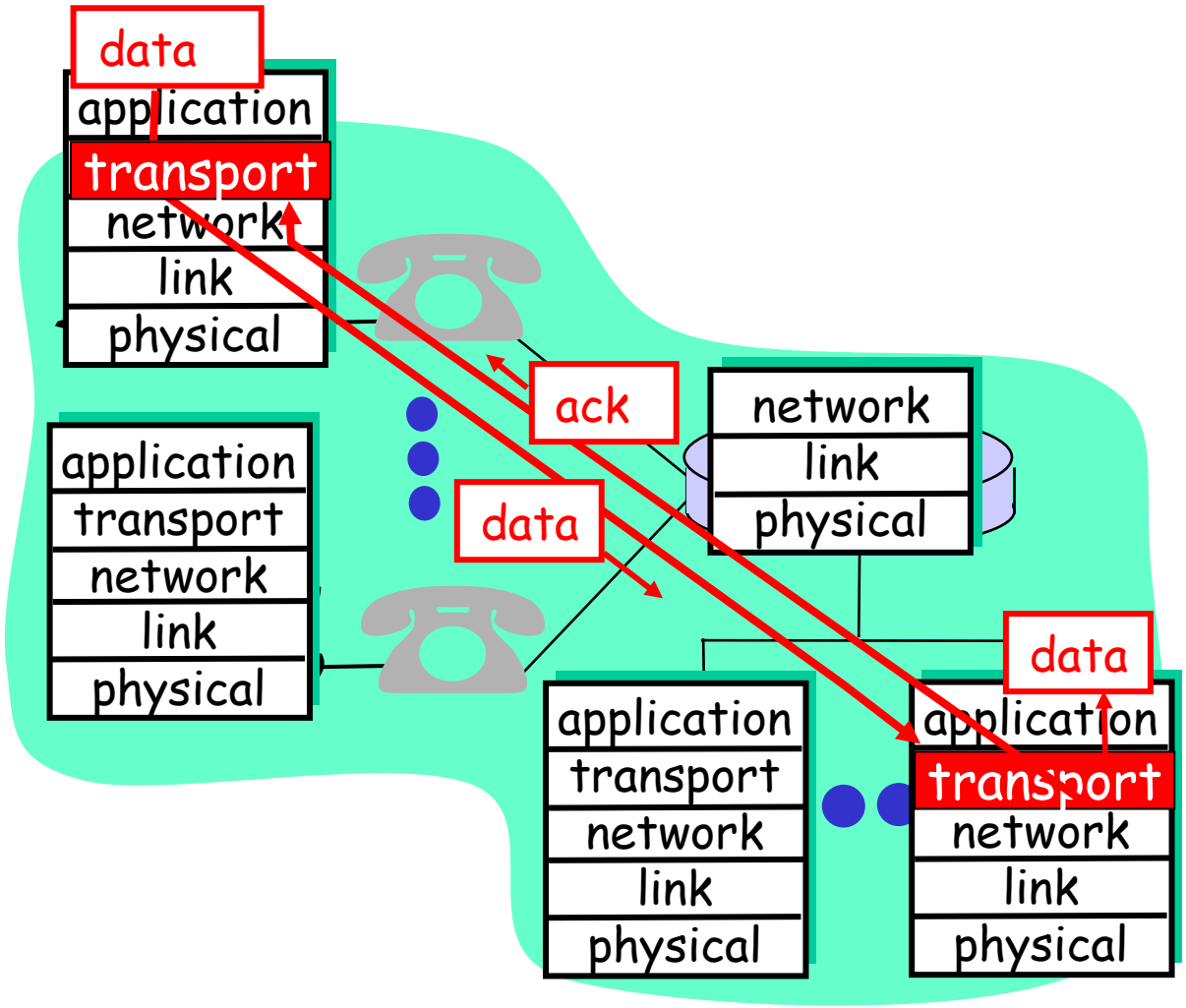
- Phân tán
- Các "thực thể" cài đặt chức năng của tầng nằm trên các thiết bị
- Các "thực thể" gửi và nhận thông điệp từ các đối tác tương đương



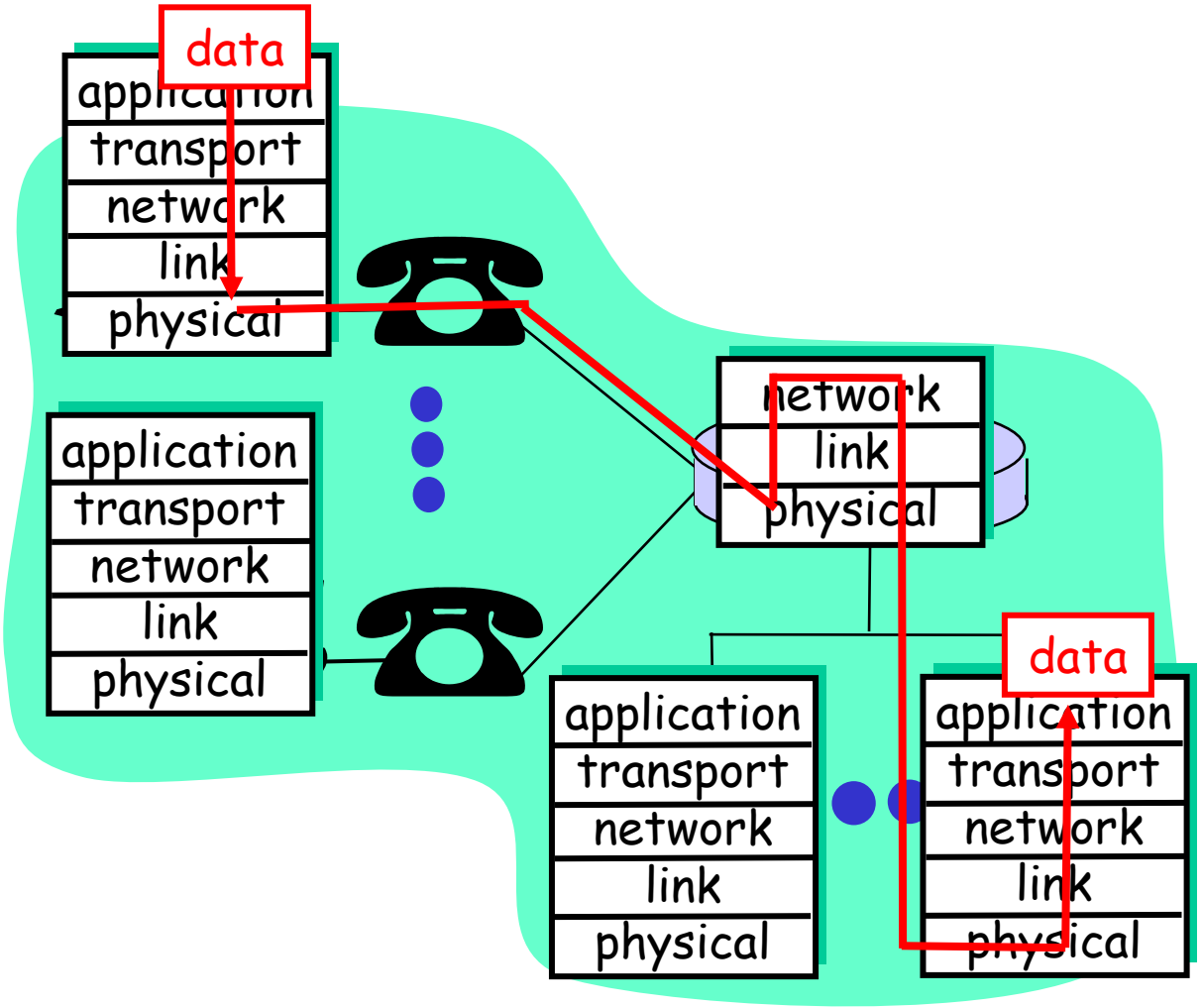
Phân tầng : Kênh truyền Logic

Ví dụ: Tầng giao vận

- Lấy dữ liệu từ tầng Ứng dụng
- Bổ sung Địa chỉ và các Thông tin Kiểm tra Tính Tin cậy để tạo thành "datagram"
- Gửi datagram tới đối tác bên kia
- Đợi đối tác bên kia gửi biên nhận
- Ví dụ: Hệ thống Bưu cục



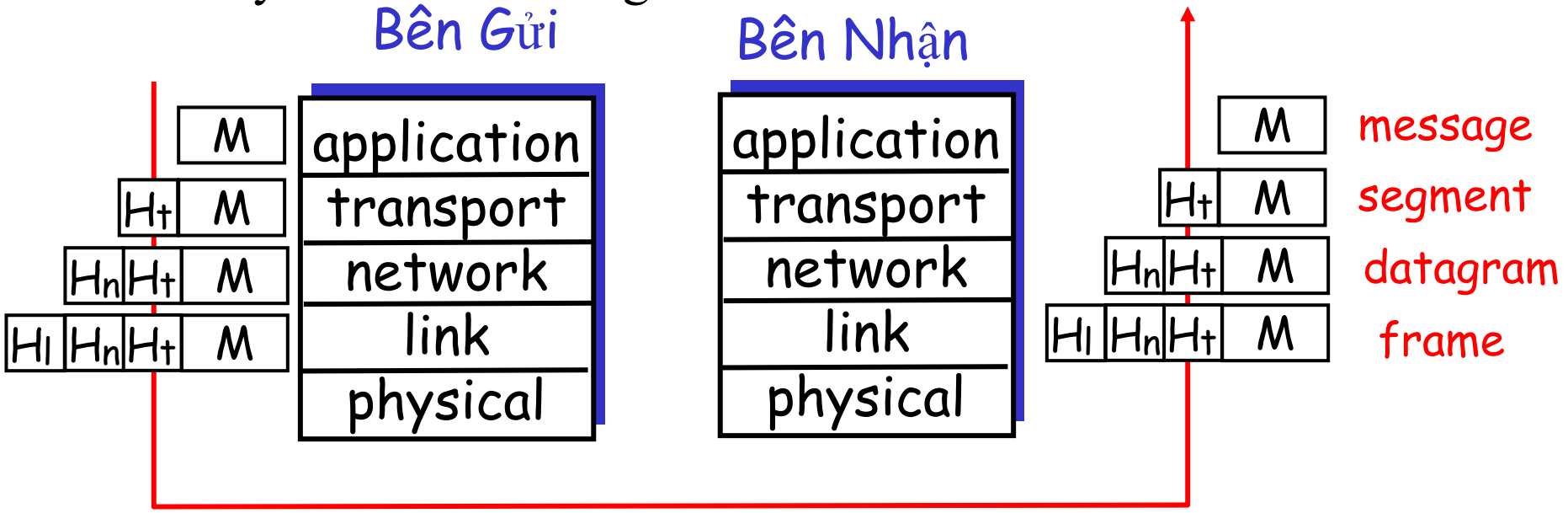
Phân tầng : Đường truyền Vật lý



Dữ liệu đi qua Hệ thống giao thức

Các tầng lấy dữ liệu từ tầng bên trên

- Bổ sung thông tin ***tiêu đề*** để tạo ra đơn vị dữ liệu mới (***PDU***)
- Chuyển PDU cho tầng bên dưới

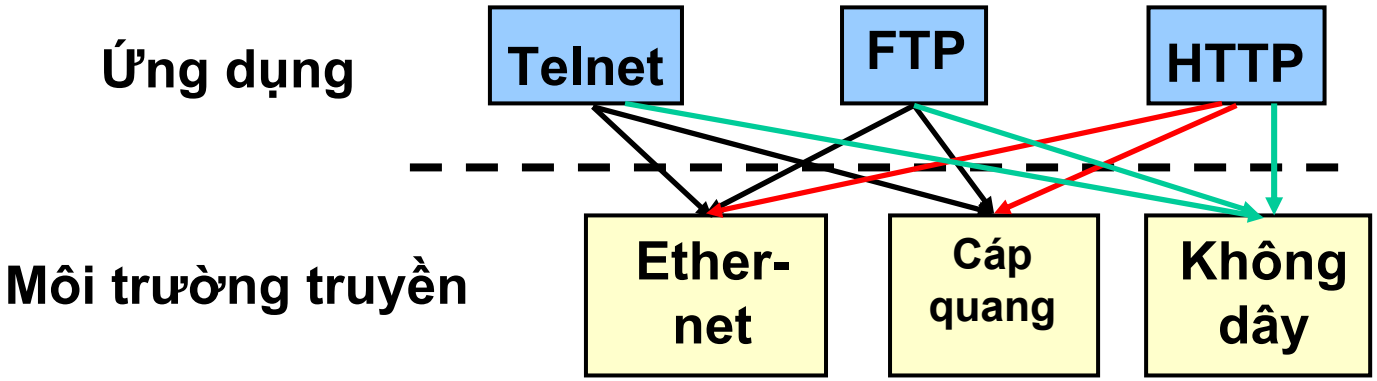


Tại sao phân tầng ?

Với các Hệ thống cực kỳ phức tạp:

- Cấu trúc tường minh cho phép xác định cụ thể quan hệ giữa các thành phần một cách rõ ràng
  - Mô hình tham chiếu phân tầng đã trình bày trên
- Chia nhỏ chi phép bảo trì, nâng cấp dễ dàng
  - Thay thế hoạt động nội tại của một tầng không ảnh hưởng đến toàn bộ Hệ thống
  - Ví dụ: thay đổi thủ tục kiểm tra hành lý không ảnh hưởng đến các quy trình khác
- ***NHƯNG*** nhược điểm của phân tầng ?

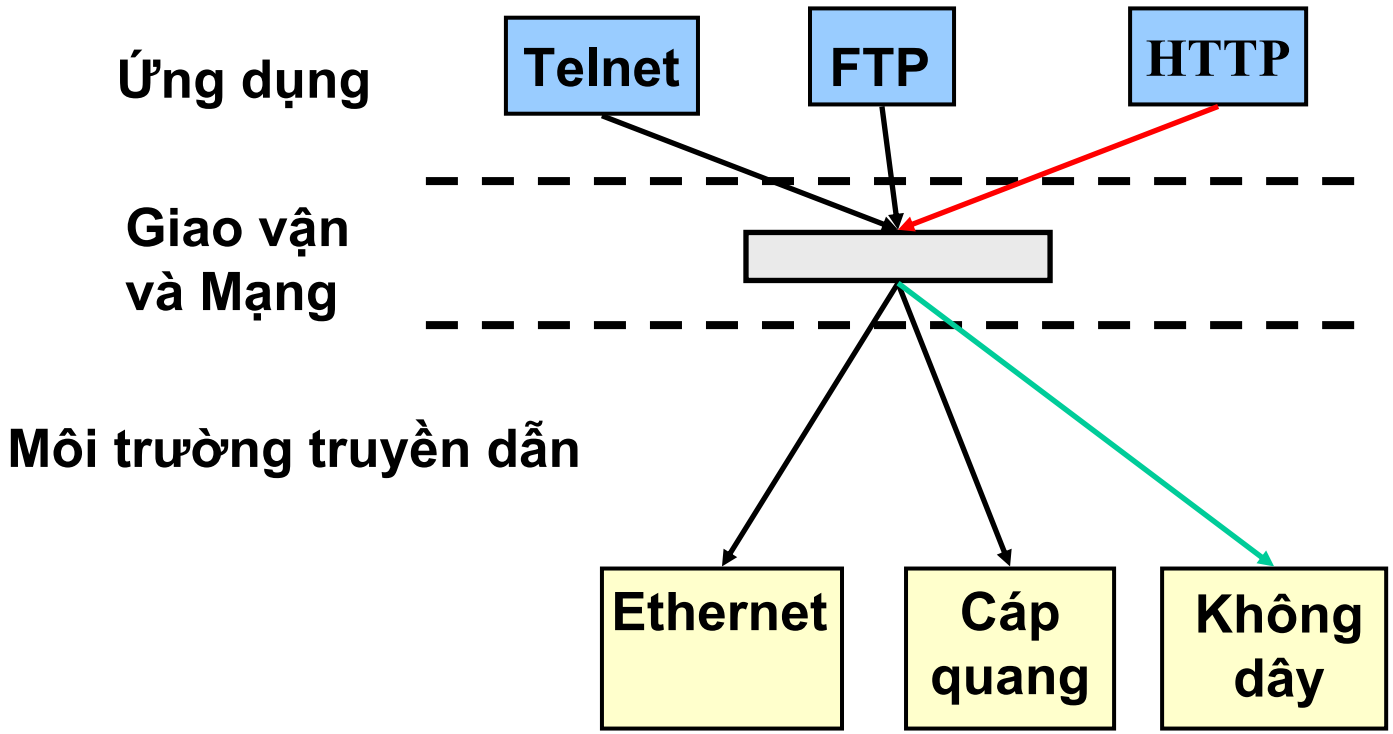
Giả sử Không Phân tầng



- Nếu không phân tầng: Khi có công nghệ Mạng mới phải viết lại các Ứng dụng.
  - ***Cực kỳ Tốn kém !***

Ví dụ : Lợi ích của Phân tầng

- Tầng ở giữa : Cung cấp lớp trừu tượng chung cho tất cả các Công nghệ truyền dẫn khác nhau



Nhược điểm ?

The End-to-End Arguments

Chức năng Ứng dụng chỉ có thể cài đặt một cách chính xác và đúng đắn với sự trợ giúp từ chính các ứng dụng chạy tại các thiết bị đầu cuối của Hệ thống truyền thông. Do vậy Không thể cài đặt chức năng như một đặc tính của Hệ thống truyền thông

The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication systems. Therefore, providing that questioned function as a feature of the communications systems itself is not possible.

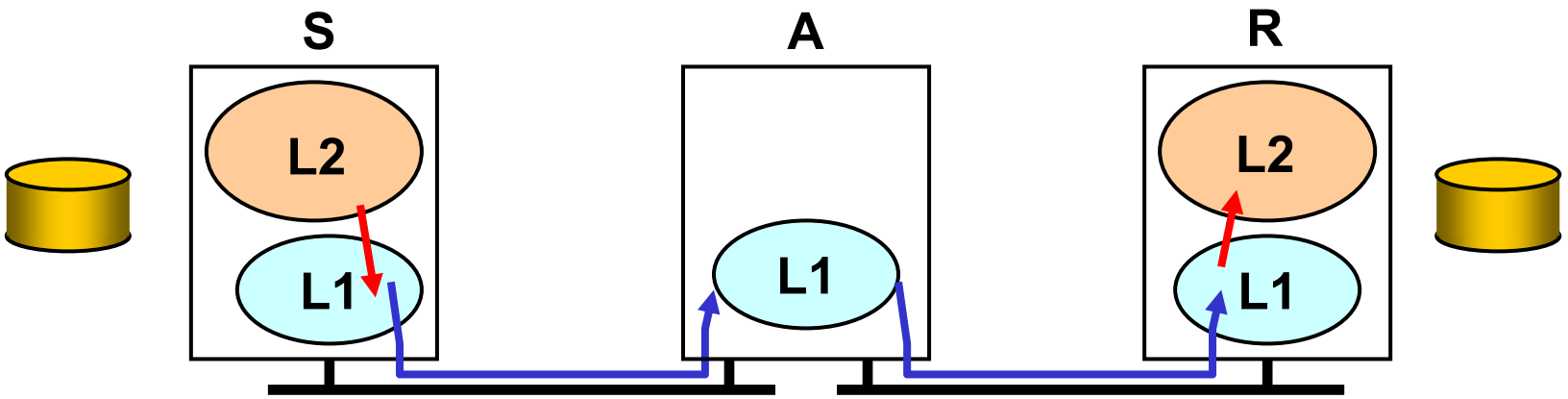
J. Saltzer, D. Reed, and D. Clark, 1984

Ý nghĩa của Nguyên lý này

- Ứng dụng biết rõ nhất về các yêu cầu của mình. Do đó nên đặt các chức năng ở tầng cao nhất có thể

- Suy nghĩ thật cẩn trọng nếu cài đặt chức năng ở tầng thấp hơn, kể cả khi bạn nghĩ rằng điều này hữu ích cho Ứng dụng

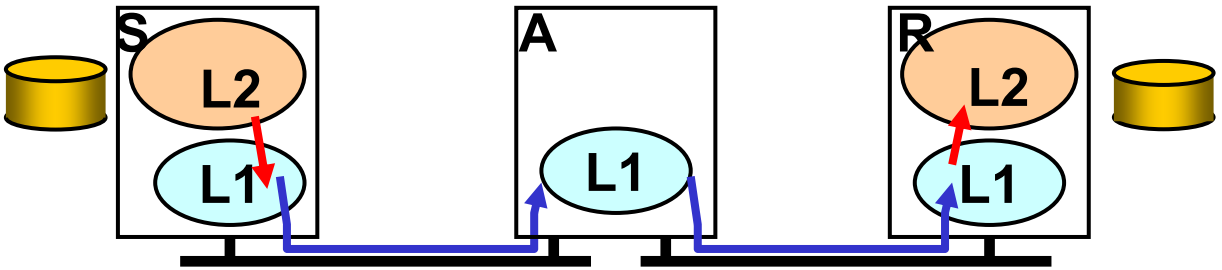
Ví dụ : Cài đặt tính Tin cậy ở đâu ?



- Giải pháp 1: Tầng mạng (tầng thấp L1) cung cấp tính tin cậy, nghĩa là mỗi chặng đều phải truyền tin cậy.
- Giải pháp 2: Hai điểm đầu cuối (tầng cao L2) cung cấp tính tin cậy, nghĩa là phải kiểm tra tại hai đầu và có thể gửi lại.

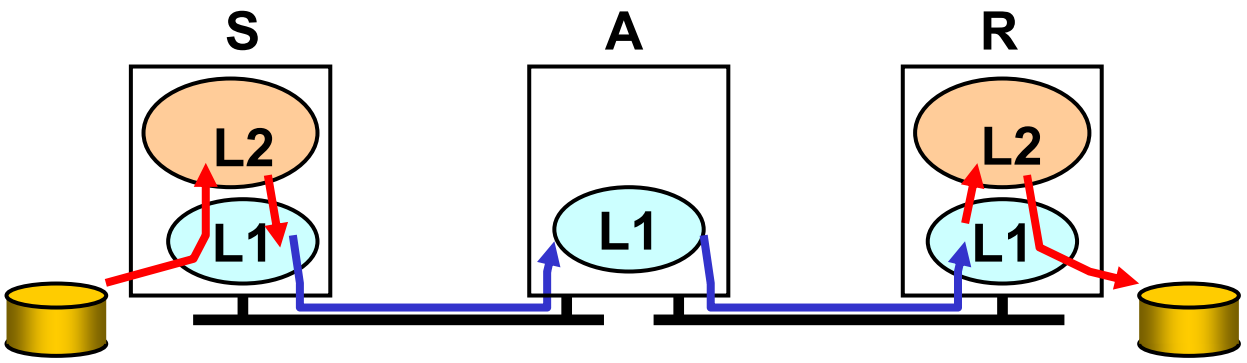
Tại sao cài đặt Tính tin cậy ở Tầng cao ?

- Tầng thấp hoàn toàn không thể cung cấp khả năng này
  - Kiểu gì bên Nhận cũng phải kiểm tra lại !
- Nếu cài đặt ở tầng thấp
  - Tăng độ phức tạp (khó lập trình hơn)
  - Tăng chi phí phụ trội (hiệu suất/ giá thành) ở các tầng thấp – tất cả tầng cao phải trả giá
- Tầng cao
  - Hiểu rõ điều kiện hơn và có thể lựa chọn giải pháp cài đặt tối ưu



Tại sao cài đặt Tính tin cậy ở Tầng thấp ?

- Tăng Hiệu suất, ví dụ nếu từ A đến R có tỷ lệ lỗi rất cao, đặt chức năng đảm bảo tin cậy ở L1 sẽ:
  - Hiệu quả hơn
  - Giảm độ trễ
    - Đặc biệt khi kênh truyền từ S đến A có độ trễ lớn
- Chia sẻ các đoạn mã dùng chung, ví dụ tính tin cậy được nhiều ứng dụng yêu cầu



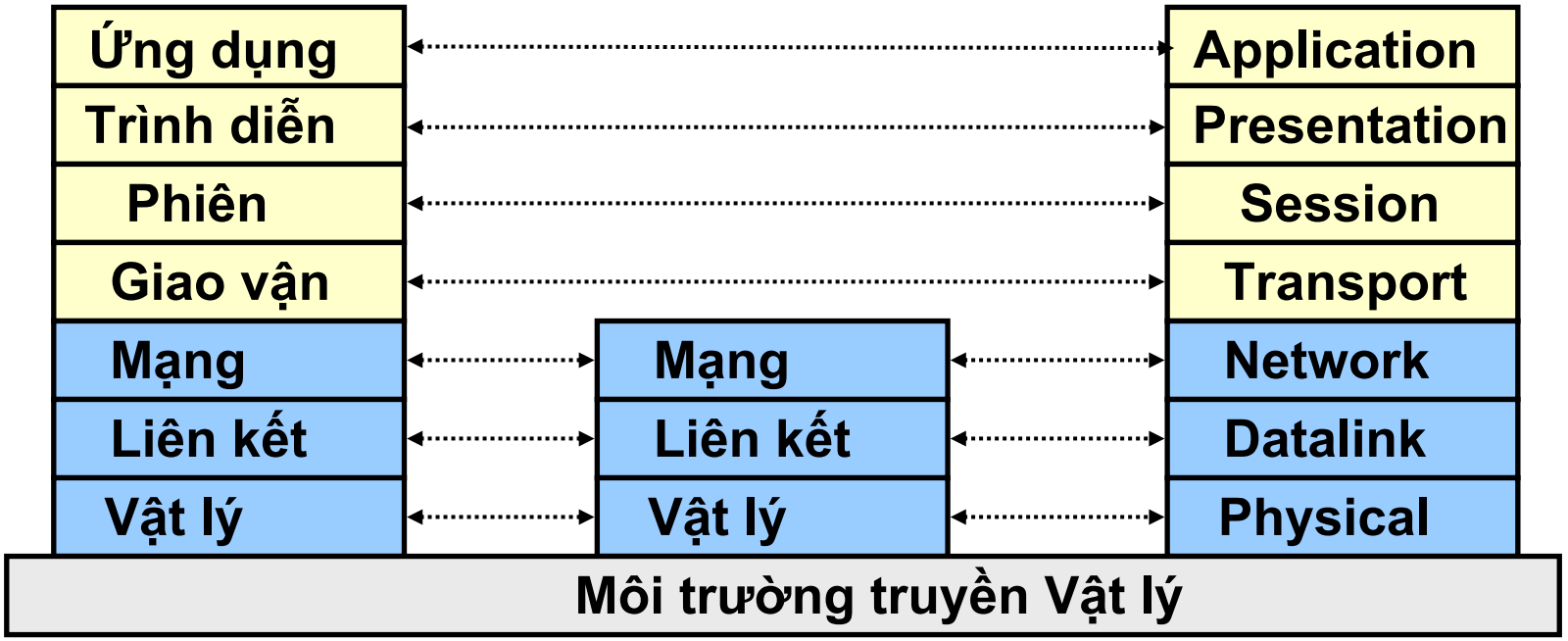
Thỏa hiệp

- Ứng dụng biết nhiều thông tin về dữ liệu và ngữ nghĩa của dịch vụ
  - Ví dụ : Tính tin cậy
  - Ví dụ nào cài đặt chức năng tại các điểm đầu cuối ?

- Tầng thấp có nhiều thông tin về các ràng buộc trên kênh truyền dữ liệu hơn (chẳng hạn tốc độ, tỷ lệ lỗi)

Mô Hình Tham Chiếu ISO/OSI

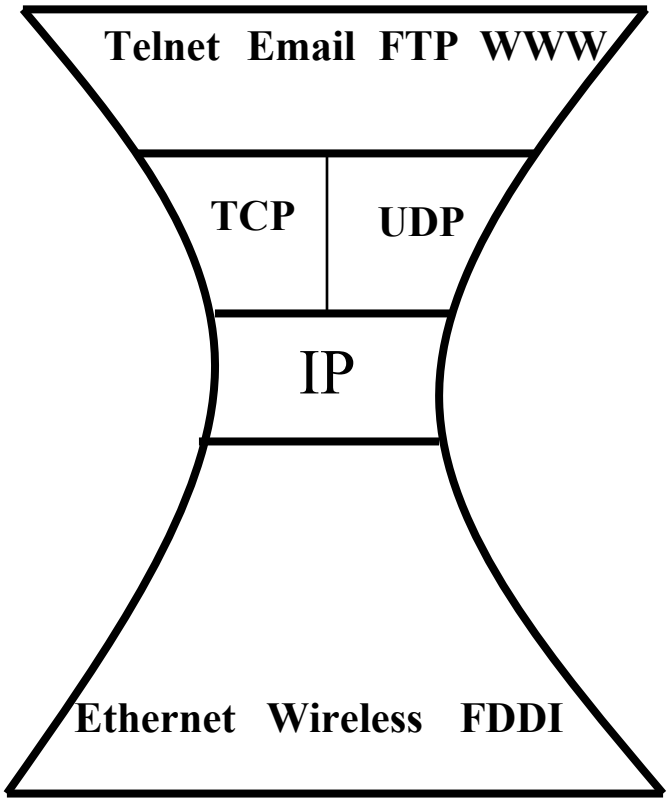
- Bảy tầng
  - Ba tầng thấp trên từng chặng
  - Bốn tầng cao cho các điểm đầu cuối



Mô Hình Giao thức Internet

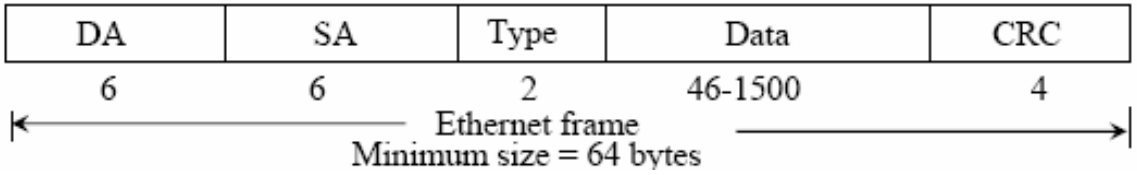
- Ứng dụng: Hỗ trợ các ứng dụng Mạng
  - FTP, SMTP, HTTP
- Giao vận: Truyền dữ liệu giữa hai tiến trình đầu cuối
  - TCP, UDP
- Mạng: Định tuyến các gói tin giữa hai thiết bị
  - IP, Các giao thức định tuyến (BGP, OSPF)
- Liên kết dữ liệu: Truyền dữ liệu giữa hai thực thể chung nhau môi trường truyền
  - PPP, Ethernet
- Vật lý: tín hiệu trên môi trường truyền

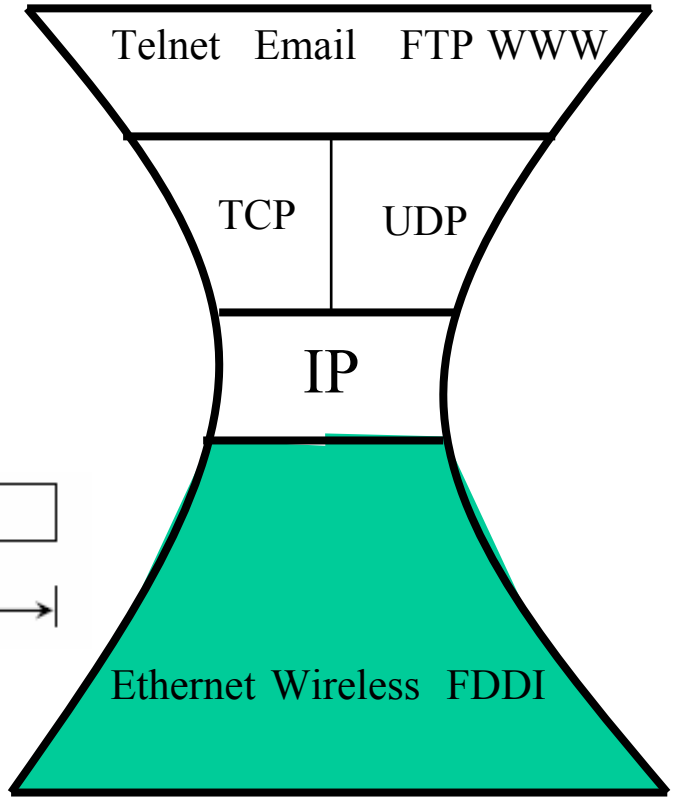
Mô hình Kiến trúc Internet – Đồng hồ cát



Tầng Liên kết Dữ liệu: Dịch vụ của Ethernet

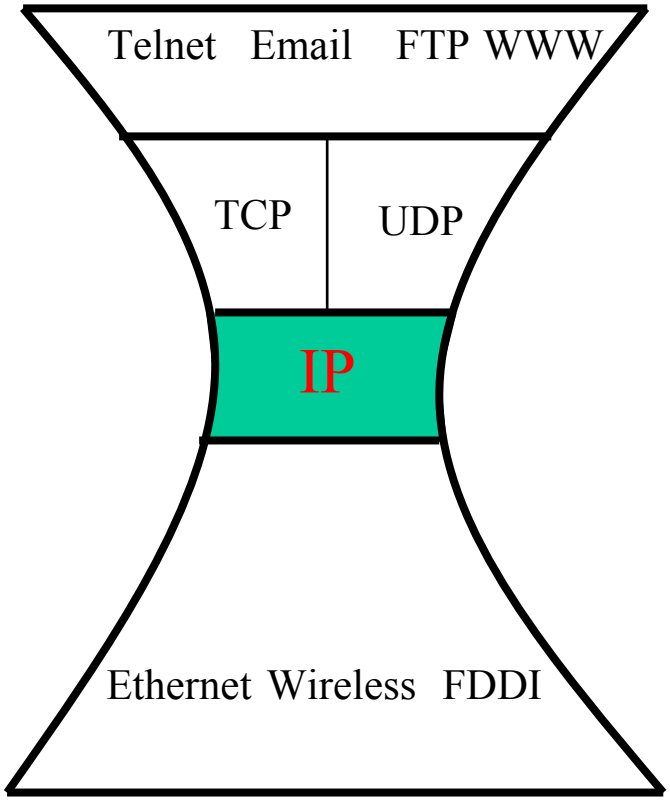
- *Phân kênh / Dồn kênh*
  - Gửi frames cho tầng Mạng
- *Đa truy cập*
  - Gửi frame cho các nút ngang hàng qua kênh truyền dùng chung
- *Phát hiện lỗi*



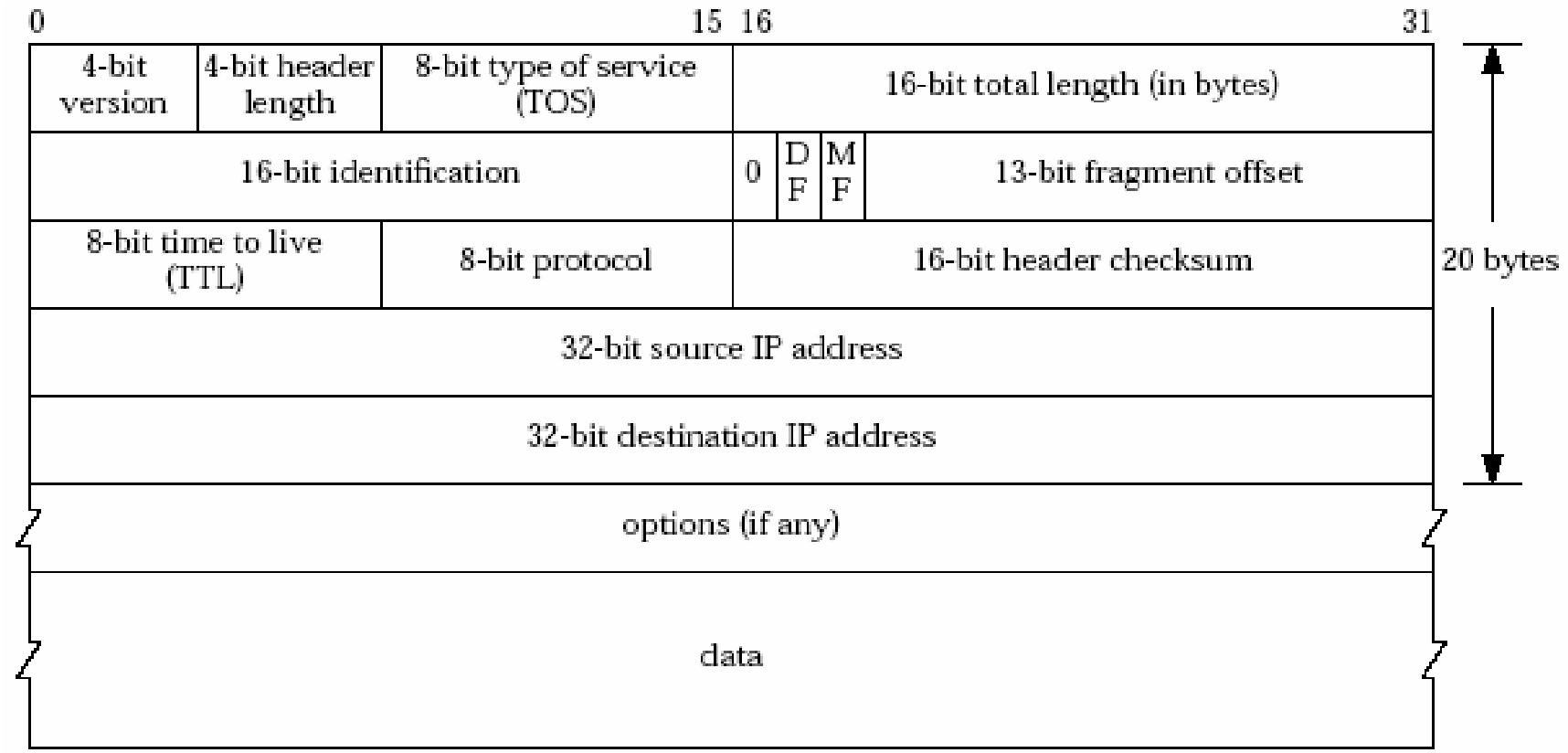


Tầng Mạng: Dịch vụ của IP

- *Phân kênh / Dồn kênh*
  - Chuyển packet cho Tầng Giao vận
- *Định tuyến*
  - Cố gắng tối đa để chuyển gói tin từ nơi Gửi đến nơi Nhận
- *Phân mảnh và Hợp nhất*
  - Chia gói to ra nhiều gói con
  - Bị loại bỏ trong IPv6
- *Phát hiện lỗi*
- *Không cung cấp*
  - Tính tin cậy, Đặt chỗ trên đường truyền

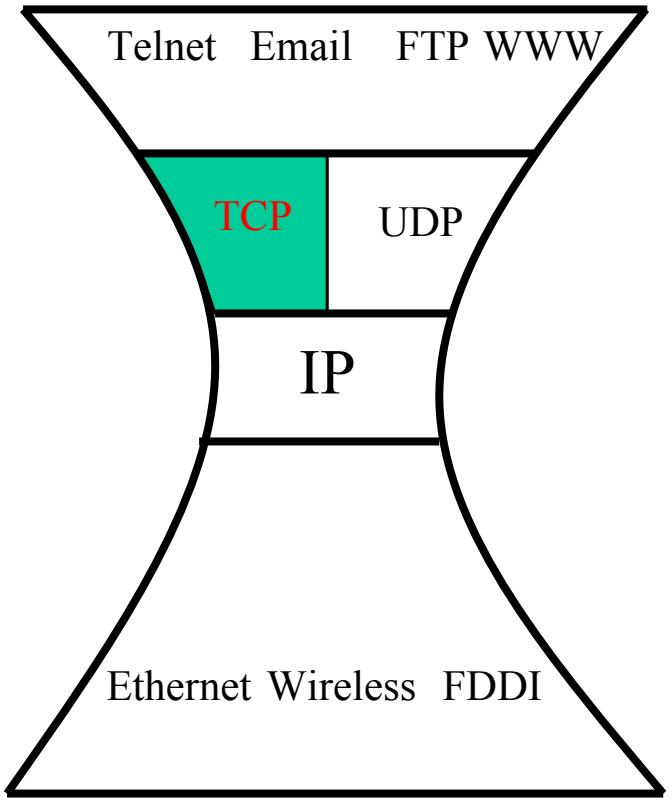


Tầng Mang: Tiêu đề IPv4

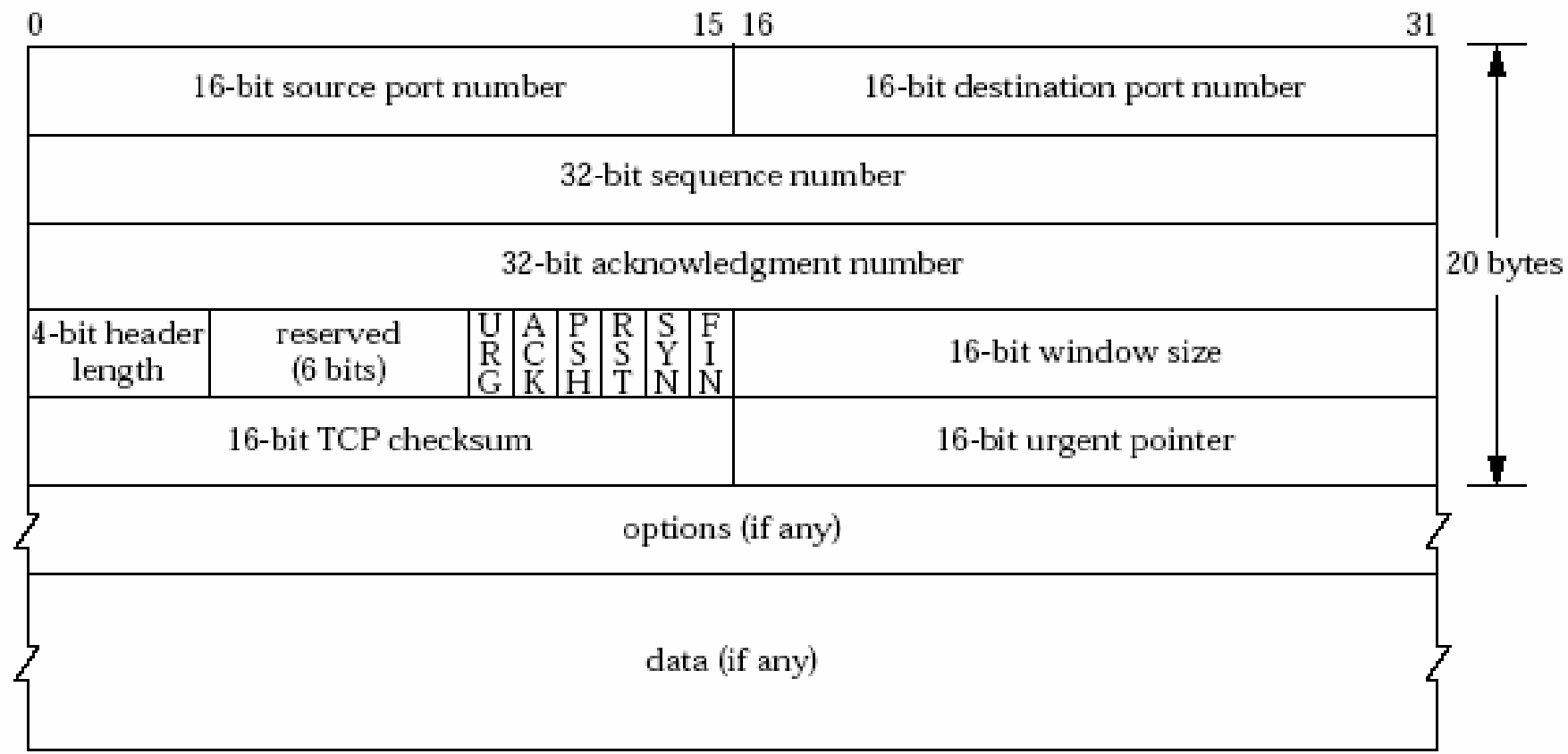


Tầng Giao vận : Dịch vụ TCP

- *Phân kênh / Dồn kênh*
- *Truyền Tin cậy*
  - Giữa tiến trình Gửi và tiến trình Nhận
  - Hai bên phải thiết lập trước kết nối: Dịch vụ hướng kết nối
- *Điều khiển lưu lượng*
  - Bên Gửi không gửi quá nhiều
- *Kiểm soát tắc nghẽn*
  - Giảm tốc độ gửi khi mạng quá tải
- *Phát hiện lỗi*
- *Không cung cấp*
  - Đảm bảo về Thời gian và Băng thông

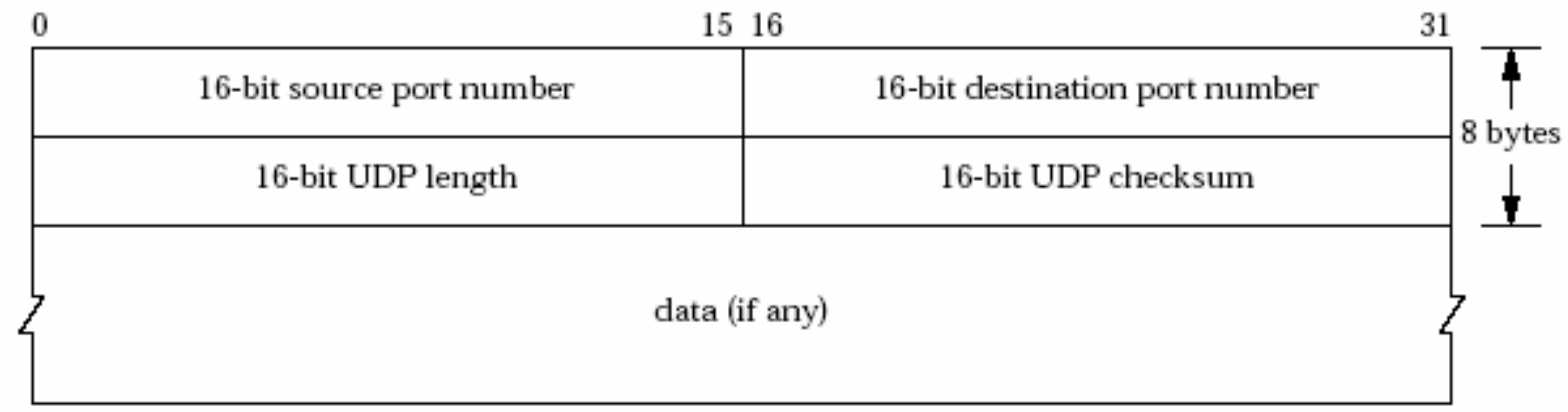


TCP Header


0
15 16
31
16-bit source port number
16-bit destination port number
32-bit sequence number
32-bit acknowledgment number
20 bytes
4-bit header length
reserved (6 bits)
URG
ACK
PSH
RST
SYN
FIN
16-bit window size
16-bit TCP checksum
16-bit urgent pointer
options (if any)
data (if any)

Dịch vụ của UDP

- Dịch vụ Không hướng nối
- Không cung cấp: Thiết lập kết nối, Tính tin cậy, Điều khiển lưu lượng, Kiểm soát tắc nghẽn, Đảm bảo về Thời gian và Băng thông
  - Vậy tại sao sử dụng UDP?



Chúng ta đã học những gì ?

- Hai ví dụ
- Phân tầng là gì?
- Tại sao phải Phân tầng?
- Cách xác định phạm vi của tầng: Nguyên lý đầu cuối
- Mô hình 7 tầng ISO/OSI khác với Mô hình Internet như thế nào

➢ *Tổng kết*

Phân tầng

- Kỹ thuật chính để thiết kế giao thức truyền thông; nhờ đó
  - Chia nhỏ các Hệ thống phức tạp

- Vấn đề Thiết kế cơ bản : Đặt chức năng vào đâu ?

Nguyên lý Đầu cuối

- Nếu tầng Ứng dụng có thể làm được, đừng đặt chức năng ở tầng thấp hơn - Ứng dụng biết rõ nhất về nhu cầu của mình
  - Chỉ cài đặt chức năng ở tầng thấp hơn, nếu :
    (1) Cải thiện Hiệt suất của nhiều ứng dụng – kể cả các ứng dụng tiềm tàng
    (2) Không ảnh hưởng đến các ứng dụng khác, and
    (3) Không quá phức tạp

- Ví dụ : Internet

Thách thức

- Khi xây dựng một Hệ thống (Mạng máy tính) hiệu quả: Cân bằng giữa:

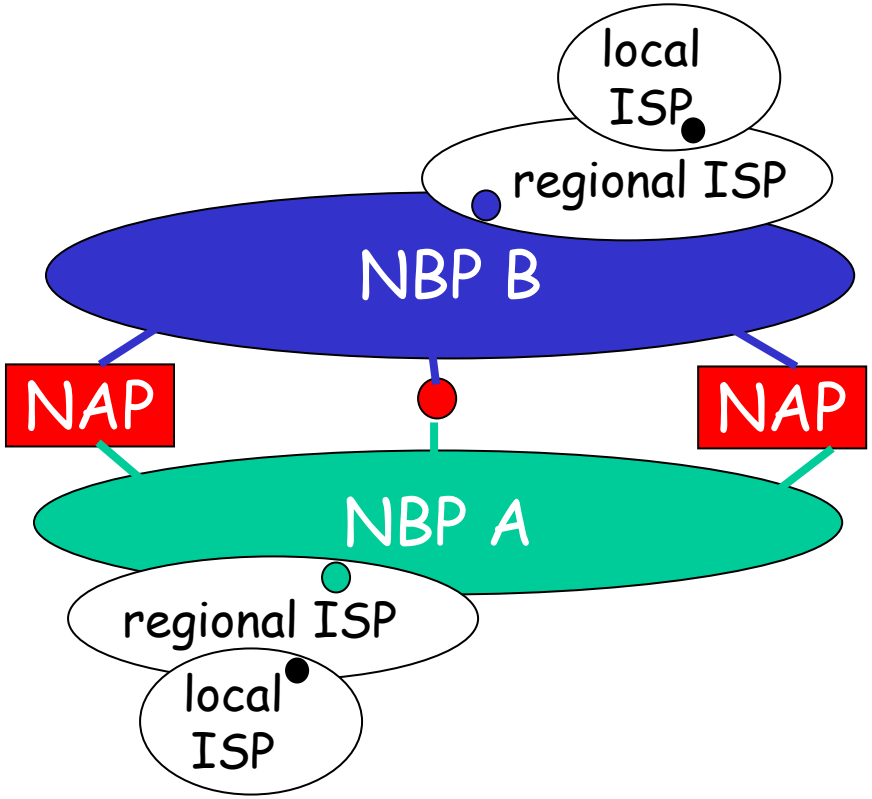
Nguyên lý Đầu cuối

Hiệu quả sử dụng ⟷ Tái sử dụng, Liên tác, Công sức cài đặt (áp dụng nguyên tắc phân tầng)

Không có một câu trả lời dùng chung: Phụ thuộc vào Mục tiêu và Giả định

Cấu trúc Internet: Mạng các Mạng

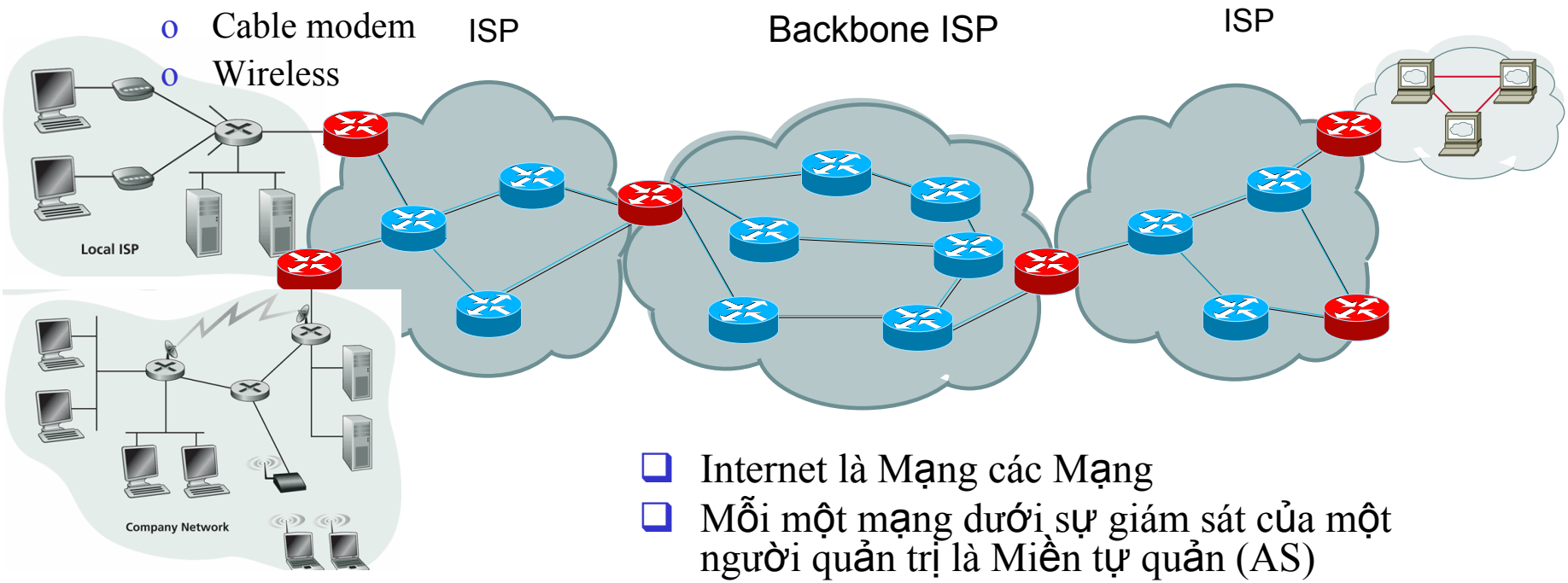
- Gần như phân cấp
- National/International Backbone Providers (NBPs)
  - Ví dụ: BBN/GTE, Sprint, AT&T, IBM, UUNet
  - Kết nối với nhau hoặc kết nối với Network Access Point (NAPs)
- ISP khu vực
  - Kết nối vào NBPs
- ISP cơ sở, công ty
  - Kết nối vào ISP khu vực



Cơ sở Hạ tầng Vật lý của Internet

Truy cập từ nhà

- Modem
- DSL
- Cable modem
- Wireless



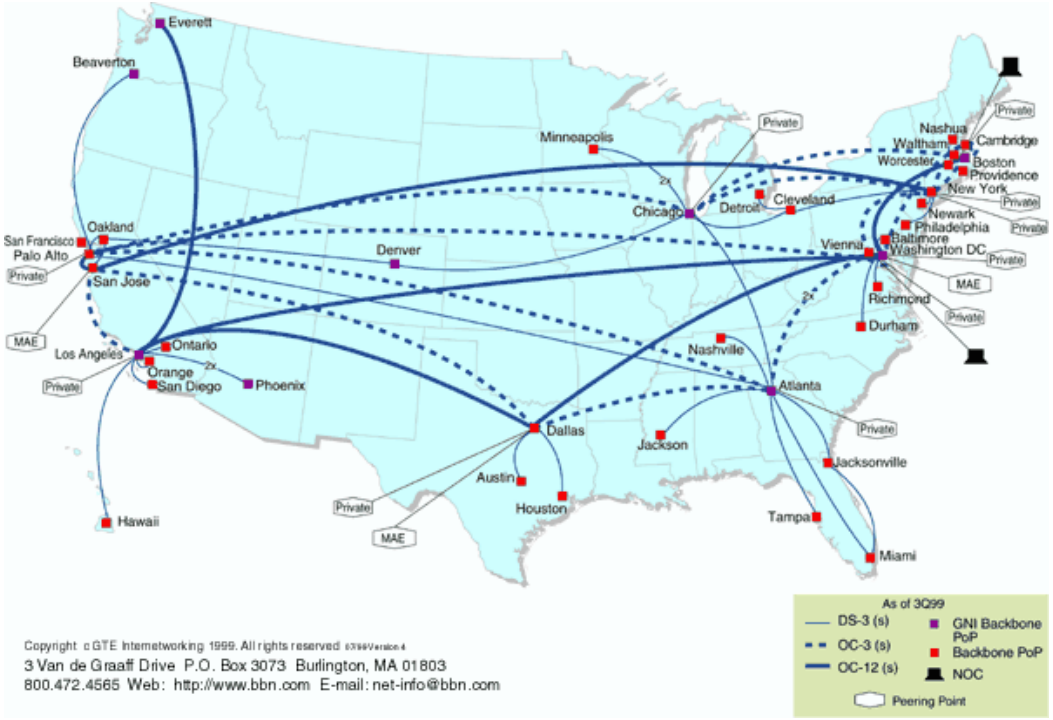
Truy cập từ cơ quan

- Ethernet
- FDDI
- Wireless

- Internet là Mạng các Mạng
- Mỗi một mạng dưới sự giám sát của một người quản trị là Miền tự quản (AS)
- Mạng truy cập và Mạng chuyển tiếp

National Backbone Provider ở Mỹ

Ví dụ Trục chính BBN/GTE US

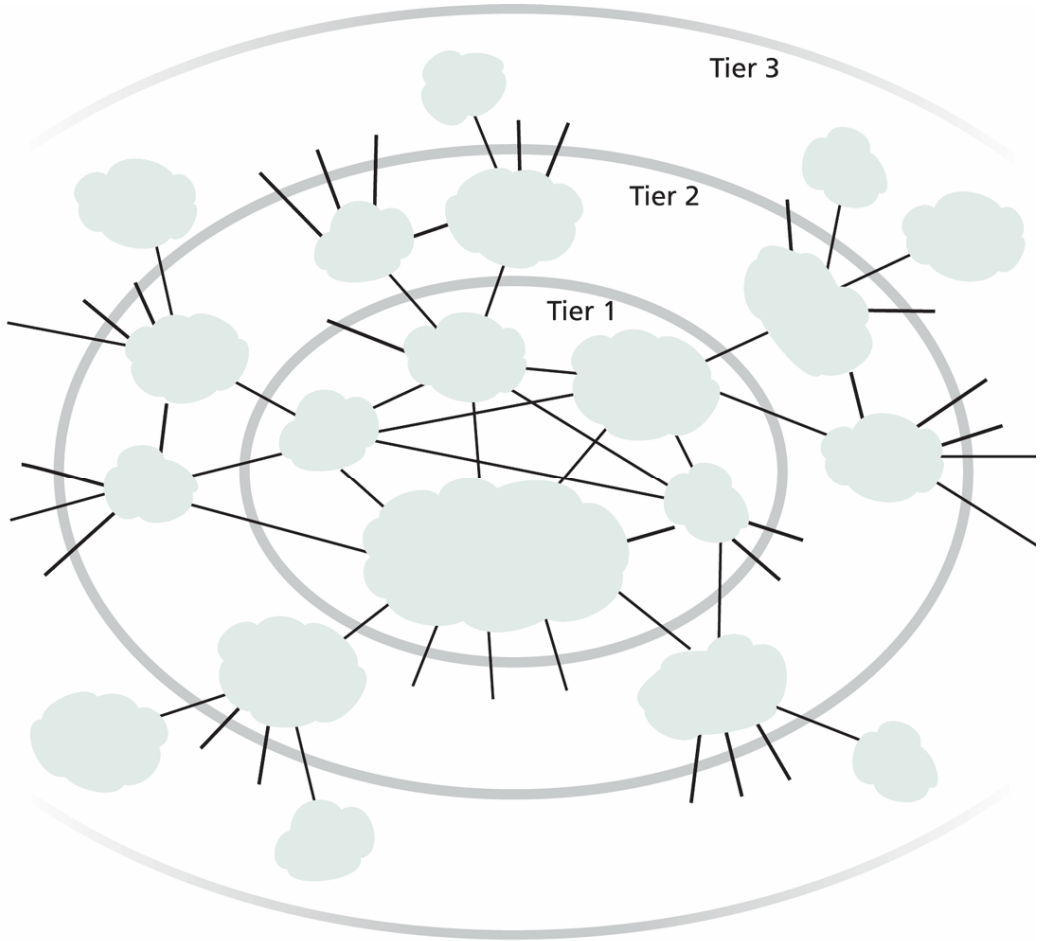


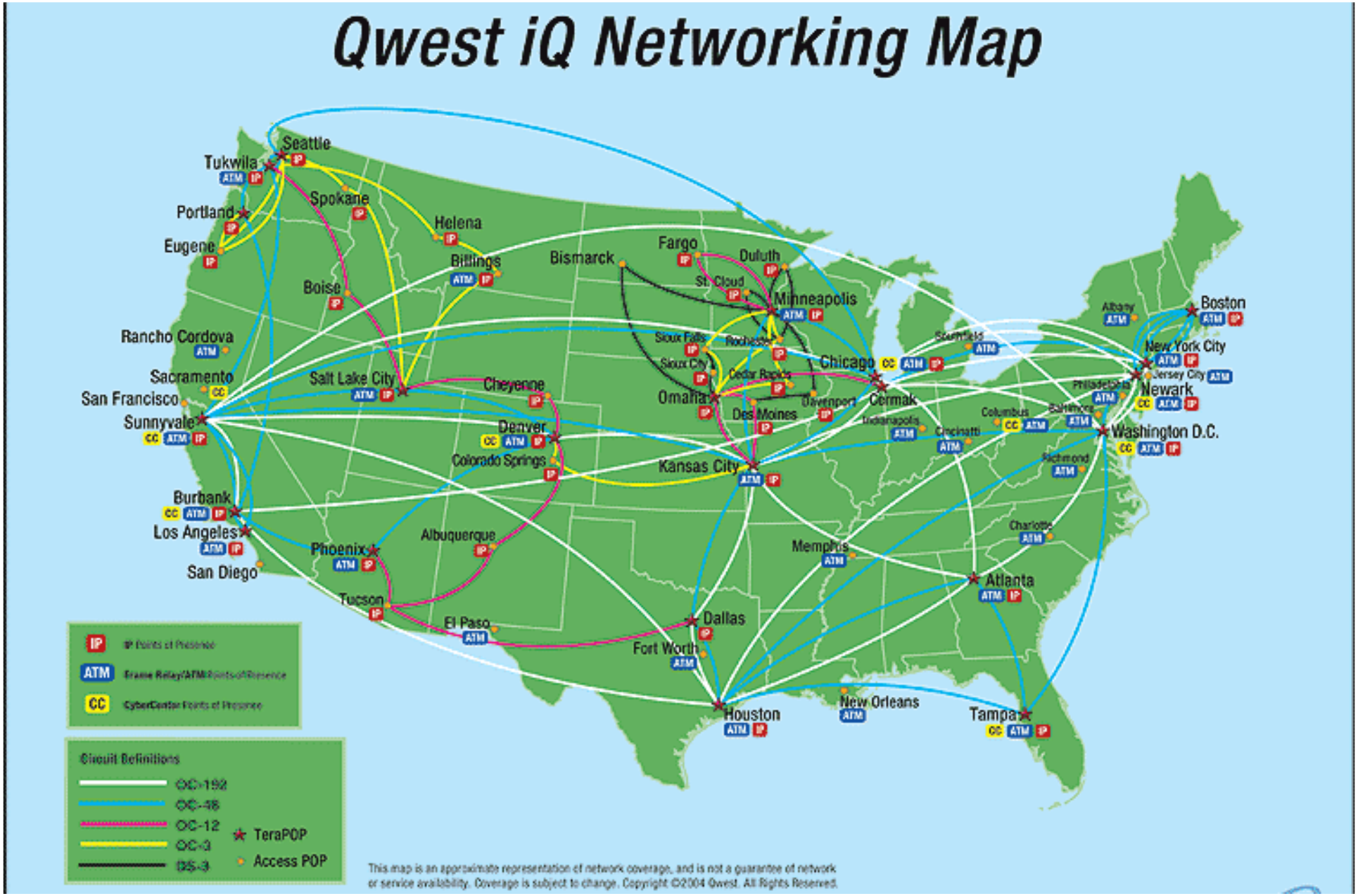
Abilene I2 Backbone

http://abilene.internet2.edu/maps-lists/

Kết nối các ISP

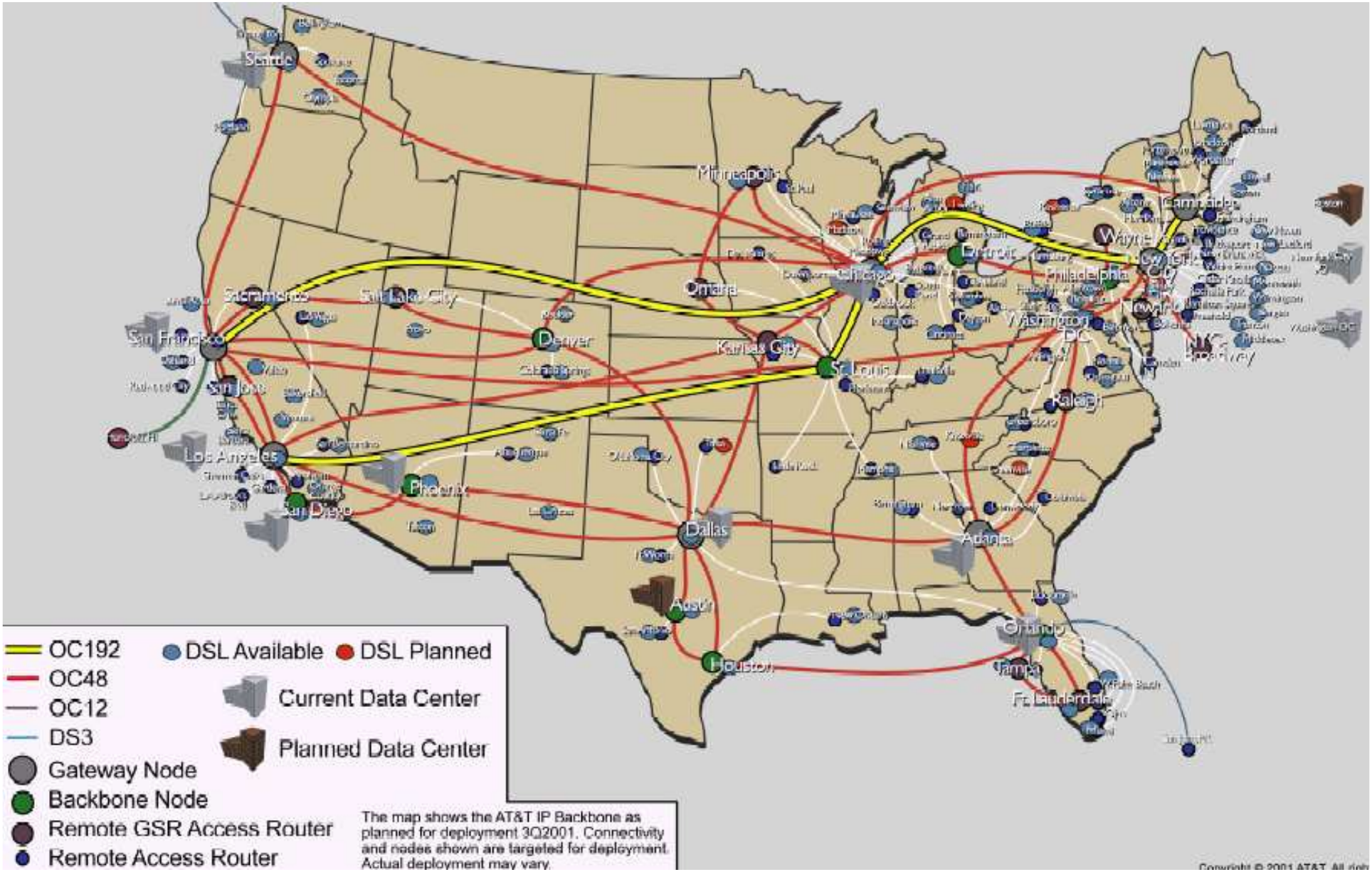
- Gần như Phân cấp
  - Chia thành nhiều lớp
  - ISP ở Lớp 1 còn gọi là Backbone Providers. Ví dụ: AT&T, Sprint, UUNet, Level 3, Qwest, Cable & Wireless
- ISP ở các lớp khác được khách hàng và các ISP khác kết nối vào
  - Ví dụ: MCI có 4,500 ISP con
- ISP cũng kết nối vào Network Access Point (NAP)





Qwest Backbone Map http://www.qwest.com/about/qwest/network.html

AT&T USA Backbone Map



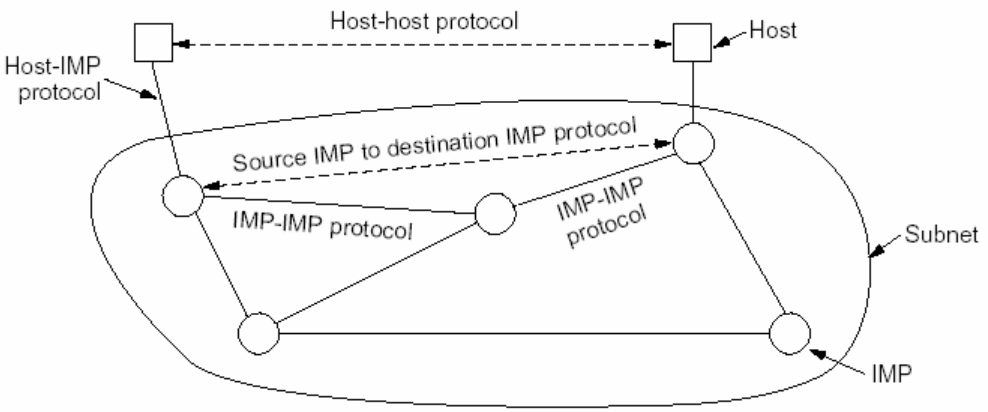
From AT&T web site.

Lịch sử Internet

1957-1972: Các nguyên tắc Chuyển mạch Gói đầu tiên

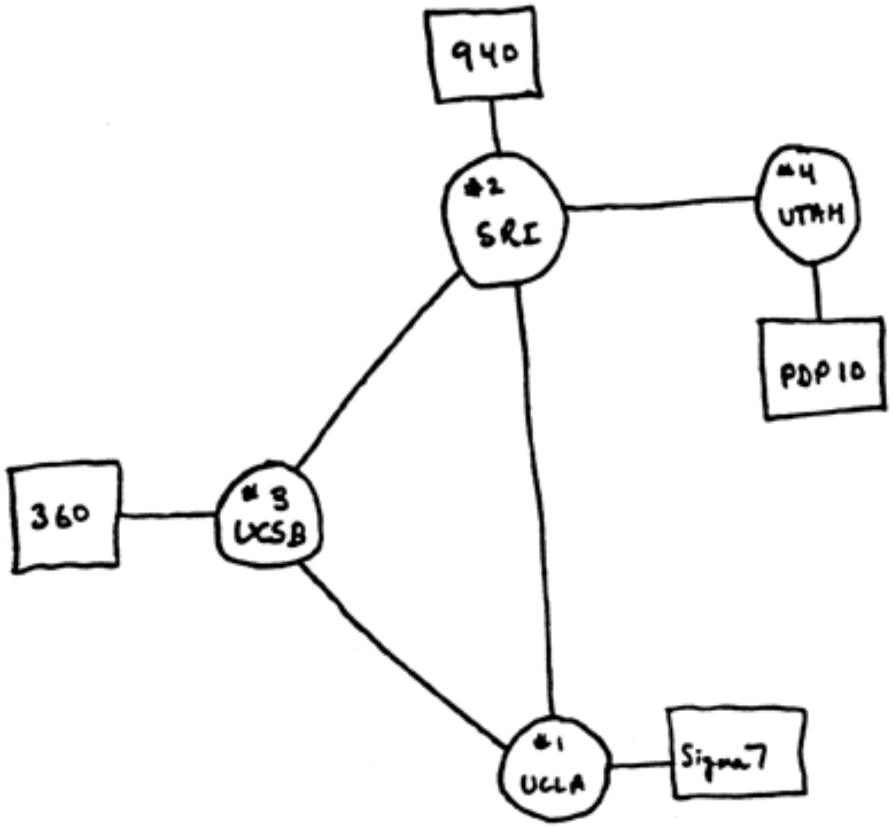
- **1957**: Liên Xô phóng Sputnik, Bộ quốc phòng Hoa Kỳ thành lập ARPA.
- **1961**: Kleinrock – Lý thuyết Hàng đợi chứng minh hiệu suất mạng chuyển mạch gói
- **1964**: Baran – Mạng chuyển mạch đầu tiên áp dụng trong quân sự
- **1967**: ARPAnet nhận được tài trợ từ Advanced Reearch Projects Agency
- **1969**: Nút mạng ARPAnet đầu tiên vận hành

- **1972**:
  - ARPAnet được công bố chính thức
  - NCP (Network Control Protocol) giao thức mạng nối 2 nút đầu tiên
  - Chương trình email đầu tiên
  - ARPAnet có 15 nút

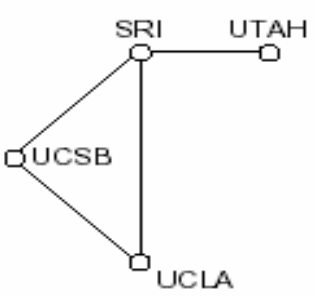


Mang ARPANET đầu tiên

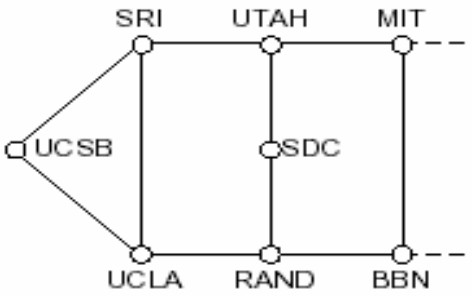
- **1969**
  - ARPANET : 4 nút, 50kbps



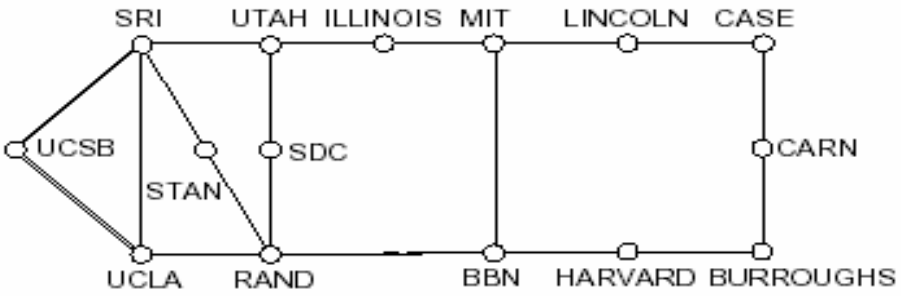
Những Mở rộng đầu tiên của ARPANET



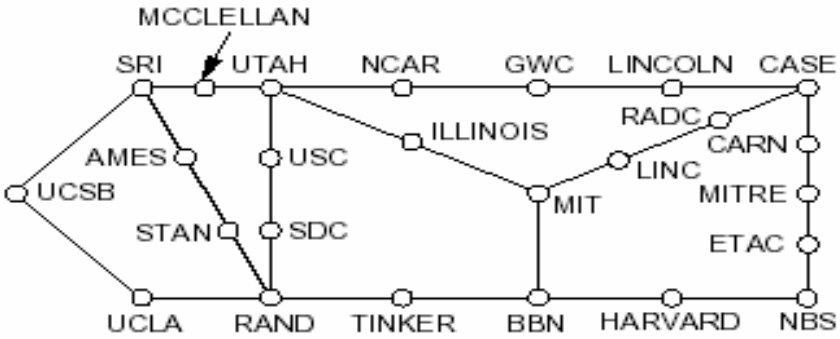
Tháng 12/1969



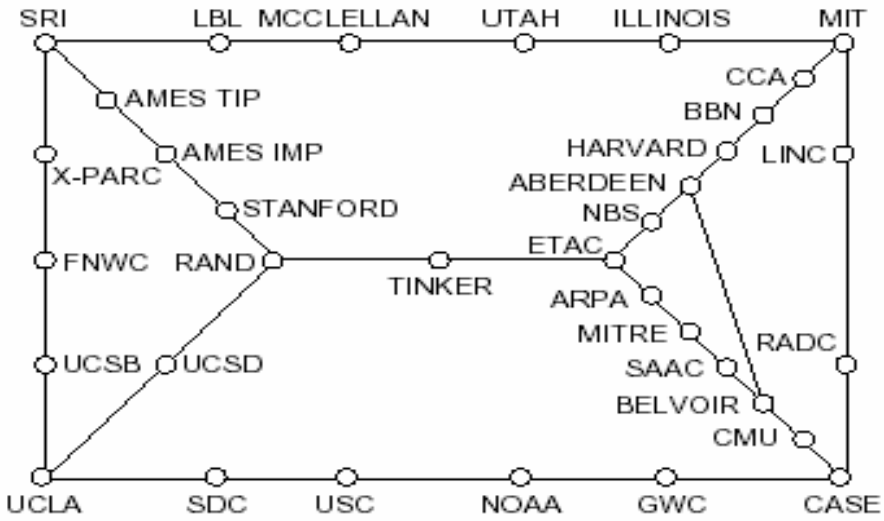
Tháng 7/1970



Tháng 3/1971



Tháng 4/1972



Tháng 9/1972

Lịch sử Internet

1972-1980: Kết nối các mạng độc quyền

- **1970**: Mạng vệ tinh ALOHAnet ở Hawaii
- **1973**: Metcalfe đề xuất Luận văn Tiến sỹ về Ethernet
- **1974**: Cerf và Kahn – đưa ra kiến trúc kết nối các mạng máy tính
- **Cuối 70's**: Các kiến trúc độc quyền: DECnet, SNA, XNA
- **Cuối 70's** : Mạng chuyển mạch gói có kích thước cố định (Tiền thân của ATM)
- **1979**: ARPAnet có 200 nút

Quan điểm về Kết nối các mạng của Cerf và Kahn:

- Giản thiểu, Tự trị - Đòi hỏi ít sự thay đổi khi kết nối vào Mạng máy tính
- Mô hình dịch vụ "Cố gắng tối đa"
- Router không lưu lại trạng thái
- Kiểm soát phân tán

Định hình nên Kiến trúc Internet ngày nay

Lịch sử Internet

1980-1990: Nhiều giao thức mới. Thời kỳ thịnh vượng

- **1983**: TCP/IP đi vào thực tế
- **1982**: Giao thức gửi thư điện tử SMTP
- **1983**: DNS giải mã tên thành địa chỉ IP
- **1985**: Giao thức FTP
- **1988**: Kiểm soát tắc nghẽn trong TCP

- Nhiều mạng cấp quốc gia: Csnet, BITnet, NSFnet, Minitel
- 100,000 máy tính kết nối vào mạng nào đó

Lịch sử Internet

Thập niên 1990 : Thương mại hóa và WWW

- **Đầu thập niên 1990**: ARPAnet giải thể
- **1991**: NSF bãi bỏ hạn chế cấm hoạt động thương mại trên mạng NSFnet (cũng giải thể vào 1995)
- **Đầu thập niên 1990**: WWW
  - hypertext [Bush 1945, Nelson 1960's]
  - HTML, http: Berners-Lee
  - 1994: Mosaic, later Netscape
  - Cuối thập niên 1990: Thương mại hóa trên WWW

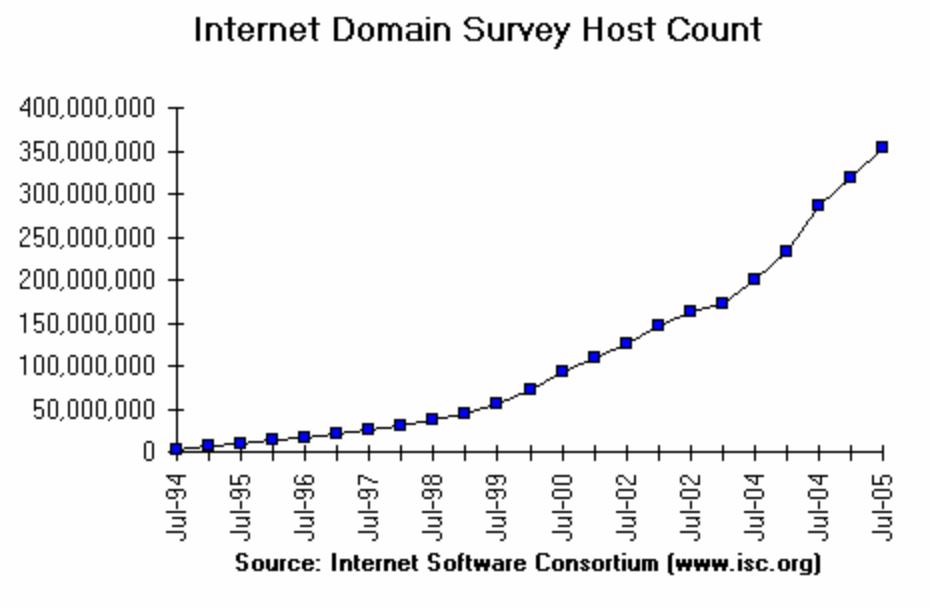
Cuối thập niên 1990:

- Khoảng 50 triệu máy tính kết nối vào Internet
- Khoảng 100 triệu người dùng
- Tốc độ kết nối trên đường trục chính là 1 Gbps

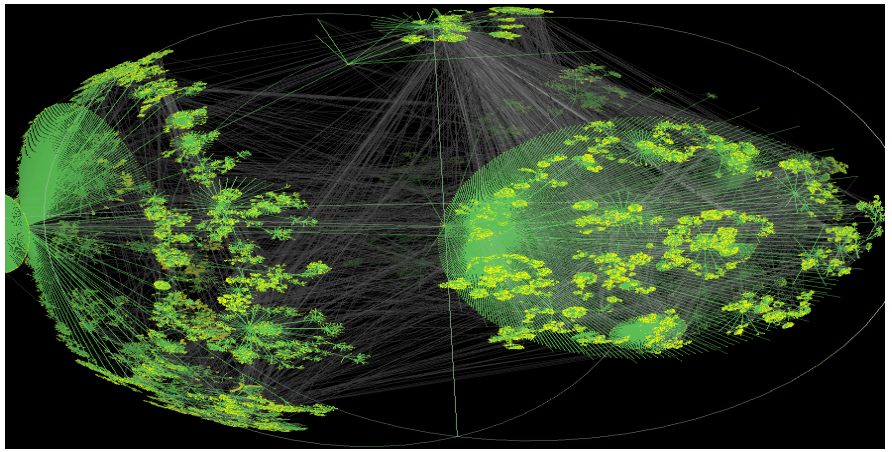
Tốc độ tăng trưởng : Tính theo số lượng

Số lượng Máy tính trên Internet:

Tháng 08/1981	213
Tháng 10/1984	1,024
Tháng 12/1987	28,174
Tháng 10/1990	313,000
Tháng 07/1993	1,776,000
Tháng 07/1996	19,540,000
Tháng 07/1999	56,218,000
Tháng 07/2004	285,139,000
Tháng 01/2005	317,646,000
Tháng 07/2005	353,284,000



Cấp độ kết nối Router



ATM: Asynchronous Transfer Mode

Internet:

- Là phương thức trao đổi dữ liệu được sử dụng rộng rãi nhất trên toàn cầu

Cuối thập niên 1980:

- Các nhà viễn thông phát triển ATM: Cạnh tranh để truyền dữ liệu và âm thanh với tốc độ cao
- Cơ quan định chuẩn:
  - ATM Forum
  - ITU

Nguyên lý của ATM:

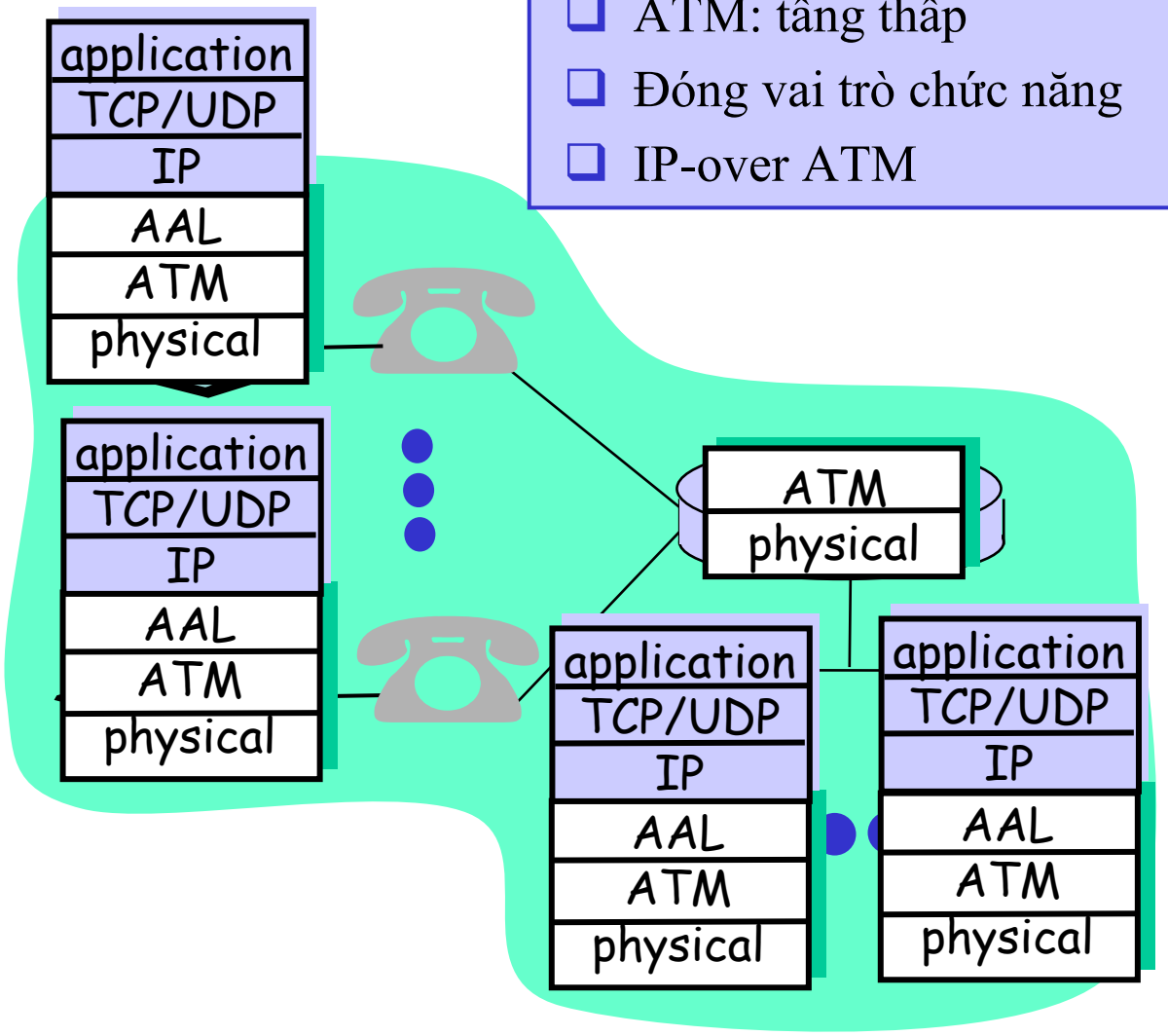
- Kích thước nhỏ (48 byte payload, 5 byte header) và có độ lớn cố định gọi là *tế bào cells* (giống packet)
  - Chuyển mạch nhanh
  - Kích thước nhỏ thuận tiện cho âm thanh
- Mạng chuyển mạch ảo: switch duy trì trạng thái cho mỗi kết nối
- Giao diện được định nghĩa rõ ràng giữa "mạng" và "người sử dụng" (giống công ty điện thoại)

Phân tầng trong ATM

- **ATM Adaptation Layer (AAL)**: Giao diện với các tầng trên
  - Hệ thống đầu cuối
  - Phân đoạn/ Ráp đoạn
- **ATM Layer**: Chuyển mạch tế bào
- **Physical**

Ứng dụng nằm ở đâu?

- ATM: tầng thấp
- Đóng vai trò chức năng
- IP-over ATM



Chương 1: Tổng kết

Quá nhiều và Phức tạp !

- Tổng quan về Internet
- Giao thức là gì ?
- "Rìa", "Lõi" và Truy cập tới mạng
- Hiệu suất: Mất mát, Độ trễ
- Phân tầng và Mô hình dịch vụ
- Trục chính, NAP, ISP
- Lịch sử
- Mạng ATM

Hy vọng rằng

- Phạm vi, Tổng quan, "cảm giác" về Mạng
- Chi tiết và sâu hơn sẽ được trình bày trong các chương sau

Chương 2: MÔ HÌNH DỮ LIỆU QUAN HỆ

I. CÁC KHÁI NIỆM CƠ BẢN

- Thuộc tính
- Lược đồ quan hệ
- Miền trị
- Quan hệ

II. CƠ SỞ DỮ LIỆU QUAN HỆ VÀ HỆ CƠ SỞ DỮ LIỆU QUAN HỆ

III. ĐỊNH NGHĨA SIÊU KHOÁ VÀ KHOÁ

- Siêu khoá
- Khoá
- Khoá chính và khoá dự phòng
- Khoá ngoại

IV. CÁC PHÉP TÍNH TRÊN CƠ SỞ DỮ LIỆU QUAN HỆ

- Phép chèn
- Phép loại bỏ
- Phép thay đổi

I. CÁC KHÁI NIỆM CƠ BẢN

1.1. Thuộc tính

Mỗi đối tượng quản lý đều có những đặc tính riêng biệt. Các đặc tính riêng biệt này được gọi là các thuộc tính. Ví dụ, mỗi sinh viên đều có ***mã sinh viên***, ***họ và tên***, ***quê quán, năm sinh***. Các thuộc tính của một đối tượng được phân biệt với nhau qua tên của chúng. Trong thực tế, các nhà phân tích - thiết kế thường đặt tên thuộc tính mang tính gợi nhớ. Trong giáo trình này khi không cần quan tâm đến ngữ nghĩa ta thường ký hiệu tên các thuộc tính bởi các chữ cái hoa ở đầu bảng chữ cái **A, B, C,** ...

Ký hiệu tập thuộc tính

Giả sử có 3 thuộc tính A, B, C

Ta viết:

$$\{A,B,C\} \equiv A,B,C \equiv ABC$$

I. CÁC KHÁI NIỆM CƠ BẢN (tiếp)

1.2. Lược đồ quan hệ (Relational Schema)

Một lược đồ quan hệ được đặc trưng bởi một tên riêng là tên của lược đồ và một tập hữu hạn các thuộc tính U={A1 , A2 ,..., An}. Lược đồ R với tập thuộc tính U={A1, A2 ,..., An} được ký hiệu là R(U) hay R(A1 , A2,..., An).

Ví dụ:

SINHVIEN(MaSV, Hoten, NgaySinh, QueQuan, Makhoa)

KHOA(Makhoa, Tenkhoa)

1.3. Miền trị (Domain)

Cho tập hữu hạn các thuộc tính U= {A1, A2,..., An} ($U \neq \varnothing$). Ứng với mỗi thuộc tính $A_i \in U$ (i=1, 2,..., n) là một tập di chứa ít nhất hai phần tử, được gọi là miền trị của thuộc tính Ai , ta viết di= dom(Ai).

I. CÁC KHÁI NIỆM CƠ BẢN (tiếp)

1.4. Quan hệ (Relational)

Một quan hệ r với tập thuộc tính U = {A1, A2,..., An} là tập con của tích đề các $dom(A1) \times dom(A2) \times .... \times dom(An)$. Mỗi phần tử của r là một n-bộ (a1, a2,..., an) và được gọi là một bộ của quan hệ r. Khi đó ta ký hiệu r(U) hoặc r(A1, A2,..., An)

Ví dụ: Quan hệ **SINHVIEN** với tập thuộc tính

U={MaSV, Hoten, QQ, NS} có dạng:

MaSV	*Hoten*	*QQ*	*NS*
01	Lê Anh	Nghệ An	1970
02	Trần Trung	Hà nội	1975
03	Đỗ Hà	Hải phòng	1972
04	Lê Anh	Hà nội	1975
…			

I. CÁC KHÁI NIỆM CƠ BẢN (tiếp)

1.5. Hạn chế của bộ

Giả sử u là một bộ của quan hệ r(U) và $X \subseteq U$. Khi đó ký hiệu: u[X]= u|X là hạn chế của bộ u trên tập X.

Ví dụ: Quan hệ **SINHVIEN** với tập thuộc tính

U={MaSV, Hoten, QQ, NS} có dạng:

MaSV	*Hoten*	*QQ*	*NS*
01	Lê Anh	Nghệ An	1970
02	Trần Trung	Hà nội	1975
03	Đỗ Hà	Hải phòng	1972
04	Lê Anh	Hà nội	1975
…			

t1=(’01’, ‘Lê Anh’, ‘Nghệ An’, 1970)

t1[MaSV, Hoten] =(’01’, ‘Lê Anh’) ;

t1[Hoten, QueQuan] =(‘Lê Anh’,’Nghệ An’) ;

II. CSDL QUAN HỆ VÀ HỆ QTCSDL QUAN HỆ

2.1. Các khái niệm

- Cơ sở dữ liệu quan hệ là cơ sở dữ liệu mà lược đồ là tập các lược đồ quan hệ và các thể hiện là các n-bộ của các lược đồ quan hệ này.
- Hệ quản trị cơ sở dữ liệu quan hệ là hệ quản trị cơ sở dữ liệu có chức năng tạo lập và khai thác cơ sở dữ liệu quan hệ.

II. CSDL QUAN HỆ VÀ HỆ QTCSDL QUAN HỆ

2.2.Tạo lập cơ sở dữ liệu quan hệ

Để tạo lập cơ sở dữ liệu quan hệ ta phải tạo lập các quan hệ. Mỗi quan hệ ta cần khai báo các mục sau:

- Khai báo tên của quan hệ
- Khai báo tên thuộc tính, miền trị của các thuộc tính
- Các ràng buộc dữ liệu cho thuộc tính hay giữa các thuộc tính.

II. CSDL QUAN HỆ VÀ HỆ CSDL QUAN HỆ

2.2.Tạo lập cơ sở dữ liệu quan hệ (tiếp)

Ràng buộc toàn vẹn (Integrity Constraints): Là những quy định mà dữ liệu trong một cơ sở dữ liệu phải thoã mãn. Việc đặt ra các ràng buộc nhằm đảm bảo cho dữ liệu trong cơ sở dữ liệu phản ánh đúng thực tế. Nó thường được mô tả lúc tạo lập quan hệ.

Các ràng buộc gồm nhiều loại:

- *Ràng buộc về miền trị (Domain)*
- *Ràng buộc xác nhận (Assertion)*
- *Ràng buộc về khóa (Key)*

III. SIÊU KHOÁ VÀ KHOÁ

3.1. Siêu khoá (Super key)

Cho quan hệ r định nghĩa trên lược đồ R với tập thuộc tính U={A1,...,An}. Tập $K \subseteq U$ được gọi là siêu khoá của r (hoặc của R) nếu với mọi t1, t2 $\in$ r , t1 $\neq$ t2 thì t1[K] $\neq$ t2[K].

Nói cách khác một tập con $K \subseteq U$ được gọi là siêu khoá của r nếu K có thể dùng làm cơ sở để phân biệt các bộ của r.

⇨ ***Một lược đồ quan hệ R luôn có ít nhất một siêu khoá***

III. SIÊU KHOÁ VÀ KHOÁ (tiếp)

3.1. Siêu khoá (Tiếp)

Ví dụ: Quan hệ **SINHVIEN** với tập thuộc tính

U={MaSV, Hoten, QQ, NS} có dạng:

MaSV	*Hoten*	*QQ*	*NS*
01	Lê Anh	Nghệ An	1970
02	Trần Trung	Hà nội	1975
03	Đỗ Hà	Hải phòng	1972
04	Lê Anh	Hà nội	1975
…			

Với lược đồ quan hệ SINHVIEN(MaSV, Hoten, QQ, NS) có các siêu khoá:

K1={MaSV}

K2={MaSV, Hoten}

K3={MaSV, Hoten, QQ}

K4={MaSV, Hoten, NS}

III. SIÊU KHOÁ VÀ KHOÁ (tiếp)

3.2. Khoá (Key)

Cho lược đồ quan hệ **R** với tập thuộc tính U = [A1,..., An]. Tập K⊆U được gọi là khoá của **R** nếu:

1. K là siêu khoá của R
2. ∀K' ⊂ K thì K' không phải là siêu khoá của R.

Ví dụ:

Với lược đồ quan hệ **SINHVIEN(**MaSV, Hoten, QQ, NS)

ta có khoá K={MaSV}.

⇨ ***Một lược đồ quan hệ R luôn có ít nhất một khoá***

III. SIÊU KHOÁ VÀ KHOÁ (tiếp)

3.3. Khoá chính và khoá dự phòng

Một lược đồ quan hệ R có thể có nhiều khoá. Ta có thể chọn một trong các khoá của R làm ***khoá chính*** (primary key). Các khoá còn lại được gọi là ***khoá dự phòng***.

3.4. Khoá ngoại (Foreign key)

Một tập K được gọi là khoá ngoại của lược đồ quan hệ R nếu nó không phải là khoá chính của R nhưng nó lại là khoá chính của một lược đồ quan hệ khác.

Ví dụ: Cho 2 lược đồ

Khoá chính

KHOA(**MaKhoa**, TenKhoa)

SINHVIEN(**MaSV**, HoTen, NgaySinh, **MaKhoa**)

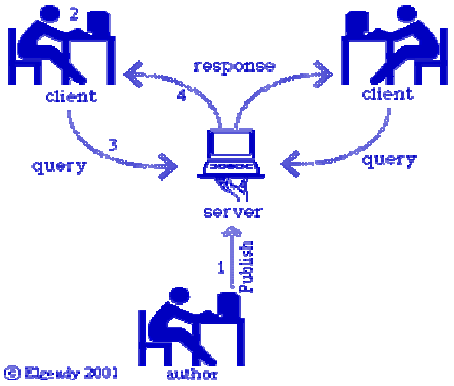
Khoá chính

Khoá ngoại

Chương 2 : Tầng Ứng Dụng

Mục tiêu:

- Định nghĩa, Hoạt động các giao thức trong tầng Ứng dụng.
  - Mô hình Khách hàng - Người phục vụ (client-server)
  - Mô hình đồng đẳng
  - Tìm hiểu một số giao thức tầng Ứng dụng thông qua các ví dụ cụ thể.



Sẽ học cái gì ?

- Các giao thức cụ thể:
  - HTTP
  - FTP
  - SMTP
  - POP
  - DNS
  - MSN, Gnutella
- Lập trình các ứng dụng mạng : socket API

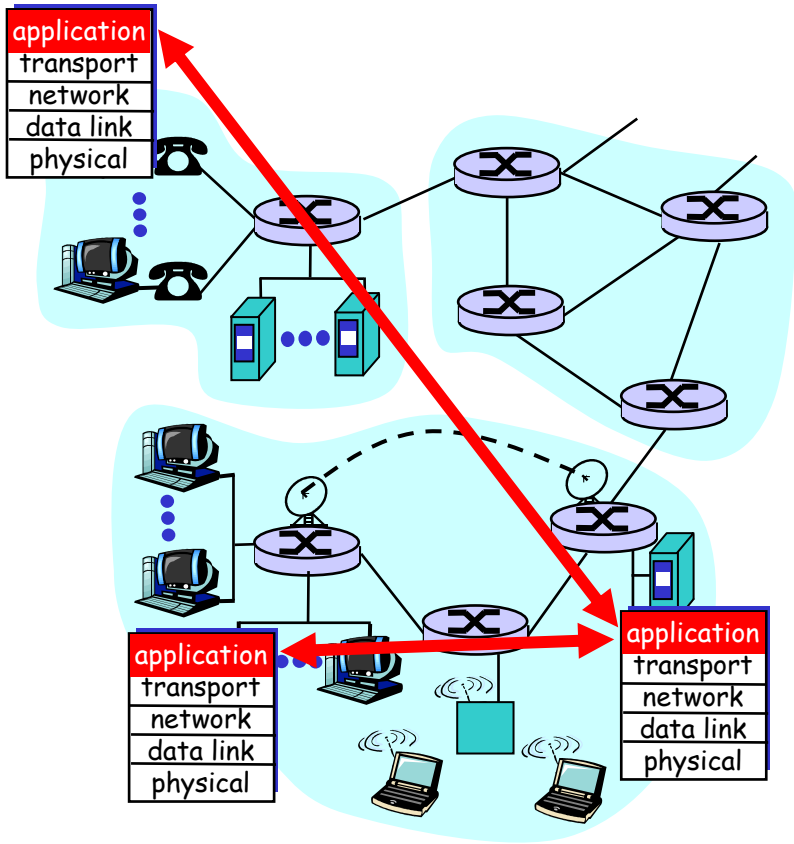
Trình Ứng dụng và Giao thức tầng Ứng dụng

Trình ứng dụng : Các tiến trình phân tán truyền thông với nhau

- Chạy trên các thiết bị đầu cuối
- Trao đổi thông điệp với nhau.
- email, FTP, Web

Giao thức tầng ứng dụng

- Là một phần của trình ứng dụng.
- Xác định thông điệp trao đổi giữa các ứng dụng.
- Sử dụng dịch vụ truyền thông do giao thức tầng dưới (TCP, UDP) cung cấp.



Một số thuật ngữ

- **Tiến trình :** là chương trình chạy trên thiết bị đầu cuối.
- Trên cùng máy tính, hai tiến trình có thể truyền thông với nhau thông qua **truyền thông liên tiến trình** (HĐH quản lý).
- Các tiến trình chạy trên các máy tính khác nhau : tuân thủ giao thức tầng ứng dụng.

- User agent: phần mềm đóng vai trò giao diện giữa người dùng và mạng.
  - Cài đặt Giao thức tầng Ứng dụng.
  - Web: browser
  - E-mail: mail reader
  - streaming audio/video: media player

Mô hình Client-Server

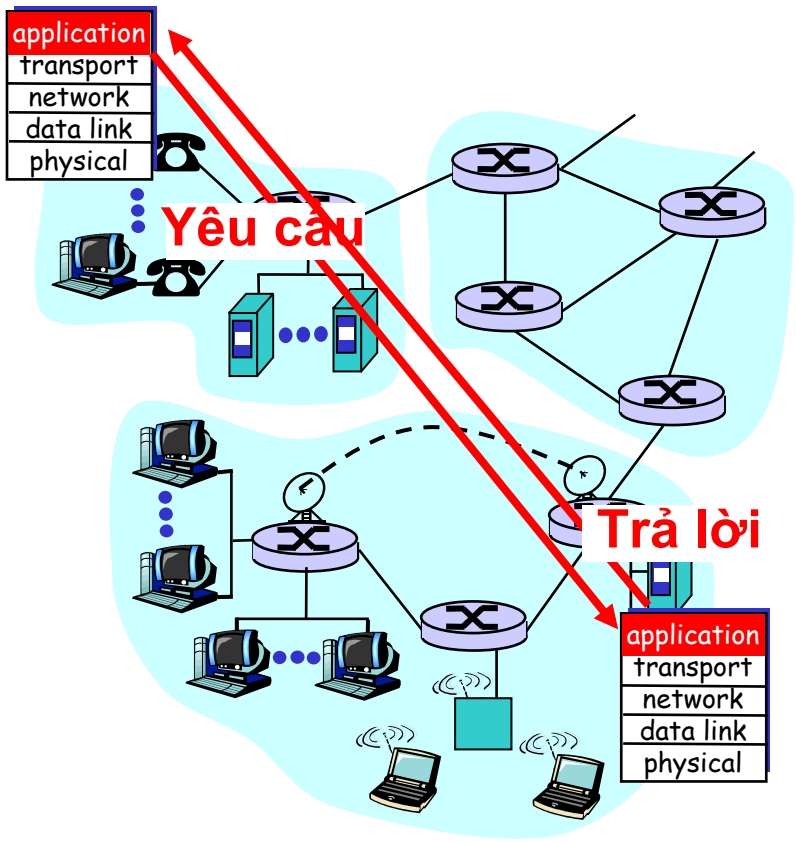
Ứng dụng gồm hai thành phần :
Client và *Server*

Client:

- Khởi tạo giao tiếp với server ("speaks first")
- Yêu cầu dịch vụ từ server.
- Web : client nhận và hiển thị web page ; e-mail: các trình đọc thư.

Server:

- Cung cấp dịch vụ theo yêu cầu từ Client
- Web server gửi Web page, Mail server gửi và nhận E-mail



Giao thức tầng Ứng dụng

API: Application Programming Interface

- Là giao diện giữa tầng Ứng dụng và tầng Giao vận.
- Socket : Internet API
  - Hai tiến trình truyền thông bằng cách gửi/nhận dữ liệu vào/từ socket.

Vậy thì Phân biệt các tiến trình bằng cách nào?

- **Địa chỉ IP** của máy.
- **Số hiệu Cổng** – cho phép bên máy tính nhận xác định tiến trình nào nhận thư.

Còn tiếp …

Ứng dụng cần Dịch vụ Giao vận gì?

Mất mát dữ liệu (Data loss)

- Một số ứng dụng có thể chấp nhận một số mất mát.
- Một số ứng dụng lại đòi hỏi 100% dữ liệu truyền tin cậy (web).

Thời gian (Timing)

- Một số ứng dụng (hội thoại qua Internet, các trò chơi trực tuyến) đòi hỏi độ trễ thấp.

Băng thông (Bandwidth)

- Một số ứng dụng (đa phương tiện) yêu cầu băng thông tối thiểu để có thể hoạt động được.
- Một số ứng dụng khác có thể sử dụng bất cứ băng thông nào được cấp phát.

Yêu cầu với một số Ứng dụng cụ thể

Ứng dụng	Mất mát	Băng thông	Thời gian
Truyền file	Không	Co dãn	Không
Thư tín điện tử	Không	Co dãn	Không
Web	Có thể	Co dãn	Không
Đa phương tiện Thời gian thực	Có thể	audio: 5Kb-1Mb video:10Kb-5Mb	Có, 100's msec
stored audio/video	Chấp nhận	same as above	Có, few secs
Trò chơi tương tác	Chấp nhận	few Kbps up	Có, 100's msec
Ứng dụng Tài chính	Không	Co dãn	Không xác định

Dịch vụ Giao thức Giao vận Internet

Dịch vụ TCP:

- ***Hướng nối*** *:* Yêu cầu thiết lập kết nối giữa client và server.
- ***Truyền dữ liệu tin cậy*** giữa tiến trình gửi và nhận
- ***Điều khiển lưu lượng*** *:* bên gửi sẽ không làm "lụt" bên nhận.
- ***Kiểm soát tắc nghẽn****:* điều chỉnh tốc độ gửi khi mạng quá tải.
- **Không hỗ trợ** *:* thời gian, băng thông tối thiểu.

Dịch vụ UDP:

- Truyền dữ liệu không tin cậy giữa các tiến trình gửi và nhận.
- **Không hỗ trợ** : thiết lập kết nối, độ tin cậy, điều khiển lưu lượng, kiểm soát tắc nghẽn, thời gian, băng thông tối thiểu.

Tại sao một số ứng dụng sử dụng UDP

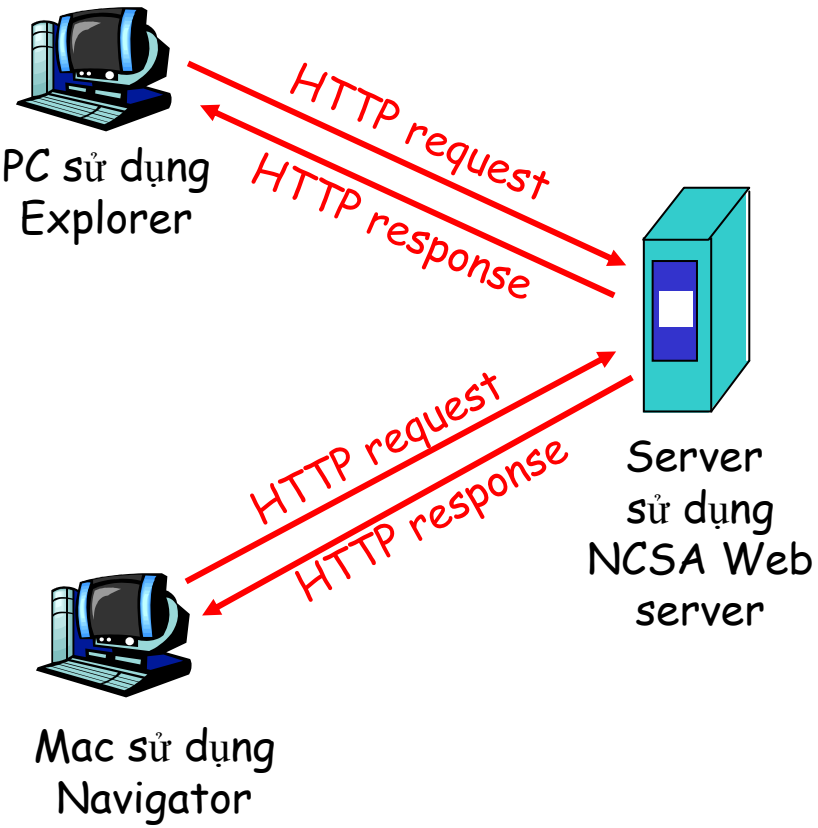
Các ứng dụng trên Internet và giao thức giao vận tương ứng

Ứng dụng	Giao thức Ứng dụng	Giao thức Giao vận tương ứng
Thư tín Điện tử	SMTP [RFC 821]	TCP
Truy cập từ xa	telnet [RFC 854]	TCP
Web	HTTP [RFC 2068]	TCP
Truyền file	FTP [RFC 959]	TCP
Đa phương tiện luồng	Độc quyền (vd. RealNetworks)	TCP hoặc UDP
file server ở xa	NSF	TCP hoặc UDP
Điện thoại Internet	Độc quyền (vd. Vocaltec)	Thường là UDP

Giao thức Web : HTTP

HTTP: Giao thức truyền siêu văn bản.

- Là giao thức tầng ứng dụng cho ứng dụng Web.
- Sử dụng mô hình client/server
  - ***client:*** browser yêu cầu, nhận, hiển thị các đối tượng Web.
  - ***server:*** Web server gửi các đối tượng khi có yêu cầu
- HTTP1.0 : RFC 1945
- HTTP1.1 : RFC 2068



Giao thức HTTP

HTTP sử dụng giao thức TCP.

- Client khởi tạo kết nối TCP (socket) tới server qua cổng 80
- Server chấp nhận kết nối TCP từ Client
- Các thông điệp HTTP : trao đổi giữa Browser (HTTP client) và Web server (HTTP server)
- Đóng kết nối TCP.

HTTP là giao thức "không trạng thái"

- server không lưu lại thông tin về yêu cầu của client

Giao thức lưu lại trạng rất phức tạp !

- Các trạng thái trước đây phải được lưu lại (tốn bộ nhớ).
- Nếu kết nối server/client bị gián đoạn, trạng thái trên chúng có thể sai khác => cần cơ chế điều chỉnh lại

Ví dụ về HTTP

Bạn đánh địa chỉ trên trình duyệt
www.someSchool.edu/someDepartment/home.index

(bao gồm file HTML tham chiếu tới 10 ảnh JPEG)

1a. HTTP client khởi tạo kết nối TCP tới HTTP server tại địa chỉ **www.someSchool.edu**. Cổng mặc định là **80**.

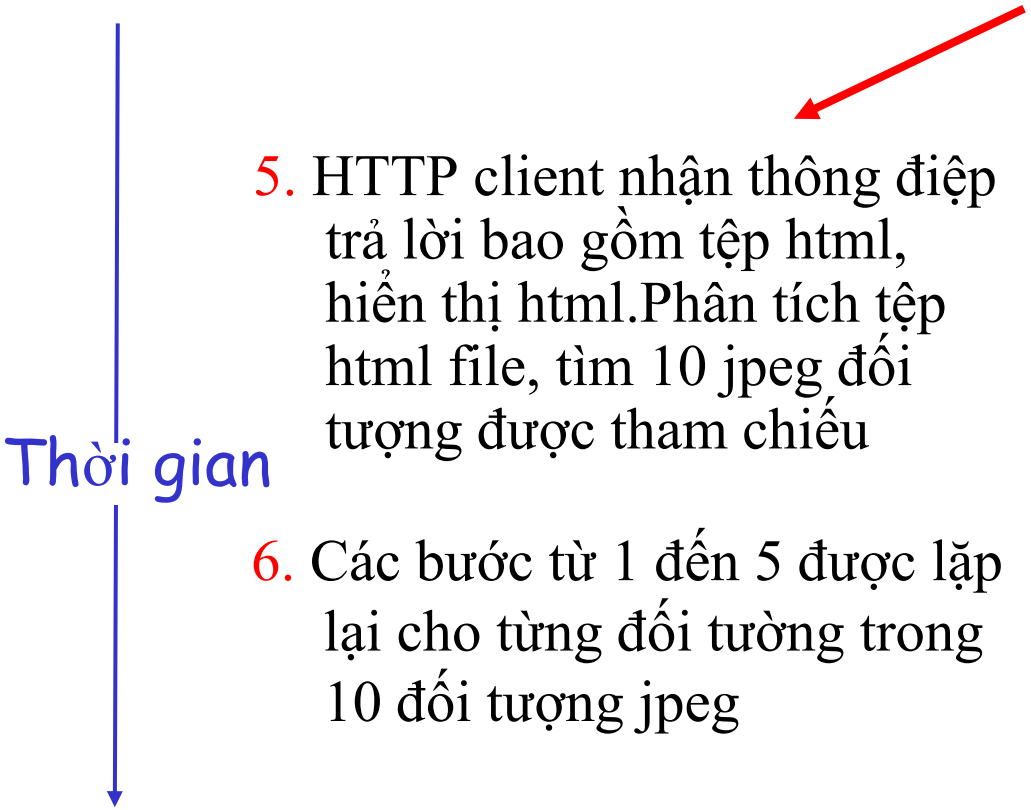
1b. HTTP server ở địa chỉ www.someSchool.edu đợi kết nối TCP ở cổng **80**, chấp nhận kết nối, thông báo lại cho client.

2. HTTP client gửi thông điệp HTTP yêu cầu (bao gồm URL) qua kết nối TCP vừa thiết lập

3. HTTP server nhận thông điệp yêu cầu,lấy các đối tượng được yêu cầu gửi vào trong thông điệp trả lời, (**someDepartment/home.index**) gửi thông điệp vào socket

Thời gian

Ví dụ về HTTP



4. HTTP server đóng kết nối TCP.

5. HTTP client nhận thông điệp trả lời bao gồm tệp html, hiển thị html.Phân tích tệp html file, tìm 10 jpeg đối tượng được tham chiếu

6. Các bước từ 1 đến 5 được lặp lại cho từng đối tường trong 10 đối tượng jpeg

Kết nối liên tục và không liên tục

Không kiên trì:

- HTTP/1.0 : server phân tích yêu cầu, trả lời rồi đóng kết nối TCP.
- Một đối tượng : 2 RTT (độ trễ).
- Mỗi lần truyền, chịu một độ trễ do thiết lập kết nối.
- ***Nhưng phần lớn trình duyệt mở đồng thời nhiều kết nối TCP.***

Kiên trì:

- Mặc định cho HTTP/1.1
- server phân tích yêu cầu, trả lời, phân tích yêu cầu kế tiếp: trên cùng một kết nối TCP
- client gửi yêu cầu cho tất cả các đối tượng khi nhận được file HTML cơ sở
- Ít bước hơn, nhanh hơn.

Định dạng Thông điệp Yêu cầu HTTP

- Có hai kiểu thông điệp HTTP : **Thông điệp yêu cầu** (request) và **Thông điệp trả lời** (response)
- Thông điệp HTTP request:
  - Theo định dạng mã ASCII.

Dòng yêu cầu (Lệnh GET, POST, HEAD)

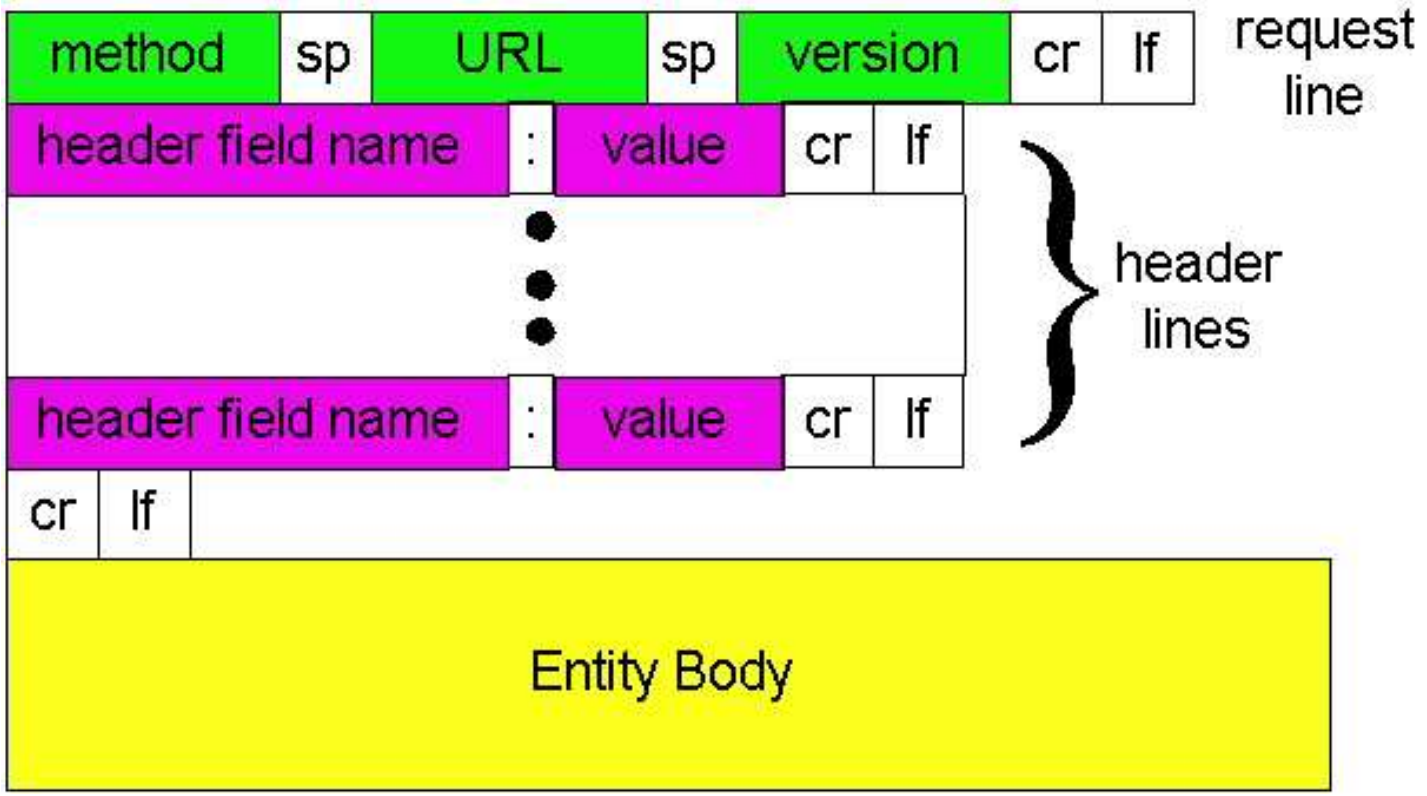
Các dòng header

```
GET /somedir/page.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif,image/jpeg
Accept-language:fr
```

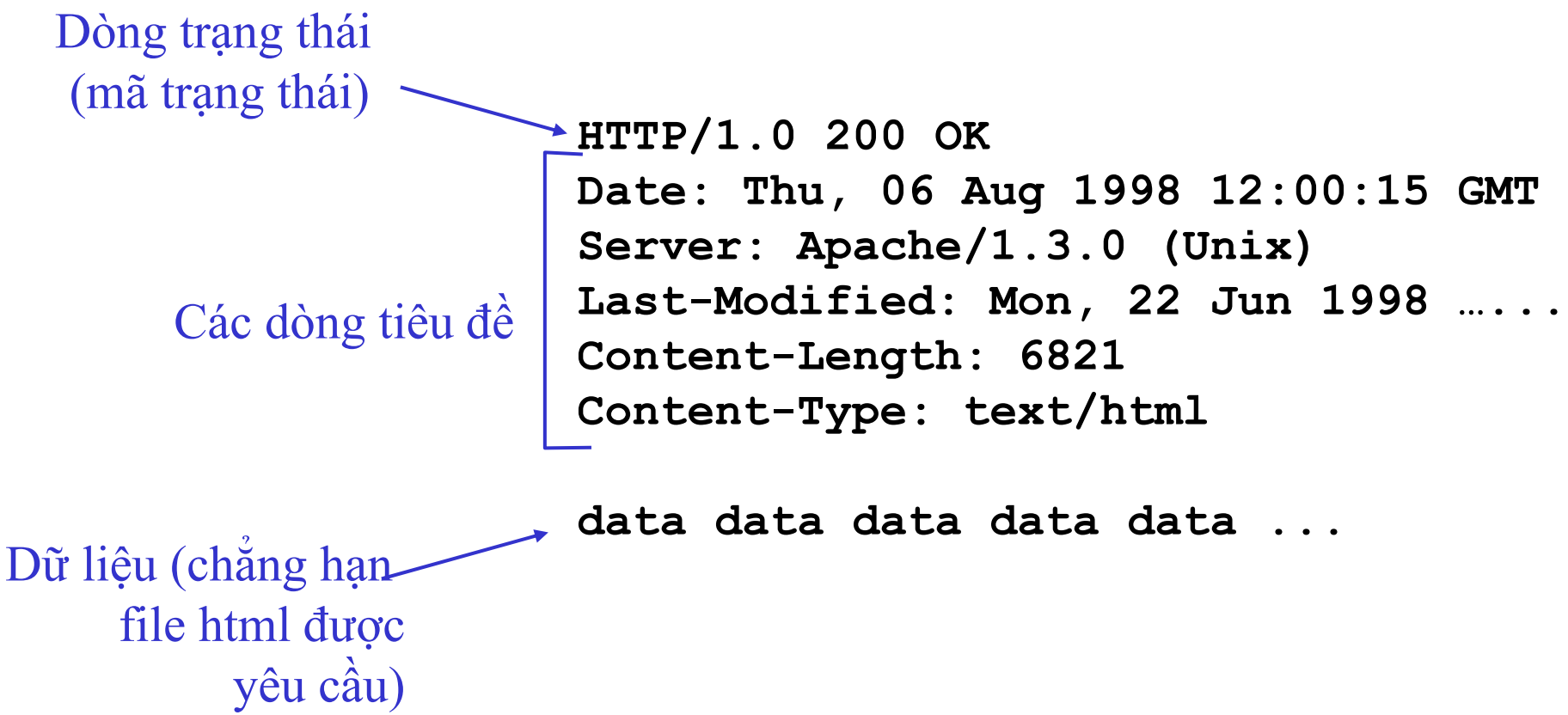
CR,LF: Kết thúc thông điệp

(CR,LF)

Định dạng Thông điệp Yêu cầu HTTP



Định dạng Thông điệp Trả lời HTTP



Mã trạng thái trong Thông điệp Trả lời

Được ghi ở dòng đầu tiên trong thông điệp Server trả lời Client

Một số mã thường gặp:

200 OK

- Yêu cầu thành công, các đối tượng được yêu cầu ở phần thân thông điệp.

301 Moved Permanently

- Đối tượng yêu cầu đã được chuyển và địa chỉ mới của đối tượng được đặt trong trường Location:

400 Bad Request

- Server không hiểu được Thông điệp yêu cầu

404 Not Found

- Đối tượng được yêu cầu không có trong server

505 HTTP Version Not Supported

- Server không hỗ trợ phiên bản giao thức HTTP.

Thử kiểm nghiệm HTTP (phía Client)

1. Telnet tới Web server:

```
telnet www.eurecom.fr 80
```

Tạo kết nối TCP tới cổng 80
(cổng mặc định cho HTTP server)
ở địa chỉ www.eurecom.fr

2. Lệnh GET trong thông điệp HTTP request:

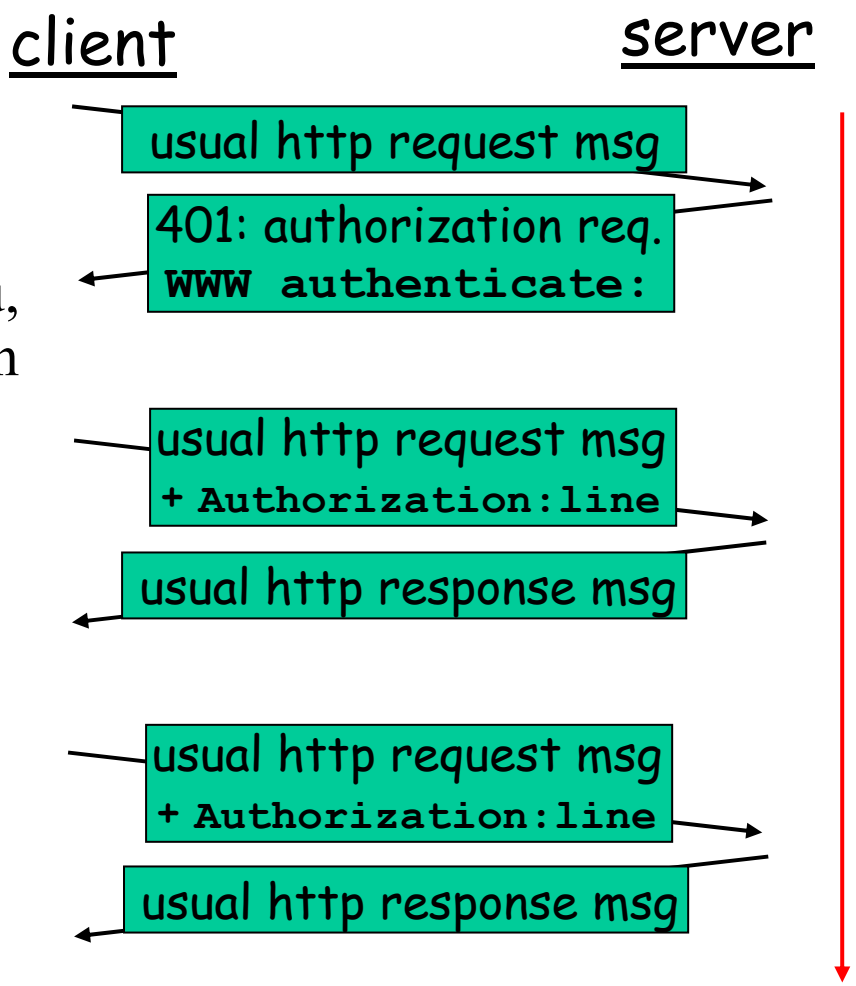
```
GET /~ross/index.html HTTP/1.0
```

Gửi thông điệp yêu cầu lấy tệp
index.html trong thư mục ~ross
cho client

3. **Xem thông điệp response do HTTP server gửi về!**

Tương tác User-Server: Kiểm chứng

- Mục tiêu : kiểm soát quyền truy cập tới đối tượng lưu trên server.
- Phương pháp : sử dụng Tên truy cập và Mật khẩu
- Không trạng thái : mỗi lần yêu cầu, client phải chứng tỏ mình có quyền
- Kiểm chứng :
  - Tiêu đề **`authorization:`** trong mỗi yêu cầu gửi đi.
  - Nếu không có quyền : server từ chối truy cập và gửi yêu cầu có trường tiêu đề **`WWW authenticate:`**



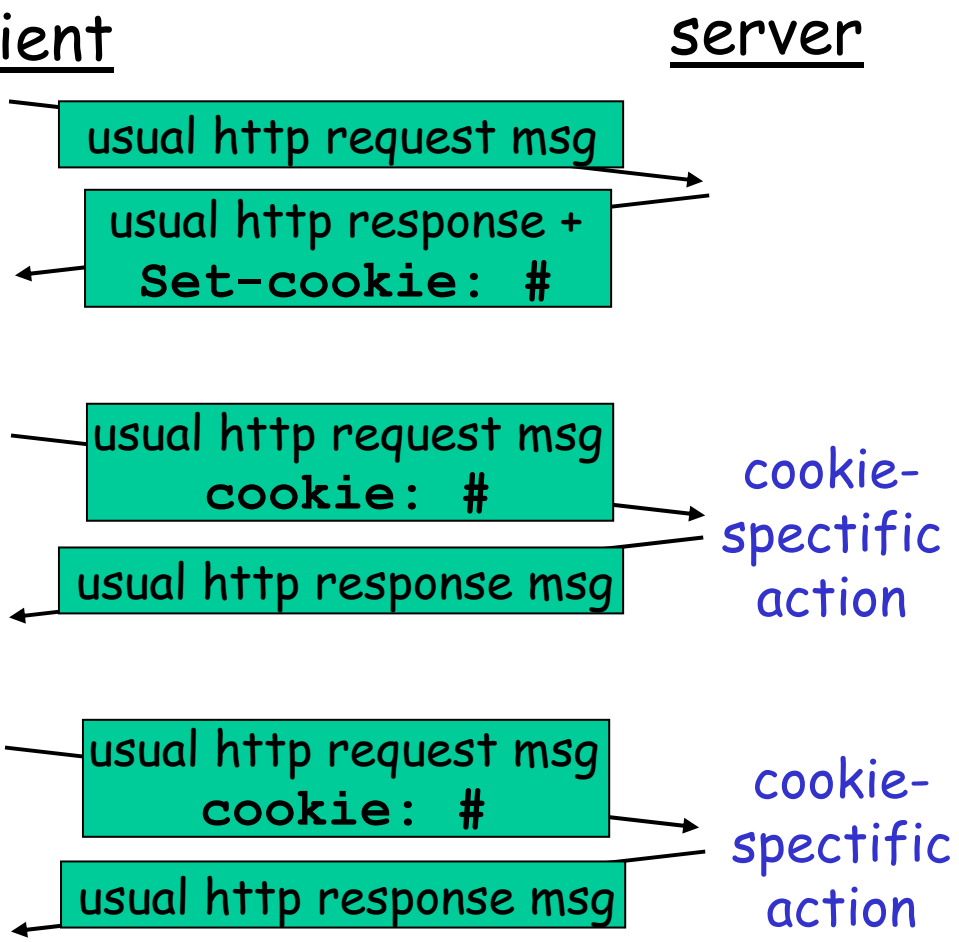
Thông thường, trình duyệt cache username/pass để người dùng không phải gõ lại trong mỗi truy cập

Lưu lại trạng thái nhờ Cookie

- Do Server tạo ra, lưu lại để sử dụng về sau:
  - Thẩm định quyền
  - Ghi nhớ các "sở thích", hoạt động của người dùng.
- Server gửi "cookie" tới client trong thông điệp response.

  Set-cookie: 1678453
- Client gửi kèm cookie trong những lần yêu cầu sau.

  cookie: 1678453

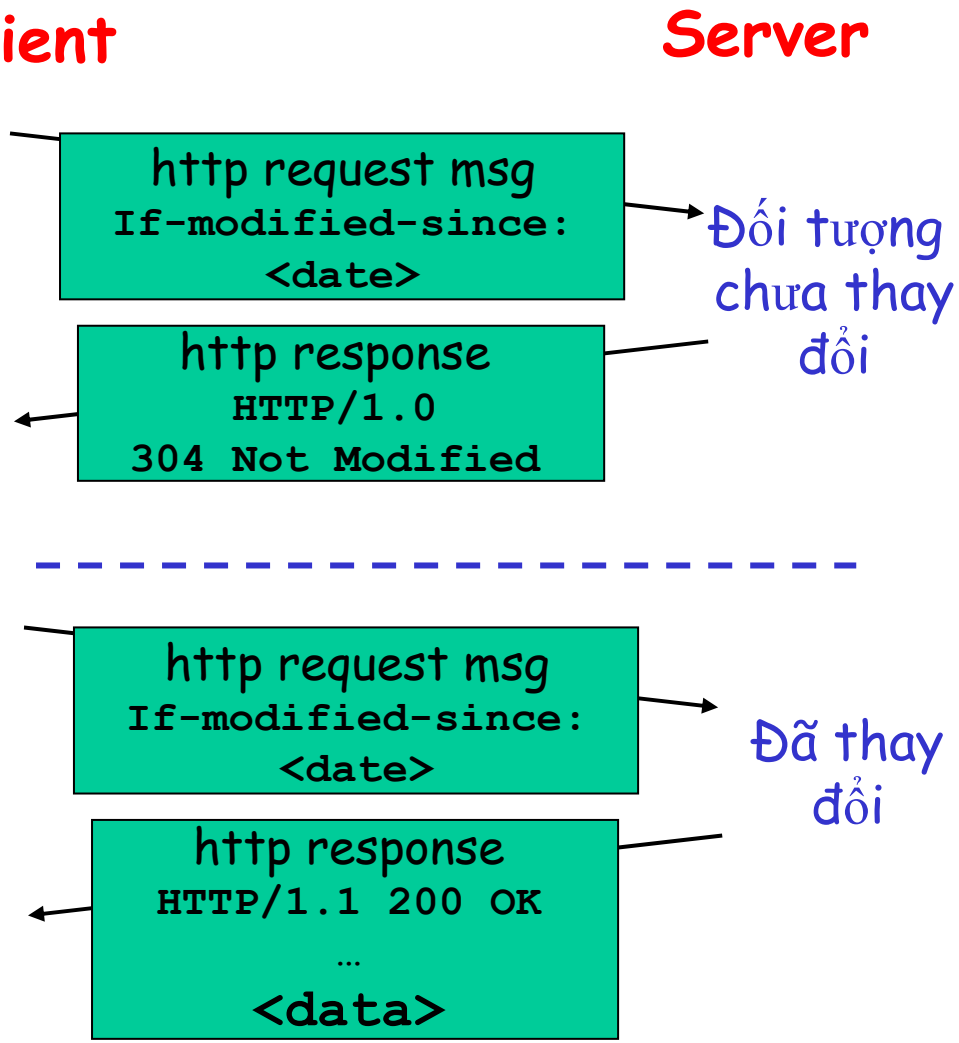


GET có điều kiện : caching bên client

- **Mục đích** : không gửi lại nếu client đã có bản mới nhất của đối tượng trong cache.
- Client: chỉ ra ngày lấy đối tượng về trong thông điệp HTTP request.

  If-modified-since: <date>
- Server: Thông điệp trả về sẽ không kèm theo đối tượng nếu đối tượng đó chưa bị thay đổi trên server:

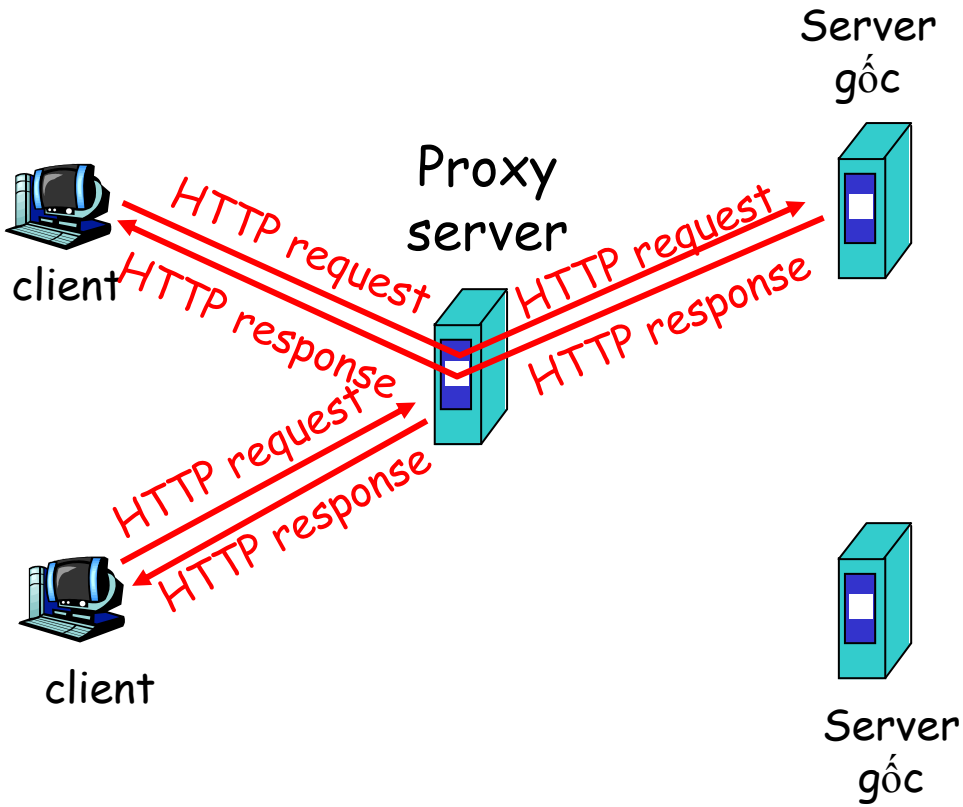
  HTTP/1.0 304 Not modified



Web Caches (Proxy Server)

Mục đích : Đáp ứng yêu cầu từ client mà không cần tới server thật sự chứa đối tượng (server gốc)

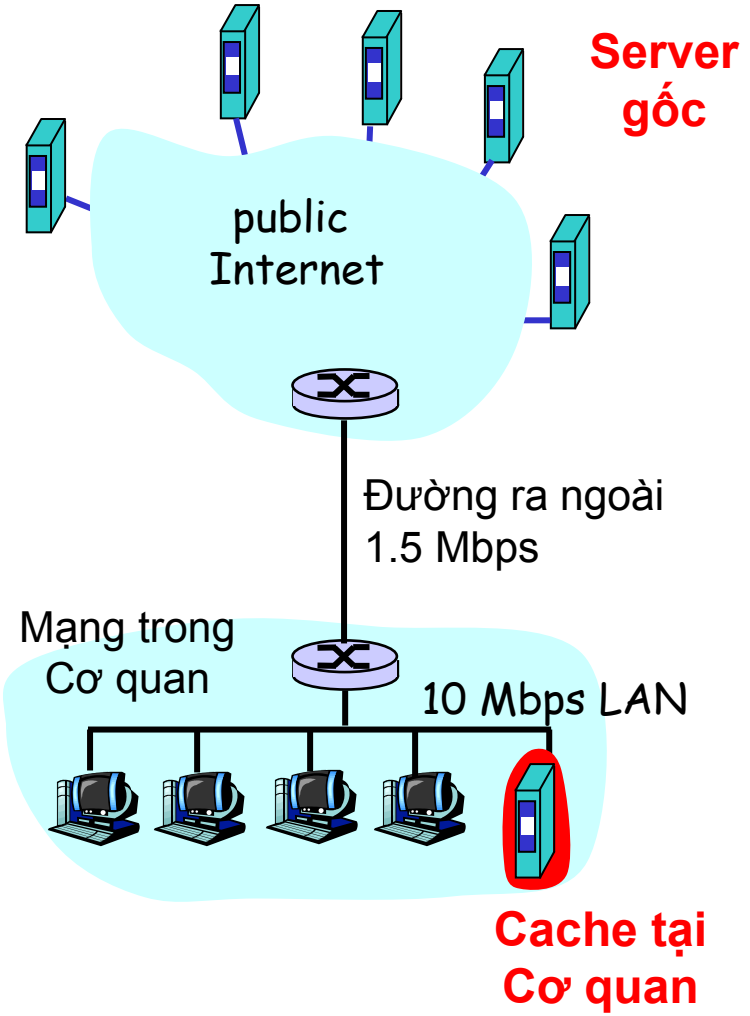
- Người dùng cấu hình browser truy nhập Web thông qua Web Cache
- Client gửi tất cả các yêu cầu HTTP request tới Web Cache
  - **Nếu có đối tượng**, Web Cache gửi đối tượng về cho client.
  - **Nếu không có**, Web Cache yêu cầu đối tượng từ server thực, sau đó gửi đối tượng cho client



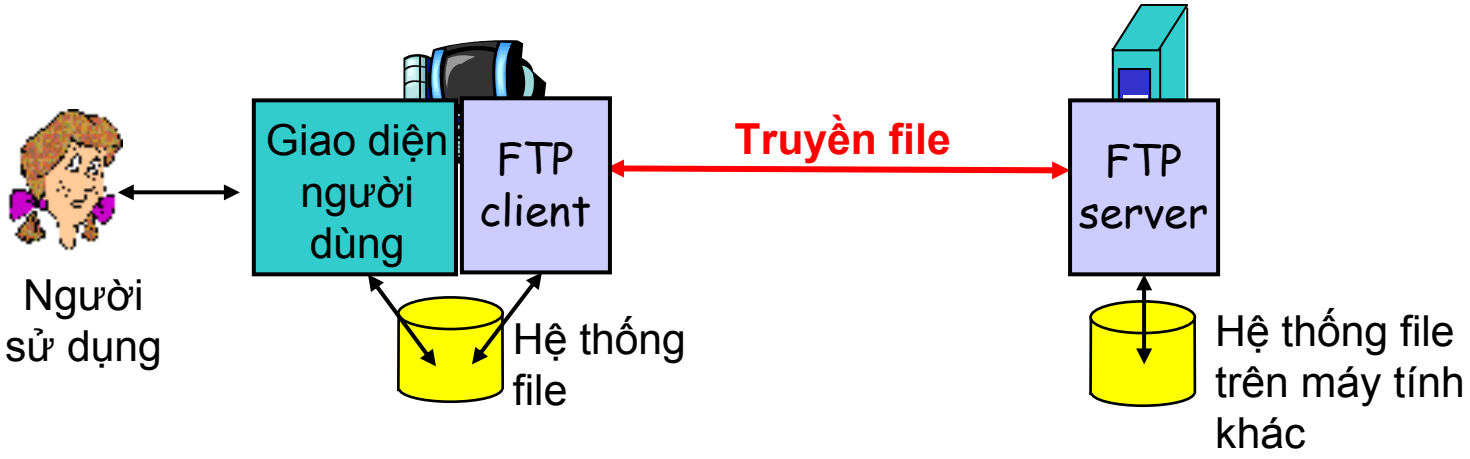
Tại sao sử dụng Web Caching ?

Giả định rằng : cache gần client hơn (có thể trong cùng mạng LAN)

- Giảm thời gian client phải đợi
- Giảm tải Mạng
  - Đường kết nối ra ngoài mạng nội bộ thường xảy ra hiện tượng "thắt nút cổ chai".



Giao thức Truyền file FTP

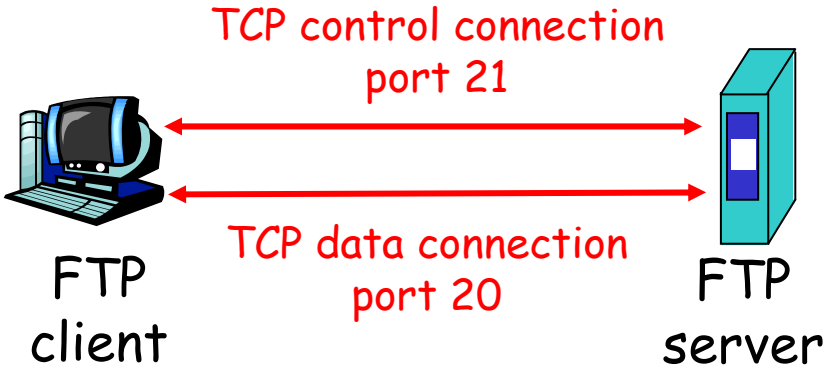


- Là giao thức trao đổi file với máy tính ở xa (remote host).
- Sử dụng mô hình client/server
  - *client:* khởi tạo kết nối.
  - *server:* máy tính ở xa (remote host)
- FTP được đặc tả trong RFC 959
- FTP server: port 21

Hai kênh kết nối FTP : Kiểm soát và Dữ liệu

- FTP client kết nối với FTP server qua cổng 21.
- Hai kết nối TCP đồng thời được tạo:
  - **Kiểm soát :** trao đổi lệnh, phản hồi giữa client và server.

    "out of band control"
  - **Dữ liệu :** kết nối tải file từ client đến server hay từ server về client.
- FTP server lưu lại trạng thái : thư mục hiện thời, lần truy nhập gần đây nhất



Các Lệnh và Trả lời thường gặp

Các lệnh thường gặp:

- Được mã hoá bằng mã ASCII
- **USER** ***username***
- **PASS** ***password***
- **LIST** trả về danh sách các file và thư mục trong thư mục hiện thời.
- **RETR filename** lấy file từ thư mục hiện thời.
- **STOR filename** Tải file vào thư mục hiện thời trên máy tính ở xa

Các mã trả về thường gặp

- Tương tự HTTP
- **331** chấp nhận username, yêu cầu password
- **125** kết nối dữ liệu được thiết lập, chuẩn bị truyền dữ liệu.
- **425** Không thể thiết lập kết nối dữ liệu
- **452** Lỗi ghi file.

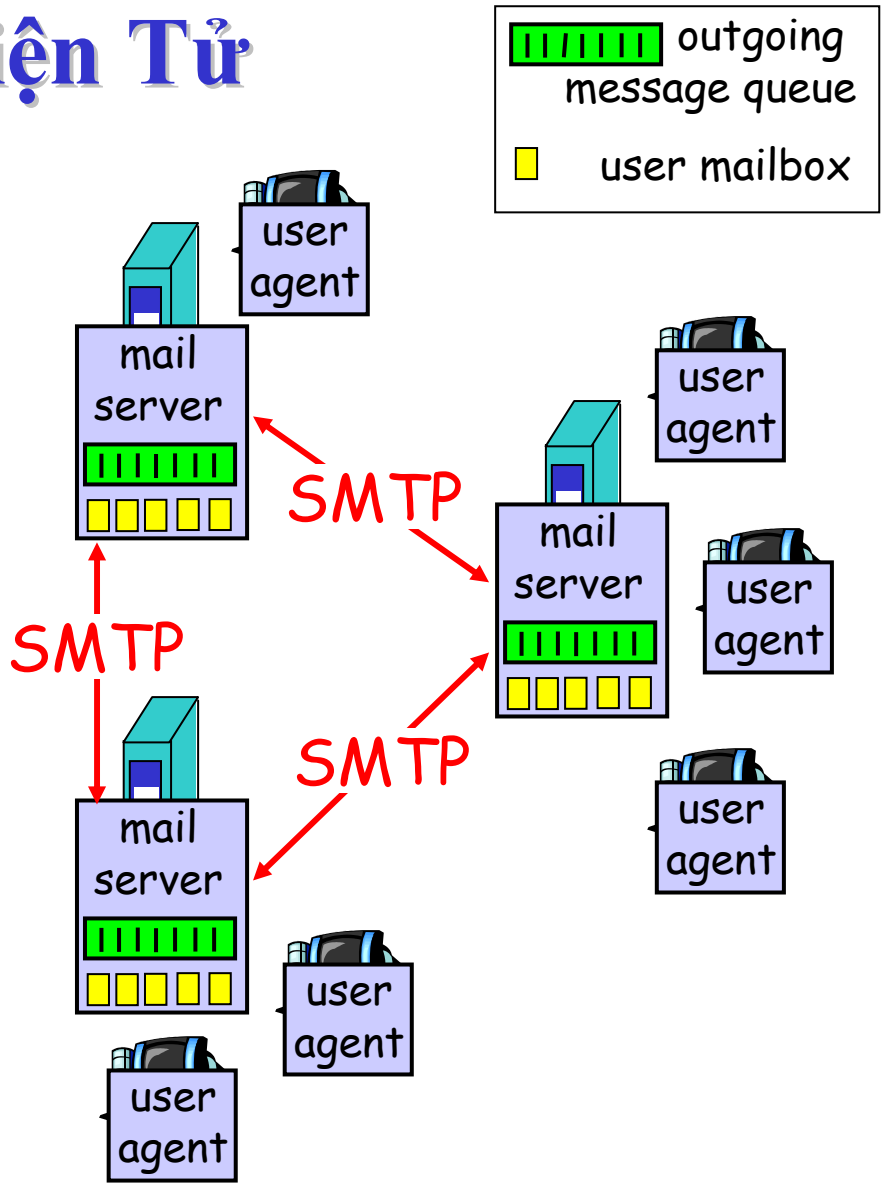
Thư Điện Tử

Gồm 3 thành phần chính:

- user agent
- mail server
- **S**imple **M**ail **T**ransfer **P**rotocol: **SMTP**

User Agent

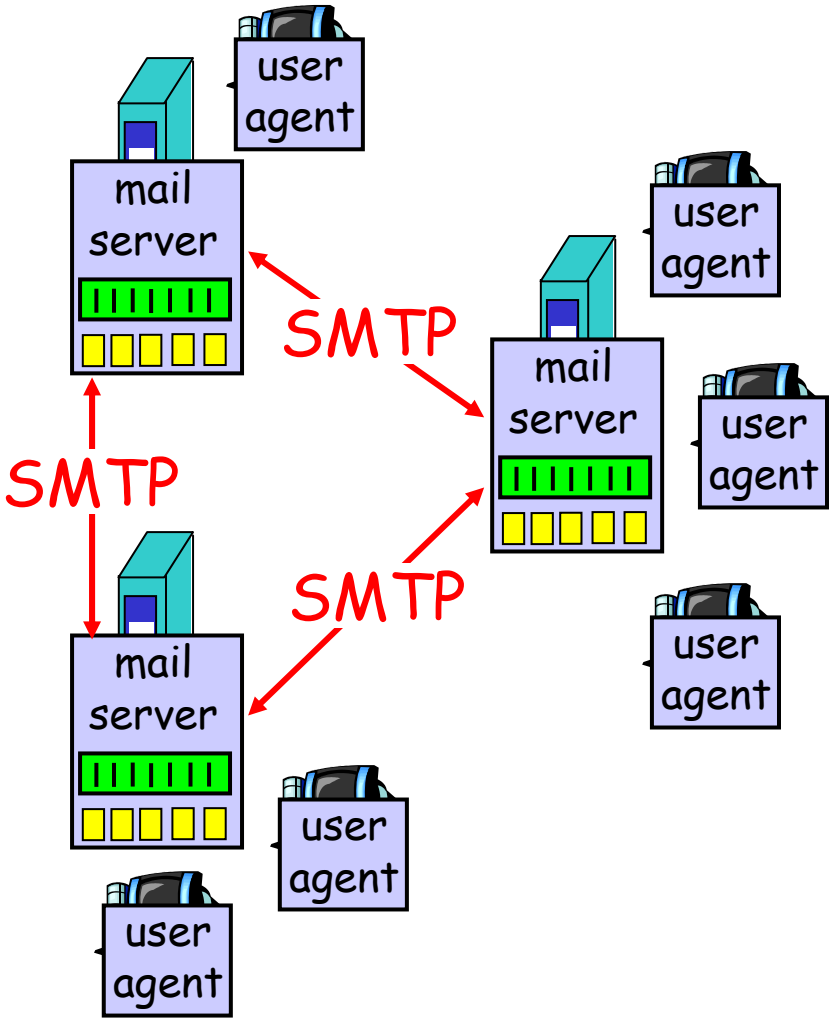
- Còn gọi là "mail reader"
- Soạn thảo và Đọc thư
- Ví dụ: Eudora, Outlook, elm, Netscape Messenger
- Các thông điệp outgoing, incoming được lưu trên server



Thư điện tử : Mail Server

Mail Server

- Hộp thư lưu lại các thư điện tử (có thể chưa được đọc) của người sử dụng.
- Hàng đợi chứa các thư sẽ được gửi đi.
- Giao thức SMTP là giao thức để các mail server trao đổi email.
  - Là client khi gửi thư.
  - Là server khi nhận thư.



Thư điện tử : Mail Server [RFC 821]

- Sử dụng dịch vụ TCP, truyền email tin cậy từ SMTP client tới SMTP server qua cổng 25
- Truyền trực tiếp : server gửi tới server nhận.
- Truyền qua ba giai đoạn :
  - "Bắt tay"
  - Truyền các thông điệp (Thư)
  - Đóng kết nối.
- Tương tác: Lệnh (client => server) và Trả lời (Server => Client)
  - **Lệnh** : mã bằng bảng mã ASCII.
  - **Trả lời** : mã trạng thái và có thể có thêm giải thích
- Các thông điệp phải được mã bằng bảng mã ASCII 7 bit

Một số tương tác SMTP

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C:   How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Tự thử nghiệm tương tác bằng SMTP

- **telnet servername 25**
- Nhận được hồi đáp có mã 220 từ server.
- Đánh các lệnh **HELO, MAIL FROM, RCPT TO, DATA, QUIT**.
- Các lệnh trên cho phép bạn gửi thư mà không cần dùng đến email client (reader)

SMTP : Kết thúc

- Giao thức SMTP sử dụng kết nối bền vững.
- SMTP đòi hỏi thông điệp (header & body) phải được định dạng bằng mã ASCII 7bit.
- Những xâu kí tự đặc biệt (ví dụ CRLF.CRLF) không được phép ghi vào thông điệp (do đó chuỗi ký tự này phải được mã hóa)
- SMTP server sử dụng CRLF.CRLF để đánh dấu kết thúc thông điệp.

So sánh với HTTP:

- HTTP: giao thức kiểu "**kéo**" ("kéo" thông tin từ server về)
- Email :là giao thức kiểu "**đẩy**" ("đẩy" thông tin lên server)
- Cả hai đều tương tác gửi **lệnh** (mã ASCII)/ **trả lời** (mã trạng thái).
- HTTP : Mỗi đối tượng nằm trong một thông điệp riêng.
- SMTP: nhiều đối tượng nằm trong cùng một thông điệp

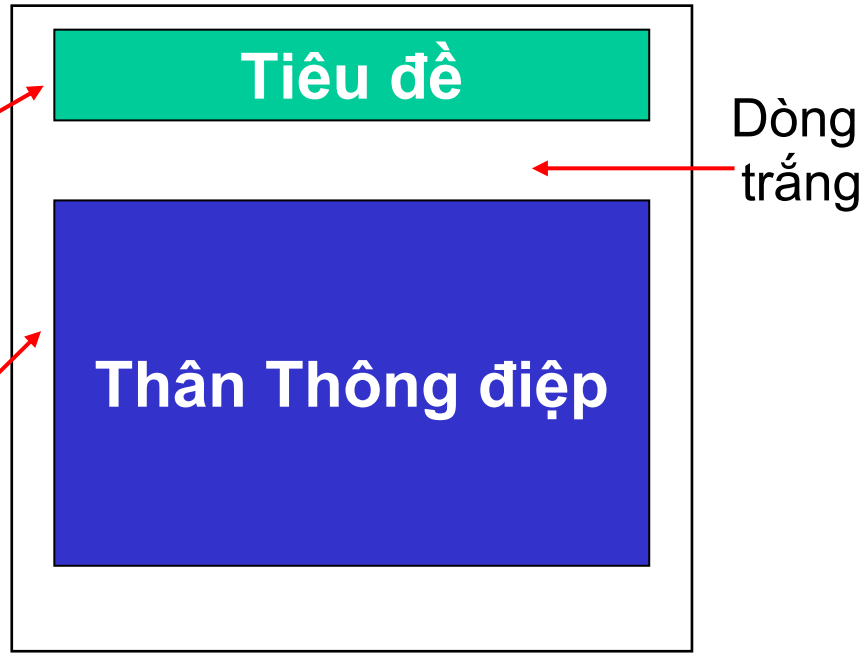
Định dạng Thông điệp eMail

SMTP : Giao thức để trao đổi email.

RFC 822 : chuẩn định dạng thông điệp email:

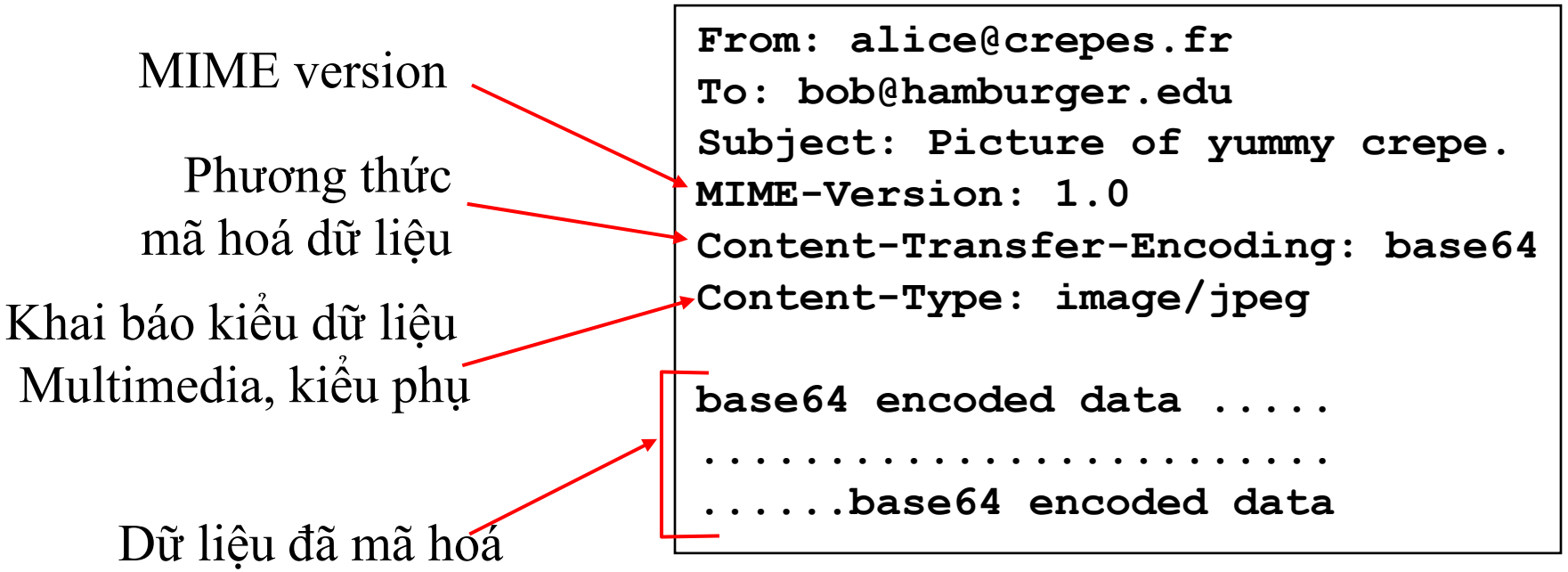
- **Tiêu đề Thông điệp :**
  - To:
  - From:
  - Subject:

  rất khác so với lệnh của SMTP !
- **Thân thông điệp**
  - Chỉ bao gồm các ký tự kiểu ASCII



Định dạng thông điệp mở rộng đa phương tiện

- **MIME** : Multimedia Mail Extension, RFC 2045, 2056
- Bổ sung thêm một dòng trong phần tiêu đề của thông điệp để mô tả kiểu MIME.



Kiểu MIME

`Content-Type: type/subtype; parameters`

Text

- Kiểu : **plain, html**

Image

- Kiểu : **jpeg, gif**

Audio

- Kiểu : **basic** (loại có quy luật hoá 8-bit), **32kadpcm** (mã hóa 32 kbps)

Video

- Kiểu : mpeg, quicktime

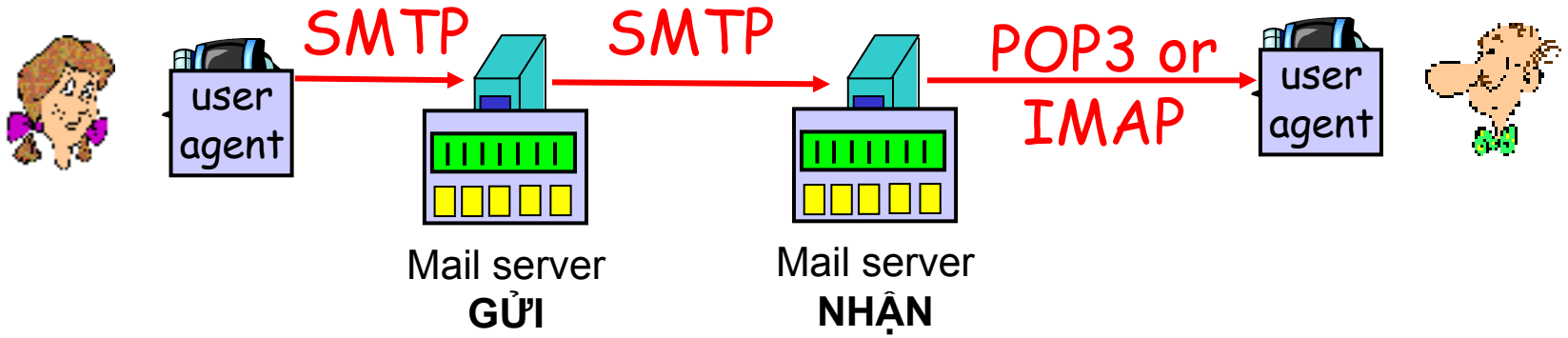
Application

- Các loại dữ liệu khác phải được xử lý bằng chương trình đọc tương ứng mới có thể đọc, xem được.
- Ví dụ các kiểu: msword, octet-stream

Kiểu chứa nhiều loại đối tượng

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Dear Bob,
Please find a picture of a crepe.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
--98766789--
```

Các giao thức truy nhập Mail



- **SMTP:** Gửi thư tới Server chứa thư của người nhận
- Giao thức đọc thư : lấy thư từ server
  - **POP : Post Office Protocol [RFC 1939]**
    - Kiểm chứng (agent <-->server) và tải thư về
  - **IMAP: Internet Mail Access Protocol [RFC 1730]**
    - Phức tạp hơn do có nhiều đặc tính hơn.
    - Thao tác trên các thư lưu trên server.
  - **HTTP : Hotmail , Yahoo! Mail, Gmail,…**

Giao thức POP3

Giai đoạn kiểm chứng

- Các lệnh Client gửi:
  - **user:** username
  - **pass:** password
- Server trả lời
  - **+OK**
  - **-ERR**

Giai đoạn xử lý,cập nhật :

- **list:** in ra danh sách các thư(được đánh số ID).
- **retr:** lấy thư có ID là số nhập vào
- **dele:** xoá thư
- **quit**

```
S: +OK POP3 server ready
C: user alice
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

DNS : Hệ thống Tên miền

Con người : có thể nhận dạng bằng nhiều cách:

- Số Chứng Minh Thư
- Tên, Biệt danh
- Số hộ chiếu

Máy tính và Router trên Internet

- Địa chỉ IP (32 bit) - sử dụng để đánh địa chỉ cho các datagram
- "Tên", ví dụ như gaia.cs.umass.edu

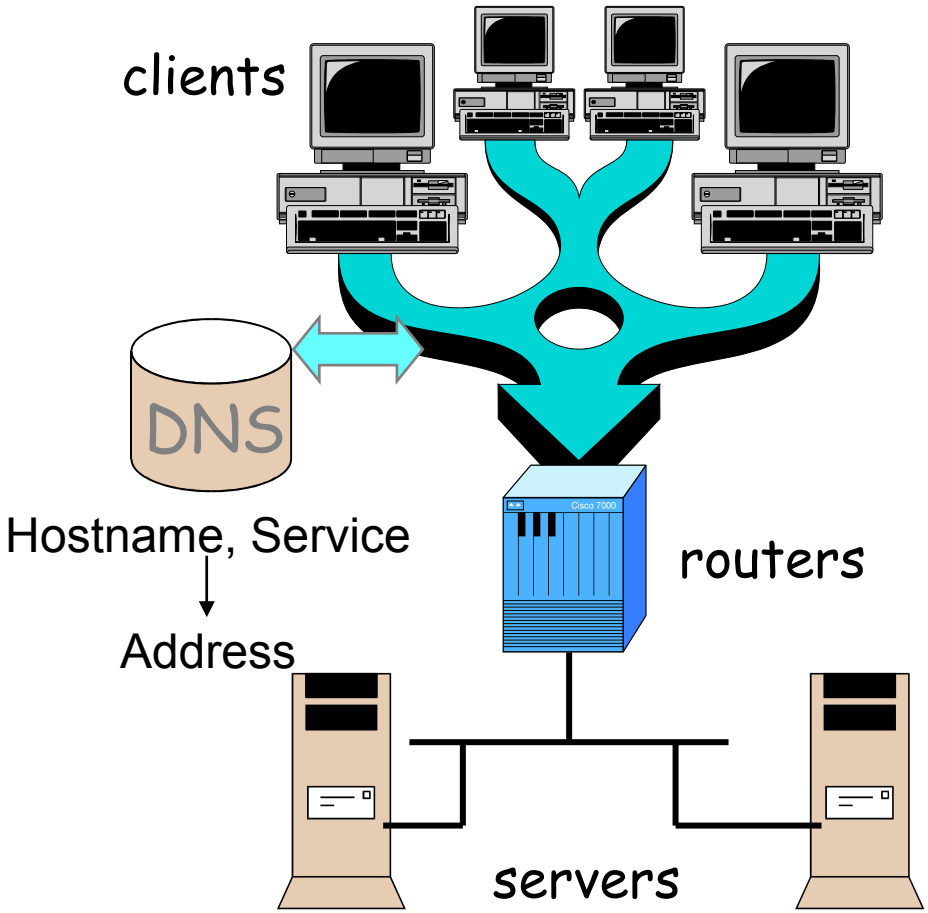
Q: Ánh xạ giữa Địa chỉ IP và Tên?

Domain Name System:

- Là **Hệ cơ sở dữ liệu phân tán** cài đặt bởi nhiều **name servers** phân cấp
- **Giao thức tầng ứng dụng** : host, routers yêu cầu name servers để giải mã tên (ánh xạ địa chỉ <->tên)
  - Chú ý : Chức năng cơ bản của Internet hoạt động như giao thức tầng Ứng dụng
  - "Phức tạp" đặt ở "rìa"

DNS: Hệ thống tên miền

- Chức năng
  - Ánh xạ (Tên miền, dịch vụ) đến một giá trị nào đó
    - (www.cs.yale.edu, Addr) -> 128.36.229.30
    - (cs.yale.edu, Email) -> netra.cs.yale.edu
    - (netra.cs.yale.edu, Addr) -> 128.36.229.21
- Tại sao không sử dụng địa chỉ IP trực tiếp ?



DNS : Name Server (Máy chủ Tên)

Tại sao tạo ra một DNS Server tập trung ?

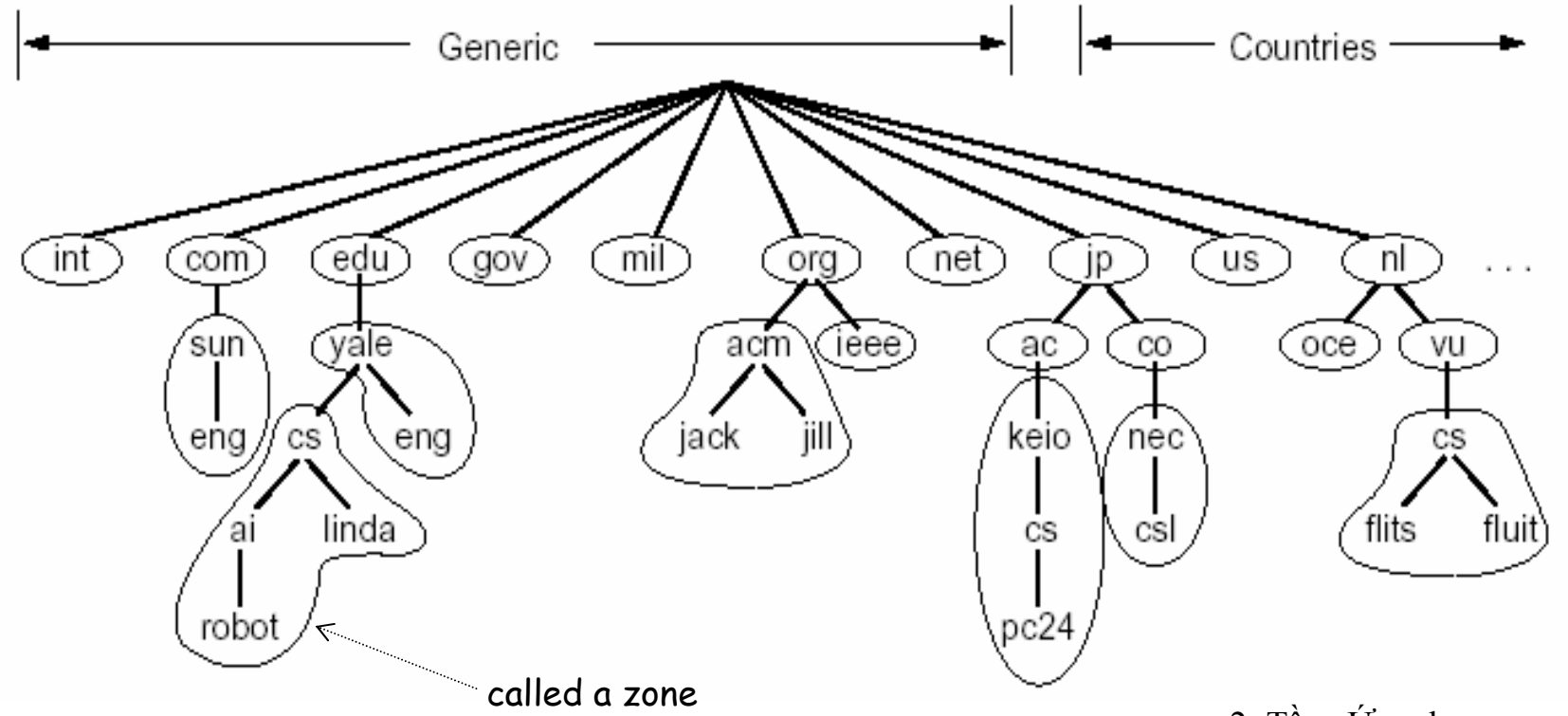
- Điểm hỏng duy nhất - nếu name-server "chết" thì cả mạng Internet sẽ "chết" theo.
- Khối lượng giao dịch tại điểm tập trung lớn.
- Cơ sở dữ liệu tập trung ở "xa" với nhiều nơi
- Bảo trì dễ hơn.

Không* MỞ RỘNG *được !

- Không có server nào có thể lưu toàn bộ được tên miền và địa chỉ IP tương ứng
- **local name servers:**
  - Mỗi ISP, công ty có *local name server (ngầm định)*
  - Câu hỏi truy vấn của host về DNS sẽ được chuyển tới local name server
- **Chức năng của name server:**
  - Đối với host: lưu địa chỉ IP và tên miền tương ứng của host
  - Có thể tìm tên miền ứng với địa chỉ IP và ngược lại

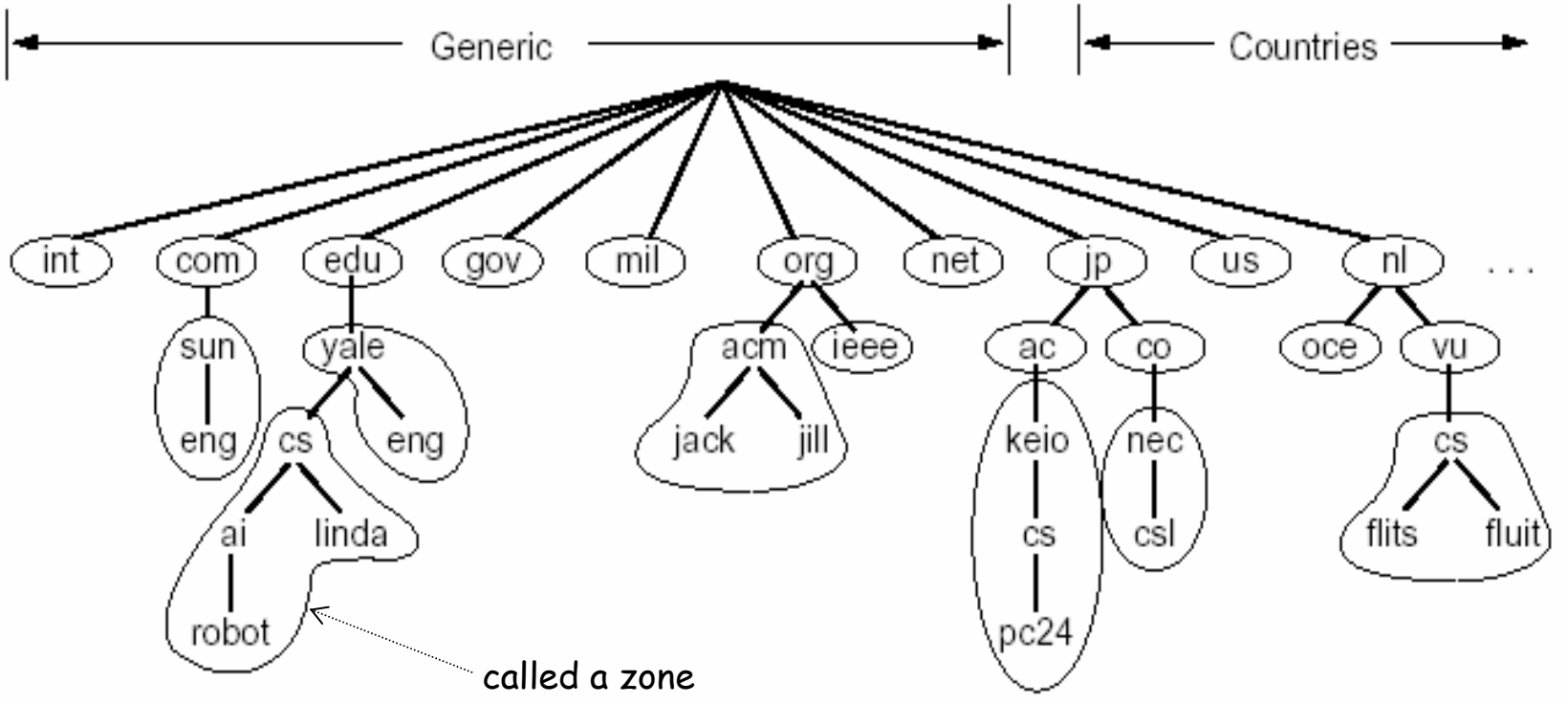
DNS: Đặt tên như thế nào ?

- Cơ chế Đặt tên
  - Không gian tên (phân cấp) được chia thành các Vùng (zone)
  - Mỗi vùng có thể được coi là Nhánh của cây tổng quát



Quản lý Phân tán Không gian Tên

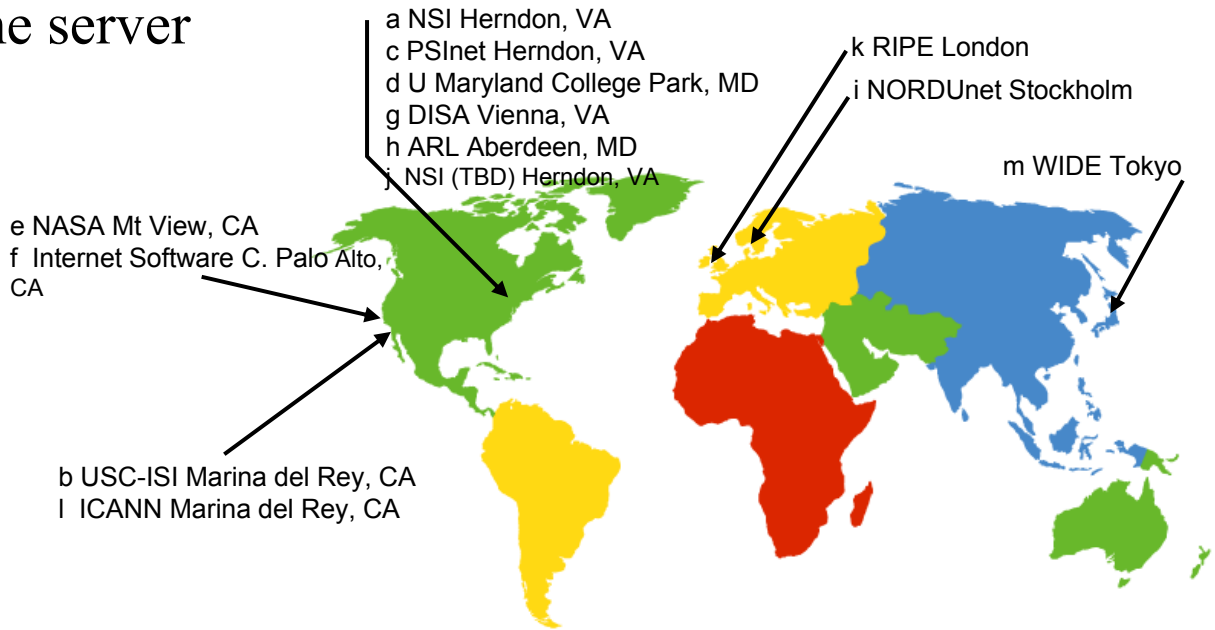
- Là Cơ sở dữ liệu phân tán được nhiều authoritative name server quản lý
  - Mỗi Vùng có một **Authoritative Name Server** riêng
  - authoritative name server of a zone có thể **trao quyền quản lý** một bộ phận trong Vùng của mình (tức là một nhánh con) cho name server khác



DNS : Root Name Server

- Local name server sẽ hỏi Root name server khi không xác định được ánh xạ.
- Root name server:
  - Hỏi authoritative name server nếu không trả lời được
  - Nhận câu trả lời từ authoritative name server
  - Trả lời local name server

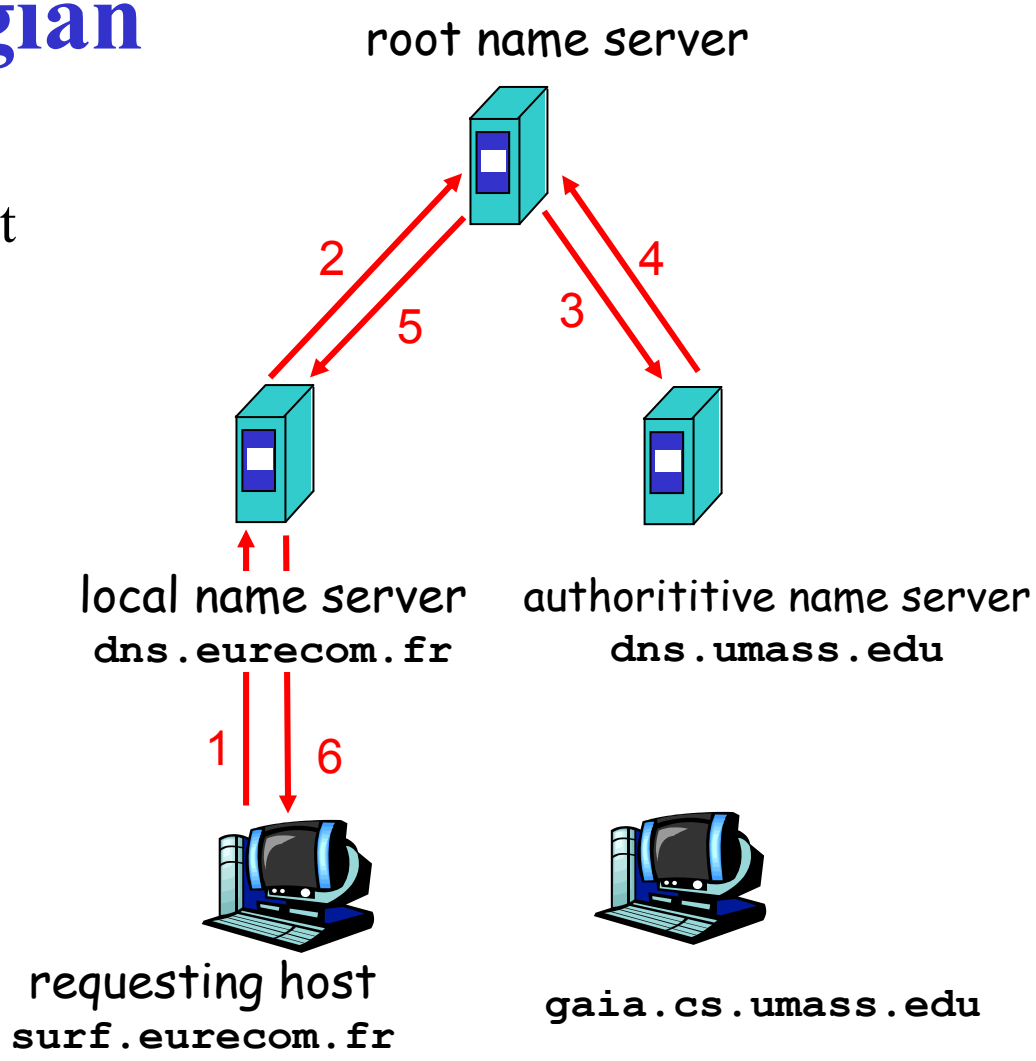
13 root name servers trên Thế giới



DNS : Ví dụ đơn giản

host **surf.eurecom.fr** muốn biết địa chỉ IP của **gaia.cs.umass.edu**

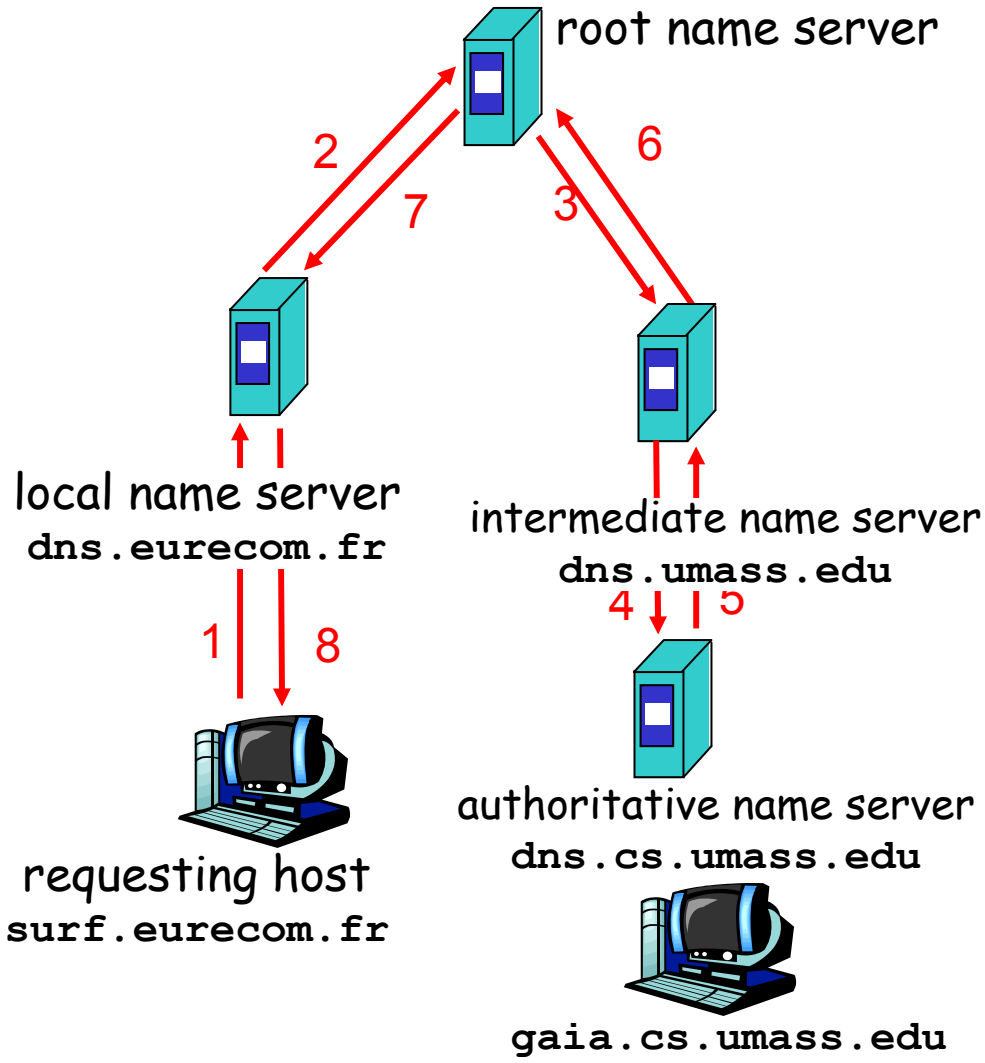
1. Hỏi local DNS server (**dns.eurecom.fr**)
2. **dns.eurecom.fr** hỏi root name server nếu cần thiết
3. root name server hỏi authoritative name server, **dns.umass.edu** nếu cần thiết.



Name Server Trung gian

Root name server:

- Có thể không biết authoritative name server
- Chỉ biết Name Server ***trung gian,*** qua đó mới tìm được authoritative name server



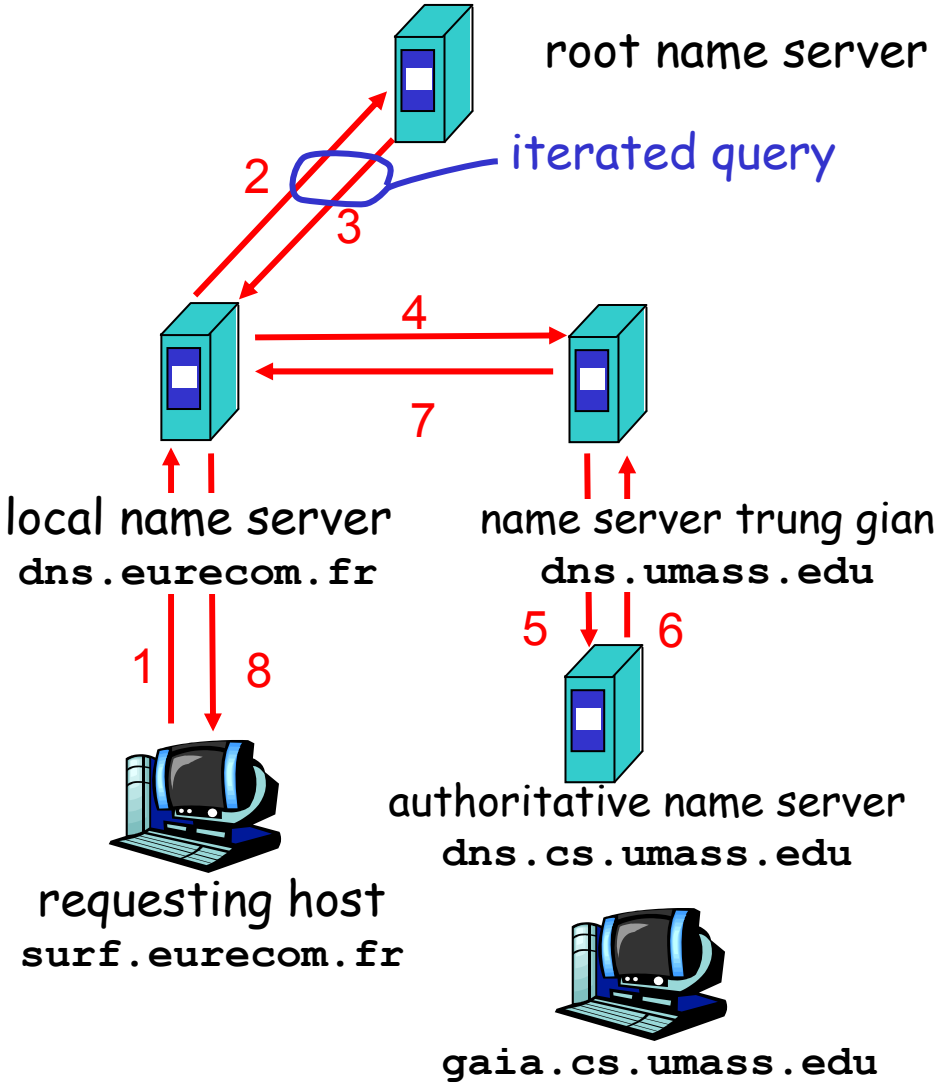
DNS: các loại truy vấn

Truy vấn đệ quy :

- Name server là nơi giải mã địa chỉ/tên. Nếu không tự mình giải mã được sẽ gửi yêu cầu đến name server khác.
- Root name server liệu có bị quá tải ?

Truy vấn tương tác:

- Nếu không phân giải được địa chỉ IP, gửi thông điệp "*Tôi không biết, hãy hỏi bạn tôi là A*". A là địa chỉ IP của name server kế tiếp.



DNS: Lưu tạm và Cập nhật bản ghi

- Khi "học" được thêm một ánh xạ, name server sẽ "ghi nhớ" ánh xạ này
  - Sau một khoảng thời gian, nếu thành phần nào trong cache không được sử dụng thì sẽ bị xóa bỏ.

- Cơ chế Cập nhật và Thông báo do IETF thiết kế:
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

Bản ghi DNS

DNS: cơ sở dữ liệu phân tán lưu các Bản ghi Tài nguyên (RR)

Định dạng RR : (name, value, type, TTL)

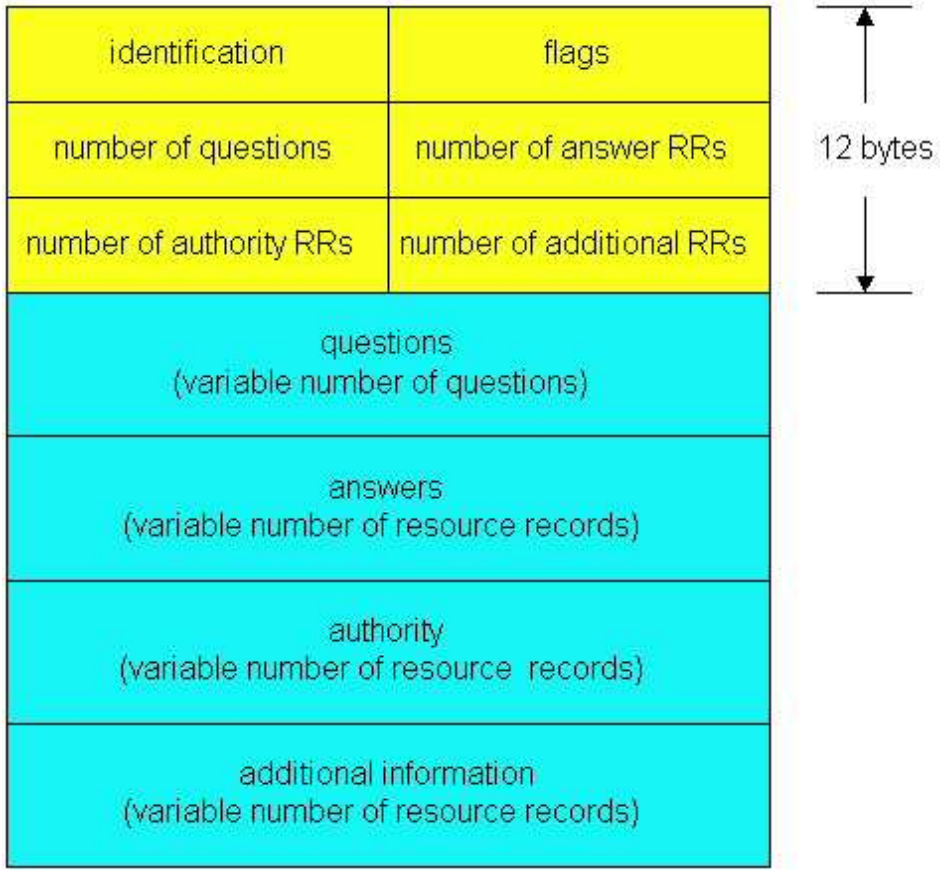
- **Type=A**
  - **name** : hostname
  - **value** : IP address
- **Type=NS**
  - **name** : domain (ví dụ foo.com)
  - **value** : địa chỉ IP của authoritative name server ứng với miền đó
- **Type=CNAME**
  - **name** : tên bí danh cho một tên thực nào đó : ví dụ *www.ibm.com* là tên bí danh của *servereast.backup2.ibm.com*
  - **value** : tên thực
- **Type=MX**
  - **value** : tên của mailserver

Giao thức và Thông điệp của DNS

Giao thức DNS : Thông điệp ***truy vấn*** và ***trả lời*** đều có chung một định dạng

Tiêu đề Thông điệp

- **identification:** 16 bit. Thông điệp trả lời và truy vấn có cùng định danh.
- **Cờ:**
  - Cờ query hay reply
  - Cờ mong muốn đệ quy
  - Cờ chấp nhận đệ quy
  - Cờ authoritative



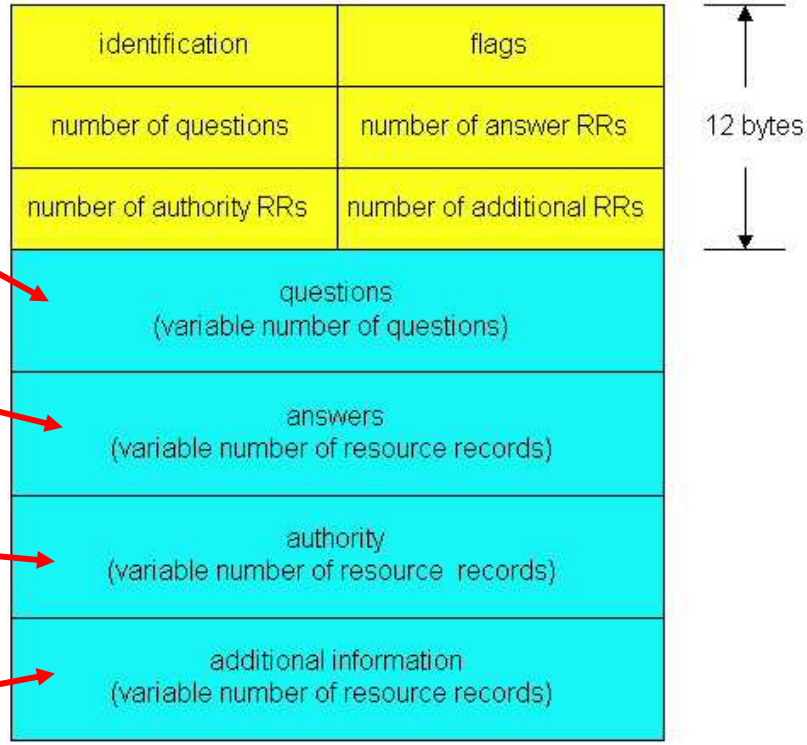
DNS – Giao thức và Thông điệp

Trường tên và Kiểu truy vấn

Thông điệp trả lời

Các bản ghi của authoritative server

Những thông tin ngoại lệ có thể hữu dụng



Quan sát DNS

- Sử dụng lệnh **dig** (Hoặc **nslookup**):
  - Yêu cầu Hỏi bằng câu hỏi tương tác:
    %dig +trace www.cnn.com

- Bắt thông điệp bằng Ethereal
  - DNS server lắng nghe ở port 53

DNS – Ưu điểm ?

- Phân cấp : nâng cao năng lực quản lý và tăng cường khả năng mở rộng
- Nhiều server : tăng khả năng phòng chống lỗi
  - Xem http://www.internetnews.com/dev-news/article.php/1486981 để thấy Tấn công từ chối Dịch vụ (DDoS) vào hệ thống root server vào tháng 10/2002 (9 trong 13 root server bị đình trệ, nhưng mạng chỉ bị chậm đi không đáng kể)
  - Xem http://www.cymru.com/DNS/index.html để thấy hiệu suất được giám sát như thế nào
- Caching làm giảm tải và giảm thời gian phản hồi

DNS – Nhược điểm

- Hệ thống Tên miền không phải là phương thức tốt nhất để đặt tên các tài nguyên khác, chẳng hạn file
- Số lượng giới hạn Kiểu tài nguyên hạn chế khả năng đưa thêm các dịch vụ mới
- Mặc dù về mặt lý thuyết có thể cập nhật bản ghi tài nguyên, nhưng trên thực tế hiếm khi làm được.
- Mô hình truy vấn đơn giản => khó cài đặt những dạng truy vấn phức tạp
- Kết nối sớm (Tách biệt truy vấn DNS với ứng dụng đưa ra truy vấn) không hiệu quả trong môi trường di động và thay đổi thường xuyên
  - Ví dụ : Cân bằng tải, Tìm máy in gần nhất

Giải pháp Phân giải Tên kiểu Linda

- Nhiều đề xuất dựa trên "Không gian làm việc Phân tán" (Linda) do David Gelernter đưa ra
  - Intentional Naming System (INS),
  - Internet Indirect Infrastructure (I3)

- Nút viết các các tuples (một dạng vector không kiểu) vào các "không gian dùng chung"
- Nút đọc các tuple phù hợp từ không gian dùng chung

Lập trình Socket

Mục đích : Nghiên cứu cách xây dựng ứng dụng client/server giao tiếp qua socket

Socket API

- BSD4.1 UNIX, 1981
- Ứng dụng Tạo, Sử dụng và Đóng socket một cách tường minh.
- Sử dụng theo mô hình Client/Server
- Hai kiểu dịch vụ ứng dụng sử dụng socket API:
  - Truyền không tin cậy
  - Tin cậy, hướng nối, đúng thứ tự.

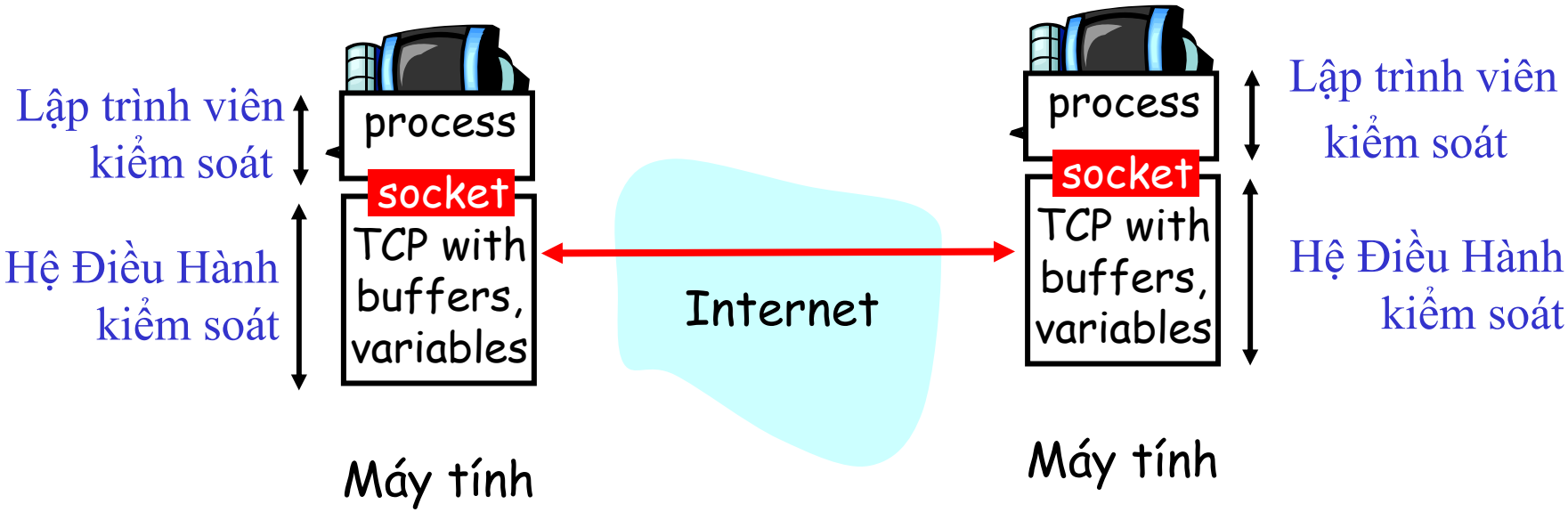
socket

Là giao diện *nằm trên máy tính,* do *ứng dụng tạo ra và quản lý,* nhưng *HĐH kiểm soát* (là "cửa") thông qua đó tiến trình vừa gửi và nhận thông điệp từ các tiến trình ứng dụng khác (ở trên máy tính khác)

Lập trình Socket TCP

Socket: Là "cửa" giữa tiến trình ứng dụng và giao thức giao vận đầu cuối (UCP/TCP)

TCP: Dịch vụ truyền byte tin cậy từ tiến trình này sang tiến trình khác.



Lập trình TCP Socket

Client phải liên lạc với server

- Tiến trình trên server phải chạy trước.
- Server phải tạo sẵn socket (door) để tiếp nhận yêu cầu từ client.

Client trao đổi với server bằng cách:

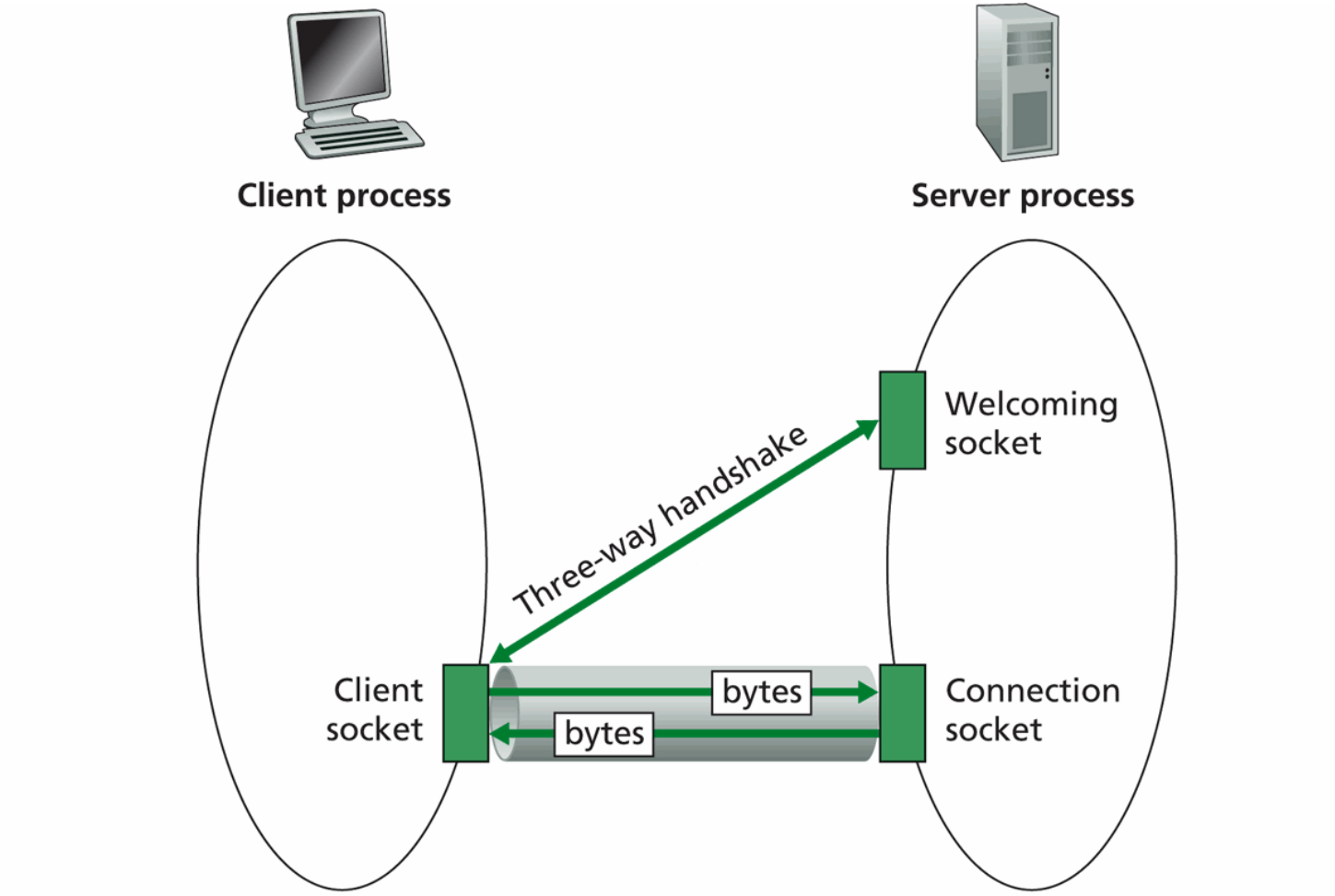
- Tạo socket TCP ở phía client
- Xác định địa chỉ IP, số hiệu cổng của tiến trình server.

- Khi client tạo socket: client TCP thiết lập kết nối tới server TCP.
- Khi nhận được yêu cầu từ client, server TCP tạo socket mới cho tiến trình trên server trao đổi dữ liệu với client
  - Cho phép server có thể đáp ứng yêu cầu của nhiều client.

Quan điểm Lập trình Ứng dụng

TCP cung cấp dịch vụ truyền dữ liệu tin cậy theo byte giữa Client và Server

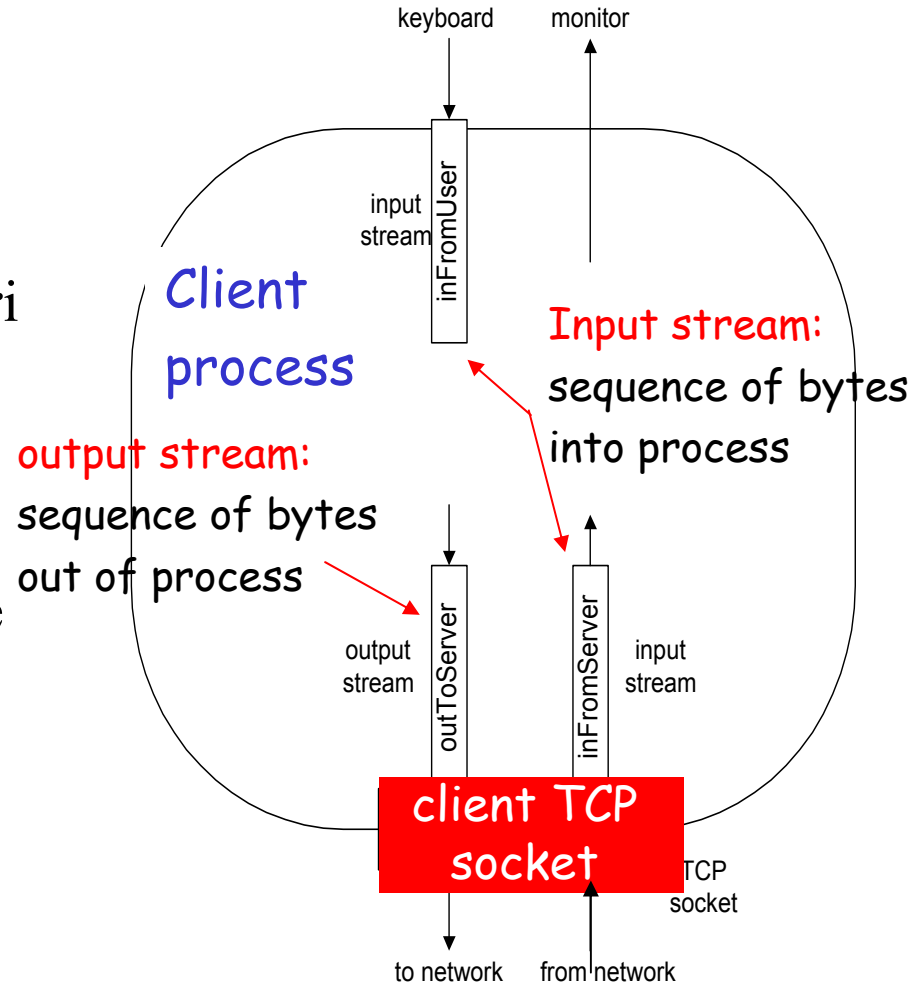
Hướng nối TCP



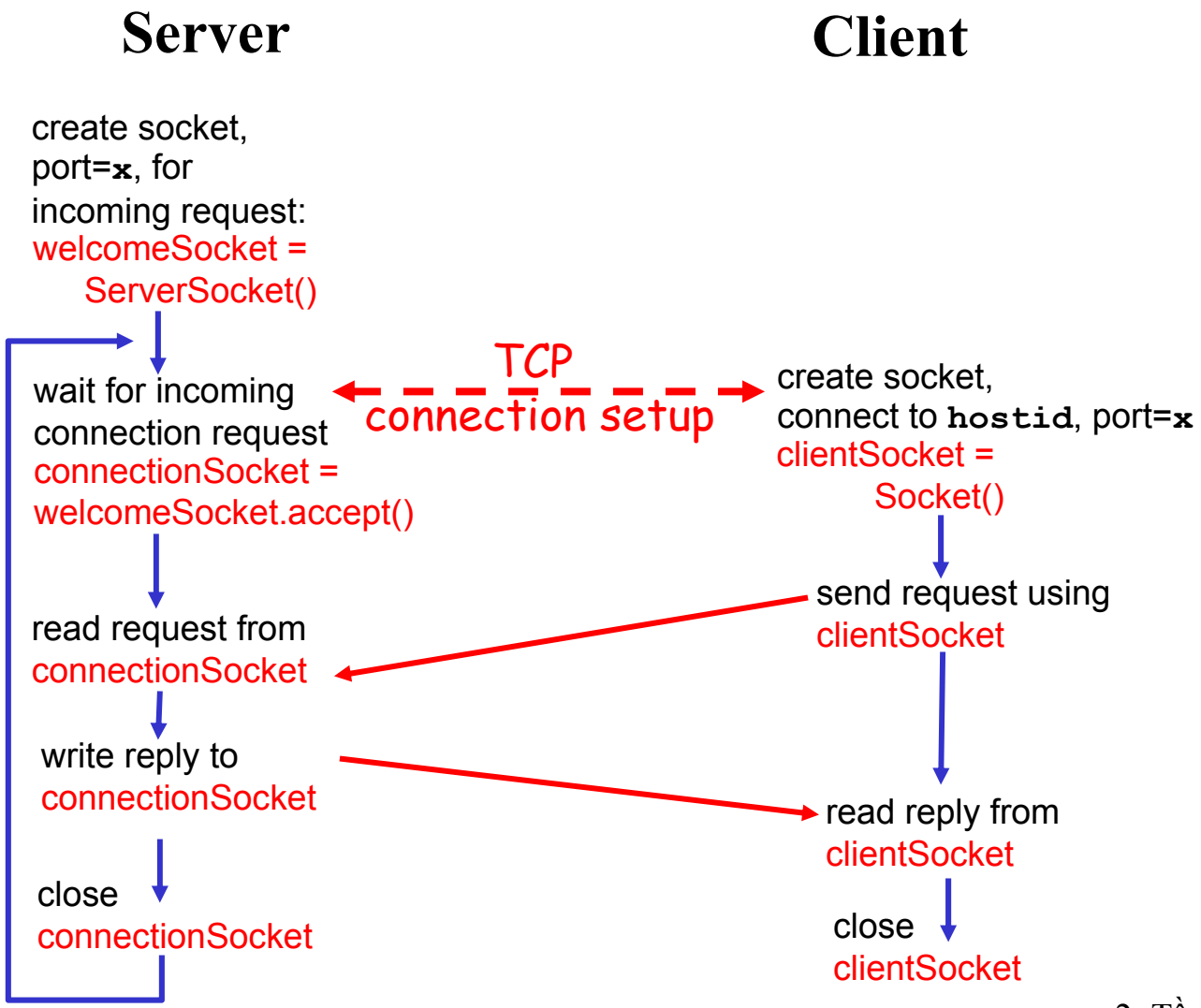
Lập trình Socket TCP

Ví dụ Ứng dụng client-server:

- Client đọc một dòng kí tự từ input chuẩn (**inFromUser** stream), gửi tới server qua socket (**outToServer**)
- server đọc dòng kí tự từ socket
- server biến đổi dòng ký tự đó (chữ thường thành chữ hoa) và gửi trả về cho client.
- client đọc dòng ký tự đã biến đổi từ socket, in ra (**inFromServer**)



Tương tác Socket Client/server : TCP



ServerSocket

- **ServerSocket**()
  - Tạo ra một socket lắng nghe kết nối từ client
- **ServerSocket(int port)**
  - Tạo ra một socket lắng nghe kết nối từ client tại cổng Port
- **ServerSocket(int port, int backlog)**
- **ServerSocket(int port, int backlog, InetAddress bindAddr)**

- **bind(SocketAddress endpoint)**
- **bind(SocketAddress endpoint, int backlog)**
- **Socket accept()**
- **close()**

Socket

- Socket(InetAddress address, int port)
- Socket(InetAddress address, int port, InetAddress localAddr, int localPort)
- Socket(String host, int port)
- bind(SocketAddress bindpoint)
- connect(SocketAddress endpoint)
- connect(SocketAddress endpoint, int timeout)
- InputStream get InputStream()
- OutputStream getOutputStream()
- close()

Ví dụ Java Client (TCP)

```
import java.io.*;
import java.net.*;
class TCPClient {

    public static void main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;

        BufferedReader inFromUser =
          new BufferedReader(new InputStreamReader(System.in));

        Socket clientSocket = new Socket ("hostname", 6789);

        DataOutputStream outToServer =
          new DataOutputStream(clientSocket.getOutputStream());
```

Tạo input stream

Tạo client socket, kết nối tới server

Tạo output stream, đính kèm vào socket

Ví dụ về Java Client (TCP)

Tạo input stream, đính kèm vào socket

```
BufferedReader inFromServer =
  new BufferedReader(new
  InputStreamReader(clientSocket.getInputStream()));

sentence = inFromUser.readLine();

outToServer.writeBytes(sentence + '\n');

modifiedSentence = inFromServer.readLine();

System.out.println("FROM SERVER: " + modifiedSentence);

clientSocket.close();

    }
}
```

Gửi dòng kí tự đến server

Đọc dòng kí tự (đã biến đổi) do server gửi về

Ví dụ về Java Server (TCP)

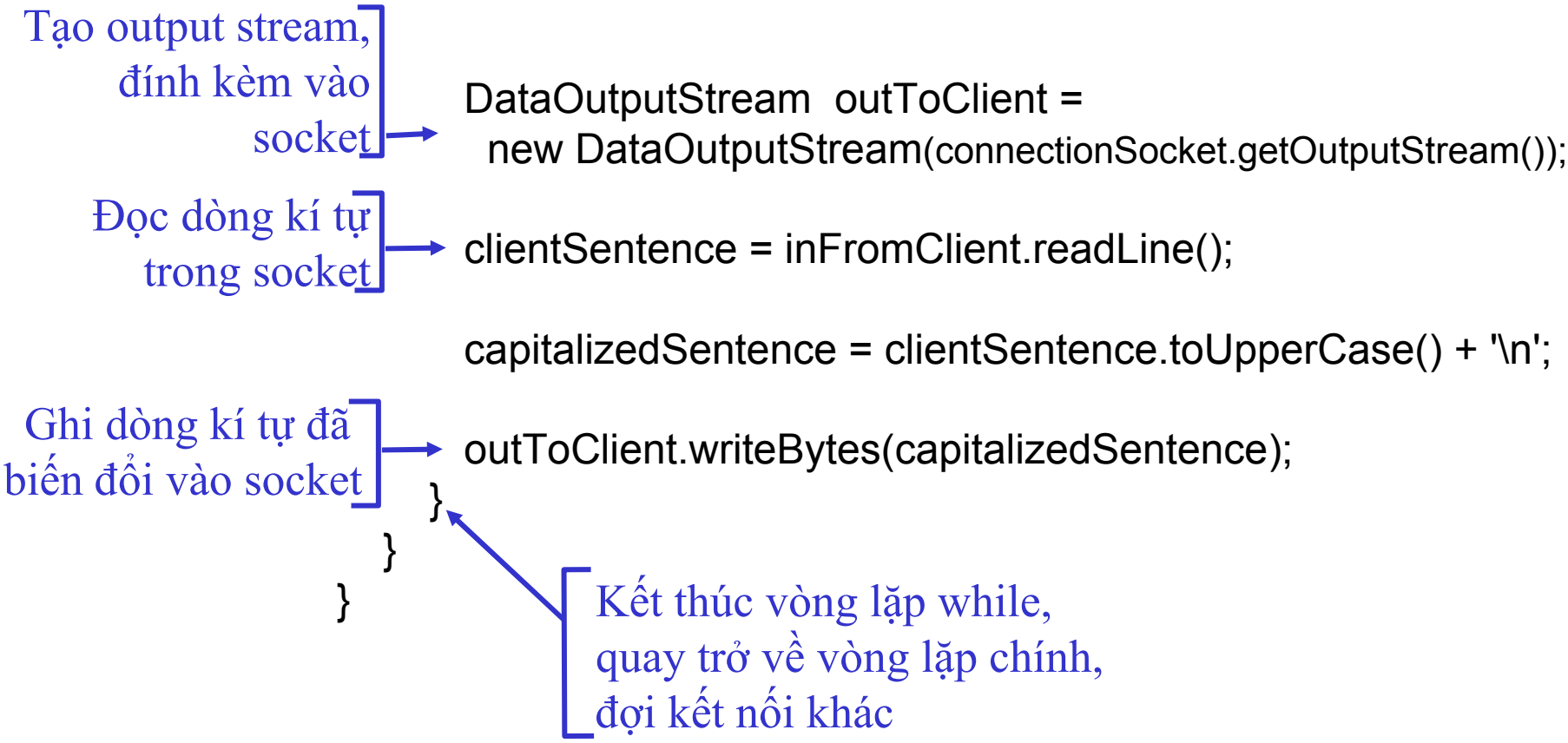
```
import java.io.*;
import java.net.*;

class TCPServer {

  public static void main(String argv[]) throws Exception
   {
     String clientSentence;
     String capitalizedSentence;

     ServerSocket welcomeSocket = new ServerSocket(6789);

     while(true) {

        Socket connectionSocket = welcomeSocket.accept();

        BufferedReader inFromClient =
          new BufferedReader(new
          InputStreamReader(connectionSocket.getInputStream()));
```

Tạo Socket để đợi ở cổng 6789

Đợi đến khi có socket từ client gửi đến

Tạo input stream, đính kèm vào socket

Ví dụ về Java Server (TCP)

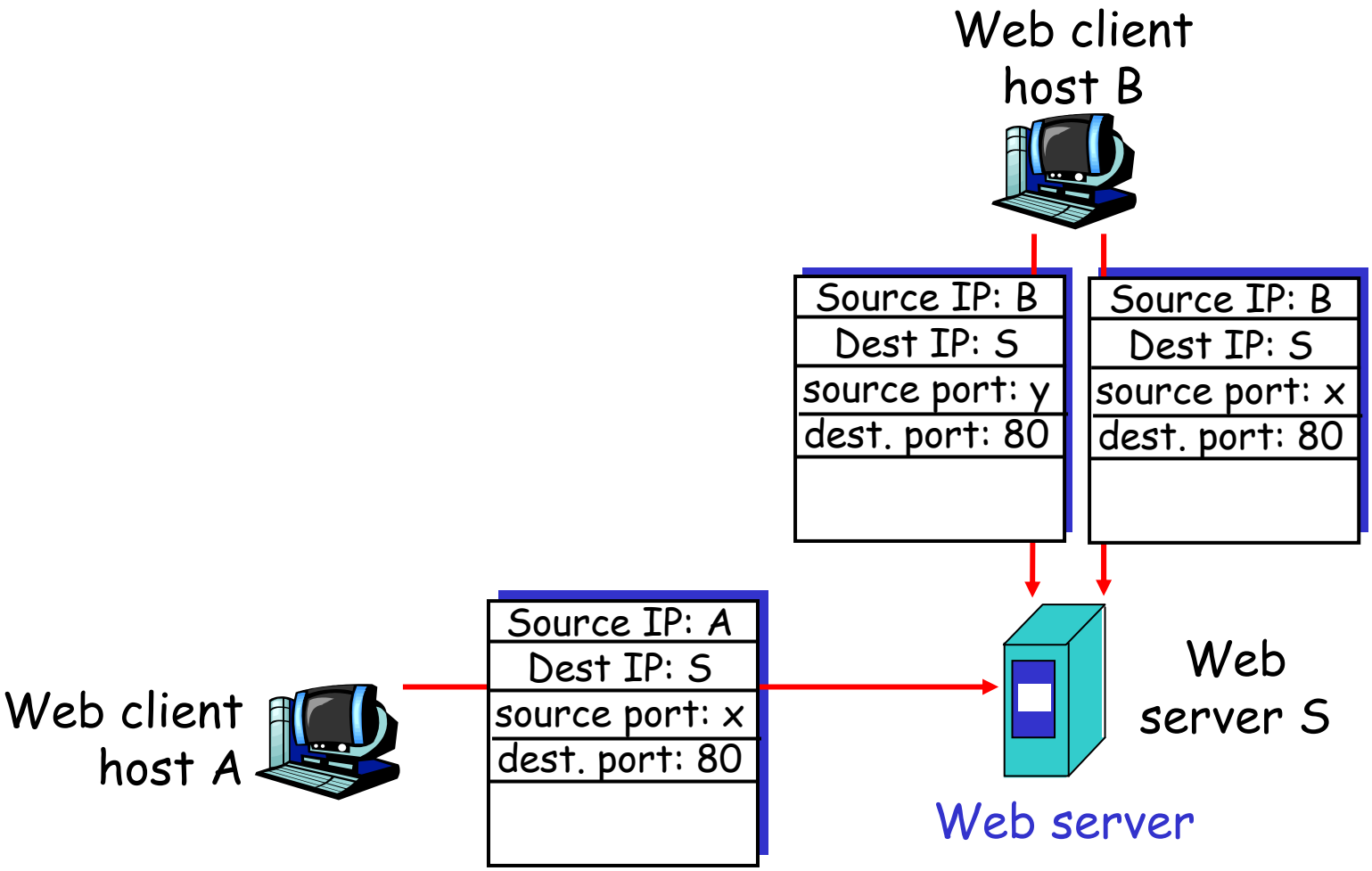


Phân kênh trong Hướng nối

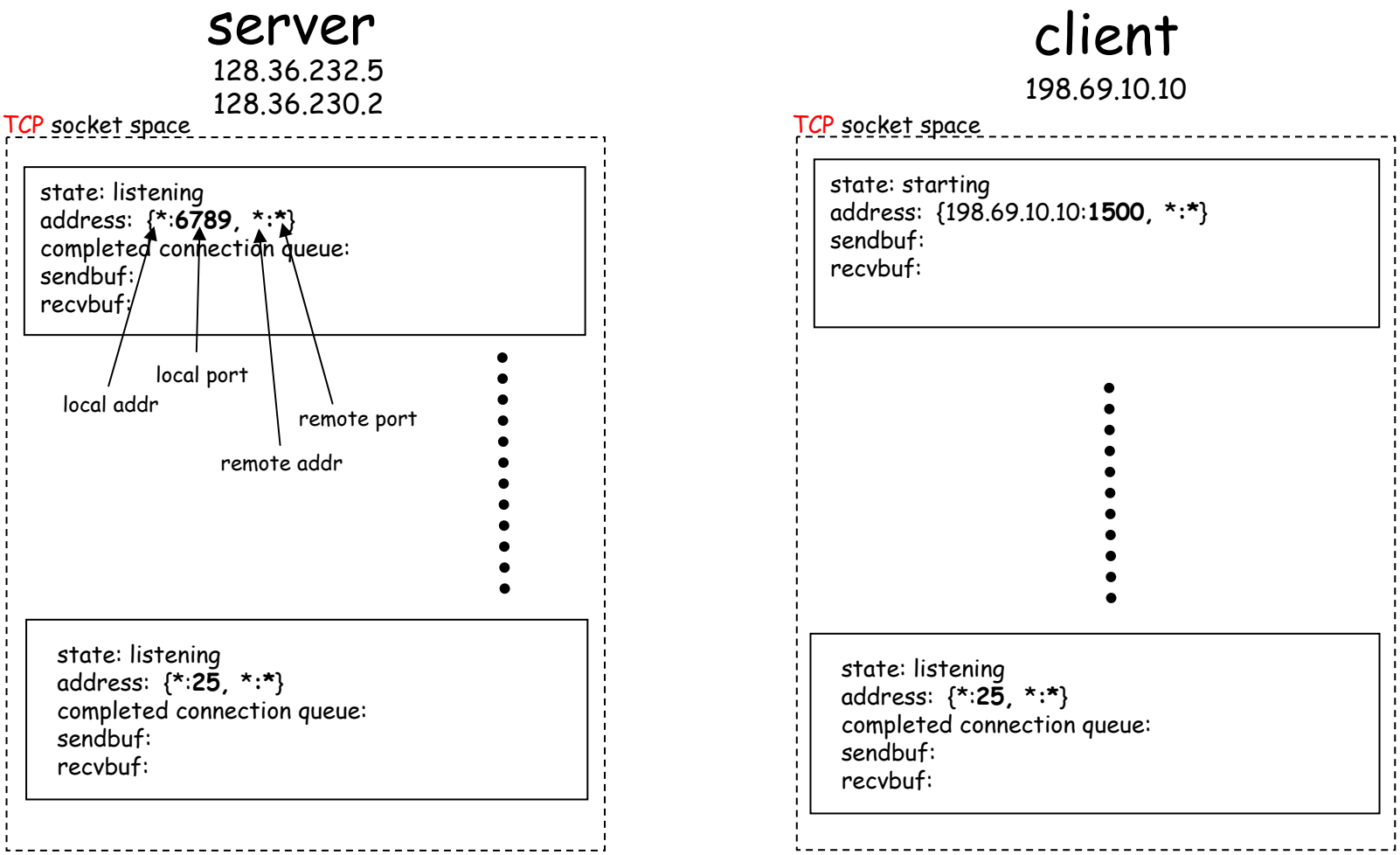
- TCP socket được xác định bởi 4 thành phần:
  - Địa chỉ IP gửi
  - Cổng tiến trình gửi
  - Địa chỉ IP nhận
  - Cổng tiến trình nhận

- Máy tính phía nhận sẽ sử dụng cả 4 thành phần này để chuyển segment đến socket phù hợp
  - Server có thể đồng thời hỗ trợ nhiều socket. Các kết nối khác nhau được tự động chuyển cho các socket tương ứng

Hướng nối : Phân kênh (tiếp)

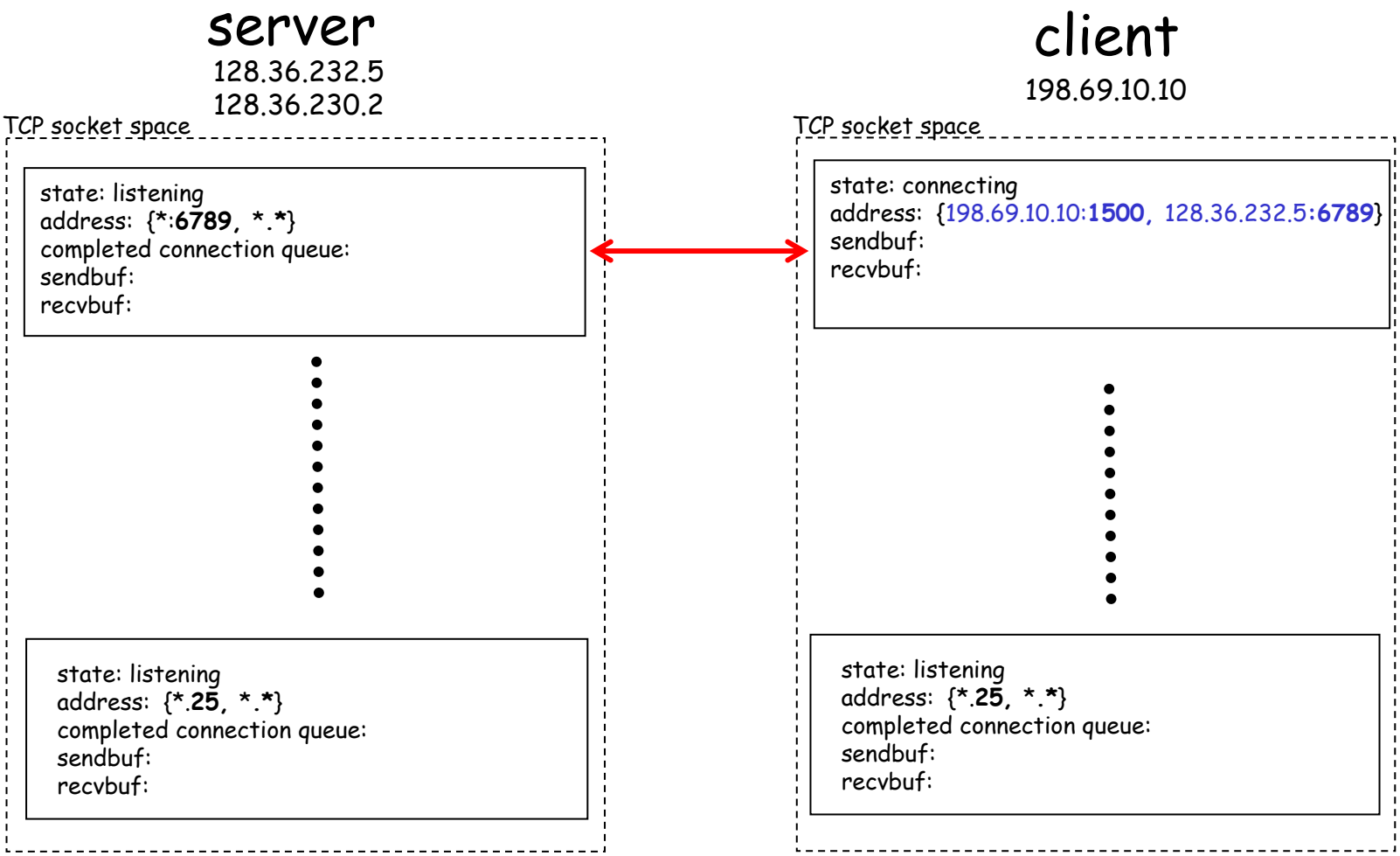


TCP sẽ làm như thế nào

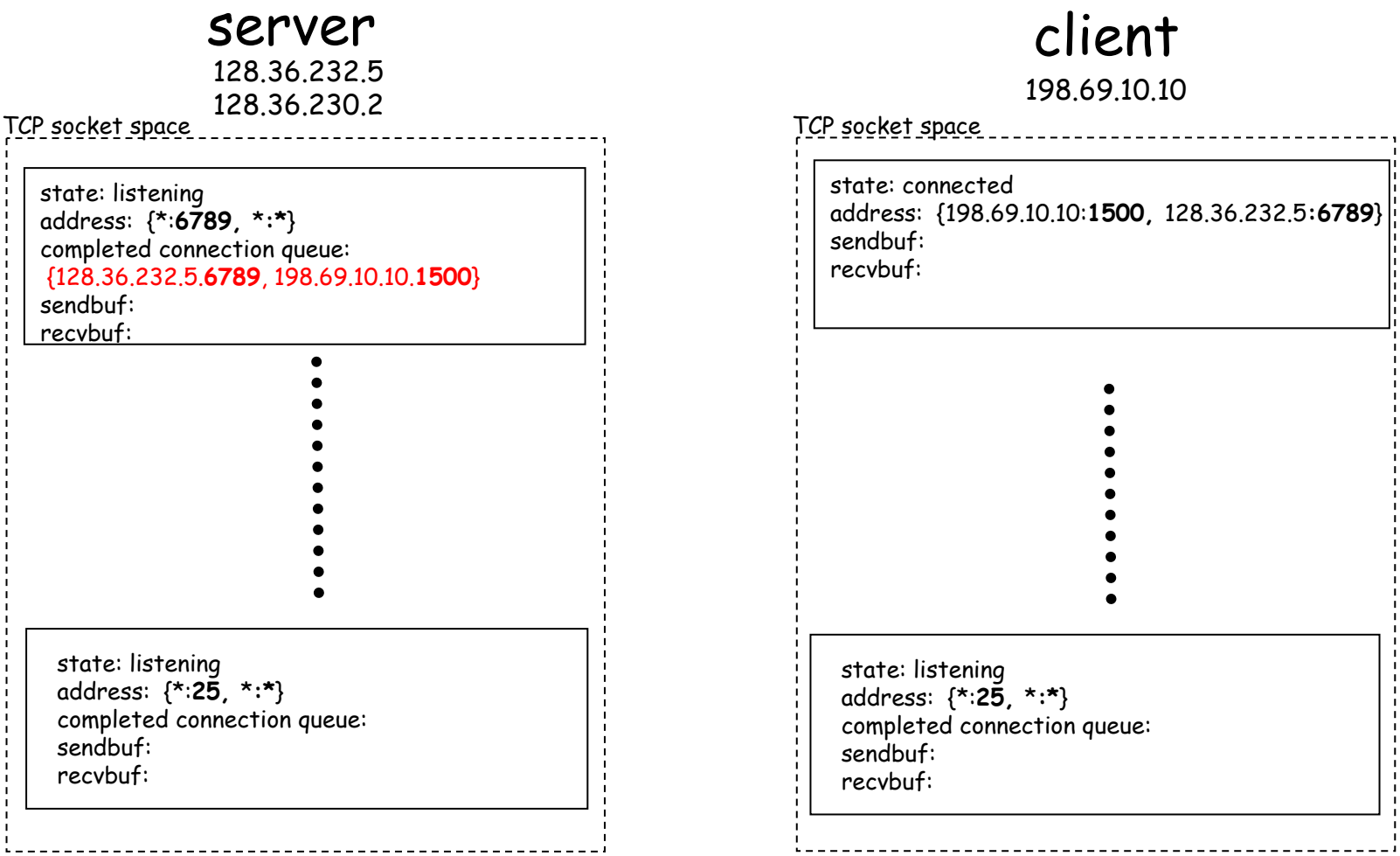


%netstat --tcp -a -l -n

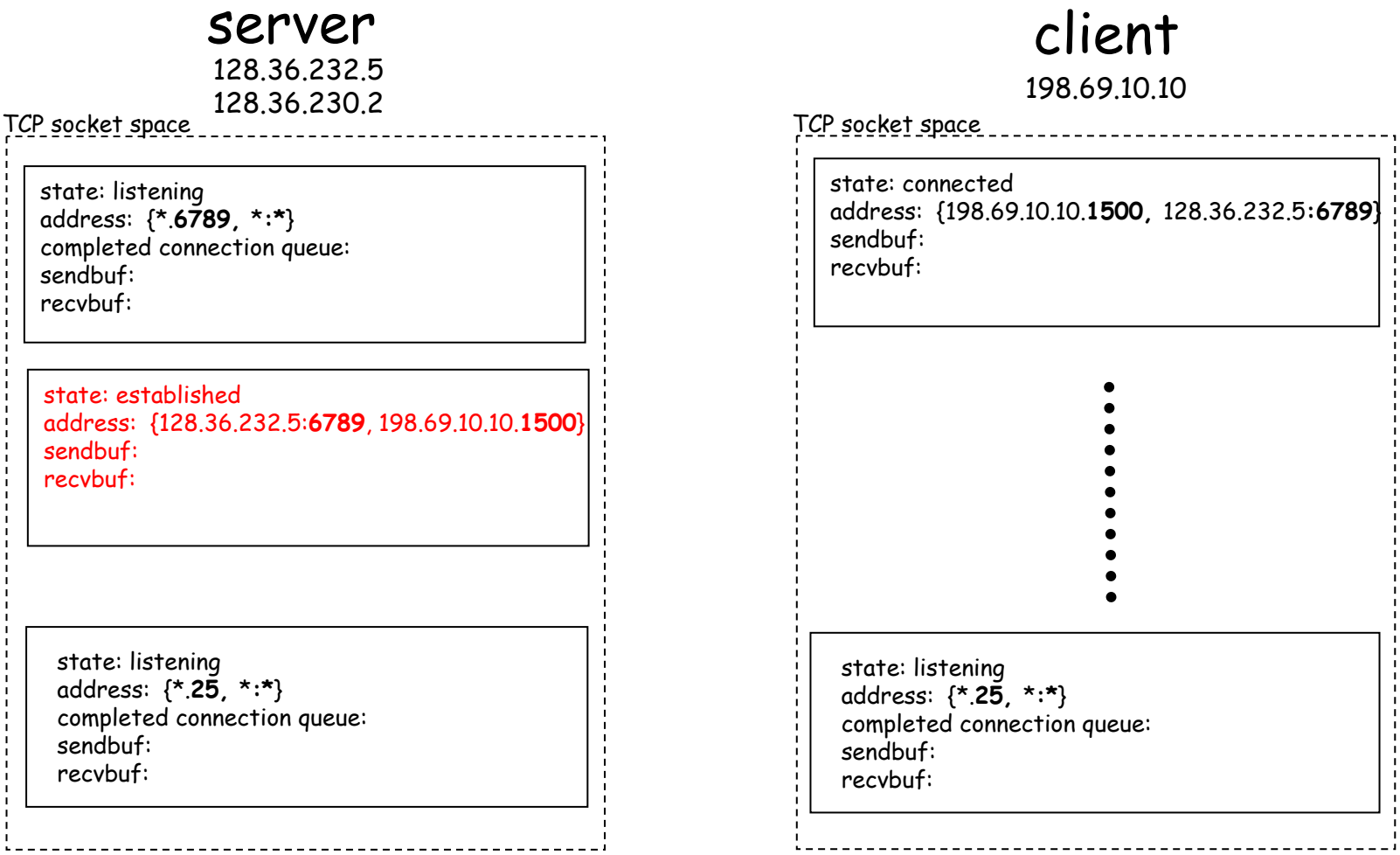
Ví du : Client khởi tạo Kết nối



Kết nối TCP thành công



Ví dụ Server chấp nhận kết nối accept()



Phân kênh cho gói tin dựa trên (IP gửi, port gửi, IP nhận, port nhận)

Packet được gửi cho socket phù hợp nhất!

Lập trình Socket UDP

UDP: không thiết lập kết nối giữa client và server

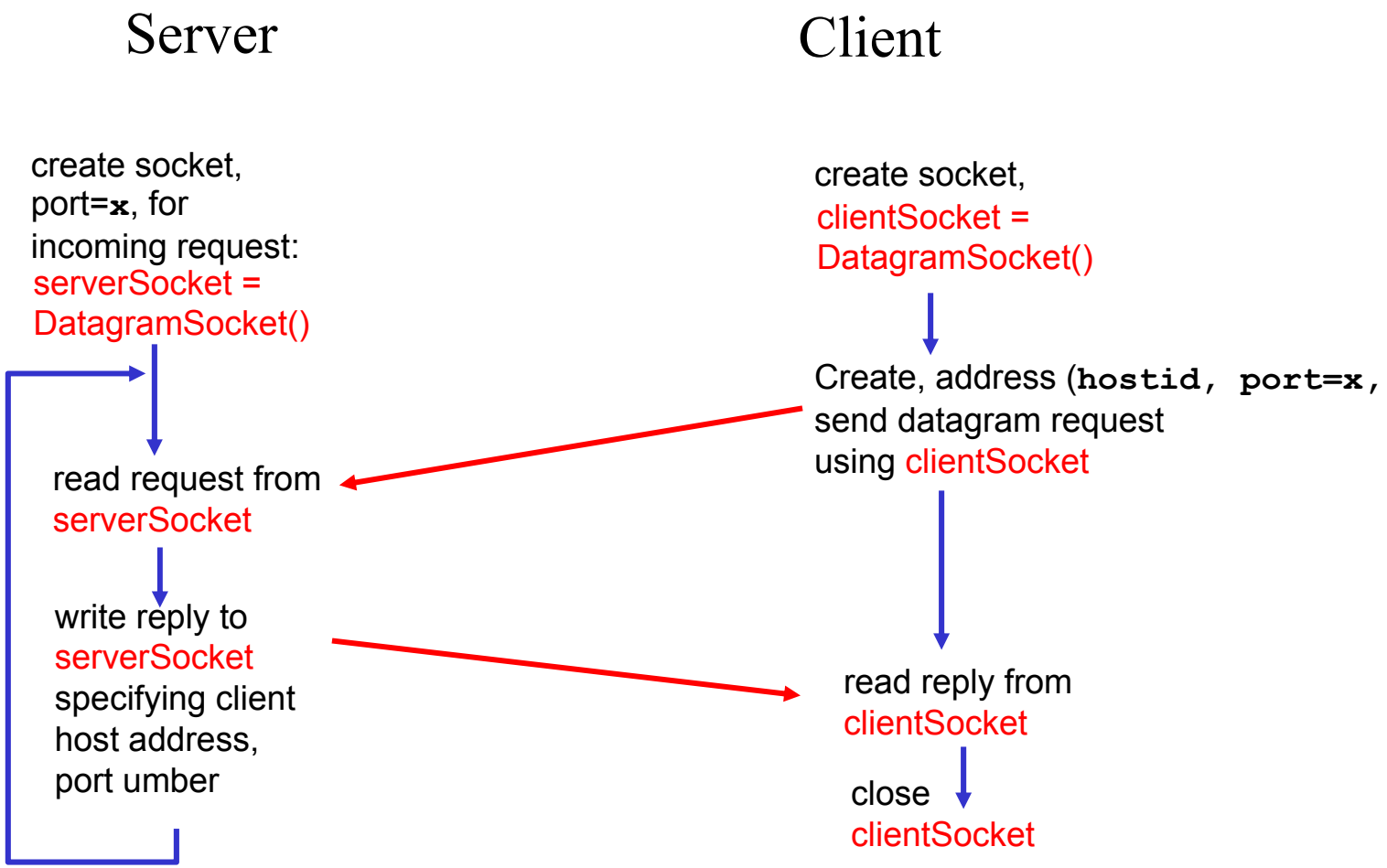
- Không "bắt tay".
- Bên gửi phải xác định chính xác địa chỉ IP và cổng của tiến trình nhận.
- Server xác định địa chỉ IP và cổng của bên gửi từ UDP datagram nhận được.

UDP : dữ liệu truyền đi có thể đến đích không theo đúng thứ tự hay mất mát.

Với Lập trình Ứng dụng

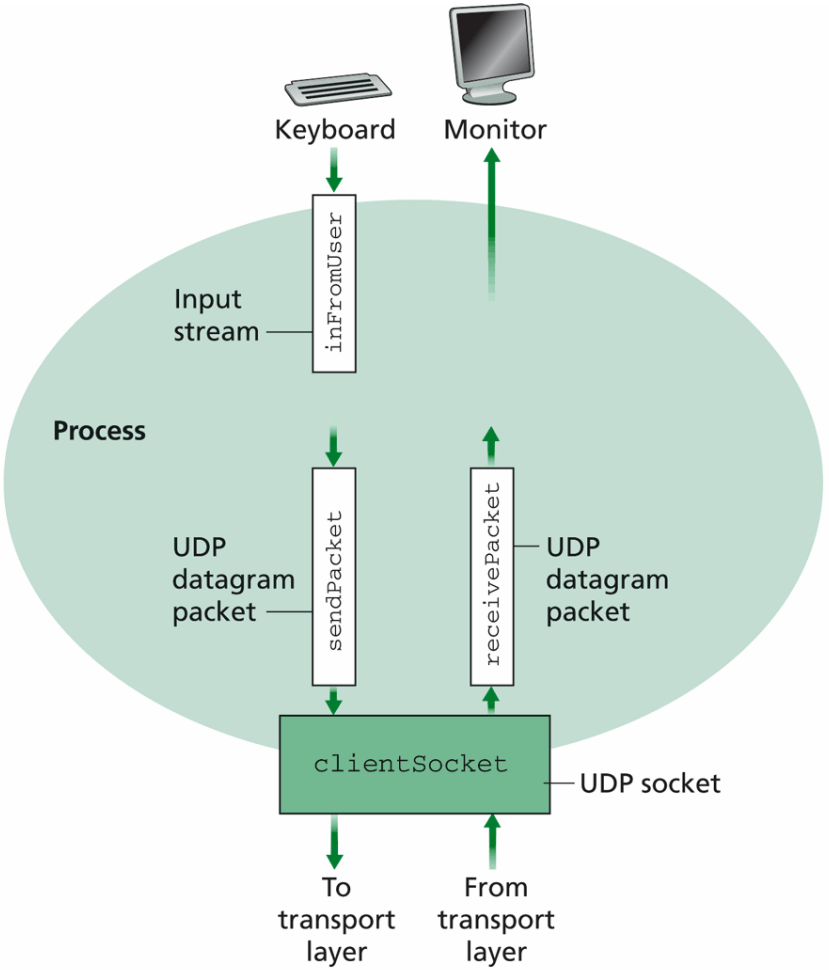
UDP cung cấp dịch vụ truyền dữ liệu theo từng nhóm byte (datagram) không tin cậy

Tương tác Socket Client/Server : UDP



Example: UDPClient.java

- UDP client đơn giản đọc input từ bàn phím, gửi qua server và đợi server trả kết quả về.



Ví dụ Java Client (UDP)

```
import java.io.*;
import java.net.*;

class UDPClient {
    public static void main(String args[]) throws Exception
    {

      BufferedReader inFromUser =
        new BufferedReader(new InputStreamReader(System.in));

      DatagramSocket clientSocket = new DatagramSocket();

      InetAddress IPAddress = InetAddress.getByName("hostname");

      byte[] sendData = new byte[1024];
      byte[] receiveData = new byte[1024];

      String sentence = inFromUser.readLine();

      sendData = sentence.getBytes();
```

Tạo input stream

Tạo client socket

Chuyển hostname sang địa chỉ IP sử dụng DNS

Ví dụ Java Client (UDP)

Tạo datagram cùng với dữ liệu,độ dài, địa chỉ IP, cổng

Gửi datagram tới server

Đọc datagram gửi về từ server

```
                DatagramPacket sendPacket =
                  new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

                clientSocket.send(sendPacket);

                DatagramPacket receivePacket =
                   new DatagramPacket(receiveData, receiveData.length);

                clientSocket.receive(receivePacket);

                String modifiedSentence =
                    new String(receivePacket.getData());

                System.out.println("FROM SERVER:" + modifiedSentence);
                clientSocket.close();
                }
            }
```

Ví dụ Java Server (UDP)

```
import java.io.*;
import java.net.*;

class UDPServer {
  public static void main(String args[]) throws Exception
    {
      DatagramSocket serverSocket = new DatagramSocket(9876);

      byte[] receiveData = new byte[1024];
      byte[] sendData  = new byte[1024];

      while(true)
        {
          DatagramPacket receivePacket =
            new DatagramPacket(receiveData, receiveData.length);
          serverSocket.receive(receivePacket);
```

Tạo datagram socket ở cổng 9876

Tạo datagram

Nhận datagram

Ví dụ về Java Server (UDP)

```
String sentence = new String(receivePacket.getData());

InetAddress IPAddress = receivePacket.getAddress();

int port = receivePacket.getPort();

                String capitalizedSentence = sentence.toUpperCase();

sendData = capitalizedSentence.getBytes();

DatagramPacket sendPacket =
   new DatagramPacket(sendData, sendData.length, IPAddress,
                      port);

serverSocket.send(sendPacket);
        }
    }
}
```

Nhận địa chỉ IP và cổng của bên Gửi

Tạo datagram để gửi tới client

Đính datagram vào socket

Kết thúc vòng lặp while, Quay trở về vòng lặp chính, đợi datagram khác đến

Lập trình Socket : Tham khảo

C-language tutorial (audio/slides):

- "Unix Network Programming" (J. Kurose),

HTTP://manic.cs.umass.edu/~amldemo/courseware/intro.

Java-tutorials:

- "All About Sockets" (Sun tutorial), HTTP://www.javaworld.com/javaworld/jw-12-1996/jw-12-sockets.html
- "Socket Programming in Java: a tutorial," HTTP://www.javaworld.com/javaworld/jw-12-1996/jw-12-sockets.html

Phần 2 : TỔNG KẾT

Đã học xong tầng ỨNG DỤNG !

- Các yêu cầu của Dịch vụ tầng Ứng dụng:
  - Sự tin cậy, Băng thông, Độ trễ.
- Mô hình Client - Server.
- Các mô hình dịch vụ của tầng giao vận trên Internet
  - Hướng kết nối, tin cậy: TCP
  - Không tin cậy, datagram: UDP

- **Các Giao thức đặc trưng:**
  - HTTP
  - FTP
  - SMTP, POP3
  - DNS

- **Lập trình Socket**
  - Giao tiếp client/server
  - Sử dụng TCP, UDP socket

Phần 2 : TỔNG KẾT

Trong chương này, chú trọng về GIAO THỨC !

- **Chủ yếu trao đổi thông điệp Request/Reply:**
  - Client yêu cầu thông tin hoặc dịch vụ.
  - Server trả lại dữ liệu cùng mã trạng thái.
- **Định dạng Thông điệp:**
  - Tiêu đề : các trường lưu giữ thông tin về dữ liệu.
  - Dữ liệu: thông tin được trao đổi.

Nhiều Dịch vụ trái ngược :

- Thông điệp mang điều khiển vs. thông điệp mang dữ liệu.
  - in-based, out-of-band
- Tập trung và Phân tán.
- Không trạng thái và Lưu trạng thái.
- Truyền thông điệp Tin cậy và Không tin cậy.
- An ninh : Kiểm chứng.

Chương 3: NGÔN NGỮ ĐỊNH NGHĨA VÀ THAO TÁC DL

I. ĐẠI SỐ QUAN HỆ

- Phép chọn (Selection)
- Phép chiếu (Projection)
- Tích Đề các (Cartesian product)
- Phép kết nối (Join)
- Phép chia (Division)
- Phép hợp (Union)
- Phép giao (Intersect)
- Phép trừ (Difference)

II. ĐẠI SỐ HỆ VÀ CÁC TÍNH CHẤT CỦA ĐẠI SỐ QUAN HỆ

III. NGÔN NGỮ SQL

- Ngôn ngữ định nghĩa dư liệu
- Truy vấn dữ liệu
- Các phép toán tập hợp (Set Operations)
- Giá trị NULL và các hàm tổng hợp của SQL
- Các hàm tổng hợp với việc nhóm dữ liệu
- Truy vấn lồng (Nested Subqueries)
- Khung nhìn (Views)
- Các lệnh cập nhật dữ liệu

IV. NGÔN NGỮ QBE

I. ĐẠI SỐ QUAN HỆ

1.1. Phép chọn (Selection)

Cho quan hệ r định nghĩa trên lược đồ quan hệ R(U), E là biểu thức chọn phát biểu trên U. Phép chọn trên quan hệ r theo điều kiện E, ký hiệu $\sigma_E(r)$ cho ta một quan hệ mới với tập thuộc tính U và các bộ là các bộ của r thoả mãn điều kiện E.

Ta viết: $\sigma_E(r) = \{t \mid t \in r \text{ và } E(t)=\text{đúng}\}$

Trong đó E(t) là giá trị của biểu thức E khi thay mọi thuộc tính Ai trong E của t bởi t[Ai].

I. ĐẠI SỐ QUAN HỆ

1.1. Phép chọn (tiếp)

Các phép toán trong E gồm:

STT	Phép toán	Ý nghĩa	Mức ưu tiên
1	>	Lớn hơn	2
2	>=, ≥	Lớn hơn hoặc bằng	2
3	<	Bé hơn	2
4	<=,≤	Bé hơn hoặc bằng	2
5	<>, ≠	Khác (không bằng)	2
6	=	Bằng	2
7	¬	Phủ định (NOT)	1
8	∧	Phép hội (AND)	3
9	∨	Phép tuyển (OR)	4

I. ĐẠI SỐ QUAN HỆ

1.1. Phép chọn (tiếp)

Ví dụ 1: Cho quan hệ KHACHHANG như sau:

MaKH	TenKH	DiaChiKH	DThoaiKH
KH01	Lê Thanh Tâm	Hà Nội	04.5557779
KH02	Nguyễn Quốc Trọng	Hà Nội	04.5557778
KH03	Lê Thị Mơ	Thanh Hoá	037.666888
KH04	Nguyễn Thanh Nam	Nghệ An	038.444555
KH05	Phan Quốc Anh	Hà Nội	04.3334445
KH06	Đỗ Văn Phúc	Thanh Hoá	037.666889
KH07	Lưu Trọng Phụng	Nghệ An	038.444556
KH08	Lê Minh Hương	Hà Tĩnh	039.444888
KH09	Lê Thanh Tâm	Hà Tĩnh	039.777899

a. Đưa ra danh sách khách hàng tên 'Lê Thanh Tâm' có địa chỉ tại 'Hà Nội'?

☞ $\sigma_{(TenKH='Lê\ Thanh\ Tâm')\ \wedge\ (DiaChiKH='Hà\ Nội')}(KHACHHANG)$

I. ĐẠI SỐ QUAN HỆ

1.1. Phép chọn (tiếp)

Ví dụ 1: Cho quan hệ KHACHHANG như sau:

MaKH	TenKH	DiaChiKH	DThoaiKH
KH01	Lê Thanh Tâm	Hà Nội	04.5557779
KH02	Nguyễn Quốc Trọng	Hà Nội	04.5557778
KH03	Lê Thị Mơ	Thanh Hoá	037.666888
KH04	Nguyễn Thanh Nam	Nghệ An	038.444555
KH05	Phan Quốc Anh	Hà Nội	04.3334445
KH06	Đỗ Văn Phúc	Thanh Hoá	037.666889
KH07	Lưu Trọng Phụng	Nghệ An	038.444556
KH08	Lê Minh Hương	Hà Tĩnh	039.444888
KH09	Lê Thanh Tâm	Hà Tĩnh	039.777899

b. Đưa ra danh sách khách hàng có địa chỉ tại 'Hà Nội' hoặc 'Nghệ An'?

☞ $\sigma_{(DiaChiKH='Hà\ Nội') \vee (DiaChiKH='Nghệ\ An')}(KHACHHANG)$

I. ĐẠI SỐ QUAN HỆ

1.2. Phép chiếu (projection)

Cho quan hệ r định nghĩa trên lược đồ quan hệ R(U) với U={A1, ..., An} và tập thuộc tính $X \subseteq U$. Phép chiếu quan hệ r lên tập thuộc tính X, ký hiệu $\Pi_X(r)$ cho ta một quan hệ mới với tập thuộc tính X và các bộ là hạn chế trên X của các bộ $t \in r$.

Ta viết $\Pi_X(r) = \{t[X] \mid t \in r\}$.

I. ĐẠI SỐ QUAN HỆ

1.2. Phép chiếu (tiếp)

Ví dụ 2:

Cho quan hệ KHACHHANG như sau:

$\Pi_{\{MaKH,\ TenKH\}}(KHACHHANG)$

MaKH	TenKH	DiaChiKH	DThoaiKH
KH01	Lê Thanh Tâm	Hà Nội	04.5557779
KH02	Nguyễn Quốc Trọng	Hà Nội	04.5557778
KH03	Lê Thị Mơ	Thanh Hoá	037.666888
KH04	Nguyễn Thanh Nam	Nghệ An	038.444555
KH05	Phan Quốc Anh	Hà Nội	04.3334445
KH06	Đỗ Văn Phúc	Thanh Hoá	037.666889
KH07	Lưu Trọng Phụng	Nghệ An	038.444556
KH08	Lê Minh Hương	Hà Tĩnh	039.444888
KH09	Lê Thanh Tâm	Hà Tĩnh	039.777899

a. Đưa ra mã, tên khách hàng ?

☞ $\Pi_{\{MaKH,\ TenKH\}}(KHACHHANG)$

I. ĐẠI SỐ QUAN HỆ

1.2. Phép chiếu (tiếp)

Ví dụ 2:

Cho quan hệ KHACHHANG như sau:

MaKH	TenKH	DiaChiKH	DThoaiKH
KH01	Lê Thanh Tâm	Hà Nội	04.5557779
KH02	Nguyễn Quốc Trọng	Hà Nội	04.5557778
KH03	Lê Thị Mơ	Thanh Hoá	037.666888
KH04	Nguyễn Thanh Nam	Nghệ An	038.444555
KH05	Phan Quốc Anh	Hà Nội	04.3334445
KH06	Đỗ Văn Phúc	Thanh Hoá	037.666889
KH07	Lưu Trọng Phụng	Nghệ An	038.444556
KH08	Lê Minh Hương	Hà Tĩnh	039.444888
KH09	Lê Thanh Tâm	Hà Tĩnh	039.777899

$\Pi_{\{DiaChiKH\}}(KHACHHANG)$

DiaChiKH
Hà Nội
Hà Nội
Thanh Hoá
Nghệ An
Hà Nội
Thanh Hoá
Nghệ An
Hà Tĩnh
Hà Tĩnh

b. Đưa ra địa chỉ khách hàng ?

☞ $\Pi_{\{DiaChiKH\}}(KHACHHANG)$

DiaChiKH
Hà Nội
Hà Tĩnh
Nghệ An
Thanh Hoá

I. ĐẠI SỐ QUAN HỆ

Ví dụ 3:

Cho quan hệ KHACHHANG như sau:

MaKH	TenKH	DiaChiKH	DThoaiKH
KH01	Lê Thanh Tâm	Hà Nội	04.5557779
KH02	Nguyễn Quốc Trọng	Hà Nội	04.5557778
KH03	Lê Thị Mơ	Thanh Hoá	037.666888
KH04	Nguyễn Thanh Nam	Nghệ An	038.444555
KH05	Phan Quốc Anh	Hà Nội	04.3334445
KH06	Đỗ Văn Phúc	Thanh Hoá	037.666889
KH07	Lưu Trọng Phụng	Nghệ An	038.444556
KH08	Lê Minh Hương	Hà Tĩnh	039.444888
KH09	Lê Thanh Tâm	Hà Tĩnh	039.777899

Đưa ra tên khách hàng có địa chỉ ở 'Hà Nội'?

☞ $\Pi_{\{TenKH\}}(\sigma_{DiachiKH='Hà\ Nội'}(KHACHHANG))$

1 $\sigma_{DiachiKH='Hà\ Nội'}(KHACHHANG)$ ⇨ 2 $\Pi_{\{TenKH\}}(\sigma_{DiachiKH='Hà\ Nội'}(KHACHHANG))$

MaKH	TenKH	DiaChiKH	DThoaiKH
KH01	Lê Thanh Tâm	Hà Nội	04.5557779
KH02	Nguyễn Quốc Trọng	Hà Nội	04.5557778
KH05	Phan Quốc Anh	Hà Nội	04.3334445

TenKH
Lê Thanh Tâm
Nguyễn Quốc Trọng
Phan Quốc Anh

I. ĐẠI SỐ QUAN HỆ

1.3. Tích Đề các (Cartesian product)

Cho hai quan hệ r định nghĩa trên lược đồ quan hệ **R(U)** và s định nghĩa trên lược đồ quan hệ **S(V)** , với U= {A1,A2, ..., An} , V= {B1,B2, ..., Bm}. Tích Đề các của r và s, ký hiệu **r×s** cho ta một quan hệ mới với tập thuộc tính $U \cup V$ và các bộ có dạng:

t= (a1,a2,..., an,b1, b2, ..., bm)

trong đó (a1 , ..., an) $\in$ r và (b1,b2, ..., bm) $\in$ s.

Ta viết:

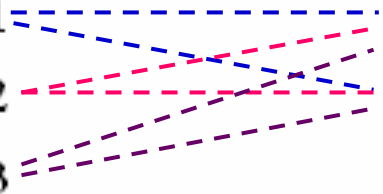
$r \times s=\{t=(a1,a2...,an ,b1,b2, ...,bm) \mid (a1,a2, ..., an) \in r$ và $(b1,b2, ..., bm) \in s\}$.

I. ĐẠI SỐ QUAN HỆ

1.3. Tích Đề các (tiếp)

Ví dụ 4:

r(A	B	C	D)	s(E	K	H)
a1	b1	1	d1	1	k1	h1
a2	b2	5	d2	2	k2	h2
a3	b3	3	d3			



r × s (A B C D E K H)

I. ĐẠI SỐ QUAN HỆ

1.4. Phép kết nối (Join)

a, Phép kết nối θ

Cho hai quan hệ r định nghĩa trên lược đồ quan hệ R(U) và s định nghĩa trên lược đồ quan hệ S(V) với U={A1, ..., An}, V ={B1, ..., Bm}. Ai và Bj là các thuộc tính tương ứng thuộc U và V sao cho Dom(Ai)=Dom(Bj). Gọi θ là một trong các phép toán {=, >, >=, <, <=, $\neq$ }.

Phép kết nối quan hệ r với s theo biểu thức Ai θ Bj , ký hiệu $r \bowtie_{A_i \theta B_j} s$ cho ta một quan hệ mới với tập thuộc tính $U \cup V$ và các bộ được xác định bởi

{(u,v) | u=(a1,a2, ..., an) $\in$ r, v=(b1,b2, ..., bm)$\in$s và u[Ai] θ v[Bj]} = đúng}.

Ta viết:

$r \bowtie_{A_i \theta B_j} s$ = {(u,v) |u=(a1,a2, ..., an)$\in$r, v=(b1,b2, ...,bm)$\in$s và u[Ai] θ v[Bj]}=đúng}.

I. ĐẠI SỐ QUAN HỆ

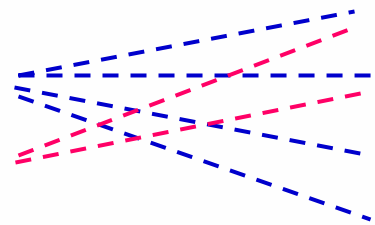
1.4. Phép kết nối (tiếp)

a, Phép kết nối θ

Ví dụ 5: Cho hai quan hệ r và s như sau:

r (A	B	C	D)
a1	b1	1	d1
a2	b2	5	d2
a3	b3	3	d3
a4	b4	1	d4

s (E	K	H)
1	k1	h1
2	k2	h2
4	k3	h3
3	k4	h4



Kết nối quan hệ r với s theo điều kiện C > E ta được:

$r \underset{C>E}{\bowtie} s$ (A B C D E K H)

I. ĐẠI SỐ QUAN HỆ

1.4. Phép kết nối (tiếp)

b, Phép kết nối tự nhiên (Natural join)

Cho hai quan hệ r định nghĩa trên lược đồ quan hệ R(U) và s định nghĩa trên lược đồ quan hệ S(V) với $U \cap V \neq \emptyset$. Phép kết nối tự nhiên giữa quan hệ r với s, ký hiệu r*s cho ta một quan hệ mới với tập thuộc tính $U \cup V$ và các bộ được xác định bởi

$$\{t \mid t[U] \in r \text{ và } t[V] \in s\}.$$

Ta viết: $r*s = \{t \mid t[U] \in r \text{ và } t[V] \in s\}.$

I. ĐẠI SỐ QUAN HỆ

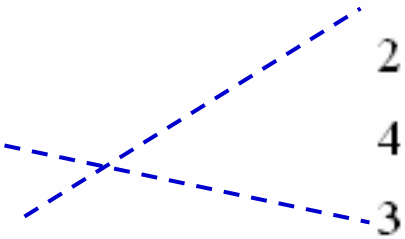
1.4. Phép kết nối (tiếp)

b, Phép kết nối tự nhiên

Ví dụ 6: Cho hai quan hệ r và s như sau:

r(A	B	C	D)
a1	b1	1	d1
a2	b2	5	d2
a3	b3	3	d3
a4	b4	1	d4

s(C	E	F)
1	e1	f1
2	e2	f2
4	e3	f3
3	e4	f4



Kết nối tự nhiên hai quan hệ r và s theo thuộc tính C ta được:

r * s (A B C D E F)

I. ĐẠI SỐ QUAN HỆ

1.4. Phép kết nối (tiếp)

Ví dụ 7:

SINHVIEN

MaSV	Hoten	QueQuan	MaKhoa
01	LE VAN AN	Nghe An	K1
02	NGUYEN AN	HT	K1
03	LE BINH	HT	K1
04	PHAM TUAN	Nghe An	K2
05	PHAN ANH	Nghe An	K2

KHOA

makhoa	tenkhoa
K1	TOAN TIN
K2	LY
K3	NGOAI NGU
K4	VAN

SINHVIEN * KHOA

MaSV	Hoten	QueQuan	MaKhoa	tenkhoa

I. ĐẠI SỐ QUAN HỆ

Ví dụ 8a:

SINHVIEN

MaSV	Hoten	QueQuan	MaKhoa
01	LE VAN AN	Nghe An	K1
02	NGUYEN AN	HT	K1
03	LE BINH	HT	K1
04	PHAM TUAN	Nghe An	K2
05	PHAN ANH	Nghe An	K2

KHOA

makhoa	tenkhoa
K1	TOAN TIN
K2	LY
K3	NGOAI NGU
K4	VAN

Tính giá trị của biểu thức $\Pi_{Hoten,\ Tenkhoa}(\sigma_{Tenkhoa='Lý'}(SINHVIEN * KHOA))$?

1 **SINHVIEN * KHOA**

MaSV	Hoten	QueQuan	MaKhoa	tenkhoa
01	LE VAN AN	Nghe An	K1	TOAN TIN
02	NGUYEN AN	HT	K1	TOAN TIN
03	LE BINH	HT	K1	TOAN TIN
04	PHAM TUAN	Nghe An	K2	LY
05	PHAN ANH	Nghe An	K2	LY

2 $\sigma_{Tenkhoa='Lý'}$**(SINHVIEN * KHOA)**

MaSV	Hoten	QueQuan	MaKhoa	tenkhoa
04	PHAM TUAN	Nghe An	K2	LY
05	PHAN ANH	Nghe An	K2	LY

3 $\Pi_{Hoten,\ Tenkhoa}$**($\sigma_{Tenkhoa='Lý'}$(SINHVIEN * KHOA))**

Hoten	tenkhoa
PHAM TUAN	LY
PHAN ANH	LY

I. ĐẠI SỐ QUAN HỆ

Ví dụ 8b:

Cho 2 quan hệ:

SINHVIEN (MaSV, Hoten, QueQuan, Makhoa)

KHOA(Makhoa, Tenkhoa)

Hãy viết biểu thức đại số quan hệ để đưa ra họ tên, tên khoa của sinh viên khoa 'Lý'?

Giải

$$\Pi_{\text{Hoten, Tenkhoa}}(\sigma_{\text{Tenkhoa='Lý'}}(\text{SINHVIEN} * \text{KHOA}))$$

I. ĐẠI SỐ QUAN HỆ

Bài tập:

Cho cơ sở dữ liệu:

HOADON(SoHD, NgayHD, MaKH)

K_HANG(MaKH, TenKH, DiaChiKH, DTKH)

M_HANG(MaMH, TenMH, DVT, DonGia)

HD_MH(SoHD, MaMH, SoLuong, ThanhTien)

Hãy trả lời các câu hỏi sau bằng biểu thức đại số quan hệ:

a. Đưa ra mã, tên các mặt hàng có giá trên 100$?
b. Cho biết tên những khách hàng đã mua hàng ngày 20/03/2009?
c. Đưa ra danh sách các mặt hàng đã bán trong tháng 4 năm 2009?
d. Đưa ra tên những mặt hàng bán trong quý 1 năm 2009?

I. ĐẠI SỐ QUAN HỆ

1.5. Phép chia

Cho hai quan hệ r định nghĩa trên lược đồ quan hệ R(U) và s định nghĩa trên lược đồ quan hệ S(V) với $V \subset U$ và $s \neq \phi$. Đặt X=U \ V. Thương của phép chia quan hệ r cho quan hệ s, ký hiệu $r \div s$ cho ta quan hệ mới với tập thuộc tính là X và các bộ được xác định bởi:

$$\{u[X] \mid (u \in r \text{ và } \forall v \in s \text{ thì } (u[X], v) \in r\}$$

Ta viết:

$$r \div s = \{u[X] \mid (u \in r \text{ và } \forall v \in s \text{ thì } (u[X], v) \in r\}.$$

I. ĐẠI SỐ QUAN HỆ

1.5. Phép chia (tiếp)

Ví dụ 9:

Cho hai quan hệ r và s như sau:

r(A	B	C	D)
a1	b1	1	d1
a1	b2	2	d1
a2	b3	3	d3
a4	b1	1	d2
a1	b3	3	d1
a2	b3	1	d3
a4	b3	3	d2
a4	b2	2	d2
a2	b1	3	d3

s (B	C)
b1	1
b2	2
b3	3

r ÷ s (A D)

I. ĐẠI SỐ QUAN HỆ

1.6. Các phép toán tập hợp

a, Quan hệ khả hợp

Hai quan hệ r và s được gọi là khả hợp (tương thích) nếu chúng có cùng tập thuộc tính.

b. Phép hợp (Union)

Cho hai quan hệ khả hợp r và s xác đinh trên lược đồ quan hệ R(U). Phép hợp 2 quan hệ r và s, ký hiệu $r \cup s$ cho ta một quan hệ mới với tập thuộc tính U và các bộ là các bộ thuộc r hoặc thuộc s.

Ta viết:

$$r \cup s = \{ t \mid t \in r \text{ hoặc } t \in s\}.$$

I. ĐẠI SỐ QUAN HỆ

1.6. Các phép toán tập hợp (tiếp)

Ví dụ 10: Cho hai quan hệ r và s như sau:

r(A	B	C)
a1	b1	1
a2	b2	5
a3	b3	3
a4	b4	1

s(A	B	C)
a1	b1	1
a5	b2	2
a7	b3	3
a4	b4	4

Hợp của r và s cho ta quan hệ:

r ∪ s (A	B	C)
a1	b1	1
a2	b2	5
a3	b3	3
a4	b4	1
a5	b2	2
a7	b3	3
a4	b4	4

I. ĐẠI SỐ QUAN HỆ

1.6. Các phép toán tập hợp (tiếp)

c. Phép giao (Intersect)

Cho hai quan hệ khả hợp r và s xác đinh trên lược đồ quan hệ R(U). Phép giao 2 quan hệ r và s, ký hiệu $r \cap s$ cho ta một quan hệ mới với tập thuộc tính U và các bộ là các bộ thuộc r và thuộc s.

Ta viết:

$$r \cap s = \{ t \mid t \in r \text{ và } t \in s\}.$$

I. ĐẠI SỐ QUAN HỆ

1.6. Các phép toán tập hợp (tiếp)

Ví dụ 11 Cho hai quan hệ r và s như sau:

r(A	B	C)
a1	b1	1
a2	b2	5
a3	b3	3
a4	b4	1

s(A	B	C)
a1	b1	1
a5	b2	2
a7	b3	3
a4	b4	4

Giao của r và s cho ta quan hệ

$r \cap s$ (A	B	C)
a1	b1	1

I. ĐẠI SỐ QUAN HỆ

1.6. Các phép toán tập hợp (tiếp)

d. Phép trừ (Difference)

Cho hai quan hệ khả hợp r và s xác đinh trên lược đồ quan hệ R(U). Phép trừ quan hệ r cho s, ký hiệu r-s cho ta một quan hệ mới với tập thuộc tính U và các bộ là các bộ thuộc r nhưng không thuộc s.

Ta viết:

$$r - s = \{ t \mid t \in r \text{ và } t \notin s\}.$$

I. ĐẠI SỐ QUAN HỆ

1.6. Các phép toán tập hợp (tiếp)

Ví dụ 12 Cho hai quan hệ r và s như sau:

r(A	B	C)
a1	b1	1
a2	b2	5
a3	b3	3
a4	b4	1

s(A	B	C)
a1	b1	1
a5	b2	2
a7	b3	3
a4	b4	4

Hiệu của r trừ s cho ta quan hệ:

r - s (A	B	C)
a2	b2	5
a3	b3	3
a4	b4	1

I. ĐẠI SỐ QUAN HỆ

Bài tập

Cho cơ sở dữ liệu:

HOADON(SoHD, NgayHD, MaKH)

K_HANG(MaKH, TenKH, DiaChiKH, DTKH)

M_HANG(MaMH, TenMH, DVT, DonGia)

HD_MH(SoHD, MaMH, SoLuong, ThanhTien)

Hãy trả lời các câu hỏi sau bằng biểu thức đại số quan hệ:

a. Đưa ra tên những mặt hàng không bán được trong quý 1 năm 2009?
b. Cho biết những mặt hàng chưa được bán lần nào?
c. Cho biết những mặt hàng bán được trong tháng 12/2008 nhưng không bán được trong tháng 1/2009?

II. ĐẠI SỐ HỆ VÀ CÁC TÍNH CHẤT CỦA ĐSQH

(Tự đọc giáo trình)

III. NGÔN NGỮ SQL (STRUCTURED QUERY LANGUAGE)

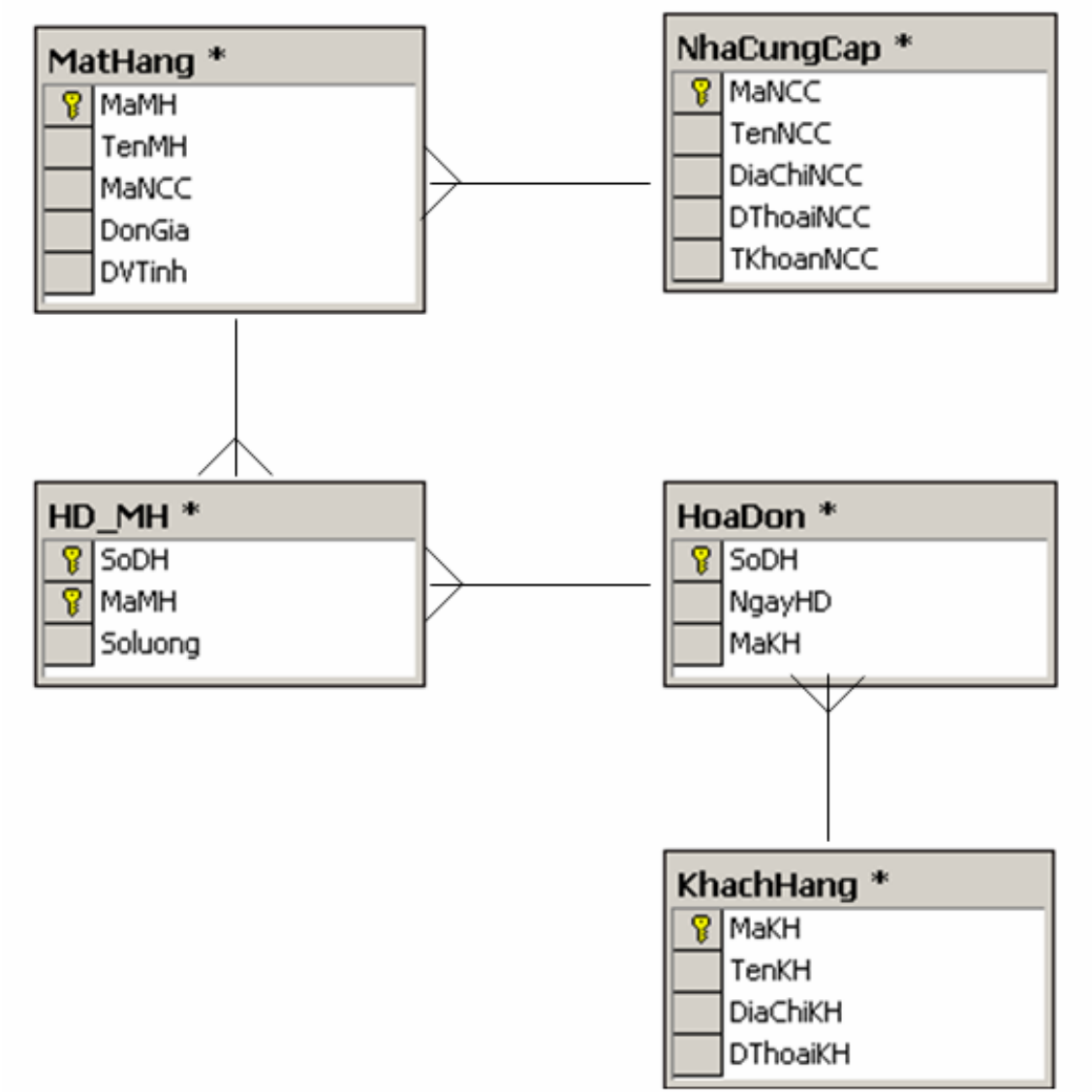
3.1. Giới thiệu

SQL là sản phẩm của công ty IBM San jose California USA. Ngôn ngữ SQL được phát triển từ ngôn ngữ SEQUEL (Strructured English Query Language) giới thiệu năm 1976. Ngay sau khi ra đời SQL đã được ứng dụng rộng rãi trong lĩnh vực khai thác cơ sở dữ liệu quan hệ. Hiện nay, hầu hết các hệ quản trị cơ sở dữ liệu đều có phần ngôn ngữ này.

Khả năng của SQL gồm:

- Định nghĩa dữ liệu : tạo cơ sở dữ liệu và cấu trúc bảng của nó,
- Truy vấn dữ liệu (select).
- Sửa đổi dữ liệu: thêm(insert) , xoá (delete) và cập nhật (update)

III. NGÔN NGỮ SQL



III. NGÔN NGỮ SQL

3.2. Ngôn ngữ định nghĩa dữ liệu

a. Tạo bảng

Cú pháp:

CREATE TABLE <Tên_bảng> **(**

<Tên_cột_1> <Kiểu_dữ_liệu> [**Not null**] [**unique**] ,

<Tên_cột_2> <Kiểu_dữ_liệu> [**Not null**] [**unique**] ,

. . .

<Tên_cột_n> <Kiểu_dữ_liệu> [**Not null**] [**unique**],

PRIMARY KEY (Khoá_chính),

[**UNIQUE** (Khoá), ...]

[**FOREIGN KEY** (Khoá_ngoại) **REFERENCES**

<Bảng_tham_chiếu>[(khoá)] [on update cascade]

[on delete cascade | on delete restrict] ,..]

[**CHECK** (<Điều_kiện_ràng_buộc>)] **);**

III. NGÔN NGỮ SQL

3.2. Ngôn ngữ định nghĩa dữ liệu (tiếp)

- **Integer**: Số nguyên từ: - 2 147 483 648 đến 2 147 483 647
- **Smallinteger**: Số nguyên từ - 32768 đến 32767
- **Decimal(n,p)**: Số thực với độ dài tối đa là n chữ số trong đó có p chữ số thập phân.
- **Float**: Số thực dạng dấu chấm động
- **Char(n):** Xâu ký tự có độ dài cố định n (n <=255)
- **Varchar(n)**: Xâu ký tự có độ dài biến đổi từ 0 đến n . Độ dài <= n.
- **Date**: Kiểu ngày tháng.

III. NGÔN NGỮ SQL

3.2. Ngôn ngữ định nghĩa dữ liệu (tiếp)

b. Xoá bảng

Cú pháp:

DROP TABLE <Tên_bảng>;

Ví dụ 1: Xoá bảng HD_MH.

DROP TABLE HD_MH;

III. NGÔN NGỮ SQL

3.2. Ngôn ngữ định nghĩa dữ liệu (tiếp)

c. Thêm cột của bảng

Cú pháp:

ALTER TABLE <Tên_bảng>

ADD <Tên_cột> <Kiểu_dữ_liệu> [**not null**];

Ví dụ 2:

Thêm cột MoTaMH (mô tả mặt hàng) với kiểu dữ liệu Varchar(100) cho bảng MatHang.

ALTER TABLE MatHang

ADD MoTaMH varchar(100);

III. NGÔN NGỮ SQL

3.2. Ngôn ngữ định nghĩa dữ liệu (tiếp)

d. Xoá cột của bảng

Cú pháp:

ALTER TABLE <Tên_bảng>

DROP <Tên_cột>;

Ví dụ 3: Xoá cột TKkhoanNCC của bảng NhaCungCap.

ALTER TABLE NhaCungCap

DROP TKhoanNCC;

III. NGÔN NGỮ SQL

3.3. Truy vấn dữ liệu

a. Cấu trúc cơ bản của câu lệnh SELECT...

SELECT $A_1, A_2, \ldots, A_n$

FROM $r_1, r_2, \ldots, r_m$

WHERE P;

Trong đó:

- A_i (i=1..n) là tên các thuộc tính có mặt trong kết quả truy vấn,
- r_i (i=1..m) là tên các quan hệ,
- P là biểu thức logic mà các bộ trong kết quả truy vấn phải thoã mãn.

Truy vấn trên tương đương với biểu thức đại số quan hệ sau:

$$\Pi_{\{A1, A2, \ldots, An\}}(\sigma_P (r_1 \times r_2 \times \ldots \times r_m))$$

III. NGÔN NGỮ SQL

3.3. Truy vấn dữ liệu (tiếp)

b, Mệnh đề SELECT

Mệnh đề SELECT liệt kê các thuộc tính cần đưa ra trong kết quả truy vấn. Mệnh đề này tương ứng với phép chiếu trong đại số quan hệ.

Ví dụ 3.5.5:

Đưa ra mã, tên của tất cả các mặt hàng có bán tại của hàng

```
SELECT MaMH, TenMH
FROM MATHANG;
```

Chú ý: Các tên (thuộc tính, quan hệ) trong SQL không phân biệt chữ hoa và chữ thường; chỉ bao gồm chữ cái, chữ số và dấu gạch dưới.

III. NGÔN NGỮ SQL

3.3. Truy vấn dữ liệu

b, Mệnh đề SELECT (tiếp)

- SQL cho phép lặp các bộ trong kết quả truy vấn. Để loại bỏ các bộ lặp ta viết thêm từ khoá **DISTINCT** vào sau **SELECT**.

Ví dụ 3.5.6:

Đưa ra danh sách tên tỉnh/thành phố có khách hàng đã mua hàng?

```
SELECT DISTINCT DiaChiKH
FROM KHACHHANG;
```

III. NGÔN NGỮ SQL

3.3. Truy vấn dữ liệu

b, Mệnh đề SELECT (tiếp)

- Để giữ lại các bộ lặp ta thêm từ khoá **ALL** và sau **SELECT**.

Ví dụ 3.5.7:

```
SELECT ALL DiaChiKH
FROM KHACHHANG;
```

III. NGÔN NGỮ SQL

3.3. Truy vấn dữ liệu

b, Mệnh đề SELECT (tiếp)

- Nếu giữa các quan hệ trong mệnh đề FROM có thuộc tính chung, thì ta phải chỉ rõ thuộc tính đó là của quan hệ nào, bằng cách viết tên quan hệ ngay trước tên thuộc tính, giữa chúng được ngăn cách bởi dấu chấm (.).

Ví dụ 3.5.8:

```
SELECT KHACHHANG.MaKH,TenKH, SoDH, NgayHD
FROM KHACHHANG, HOADON;
```

III. NGÔN NGỮ SQL

3.3. Truy vấn dữ liệu

b, Mệnh đề SELECT (tiếp)

- Trong trường hợp ta muốn đưa ra tất cả các thuộc tính của các quan hệ trong mệnh đề **FROM**, ta chỉ cần dùng dấu '*' thay cho việc liệt kê các thuộc tính.

Ví dụ 3.5.9:

```
SELECT *
FROM KHACHHANG, HOADON
WHERE (KHACHHANG.MaKH=HOADON.MaKH);
```

III. NGÔN NGỮ SQL

3.3. Truy vấn dữ liệu

b, Mệnh đề SELECT (tiếp)

- Trong mệnh đề **SELECT** có thể chứa biểu thức toán học gồm các phép toán: +, -, *, / .

Ví dụ 3.5.11:

Đưa ra danh sách các mặt hàng với đơn giá được giảm 10%

```
SELECT MaMH, TenMH,DonGia -  DonGia*0.1
FROM MATHANG;
```

III. NGÔN NGỮ SQL

3.3. Truy vấn dữ liệu (tiếp)

c, Mệnh đề FROM

Mệnh đề **FROM** liệt kê các quan hệ phải có trong truy vấn. Tương ứng với tích Đề Các trong đại số quan hệ.

III. NGÔN NGỮ SQL

3.3. Truy vấn dữ liệu (tiếp)

d, Mệnh đề WHERE

- Mệnh đề **WHERE** xác định điều kiện mà kết quả truy vấn phải thoã mãn. Tương ứng với phép chọn của đại số quan hệ.
- Sau mệnh đề **WHERE** là biểu thức logic với các phép so sánh

(>, >=, <, <=, =, <>) và các phép: AND, OR và NOT.

III. NGÔN NGỮ SQL

3.3. Truy vấn dữ liệu

d, Mệnh đề WHERE (tiếp)

Ví dụ 3.5.12:

Đưa ra danh sách các mặt hàng của nhà cung cấp 'DELL' có giá thấp hơn 1000000.

```
SELECT *
FROM  MATHANG, NHACUNGCAP
WHERE  (MATHANG.MaNCC=NHACUNGCAP.MaNCC) AND
       (TenNCC = 'DELL')  AND
       (DonGia < 1000000);
```

Ghi chú: *Các hằng ký tự và ngày tháng trong SQL được để trong cặp dấu nháy đơn (').*

III. NGÔN NGỮ SQL

3.3. Truy vấn dữ liệu

e, Một số toán tử khác

- Toán tử so sánh **BETWEEN** … **AND** …

Ví dụ 3.5.13: Tìm tất cả các mặt hàng có giá từ 100000 đến 200000.

```
SELECT *
FROM MATHANG
WHERE (DonGia  >=100000) AND (DonGia <=200000);
```

Câu lệnh trên có thể viết lại như sau:

```
SELECT *
FROM MATHANG
WHERE DonGia  between 100000 and  200000;
```

III. NGÔN NGỮ SQL

3.3. Truy vấn dữ liệu

e, Một số toán tử khác

- Toán tử so sánh xâu ký tự **like**

Ví dụ 3.5.14: Đưa ra danh sách các mặt hàng mà tên mặt hàng chứa cụm từ 'HP'.

```
SELECT *
FROM MATHANG
WHERE TenMH  like '%HP%';
```

Ta có thể sử dụng 2 ký tự '%' hay '_ ' để thay thế cho một xâu hay một ký tự bất kỳ.

- % : Thay cho một chuỗi ký tự bất kỳ
- _: Thay cho một ký tự bất kỳ

III. NGÔN NGỮ SQL

5.5. Sắp xếp kết quả truy vấn

Để sắp xếp kết quả truy vấn theo thứ tự tăng hay giảm dần giá trị của một hay một số cột nào đó, ta viết mệnh đề **ORDER BY** vào sau truy vấn.

Cách viết:

ORDER BY B1 [**DESC** | **ASC**] [,B2 [**DESC** | **ASC**]] …[,Bk [**DES**C | **ASC**]]

III. NGÔN NGỮ SQL

5.6. Các phép toán tập hợp (Set Operations)

Để thực hiện phép hợp, giao và trừ trong SQL ta dùng các phép toán: **union, intersect** và **except**.

Với các phép trên, SQL sẽ tự động loại bỏ các bộ trùng nhau; để giữ lại các bộ trùng nhau ta viết từ khoá **all** vào phía sau các phép toán, tức là: **union all, intersect all** và **except all**.

III. NGÔN NGỮ SQL

5.7. Giá trị NULL và các hàm tổng hợp của SQL (Aggregate Functions)

a, Giá trị NULL

Trong mỗi dòng của bảng, có thể có một số thuộc tính mà giá trị của chúng là chưa biết, hay không tồn tại. Trong SQL để biểu thị giá trị chưa biết hay không tồn tại người ta dùng hằng *NULL*.

Để kiểm tra giá trị của thuộc tính có là *NULL* hay không ta dùng toán tử **IS.**

III. NGÔN NGỮ SQL

5.7. Giá trị NULL và các hàm tổng hợp của SQL

b, Các phép toán trên giá trị NULL:

- Khi thực hiện các phép toán số học với giá trị *NULL*, kết quả là một giá trị *NULL*.

  Ví dụ 3..5.19: 6+ *NULL* =*NULL*

- Khi thực hiện phép so sánh (>,>=,<,<=,<>,=) với giá trị *NULL* kết quả trả về là FALSE.

III. NGÔN NGỮ SQL

5.7. Giá trị NULL và các hàm tổng hợp của SQL

c, Các hàm tổng hợp

Các hàm tổng hợp của SQL gồm:

- **SUM**(Tên_cột): Tính tổng giá trị theo cột có tên là Tên_cột của các bộ.
- **MAX**(Tên_cột): Cho giá trị lớn nhất trong cột có tên là Tên_cột.
- **MIN**(Tên_cột): Cho giá trị nhỏ nhất trong cột có tên là Tên_cột.
- **AVG**(Tên_cột): Tính giá trị trung bình theo cột có tên là Tên_cột của các bộ.

III. NGÔN NGỮ SQL

5.7. Giá trị NULL và các hàm tổng hợp của SQL

c, Các hàm tổng hợp

- **COUNT**(*|Tên_cột |**DISTINCT** Tên_cột): Đếm số bản ghi trong bảng theo tuỳ chọn:
  - * : Đếm tất cả các bản ghi trong bảng
  - Tên_cột: Đếm các bản ghi mà giá trị của cột Tên_cột khác NULL
  - **DISTINCT** Tên_cột : Đếm các bản ghi mà giá trị của cột Tên_cột khác NULL và các bản ghi giống nhau chỉ tính một.

III. NGÔN NGỮ SQL

5.8. Các hàm tổng hợp với việc nhóm dữ liệu

Để nhóm dữ liệu trên một hay một số cột, ta dùng mệnh đề **GROUP BY** được viết ngay sau mệnh đề **WHERE** hay **FROM** (nếu không có **WHERE**).

Cú pháp:

GROUP BY <danh sách cột>

[**HAVING** <điều kiện>]

Trong đó:

- <*danh sách cột*> là danh sách các cột làm cơ sở để nhóm
- Mệnh đề **HAVING** để xác định điều kiện mà các bản ghi sau khi nhóm phải thoã mãn.

III. NGÔN NGỮ SQL

5.8. Các hàm tổng hợp với việc nhóm dữ liệu

Ví dụ 1:

Cho biết số loại mặt hàng của mỗi nhà cung cấp đã cung ứng?

```
SELECT NHACUNGCAP.MaNCC, TenNCC, COUNT(*)
FROM NHACUNGCAP, MATHANG
WHERE  NHACUNGCAP.MaNCC= MATHANG.MaNCC
GROUP BY NHACUNGCAP.MaNCC, TenNCC;
```

III. NGÔN NGỮ SQL

5.8. Các hàm tổng hợp với việc nhóm dữ liệu

Ví dụ 2:

Cho biết các số hoá đơn mà trên đó bán từ 2 mặt hàng trở lên?

```
SELECT HOADON.SoHD, COUNT(*)
FROM HOADON, HD_MH
WHERE  HOADON.SoHD= HD_MH.SoHD
GROUP BY HOADON.SoHD
HAVING COUNT(*)>=2;
```

III. NGÔN NGỮ SQL

5.8. Các hàm tổng hợp với việc nhóm dữ liệu

Ghi chú:

- Các thuộc tính có trong mệnh đề SELECT đều phải có mặt trong mệnh đề GROUP BY
- Thứ tự thực hiện một câu lệnh truy vấn như sau:

FROM →WHERE →GROUP BY→HAVING→ ORDER BY→SELECT

III. NGÔN NGỮ SQL

5.9. Truy vấn lồng (Nested Subqueries)

SQL cung cấp một cơ chế cho phép cho phép lồng các truy vấn con trong mệnh đề **FROM** và **WHERE**. Mỗi truy vấn con là một truy vấn có dạng **SELECT** … **FROM**…**WHERE**.

✍ *Thường các truy vấn con được sử dụng để kiểm tra một phần tử thuộc tập hợp (quan hệ), so sánh tập hợp.*

III. NGÔN NGỮ SQL

5.9. Truy vấn lồng (Nested Subqueries)

Ví dụ 1 Đưa ra tên mặt hàng có giá cao nhất?

```
SELECT TenMH
FROM MATHANG
WHERE  DonGia in (SELECT MAX(DonGia) FROM MATHANG);
```

III. NGÔN NGỮ SQL

5.9. Truy vấn lồng (Nested Subqueries)

Ví dụ 2: Đưa ra tên các mặt hàng không được bán trong ngày '05/06/2006'

```
SELECT TenMH
FROM MATHANG
WHERE  MaMH not in (SELECT MaMH
                    FROM HD_MH, HOADON
                    WHERE (HD_MH.SoHD= HOADON.SoHD)
                          AND  (NgayHD='05/06/2006'));
```

III. NGÔN NGỮ SQL

5.10. Khung nhìn (Views)

SQL cung cấp một cơ chế cho phép che dấu đi một số dữ liệu đối với người sử dụng bằng việc sử dụng một khung nhìn. Để tạo khung nhìn ta dùng lệnh:

CREATE VIEW <tên khung nhìn> **AS**

<biểu thức truy vấn>

III. NGÔN NGỮ SQL

5.10. Khung nhìn (Views)

Ví dụ 1: Tạo khung nhìn chứa danh sách các mặt hàng được cung ứng bởi nhà cung cấp 'Sam sung'.

```
CREATE VIEW HangSS AS
        SELECT MaMH, TenMH, DonGia
        FROM MATHANG, NHACUNGCAP
        WHERE (MATHANG.MaNCC= NHACUNGCAP.MaNCC)
                AND (TenNCC='Sum sung');
```

III. NGÔN NGỮ SQL

5.10. Khung nhìn (Views)

Ví dụ 2: Cho biết tên mặt hàng bán chạy nhất?

(1)

```
CREATE VIEW TH AS
        SELECT  TenMH, SUM(Soluong) AS sl
        FROM MATHANG, HD_MH
        WHERE  MATHANG.MaMH= HD_MH.MaMH
        GROUP BY TenMH;
```

(2)

```
SELECT  *
FROM TH
WHERE sl IN (SELECT MAX(sl) FROM TH);
```

III. NGÔN NGỮ SQL

5.11. Các lệnh cập nhật dữ liệu

a. Vào dữ liệu cho bảng

Để chèn thêm một bản ghi mới vào bảng, ta dùng lệnh sau:

INSERT INTO <Tên_bảng>[(B1, B2,…,Bk)]

VALUES (C1, C2,…., Ck);

Trong đó:

- Bi (i=1,..k, k<=n _ n là số thuộc tính của <Tên_bảng>) là các thuộc tính của <Tên_bảng>. Nếu không có danh sách thuộc tính này SQL sẽ lấy toàn bộ thuộc tính.
- Ci (i=1,..k) là giá trị tương ứng với các thuộc tính Bi. Nếu Ci là giá trị Null, ta dùng hằng NULL cho Ci.

III. NGÔN NGỮ SQL

5.11. Các lệnh cập nhật dữ liệu

a. Vào dữ liệu cho bảng

Ví dụ: Thêm bản ghi (‘SO’, 'Sony’, ‘Nhật’, NULL, '13.111.02.987') vào bảng NHACUNGCAP.

```
INSERT INTO NHACUNGCAP
    VALUES ('SO', 'Sony', 'Nhật', NULL, '13.111.02.987');
```

III. NGÔN NGỮ SQL

5.11. Các lệnh cập nhật dữ liệu

a. Vào dữ liệu cho bảng

Để chèn thêm một bản ghi mới vào bảng, ta dùng lệnh sau:

INSERT INTO <Tên_bảng>[(B1, B2,…,Bk)]

VALUES (C1, C2,…., Ck);

Trong đó:

- Bi (i=1,..k, k<=n _ n là số thuộc tính của <Tên_bảng>) là các thuộc tính của <Tên_bảng>. Nếu không có danh sách thuộc tính này SQL sẽ lấy toàn bộ thuộc tính.
- Ci (i=1,..k) là giá trị tương ứng với các thuộc tính Bi. Nếu Ci là giá trị Null, ta dùng hằng NULL cho Ci.

III. NGÔN NGỮ SQL

5.11. Các lệnh cập nhật dữ liệu

b. Xoá bản ghi trong bảng

DELETE FROM Tên_bảng

[**WHERE** điều_kiện]

Ví dụ: Loại bỏ các mặt hàng được cung ứng bởi nhà cung cấp có mã 'HP'

DELETE FROM MATHANG

WHERE MaNCC=’HP’

III. NGÔN NGỮ SQL

5.11. Các lệnh cập nhật dữ liệu

c. Sửa nội dung các bản ghi trong bảng

UPDATE Tên_bảng

SET <Tên_cột 1> = <Biểu_thức 1>,

…..

<Tên_cột k> = <Biểu_thức k>

[**WHERE** Điều_kiện]

Trong đó:

- <Tên_bảng> là bảng có bản ghi cần thay đổi nội dung,
- <Tên_cột i> (i=1,..k) là các cột cần thay đổi giá trị bởi các giá trị mới là <Biểu_thứci> tương ứng.

III. NGÔN NGỮ SQL

5.11. Các lệnh cập nhật dữ liệu

c. Sửa nội dung các bản ghi trong bảng

Ví dụ: Viết lệnh để thực hiện việc giảm giá 10% cho các mặt hàng của nhà cung cấp có mã 'TO'.

```
UPDATE  MATHANG
SET DonDia = DonGia - DonGia * 0.1
WHERE MaNCC=’TO’
```

III. NGÔN NGỮ SQL

5.12. Kết nối các quan hệ

Phép kết nối cho phép kết nối 2 quan hệ và kết quả trả về là một quan hệ khác. SQL cung cấp một số loại kết nối sau:

- r **INNER JOIN** s **ON** <*điều kiện*> : Kết nối trong,
- r **LEFT OUTER JOIN** s **ON** <*điều kiện*> : Kết nối ngoài trái
- r **RIGHT OUTER JOIN** s **ON** <*điều kiện*> : Kết nối ngoài phải

Trong đó: r,s là tên quan hệ

Chương 3: Tầng giao vận

Mục tiêu :

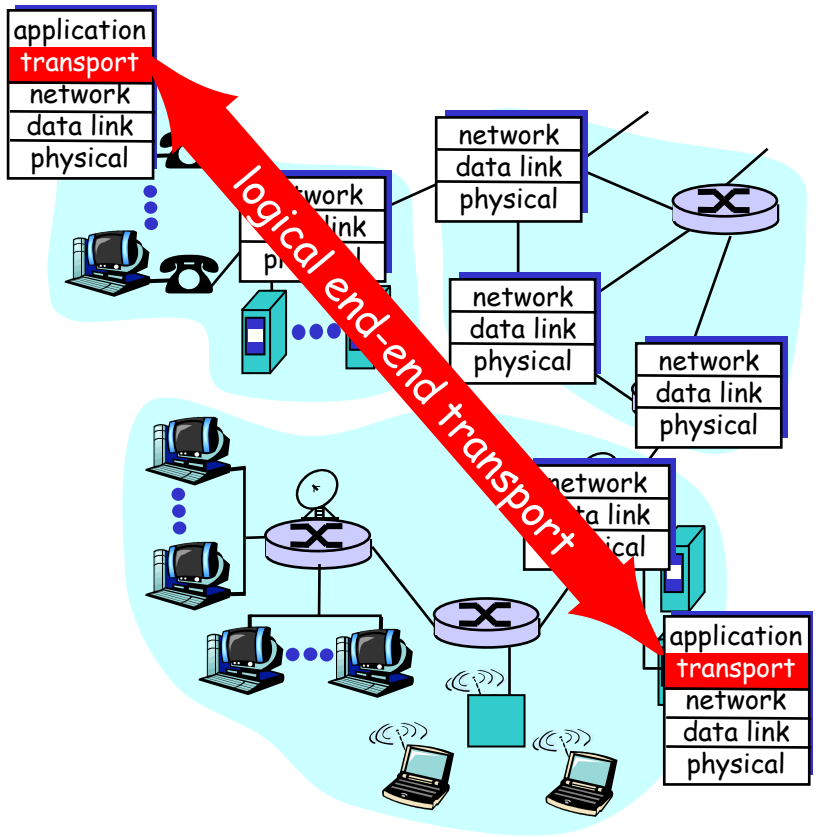
- Nguyên lý các dịch vụ của tầng giao vận:
  - Phân kênh / Dồn kênh
  - Xây dựng đường truyền tin cậy
  - Điều khiển lưu lượng
  - Kiểm soát tắc nghẽn
- Cài đặt những nguyên lý này trên Internet như thế nào ?

Sẽ học cái gì ?

- Các dịch vụ của tầng giao vận
- Phân kênh / Dồn kênh
- UDP : Giao thức không hướng nối
- Nguyên lý xây dựng đường truyền tin cậy
- TCP : Giao thức hướng nối
  - Tin cậy
  - Điều khiển lưu lượng
  - Kiểm soát tắc nghẽn
- Nguyên lý kiểm soát tắc nghẽn
- Kiểm soát tắc nghẽn trong TCP

Dịch vụ và Giao thức ở tầng Giao vận

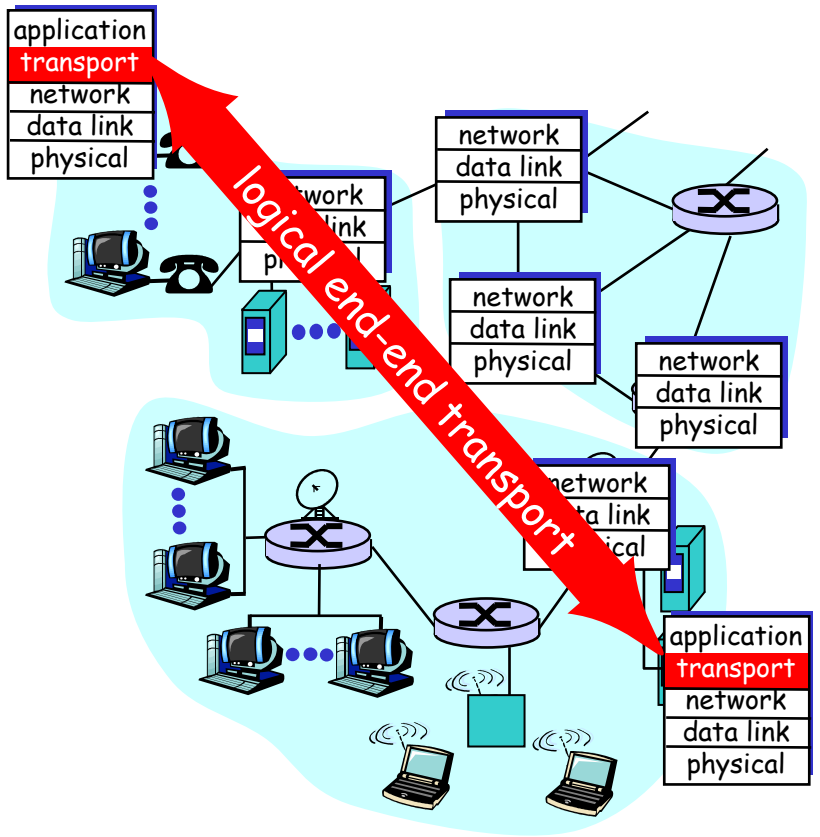
- Cung cấp *đường truyền logic* giữa các tiến trình chạy trên các thiết bị đầu cuối.
- Giao thức Giao vận chạy trên thiết bị đầu cuối
- Phân biệt dịch vụ của tầng Giao vận với tầng Mạng:
- *Tầng Mạng:* Chuyển dữ liệu giữa các thiết bị đầu cuối.
- *Tầng Giao vận:* Chuyển dữ liệu giữa các tiến trình
  - Dựa trên tầng Mạng nhưng biến đổi, tăng cường



Giao thức ở tầng Giao vận

Có hai giao thức giao vận trên Internet:

- Tin cậy, Nhận theo đúng thứ tự gửi, một người nhận (TCP)
  - Kiểm soát tắc nghẽn
  - Điều khiển lưu lượng
  - Thiết lập đường truyền
- Không tin cậy ("cố gắng tối đa"), Không theo đúng thứ tự: UDP
- Cả TCP và UDP đều không cung cấp:
  - Chuyển theo thời gian thực
  - Băng thông tối thiểu
  - Nhiều người nhận tin cậy

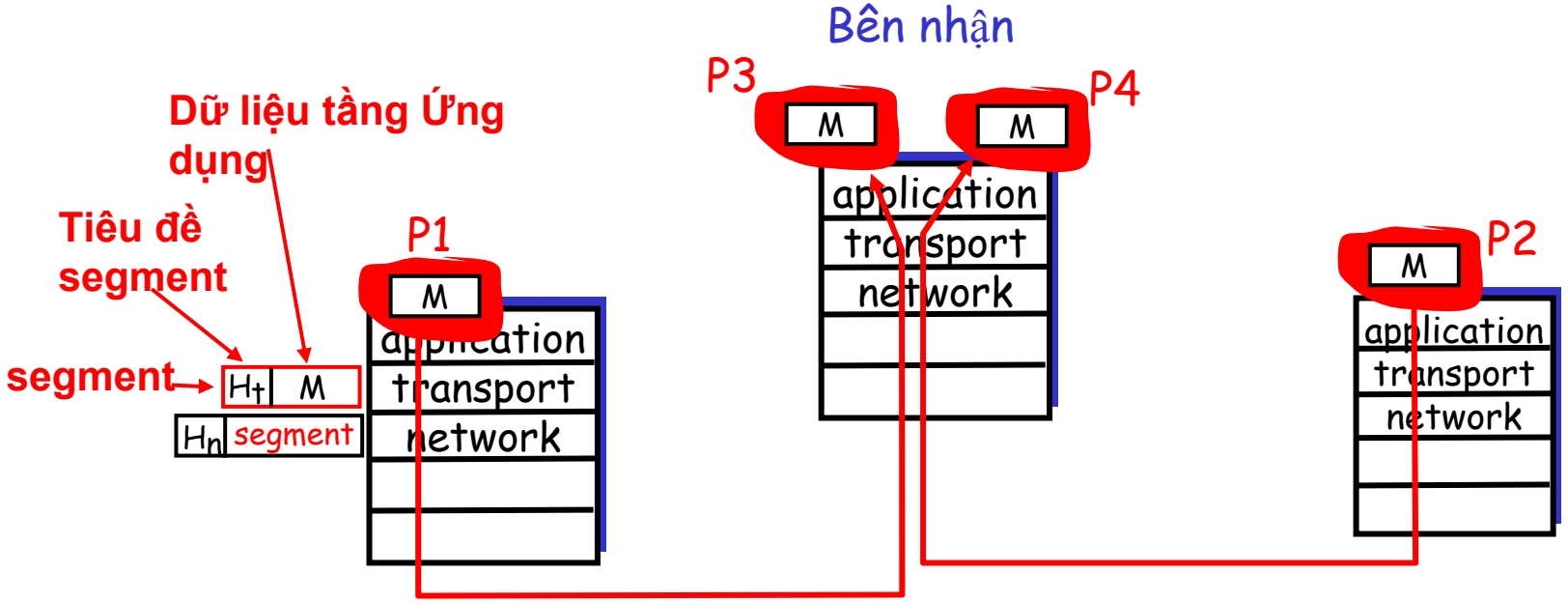


Phân kênh / Dồn kênh

Nhớ lại: *segment* – Đơn vị dữ liệu trao đổi giữa hai thực thể Giao vận

- Còn gọi là TPDU: Transport Protocol Data Unit

Phân kênh: Chuyển các segment nhận được cho Ứng dụng phù hợp

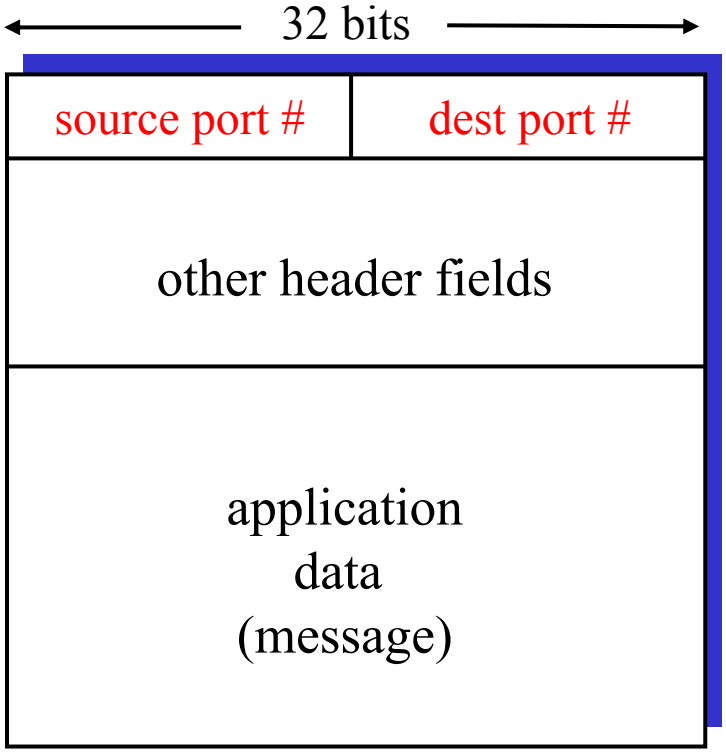


Phân kênh / Dồn kênh

Dồn kênh:
Lấy dữ liệu từ nhiều trình Ứng dụng, Bổ sung thêm các tiêu đề (để sau này Phân kênh sử dụng

Dồn kênh/Phân kênh:

- Dựa trên Địa chỉ IP, Số hiệu cổng của bên Gửi và bên Nhận
  - source, dest port trong mỗi segment
  - Nhớ lại: Các số hiệu cổng của các ứng dụng thông dụng



Khuôn dạng tổng quát của TCP/UDP

Phân biệt Phân kênh và Dồn kênh

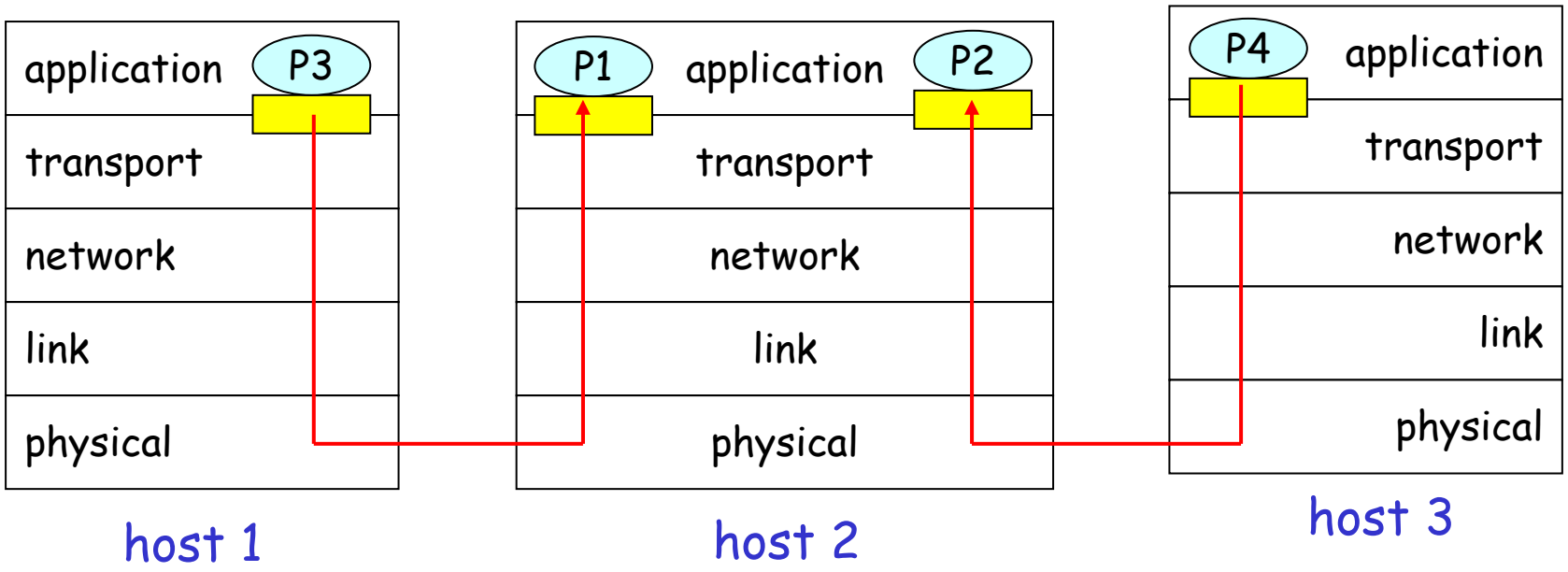
Phân kênh ở nút nhận:

Chuyển segment đến đúng socket

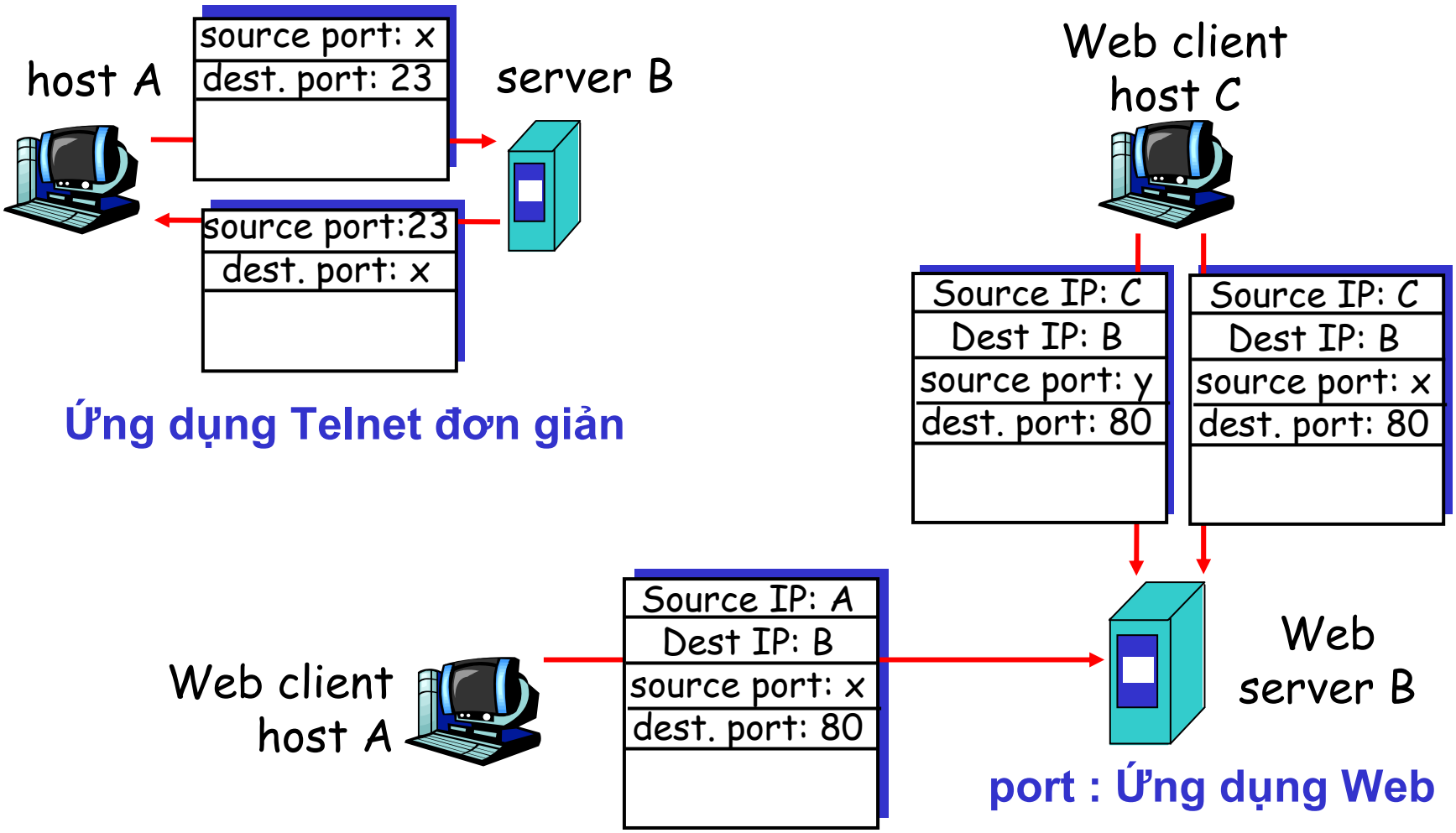
Dồn kênh ở nút gửi

Lấy dữ liệu từ nhiều socket, đặt trong các phong bì khác nhau (để phân kênh sau này)

= socket = process

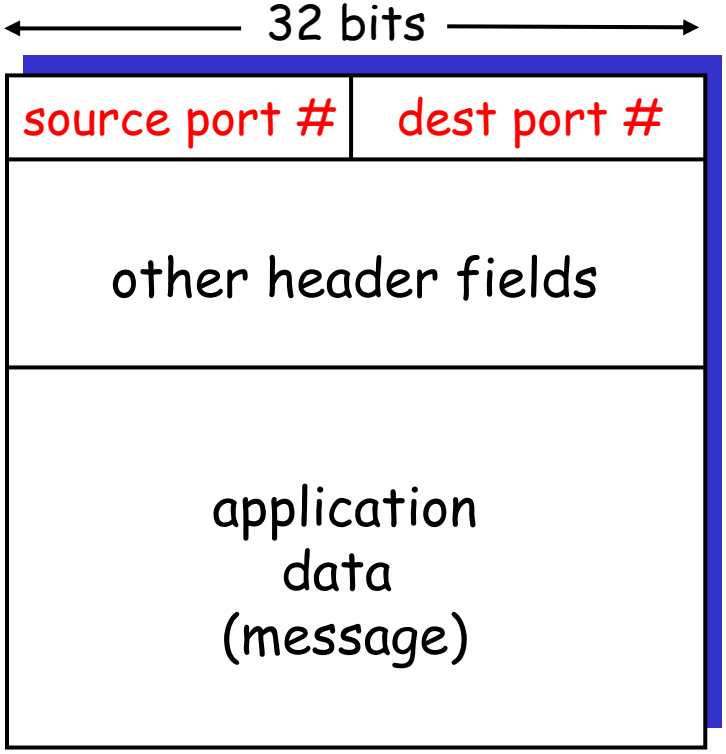


Ví dụ : Phân kênh / Dồn kênh



Phân kênh bằng cách nào ?

- Máy tính đích nhận được IP datagrams
  - Mỗi datagram có địa chỉ IP đích, IP nguồn
  - Mỗi datagram chứa một segment của tầng giao vận
  - Mỗi segment chứa địa chỉ cổng gửi, cổng nhận
    (chú ý : cổng của các ứng dụng thông dụng)
- Máy tính đích sử dụng địa chỉ IP và cổng của bên gửi để chuyển segment đến socket thích hợp



TCP/UDP segment format

UDP: User Datagram Protocol [RFC 768]

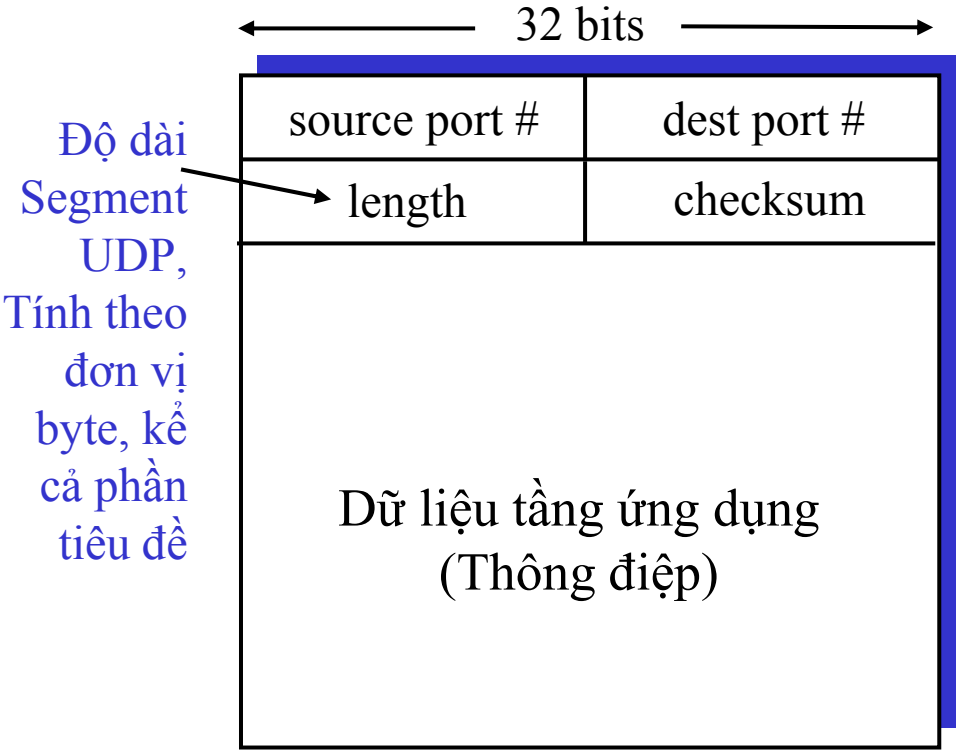
- Giao thức giao vận cực kỳ đơn giản, gần như không có gì
- Dịch vụ kiểu "cố gắng tối đa", các segment của UDP có thể :
  - Mất.
  - Tầng ứng dụng chịu trách nhiệm sắp xếp theo đúng thứ tự.
- ***Không hướng nối***:
  - Phía gửi, phía nhận không cần "bắt tay"
  - Các segment UDP được xử lý độc lập

Tại sao sử dụng UDP?

- Độ trễ nhỏ: Không có giai đoạn thiết lập đương truyền
- Đơn giản: Phía gửi và nhận không phải ghi nhớ trạng thái gửi/ nhận.
- Tiêu đề gói tin bé
- Không có cơ chế kiểm soát tắc nghẽn: Bên gửi có thể gửi dữ liệu với tốc độ tối đa.

UDP (tiếp)

- Thường được các ứng dụng đa phương tiện sử dụng
  - Chấp nhận mất dữ liệu
  - Tốc độ truyền quan trọng
- Một số ứng dụng khác cũng sử dụng (Vì sao?):
  - DNS
  - SNMP
- Muốn truyền tin cậy bằng UDP: phải đặt cơ chế tin cậy tại ứng dụng
  - Ứng dụng chịu trách nhiệm phát hiện và khắc phục lỗi!



Khuôn dạng segment UDP

UDP checksum

Mục tiêu: phát hiện "lỗi" (bit bị thay đổi giá trị) trong segment nhận được.

Bên gửi:

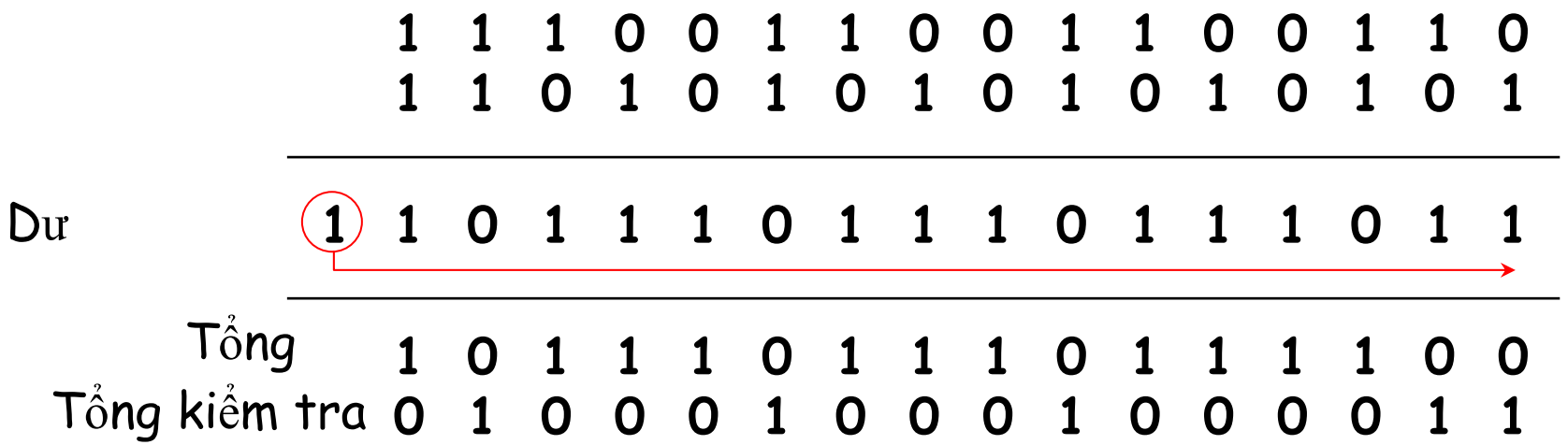
- Xem nội dung segment là các số nguyên 16-bit liên tiếp (các từ).
- checksum: tổng bù 1 của các từ
- Phía gửi sẽ đặt giá trị checksum tính được vào trường checksum trong tiêu đề gói tin UDP

Bên nhận:

- Tính giá trị checksum của segment vừa nhận được
- Kiểm tra xem giá trị vừa tính được có trùng với giá trị trong trường checkum ở tiên đề không:
  - **KHÔNG TRÙNG** – có lỗi
  - **TRÙNG** – Không phát hiện được lỗi. *Nhưng có thể có lỗi.*

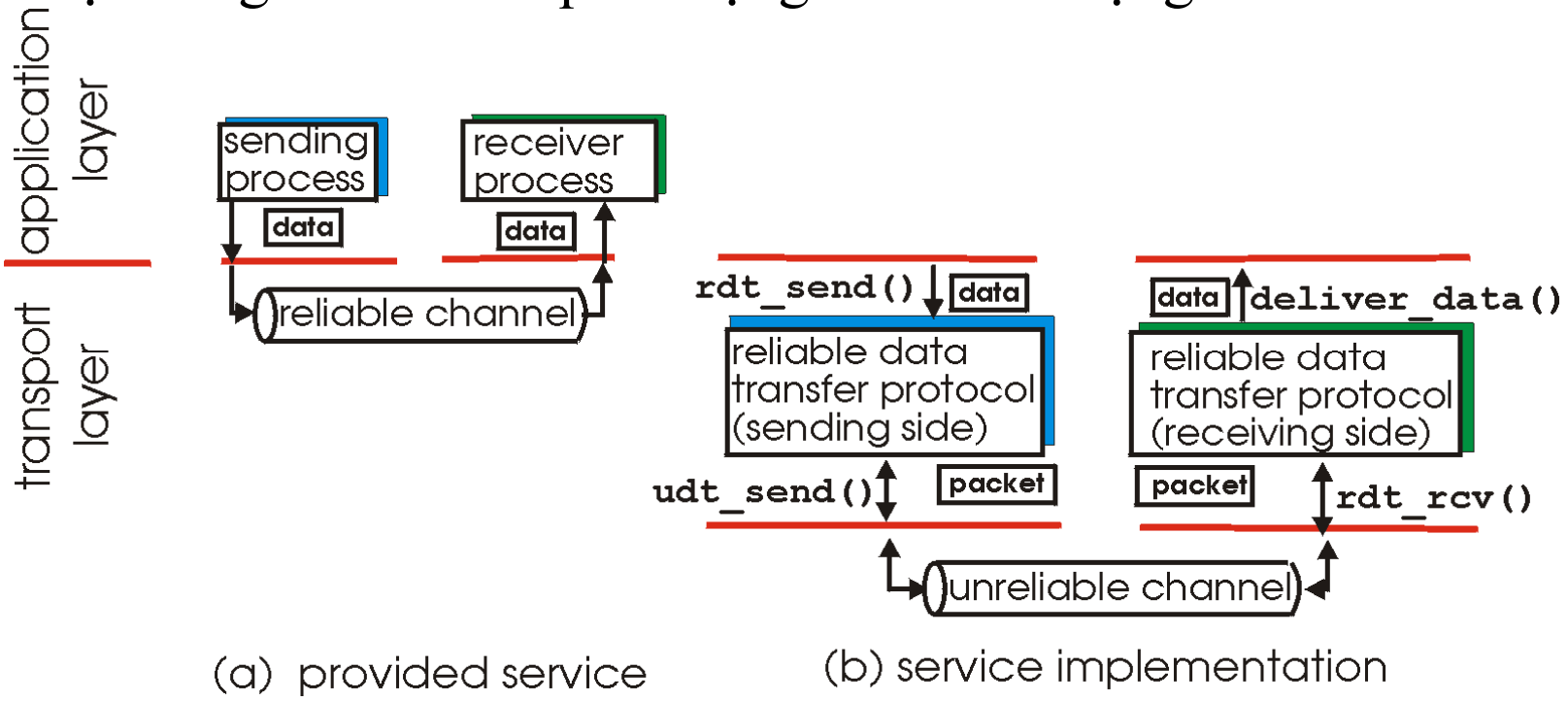
Ví dụ tính UDP Checksum

- Note
  - Khi cộng hai số, nhớ ở bit cao nhất sẽ được cộng vào kết quả
- Ví dụ: Cộng hai số nguyên 16-bit



Nguyên lý Truyền tin cậy

- Có thể áp dụng trong Tầng Giao vận và Liên kết dữ liệu
- Một trong 10 vấn đề quan trọng nhất của Mạng!

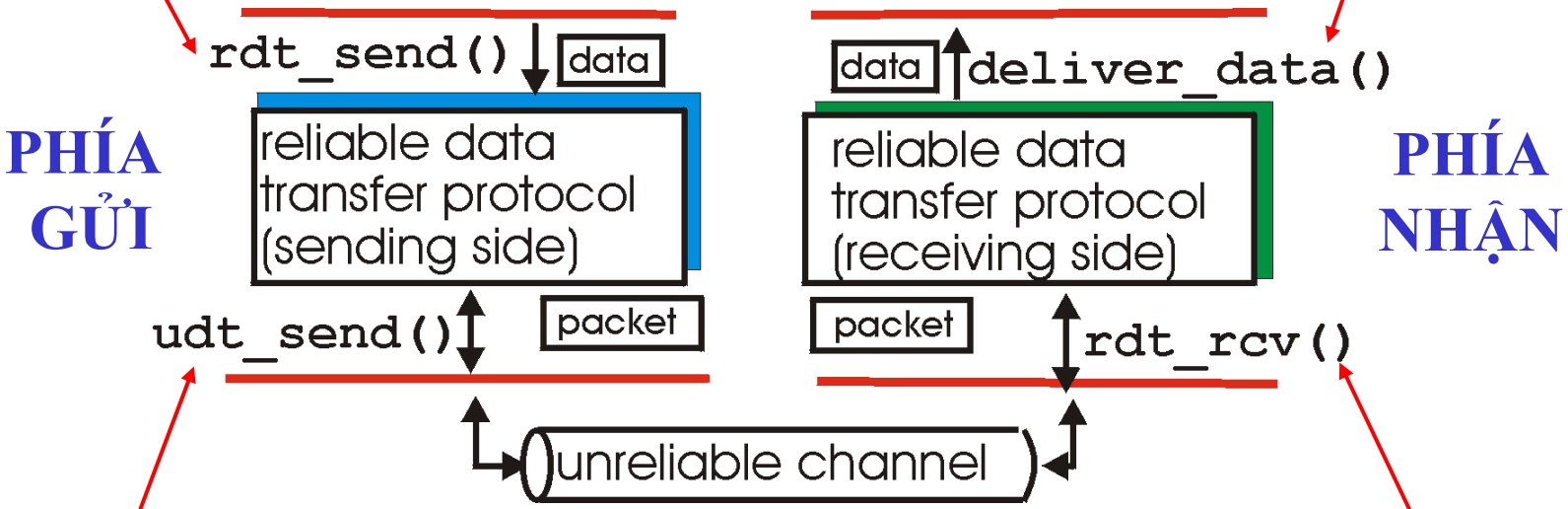


- Các đặc điểm của đường truyền không tin cậy phía dưới sẽ quyết định độ phức tạp của giao thức Truyền tin cậy (rdt)

Truyền tin cậy: Bắt đầu như thế nào ?

rdt_send(): Tầng trên gọi, (ví dụ Ứng dụng.). Yêu cầu chuyển dữ liệu tin cậy cho đối tác bên kia

deliver_data(): Rdt gọi để chuyển dữ liệu lên trên



udt_send(): rdt gọi để chuyển packet cho phía nhận qua đường truyền không tin cậy

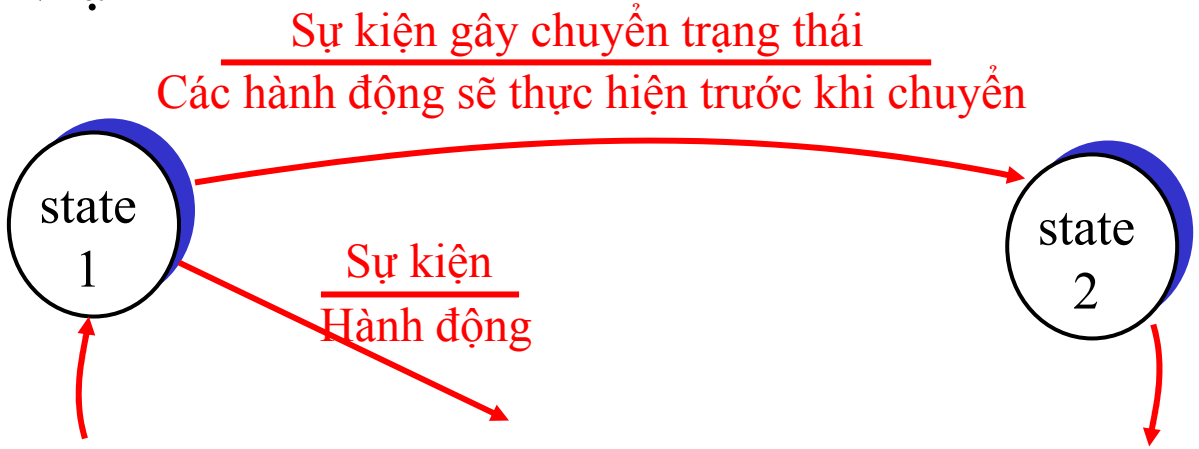
rdt_rcv(): được gọi khi packet đến phía nhận

Truyền tin cậy: Khởi đầu

Chúng ta sẽ :

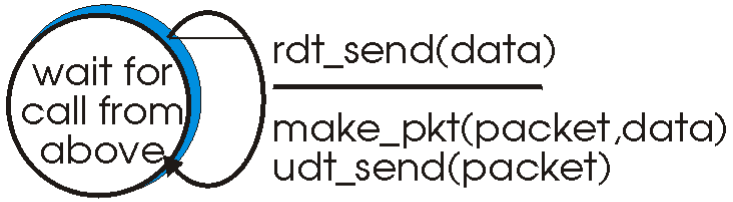
- Phát triển dần dần cả phía Gửi và phía Nhận trong Giao thức Truyền tin cậy (rdt)
- Xét dữ liệu chỉ truyền trên một hướng
  - Nhưng thông tin điều khiển có thể truyền theo cả hai hướng!
- Sử dụng máy Hữu hạn trạng thái (FSM) để miêu tả phía Gửi và phía Nhận

Trạng thái: Chỉ thay đổi sang trạng thái khác khi có sự kiện mang tính duy nhất xảy ra

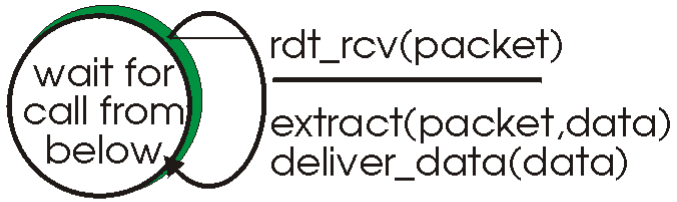


Rdt1.0: Truyền tin cậy qua kênh truyền tin cậy

- Đặc điểm của kênh truyền tin cậy
  - Bit trong gói tin không bị lỗi
  - Gói tin không bị mất
- FSM cho bên Gửi và bên Nhận độc lập nhau:
  - Phía Gửi truyền gói tin qua kênh truyền phía dưới
  - Phía Nhận đọc gói tin từ kênh truyền bên dưới



(a) rdt1.0: sending side

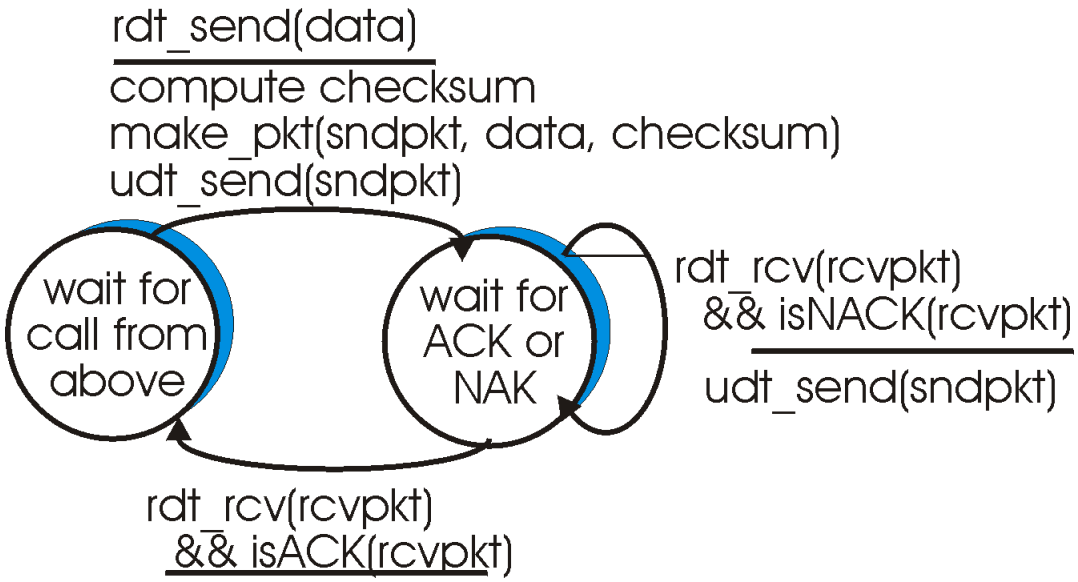


(b) rdt1.0: receiving side

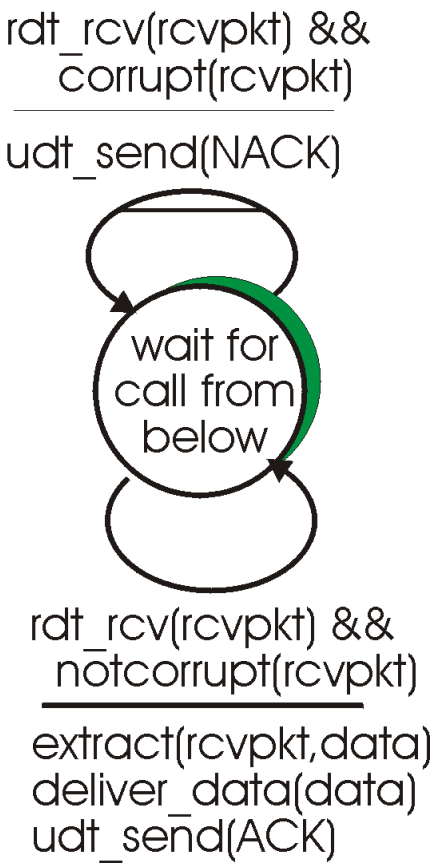
Rdt2.0: Kênh truyền có lỗi bit

- Kênh truyền bên dưới có thể khiến **bit trong gói tin lỗi**
  - Ghi nhớ: checksum có thể được sử dụng để phát hiện lỗi
- ***Vấn đề***: Khắc phục lỗi như thế nào ?
  - ***Biên nhận tích cực (ACK):*** Bên nhận thông báo tường minh cho bên gửi mình nhận đúng và chính xác gói tin.
  - ***Biên nhận tiêu cực (NAK):*** Bên nhận thông báo tường minh cho bên gửi gói tin mình nhận có lỗi.
  - Phía gửi gửi lại gói tin khi nhận được NAK
  - Ví dụ nào trong cuộc sống sử dụng ACK, NAK?
- Bổ sung thêm cơ chế mới trong **`rdt2.0`** :
  - Phát hiện lỗi
  - Phản hồi tường minh: thông điệp phản hồi (ACK,NAK) ***Nhận → Gửi***

rdt2.0: Đặc tả FSM

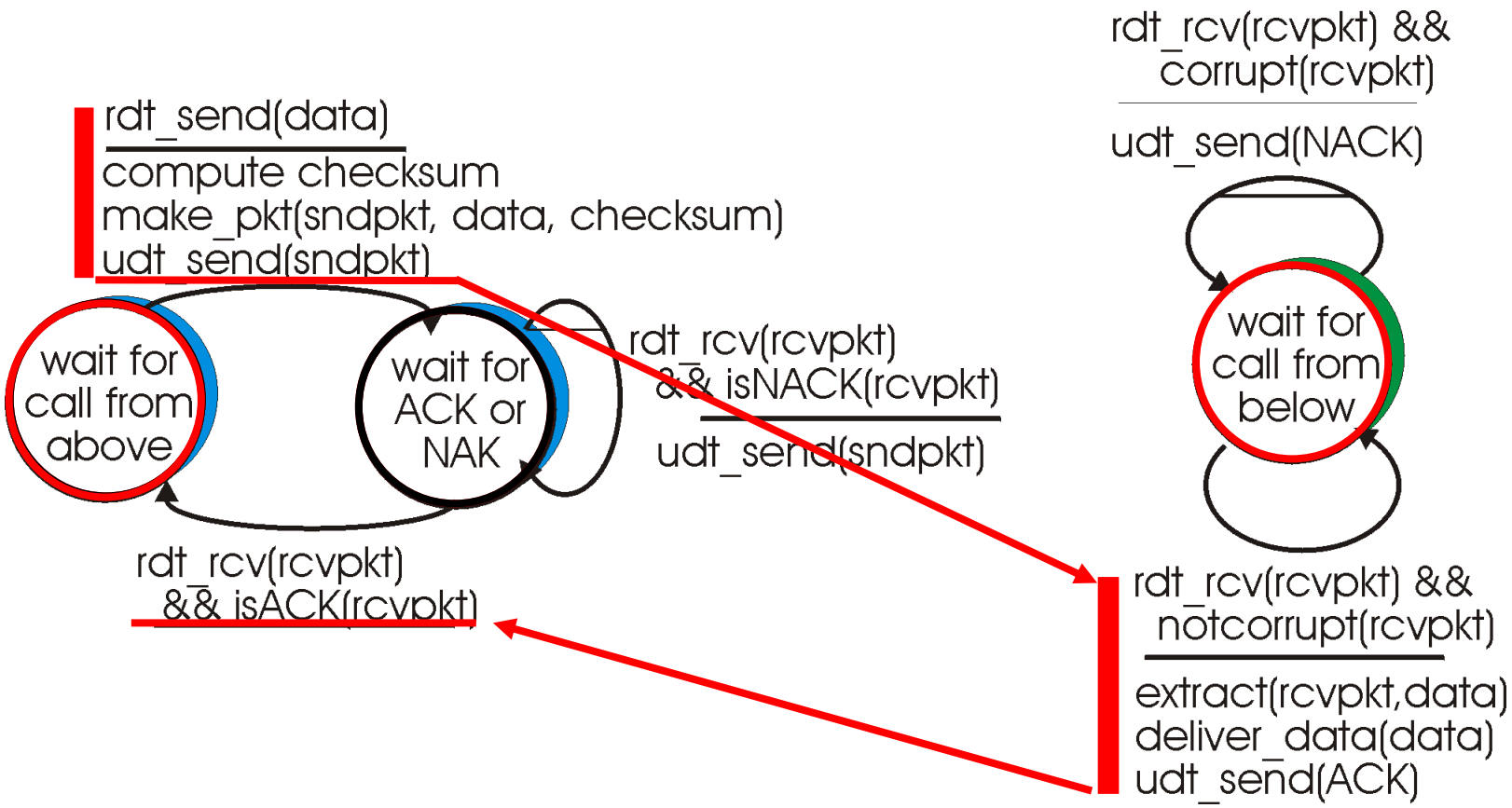


FSM Gửi



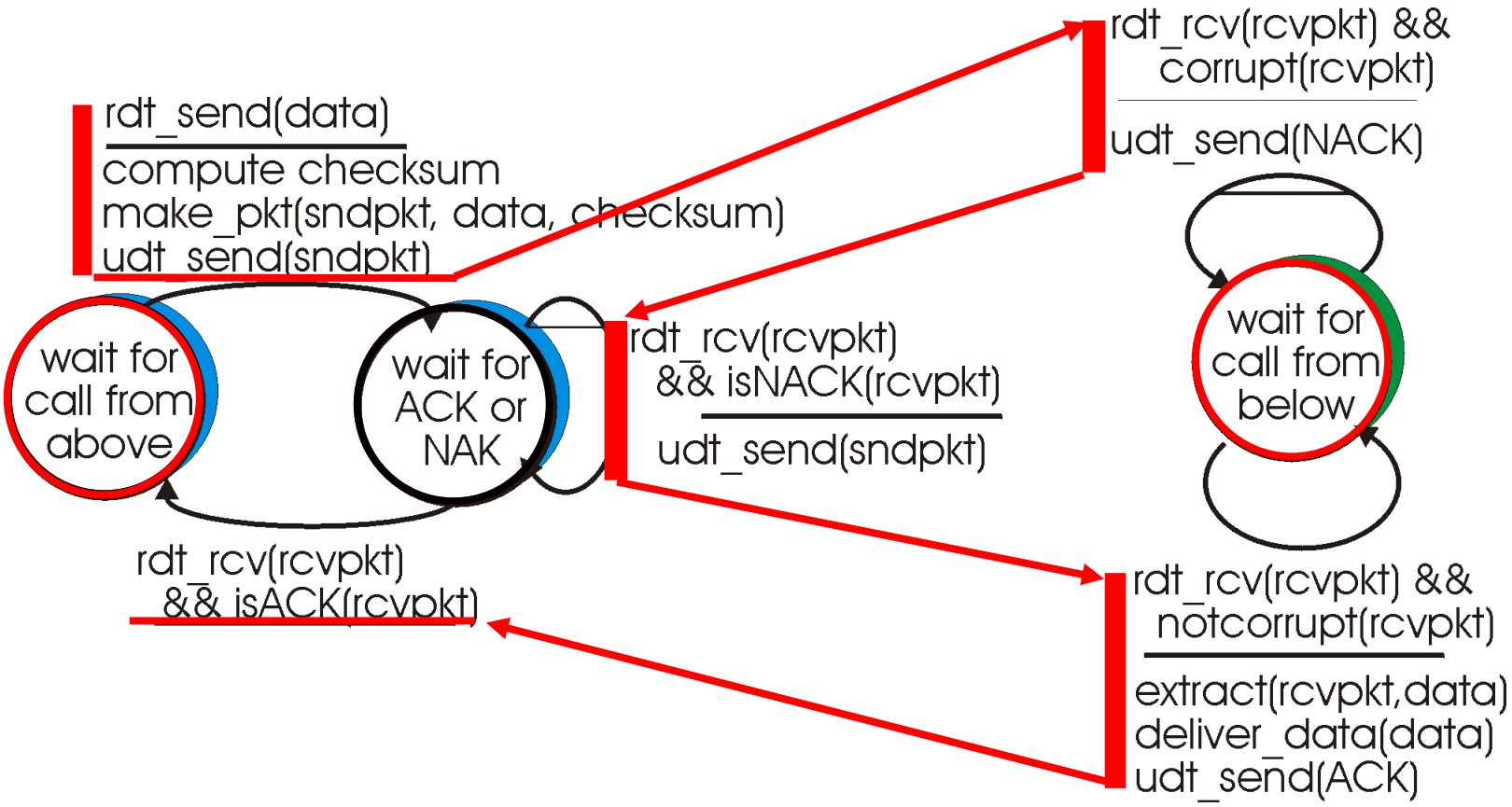
FSM Nhận

rdt2.0: Ví dụ không lỗi



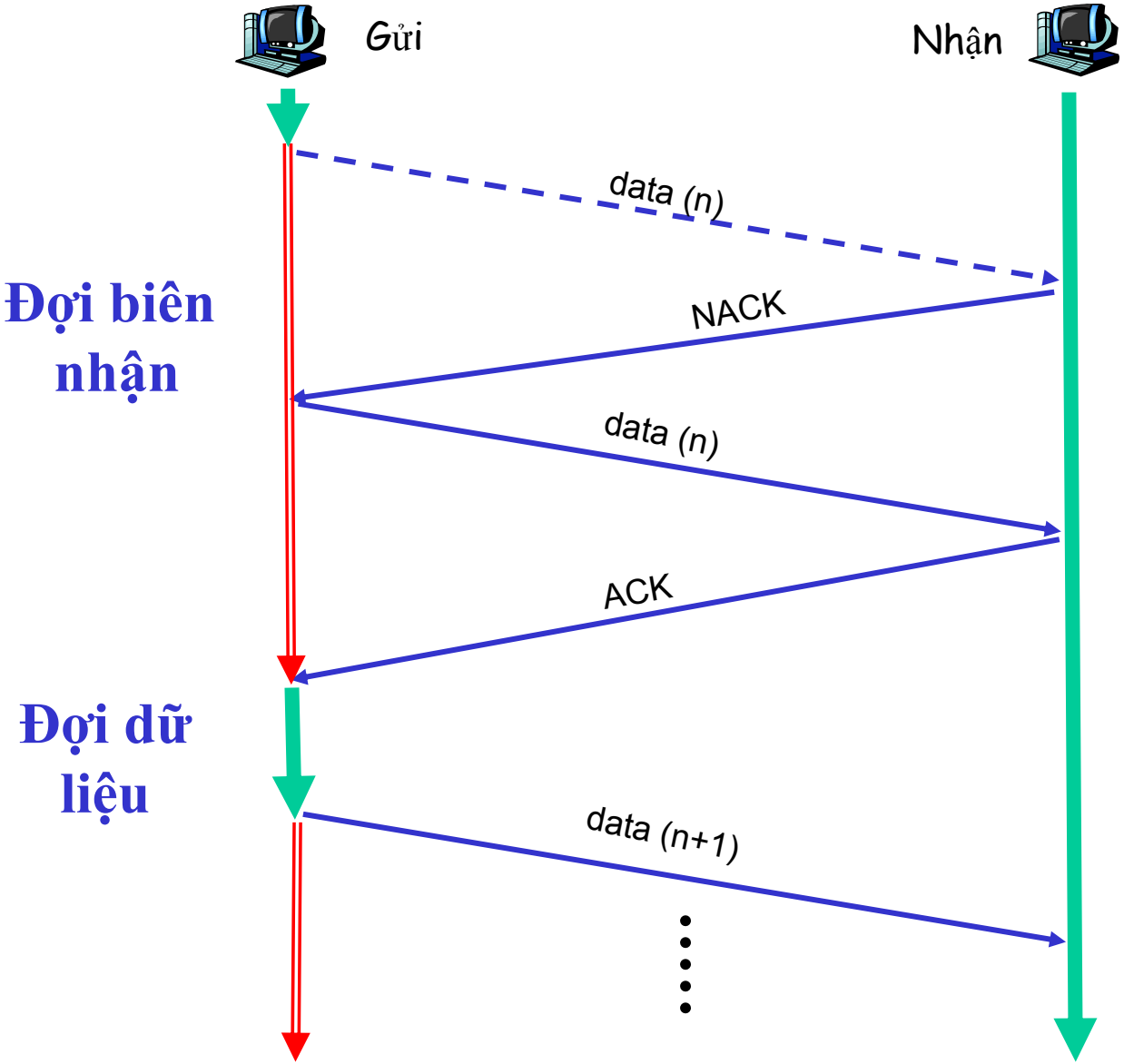
FSM Gửi **FSM Nhận**

rdt2.0: Ví dụ có lỗi



FSM Gửi **FSM Nhận**

Tổng quan về rdt2.0



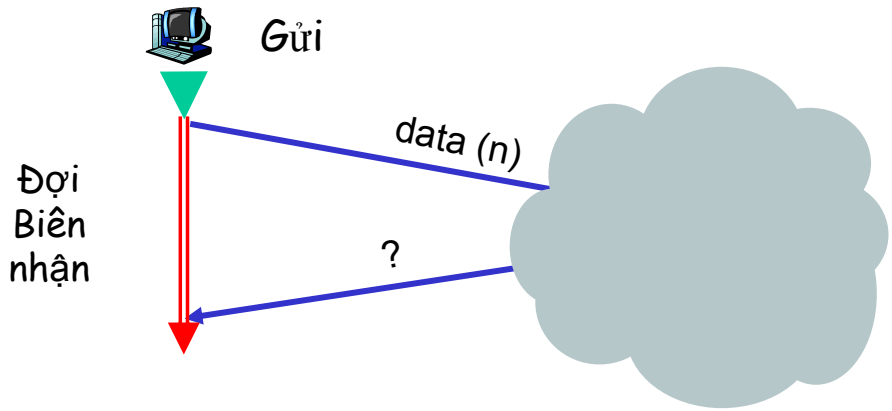
Đợi biên nhận

Đợi dữ liệu

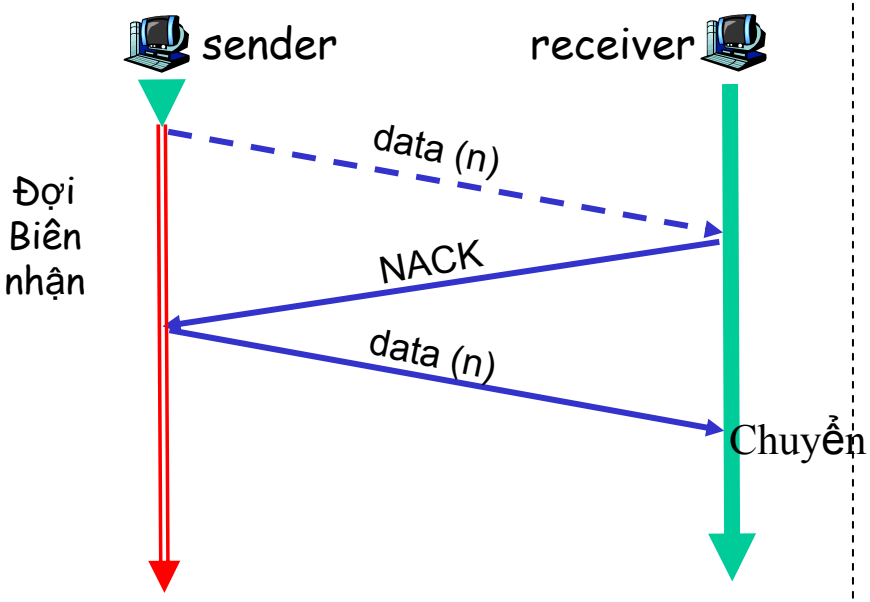
rdt2.0 chưa hoàn chỉnh !

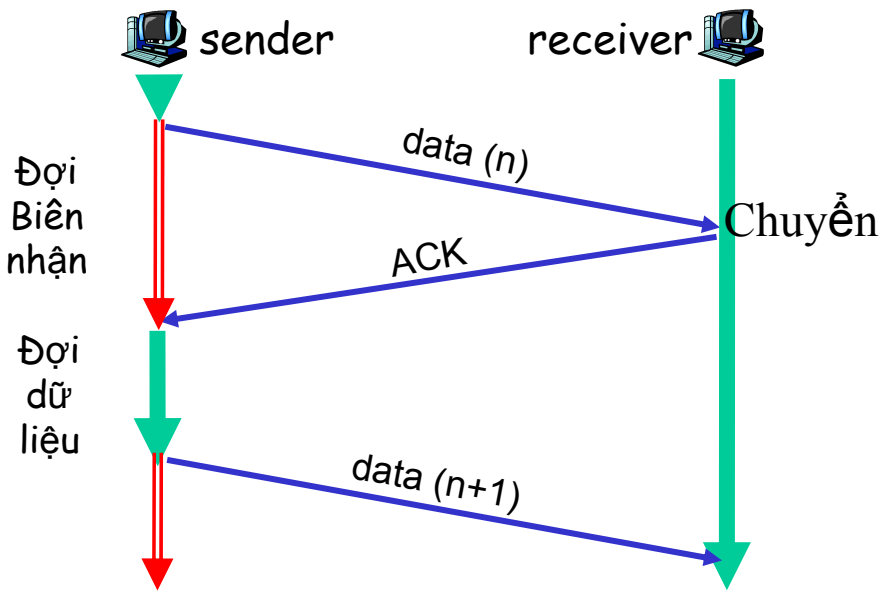
Chuyện gì xảy ra nếu ACK/NAK bị lỗi?

- Mặc dù bên Gửi nhận được phản hồi, nhưng không biết chuyện gì xảy ra ở phía Nhận !



Có hai trường hợp





rdt2.0 Có lỗi nghiêm trọng !

Như vậy nếu gói tin ACK/NAK bị lỗi

- Phía Gửi không biết gì về trạng thái phía Nhận !
- Nếu truyền lại: Dữ liệu có thể bị *trùng lặp*

Phải làm sao ?

- ***Phía gửi lại biên nhận cho phản hồi từ phía nhận?*** Nếu chính gói phản hồi bị mất thì sao ?
- ***Truyền lại ?*** Có thể truyền lại gói tin đã được nhận đúng !

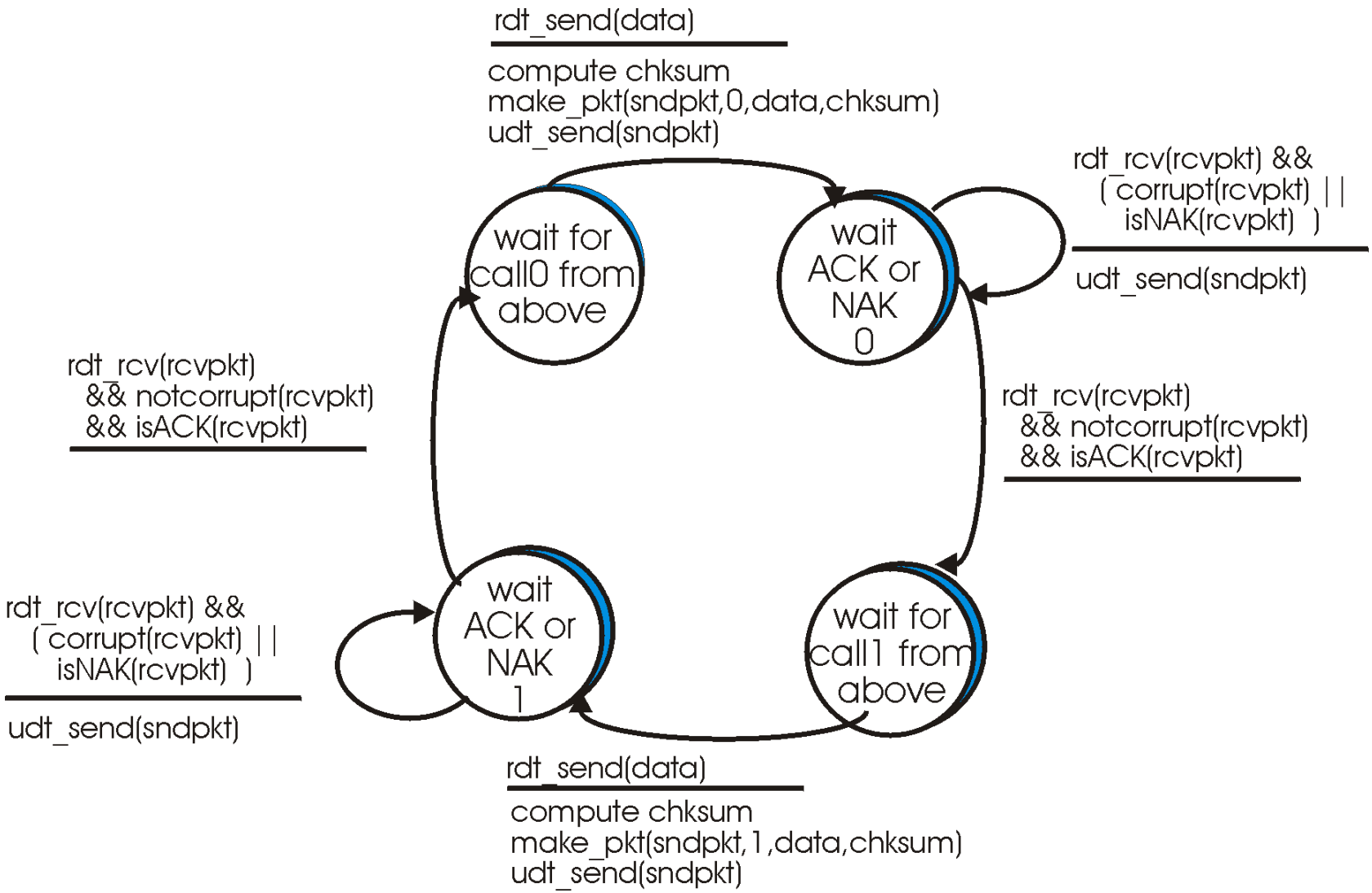
Xử lý trùng lặp ?

- Phía gửi đánh ***Số thứ tự*** cho mỗi gói tin gửi đi.
- Nếu gói tin phản hồi bị lỗi: phía Gửi truyền lại gói tin.
- Phía Nhận loại bỏ các gói tin trùng lặp (không chuyển lên trên)

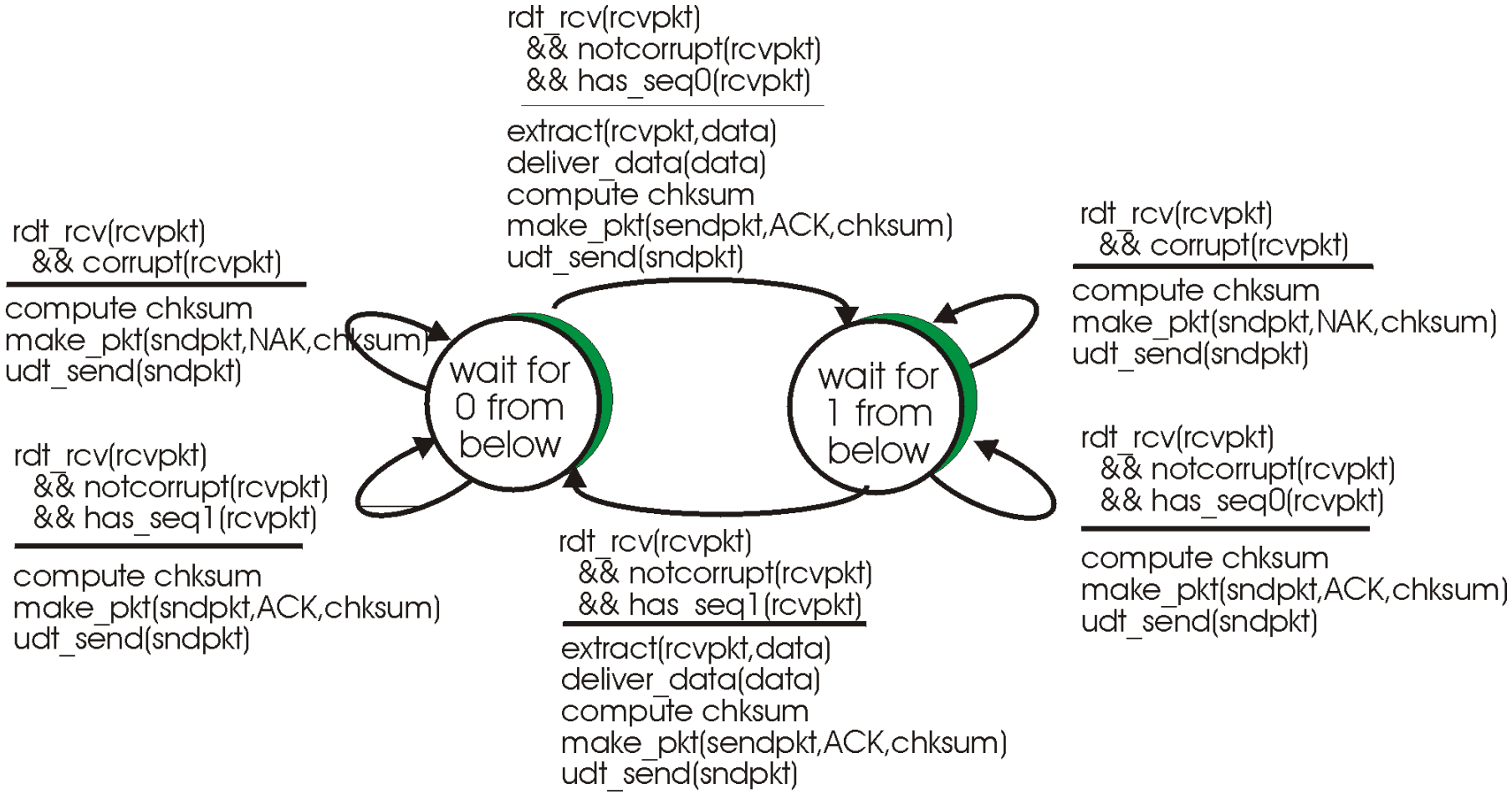
Dừng và Chờ

*Phía Gửi gửi một gói tin, **Dừng** lại và và **Chờ** phía nhận phản hồi.*

rdt2.1: Khắc phục Phản hồi bị lỗi (bên Gửi)



rdt2.1: Khắc phục Phản hồi bị lỗi (bên Nhận)



rdt2.1: Thảo luận ?

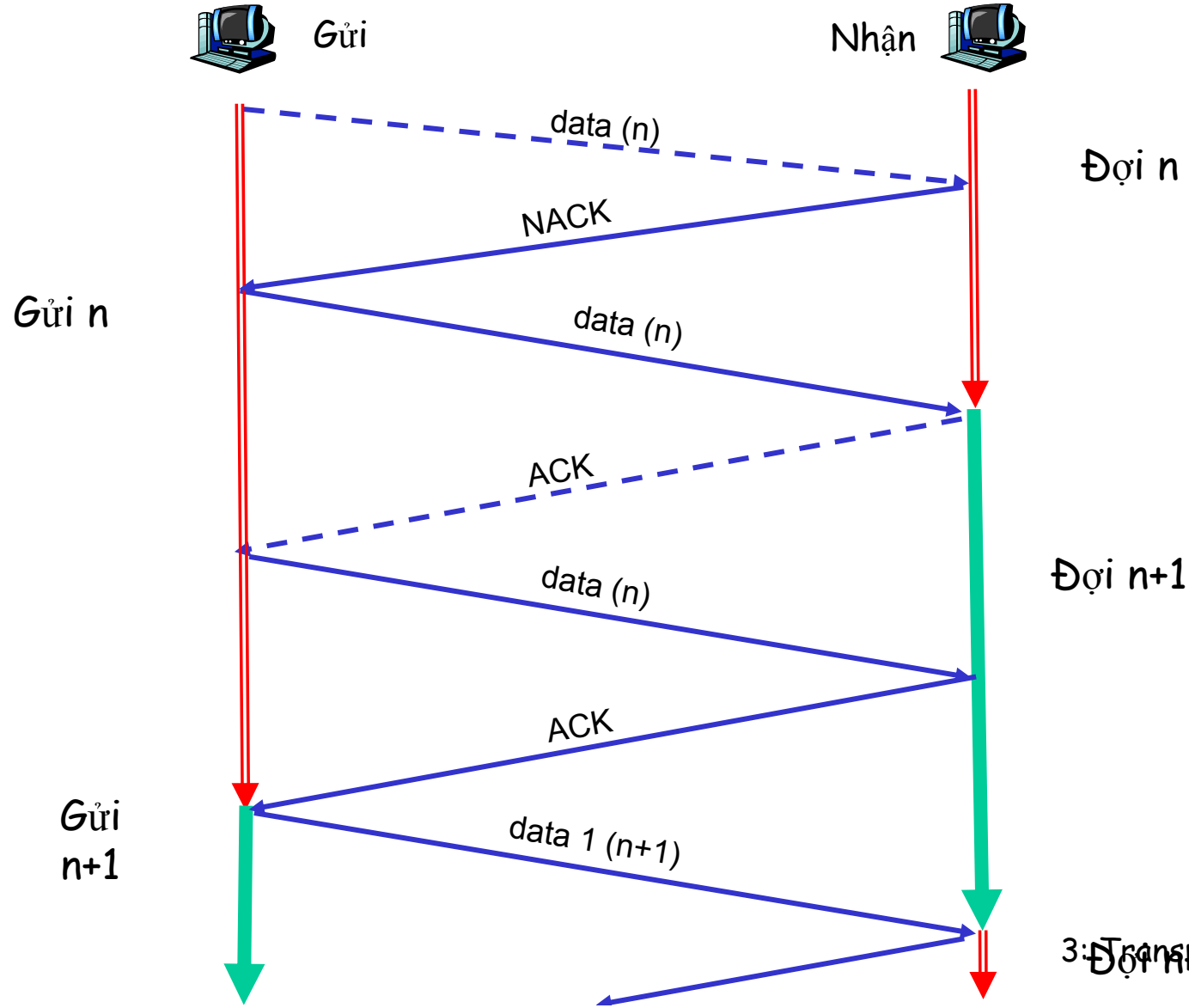
Phía Gửi:

- Đánh STT cho gói tin
- Sử dụng hai STT là (0,1) có đủ không ?
- Phải xác định được gói phản hồi có lỗi không
- Số trạng thái tăng gấp đôi
  - Trạng thái phải "ghi nhớ" mình đang gửi đi gói tin có STT là 0 hay 1.

Phía Nhận:

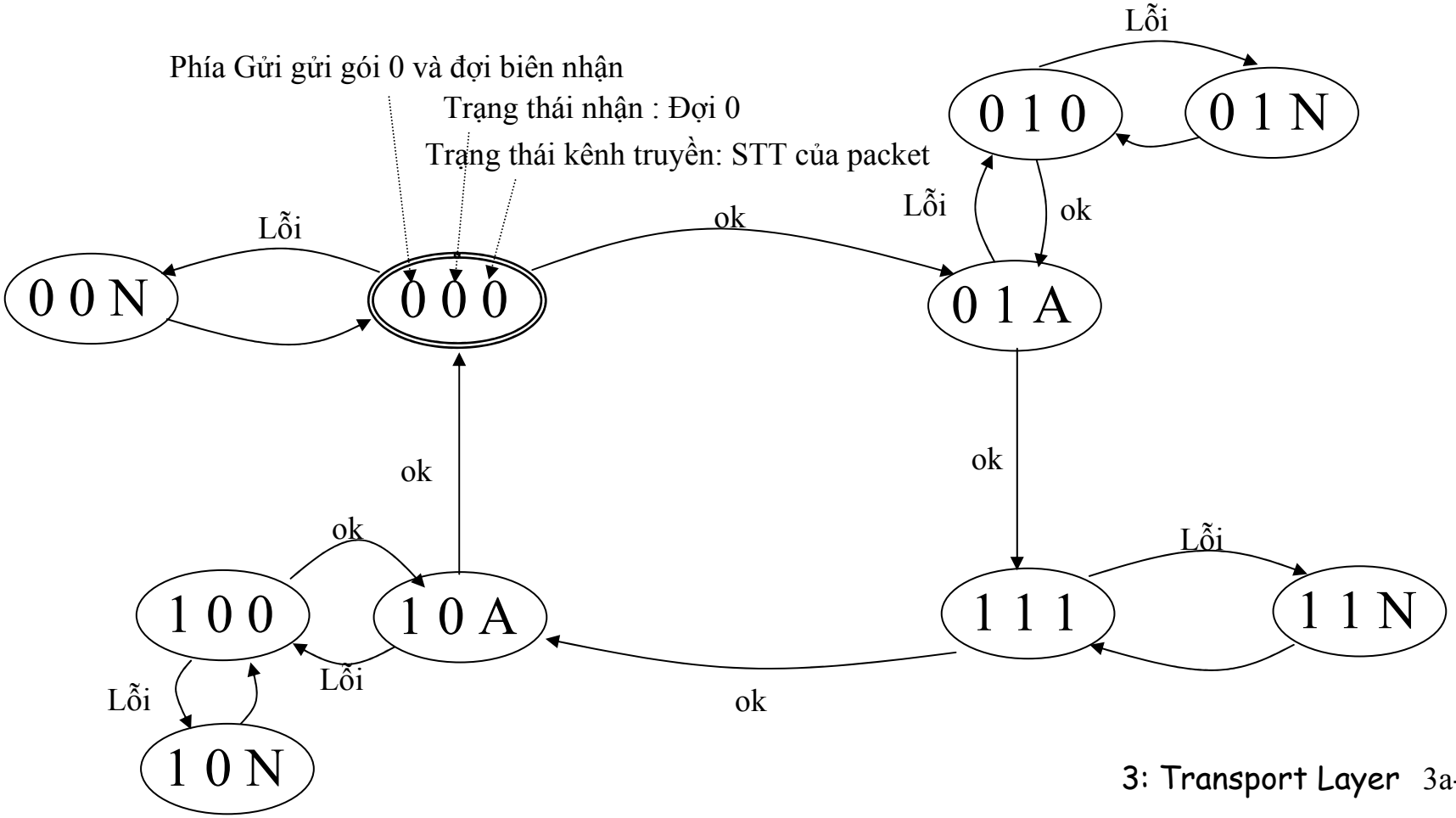
- Phải kiểm tra xem mình có nhận dữ liệu trùng lặp không
  - Trạng thái xác định mình nhận đang mong muốn nhận gói 0 hay gói 1.
- **Chú ý**: Phía Nhận không thể xác định phía Gửi có nhận đúng được thông điệp phản hồi của mình hay không.

Tổng quan về rdt2.1



Chứng minh tính đúng đắn của rdt2.1

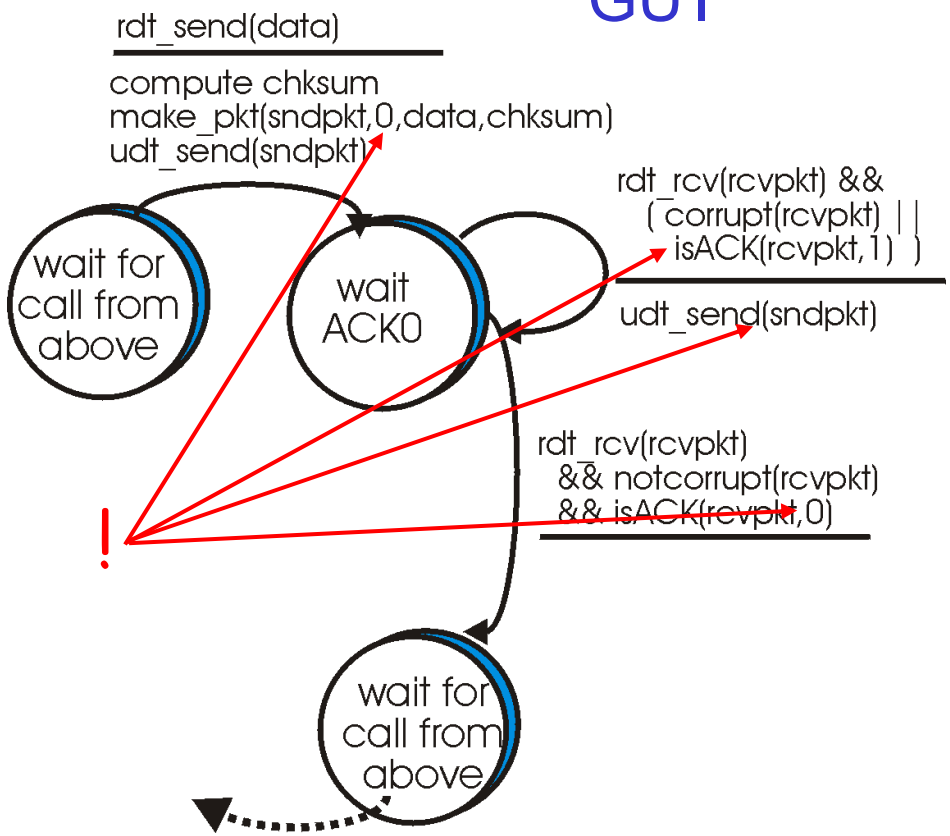
- Kết hợp trạng thái Bên Gửi và Kênh truyền.
- Giả sử luôn luôn có dữ liệu để gửi



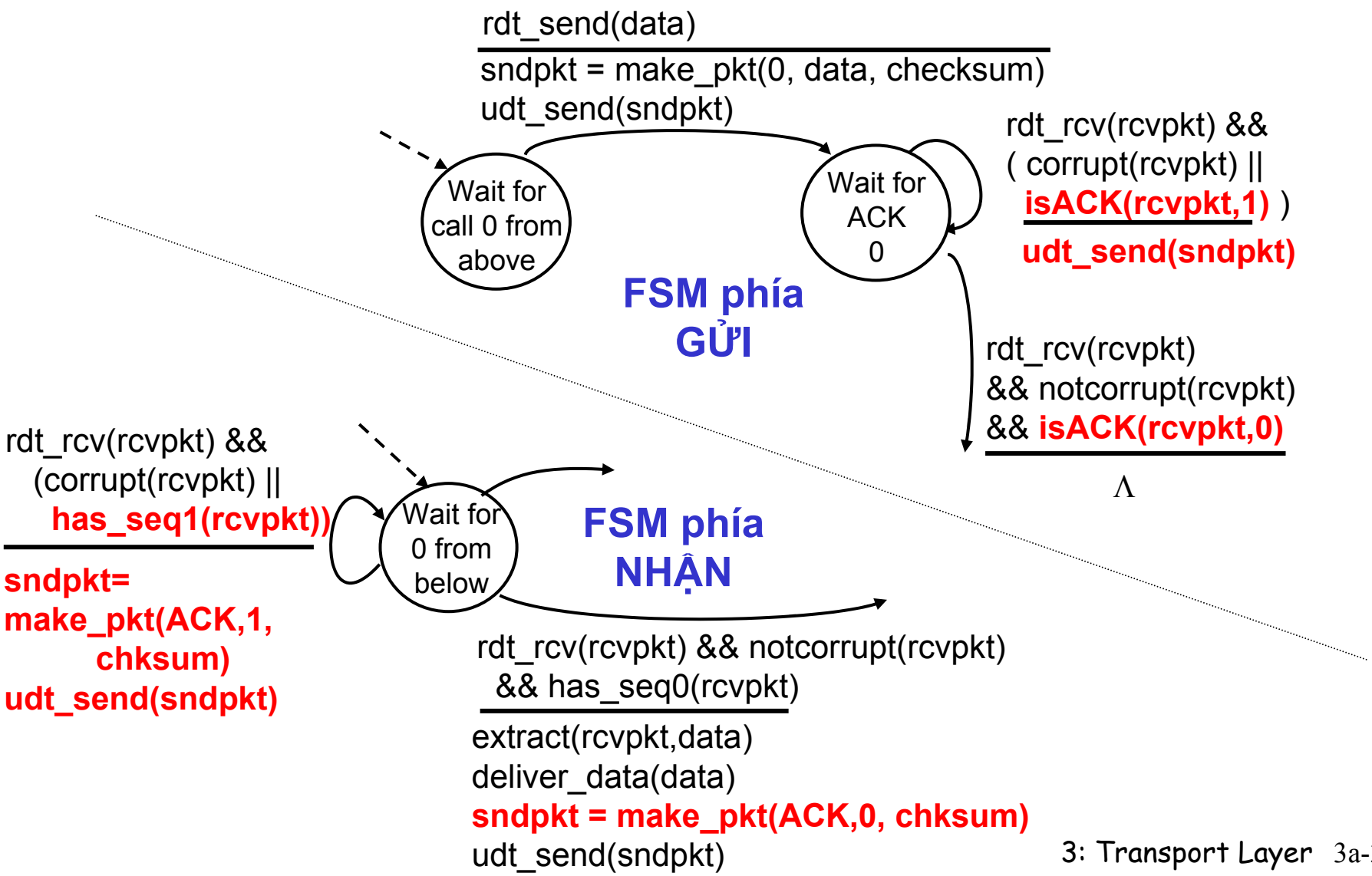
rdt2.2: Giao thức không sử dụng NAK

- Giống hệt rdt2.1, nhưng không dùng NAK
- Thay vì sử dụng NAK, phía Gửi sẽ biên nhận gói cuối cùng nhận đúng.
  - Phía Nhận phải biên nhận rõ ràng cho gói tin nào
- Ở phía Nhận, nhận được ACK trùng lặp tương đương nhận được NAK : *truyền lại.*

FSM GỬI



rdt2.2: Tách biệt phía Gửi / Nhận



rdt3.0: Kênh truyền Lỗi bit và Mất gói tin

Giả định mới: Gói tin (dữ liệu và phản hồi) có thể bị mất trên đường truyền

- checksum, STT, phản hồi, truyền lại : ***chưa đủ***.

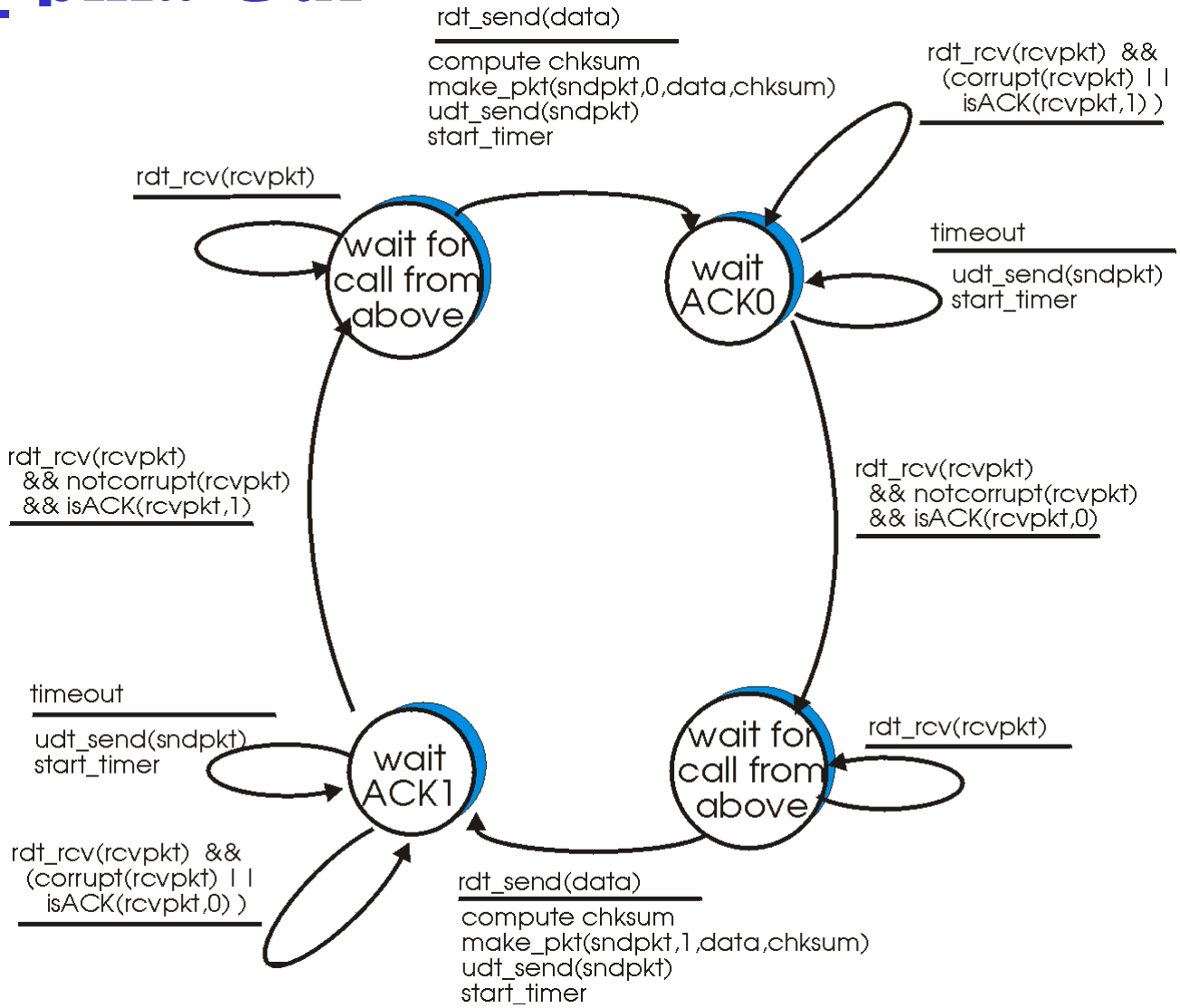
? Làm thế nào để xử lý Lỗi?

- Phía gửi sẽ đợi cho đến khi chắc chắn gói tin bị mất, rồi truyền lại.
- Nhược điểm của phương pháp này là gì ?

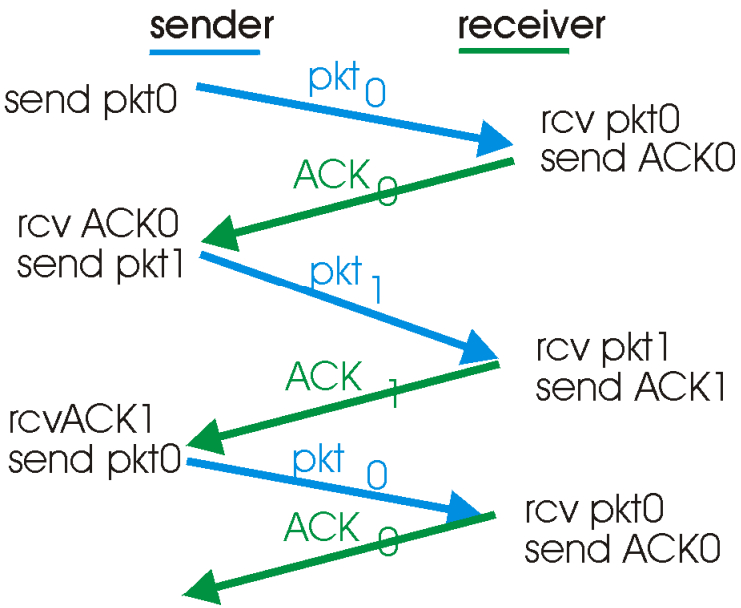
Giải pháp: Phía gửi đợi phản hồi trong một khoảng thời gian "hợp lý"

- Truyền lại nếu trong khoảng thời gian này không nhận được ACK.
- Nếu gói tin (hay ACK) chỉ bị trễ (không mất):
  - Truyền lại có thể trùng lặp, nhưng STT có khả năng giải quyết vấn đề này.
  - Bên nhận phải phản hồi tường minh gói nào nhận đúng
- Cần bộ định thời đếm ngược.

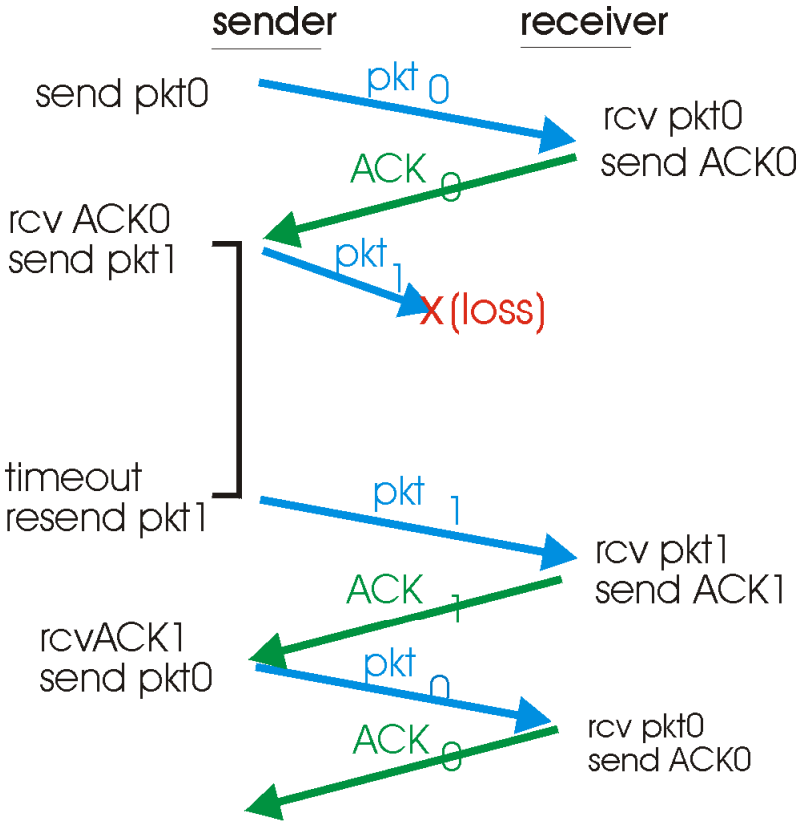
rdt3.0 phía Gửi



rdt3.0: Ví dụ

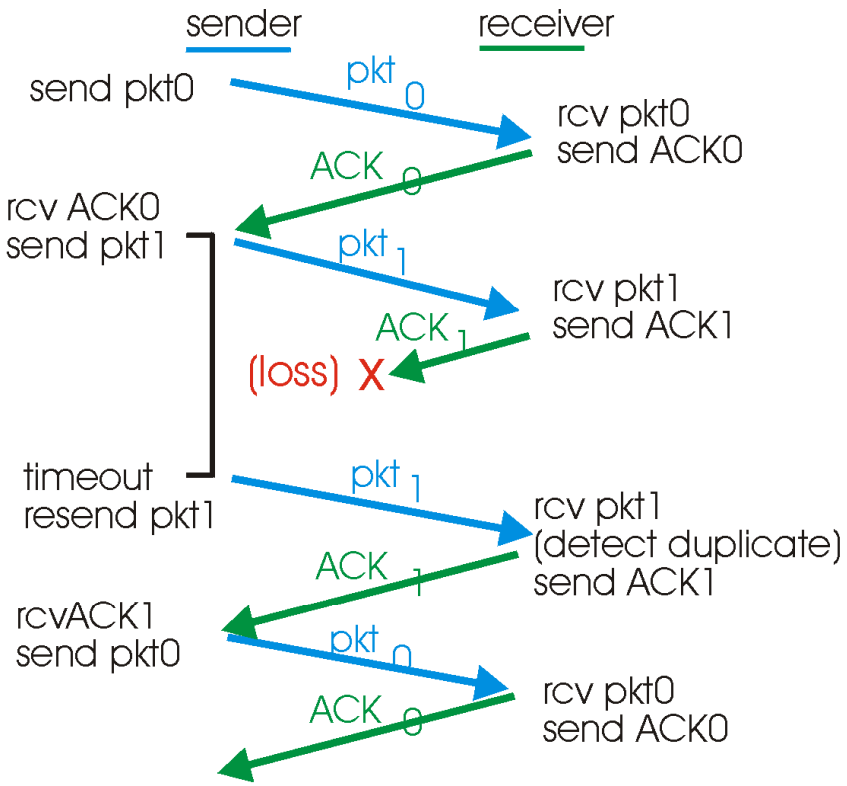


(a) operation with no loss

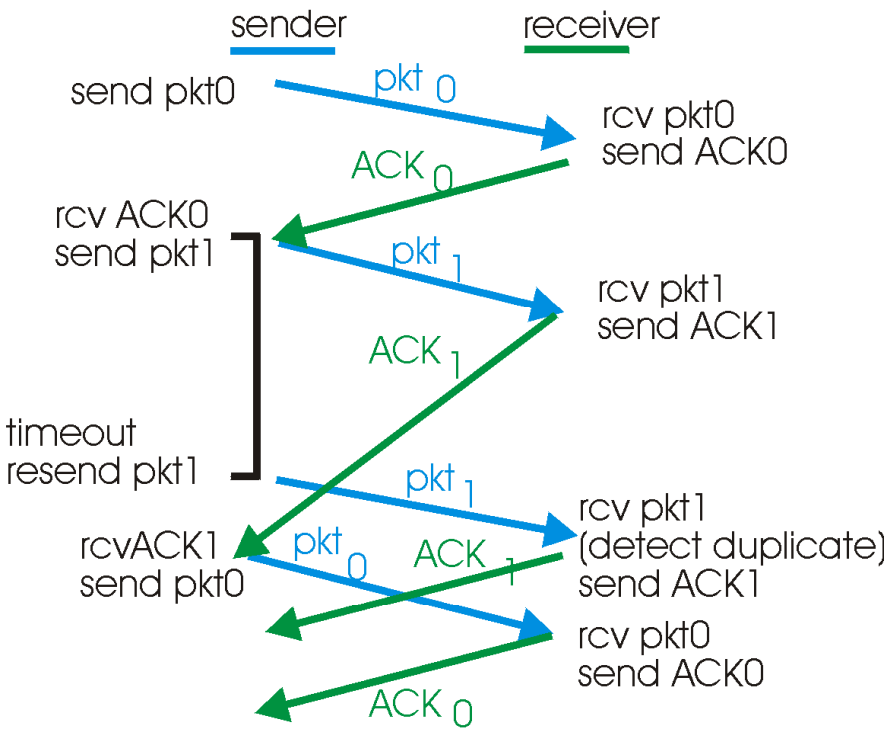


(b) lost packet

rdt3.0 : Ví dụ

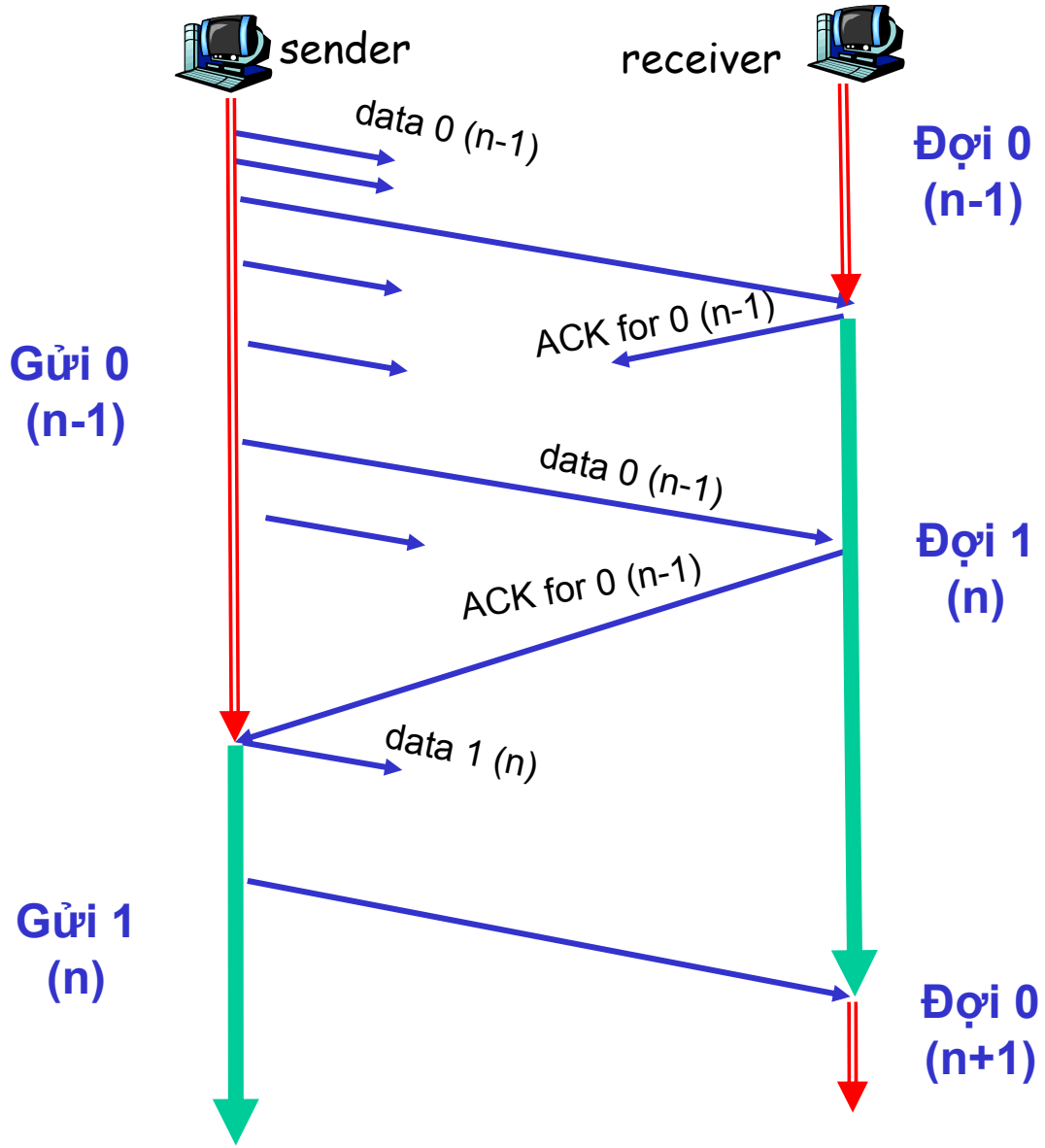


(c) lost ACK



(d) premature timeout

rdt3.0



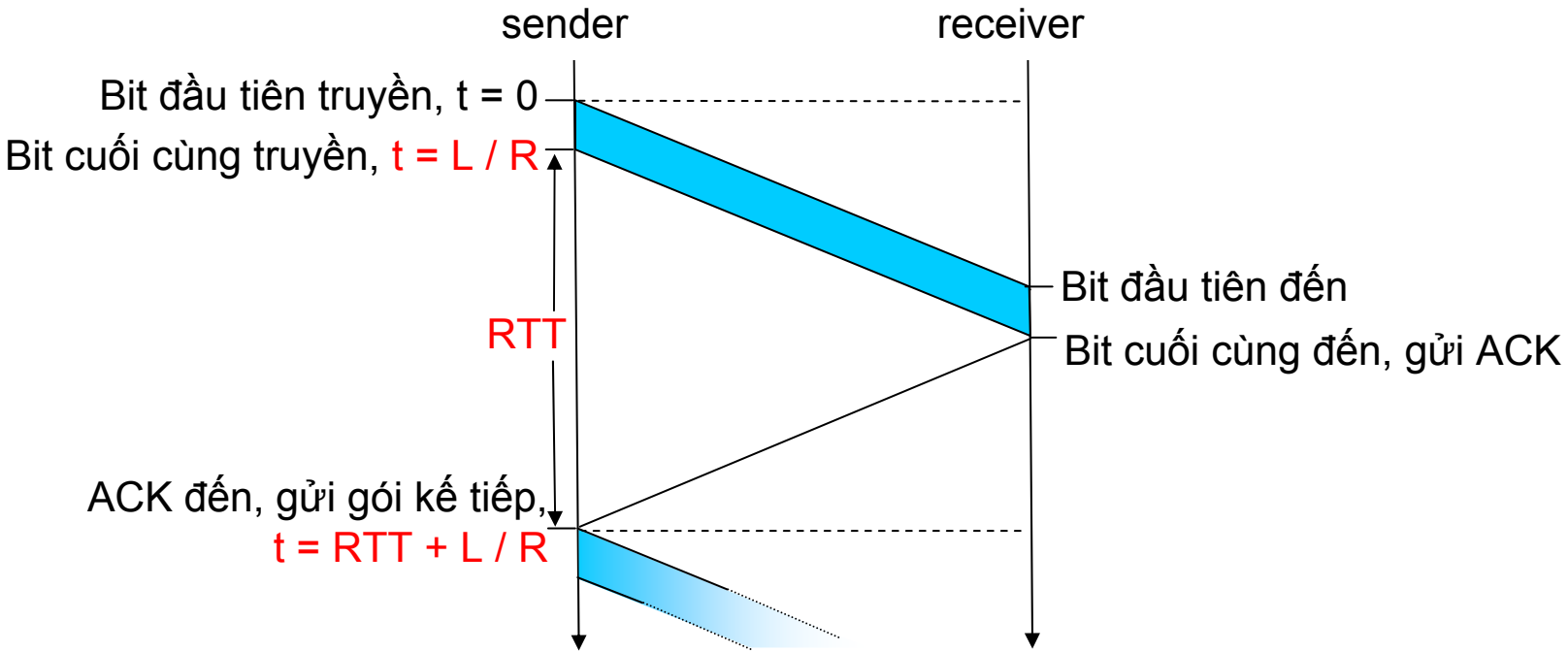
Trạng thái nhất quán

Khi trạng thái bên nhận là ***Đợi n***, trạng thái bên gửi hoặc là ***Gửi n-1*** hoặc ***Gửi n***.

Khi trạng thái bên gửi là ***Gửi n***, trạng thái bên nhận hoặc là ***Đợi n*** hoặc ***Đợi n+1***

rdt3.0: Stop-and-Wait Operation

- rdt3.0 chạy tốt, nhưng ***Hiệu suất*** rất thấp
- Ví dụ: Đường truyền 1 Gbps link, Đỗ trễ 15 ms, Gói tin 1KB :



$$U_{sender} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.027\%$$

Hiệu suất của rdt3.0

$$T_{transmit} = \frac{8kb/pkt}{10^{**}9\ b/sec} = 8\ microsec$$

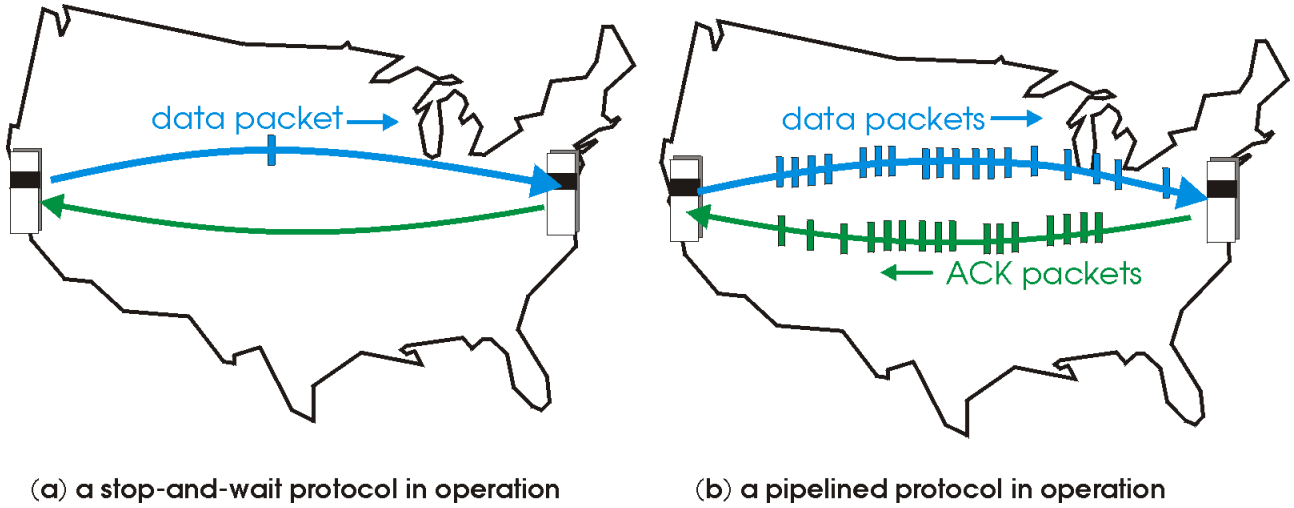
$$\text{Hiệu suất} = U = \frac{\text{Tỷ lệ thời gian phía Gửi thực sự gửi}}{} = \frac{8\ microsec}{30.016\ msec} = 0.00027$$

- Gói tin 1KB pkt truyền trong 30 msec -> thông lượng : 33kB/sec
- Giao thức tầng network gây ra hạn chế về Hiệu suất !

Giao thức kiểu Đường ống

Đường ống: phía Gửi đồng thời gửi nhiều gói tin mà không cần biên nhận.

- Tăng khoảng Số thứ tự.
- Cần bộ đệm dữ liệu tại bên Gửi/ Nhận.

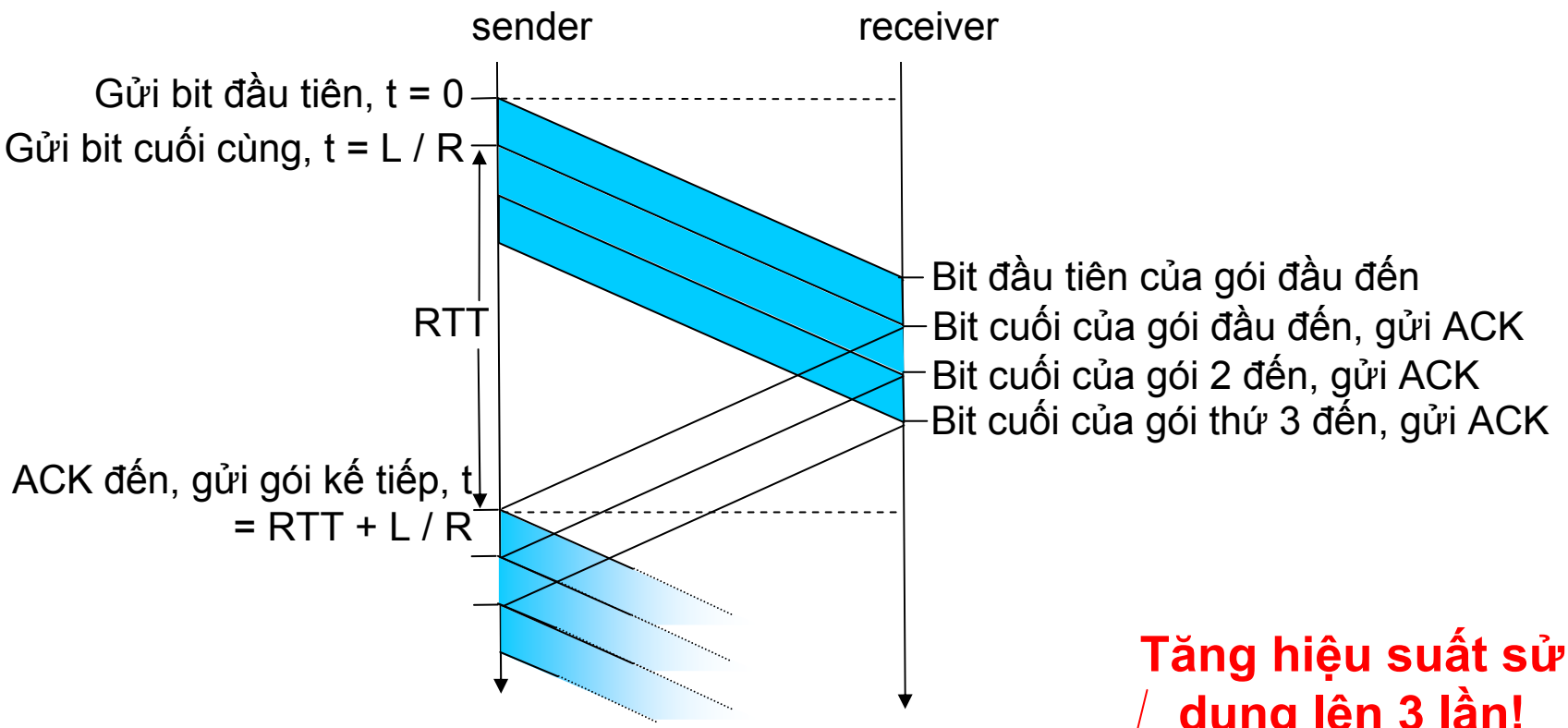


(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

- Hai lớp giao thức chính: ***Go-Back-N, Selective Repeat***

Đường ống: Tăng Hiệu suất sử dụng



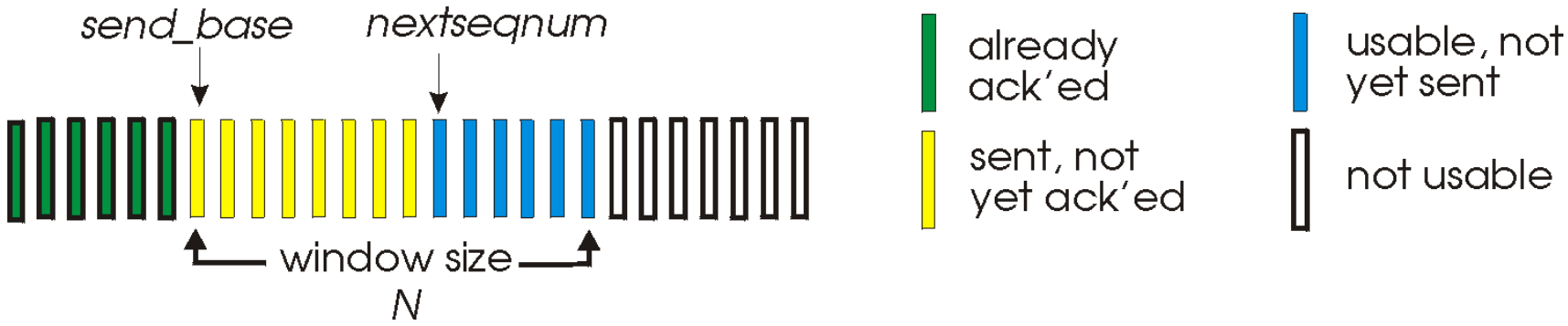
Tăng hiệu suất sử dụng lên 3 lần!

$$U_{sender} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$

Go-Back-N

Phía Gửi:

- Trường STT trong tiêu đề : k bit
- "cửa sổ" : số lượng cực đại các gói tin liên tiếp gửi mà chưa cần biên nhận.



ACK(n): Biên nhận tất cả các gói tin có STT ≤ n - "Biên nhận tích lũy"

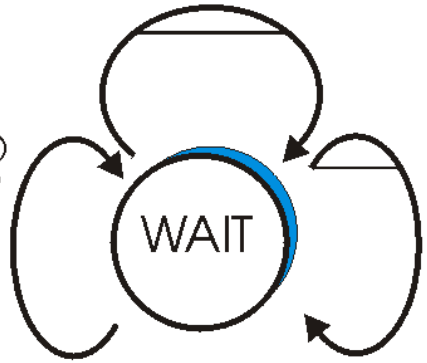
- Có thể bỏ qua các ACK trùng lặp (xem bên Nhận)

- Bộ định thời cho các gói tin gửi đi nhưng chưa biên nhận
- *timeout(n):* truyền lại gói n và tất cả các gói có STT cao hơn trong cửa sổ.

GBN: FSM mở rộng của phía Gửi

rdt_send(data)

```
if (nextseqnum < base+N) {
   compute chksum
   make_pkt(sndpkt(nextseqnum)),nextseqnum,data,chksum)
   udt_send(sndpkt(nextseqnum))
   if (base == nextseqnum)
     start_timer
   nextseqnum = nextseqnum + 1
   }
else
  refuse_data(data)
```



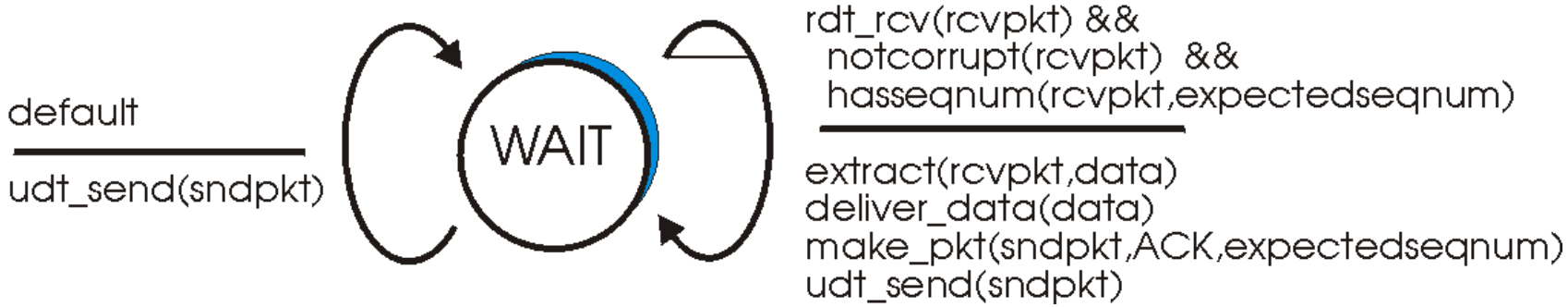
rdt_rcv(rcv_pkt) && notcorrupt(rcvpkt)

```
base = getacknum(rvcpkt)+1
if (base == nextseqnum)
   stop_timer
 else
   start_timer
```

timeout

```
start_timer
udt_send(sndpkt(base))
udt_send(sndpkt(base+1)
……
udt_send(sndpkt(nextseqnum-1))
```

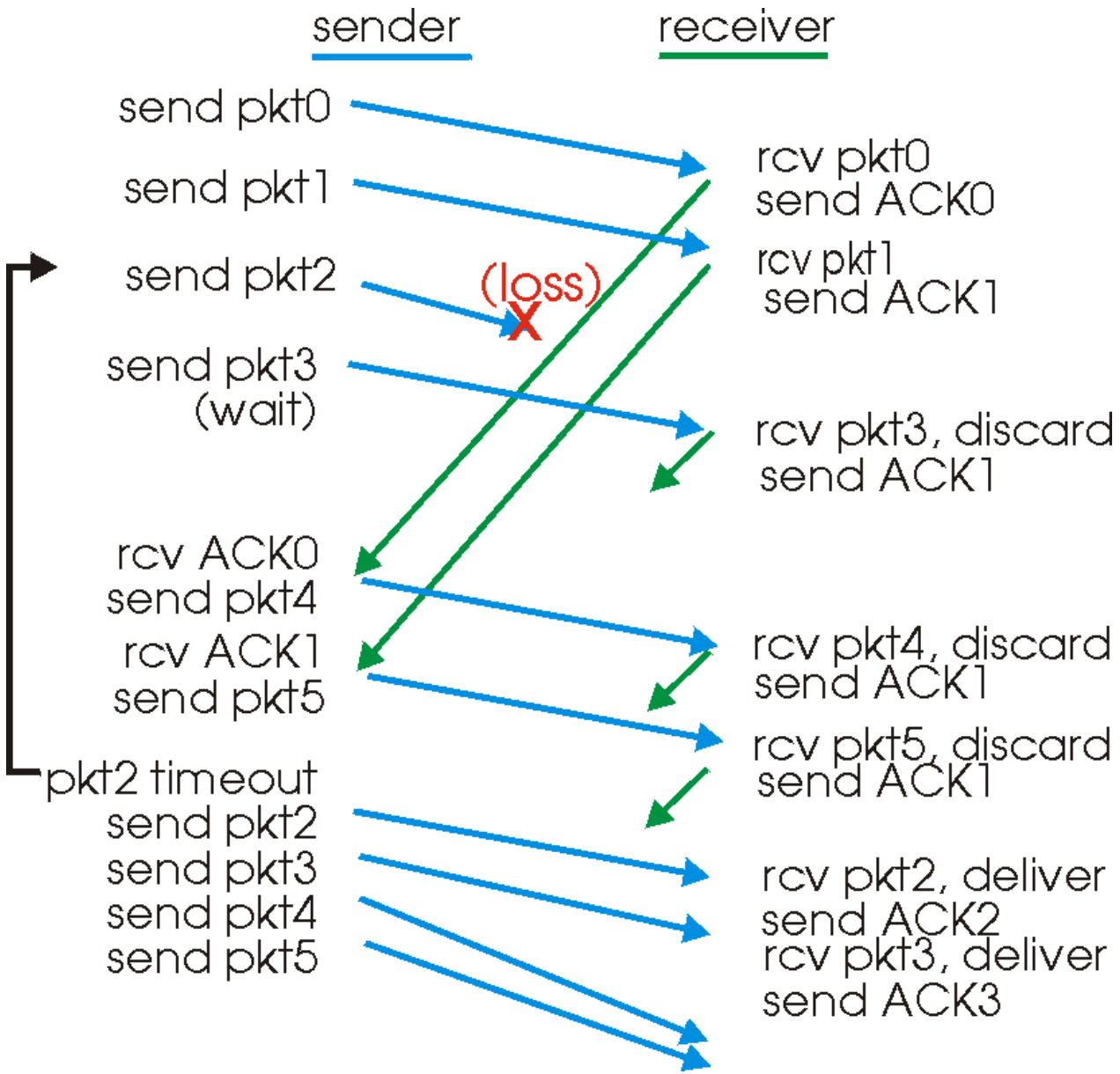
GBN: FSM mở rộng của phía Nhận



phía Nhận rất đơn giản:

- Chính sách biên nhận: Luôn luôn biên nhận cho gói tin nhận đúng. Biên nhận gói tin theo ***đúng thứ tự*** có STT lớn nhất.
  - Có thể tạo ra Biên nhận trùng lặp.
  - Phải ghi nhớ giá trị mình muốn nhận (**`expectedseqnum`**)
- Gói tin không đúng STT:
  - Loại bỏ (không lưu lại) ->
  - Biên nhận STT gói tin đúng thứ tự lớn nhất

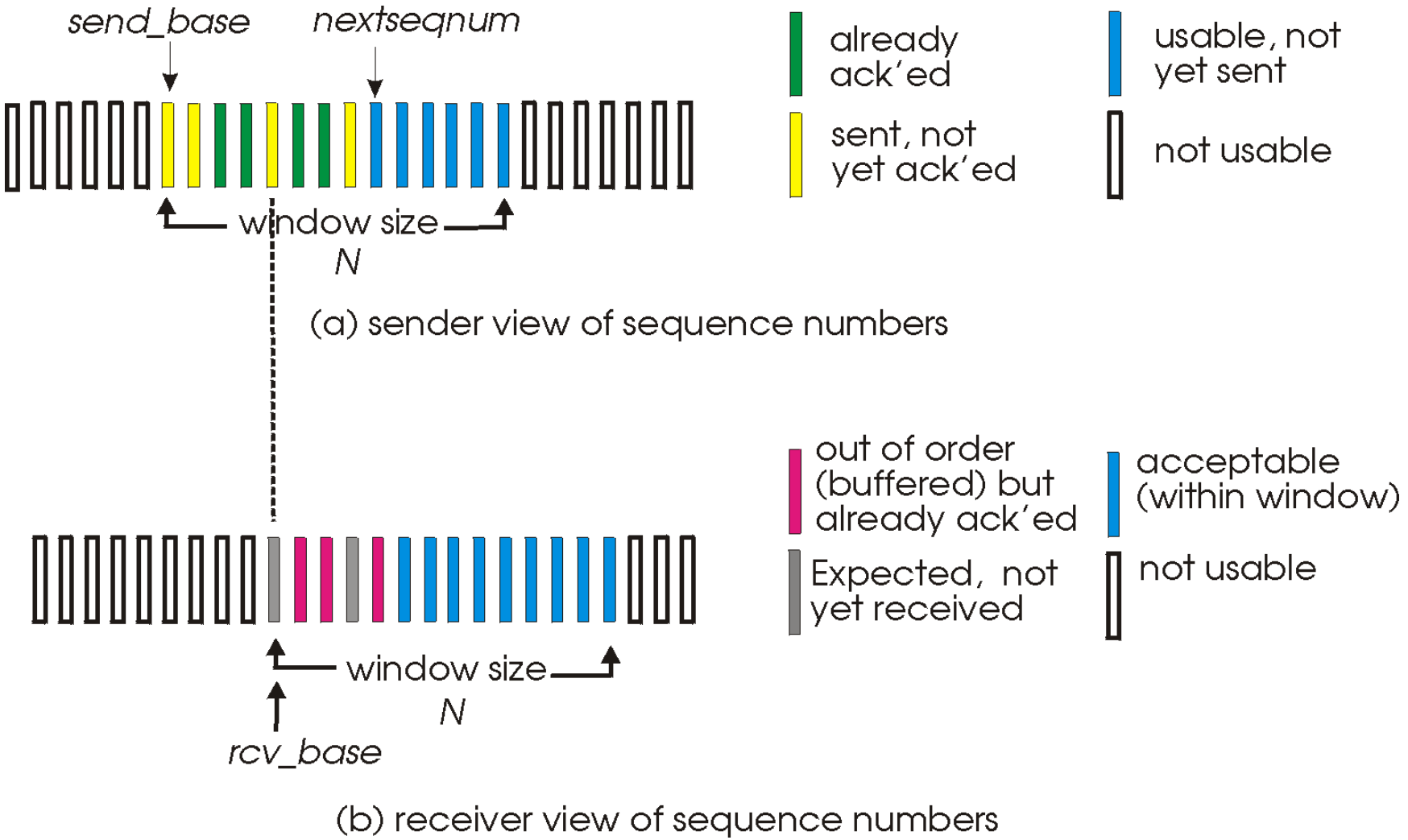
Ví dụ về GBN


sender
receiver
send pkt0
send pkt1
send pkt2
(loss)
send pkt3
(wait)
rcv pkt0
send ACK0
rcv pkt1
send ACK1
rcv pkt3, discard
send ACK1
rcv ACK0
send pkt4
rcv ACK1
send pkt5
rcv pkt4, discard
send ACK1
rcv pkt5, discard
send ACK1
pkt2 timeout
send pkt2
send pkt3
send pkt4
send pkt5
rcv pkt2, deliver
send ACK2
rcv pkt3, deliver
send ACK3

Selective Repeat (Lặp lại có Lựa chọn)

- **Phía Nhận** biên nhận ***riêng lẻ*** từng gói tin nhận đúng
  - Có thể lưu lại tạm thời các gói tin không theo đúng STT để sau này dùng lại.
- **Phía Gửi** chỉ gửi lại các gói tin chưa có biên nhận
  - phía Gửi : mỗi gói tin có bộ định thời riêng
- "Cửa sổ" phía Gửi
  - STT liên tiếp có kích thước N
  - Hạn chế số lượng gói dữ liệu đã gửi đi nhưng chưa biên nhận.

Selective Repeat: Cửa sổ phía Gửi và Nhận



Selective Repeat

phía Gửi

Dữ liệu từ bên trên xuống :

- Nếu còn có khả năng gửi, gửi tiếp dữ liệu.

timeout(n):

- Gửi lại gói n, khởi tạo lại timer

ACK(n) ∈ [sendbase,sendbase+N]:

- Đánh dấu gói n đã nhận đúng
- Nếu n là STT bé nhất chưa biên nhận, dịch chuyển cửa sổ lên STT gói tin bé nhất chưa biên nhận.

phía Nhận

pkt n ∈ [rcvbase, rcvbase+N-1]

- Gửi ACK(n)
- Không đúng thứ tự: lưu tạm.
- Đúng thứ tự: chuyển tất cả dữ liệu đã nhận đúng thứ tự lên tầng ứng dụng bên trên.

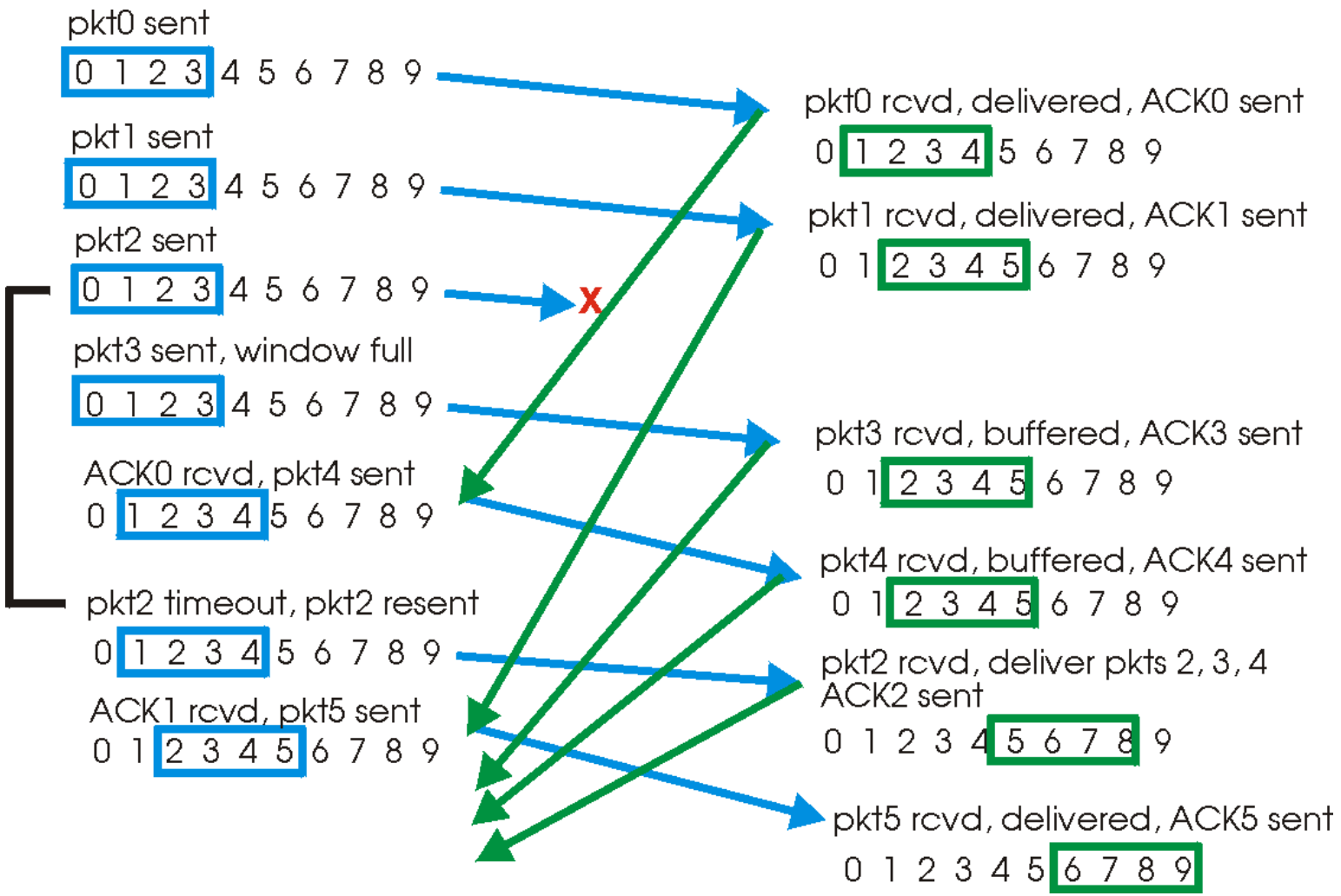
pkt n ∈ [rcvbase-N,rcvbase-1]

- ACK(n)

Các trường hợp:

- Bỏ qua

Selective Repeat : Ví dụ



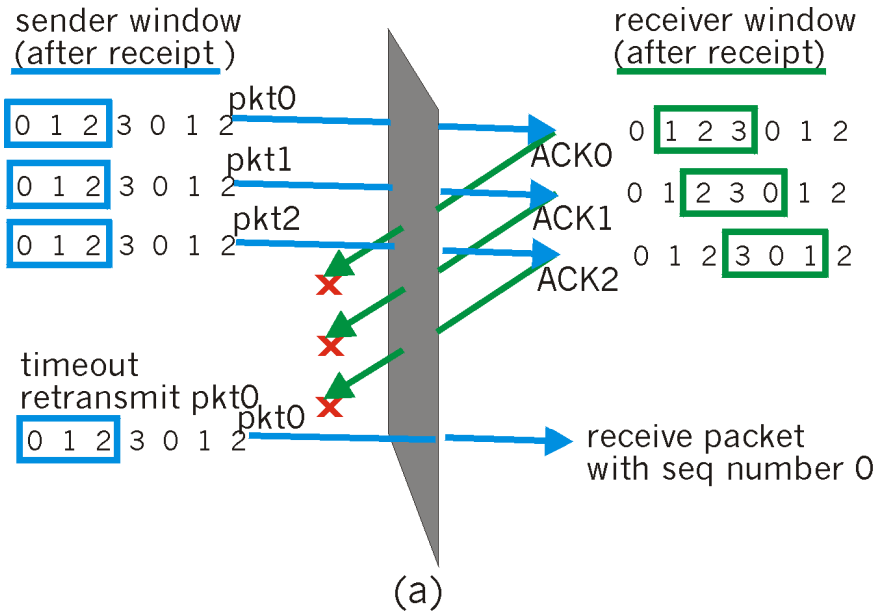
Selective Repeat: Nghịch Lý

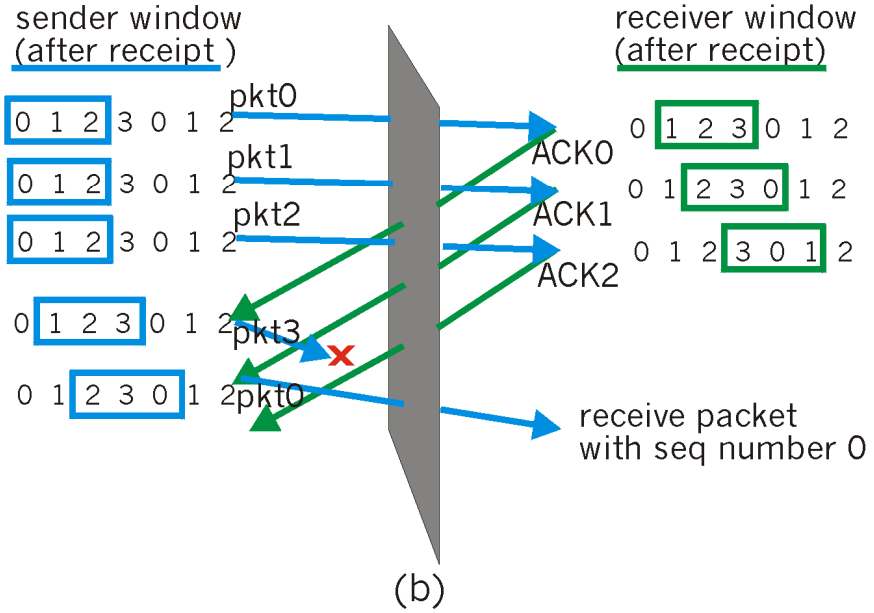
Ví dụ:

- STT: 0, 1, 2, 3
- Kích thước cửa sổ = 3

- phía Nhận không phân biệt được hai trường hợp (a) và (b)
- Chuyển dữ liệu trùng lặp lên trên (mà tưởng là dữ liệu mới) (a)

? Quan hệ giữa độ lớn cửa sổ và khoảng STT?

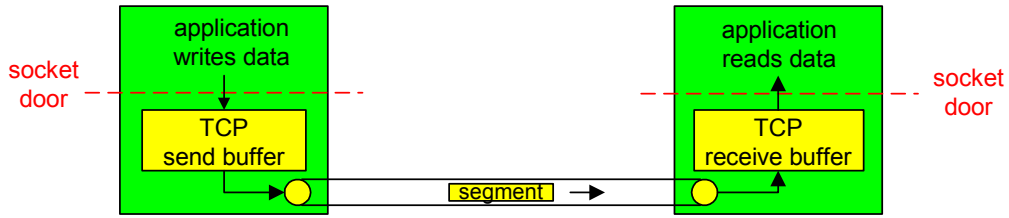




TCP: Tổng quan

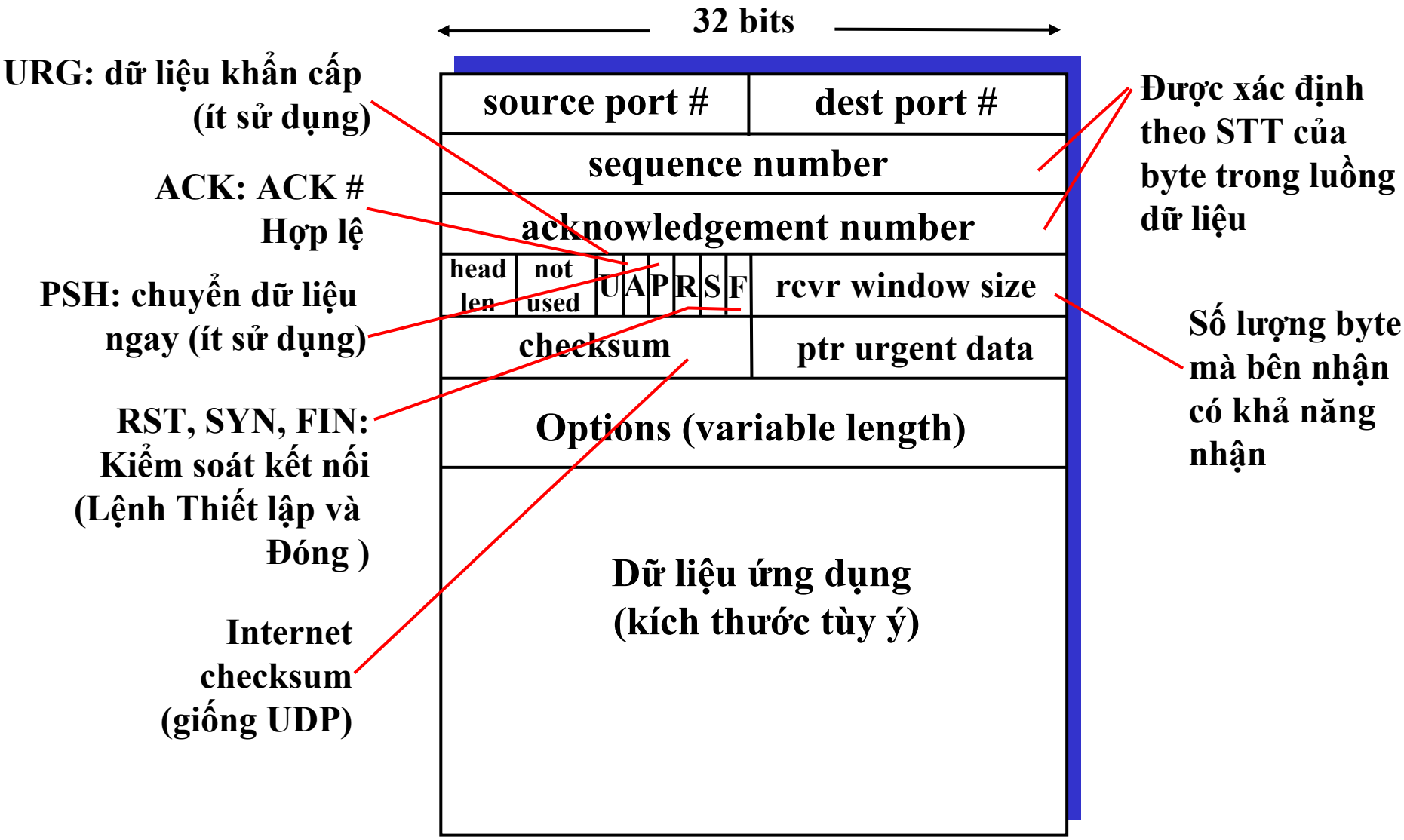
RFCs: 793, 1122, 1323, 2018, 2581

- Điểm nối điểm:
  - Một gửi, Một nhận
- Tin cậy, theo đúng thứ tự:
  - Không quan tâm đến khuôn dạng thông điệp.
- Đường ống:
  - Cửa sổ kiểm soát tắc nghẽn và điều khiển lưu lượng.
- *Bộ đệm ở phía Nhận và Gửi*



- Truyền song công:
  - Dữ liệu truyền theo cả hai hướng
  - MSS: Kích thước tối đa một segment
- Hướng nối:
  - Bắt tay, chào hỏi trước khi nói chuyện (trao đổi thông tin điều khiển). Thiết lập bộ đệm hai đầu.
- Kiểm soát lưu lượng:
  - Nói quá nhanh, nghe quá chậm

Cấu trúc TCP segment



TCP: Số thứ tự và Số biên nhận

Số thứ tự (STT):

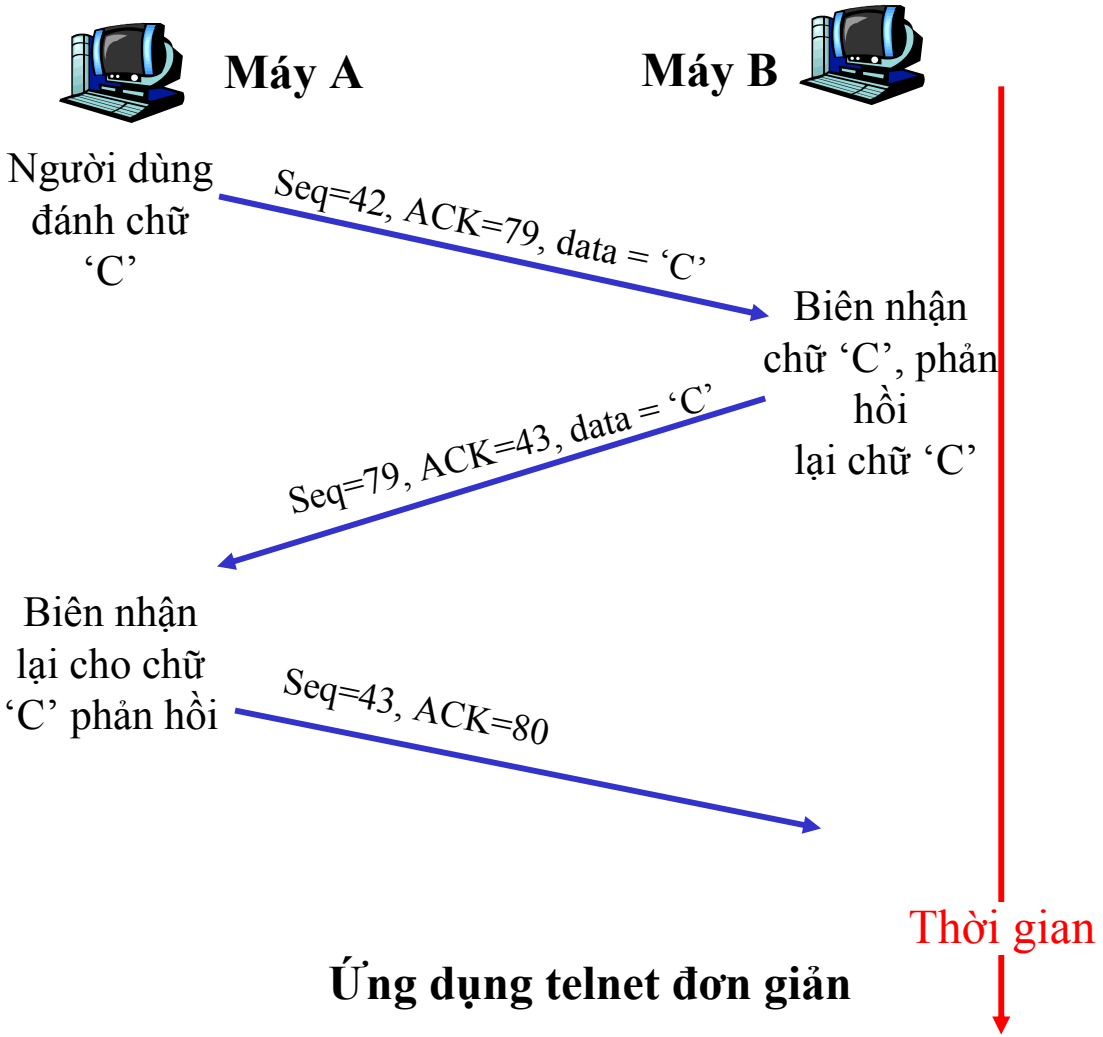
- Là Số thứ tự của byte đầu tiên trong luồng dữ liệu

Số biên nhận:

- Là Số thứ tự của byte kế tiếp mà bên nhận muốn nhận.
- Biên nhận tích lũy

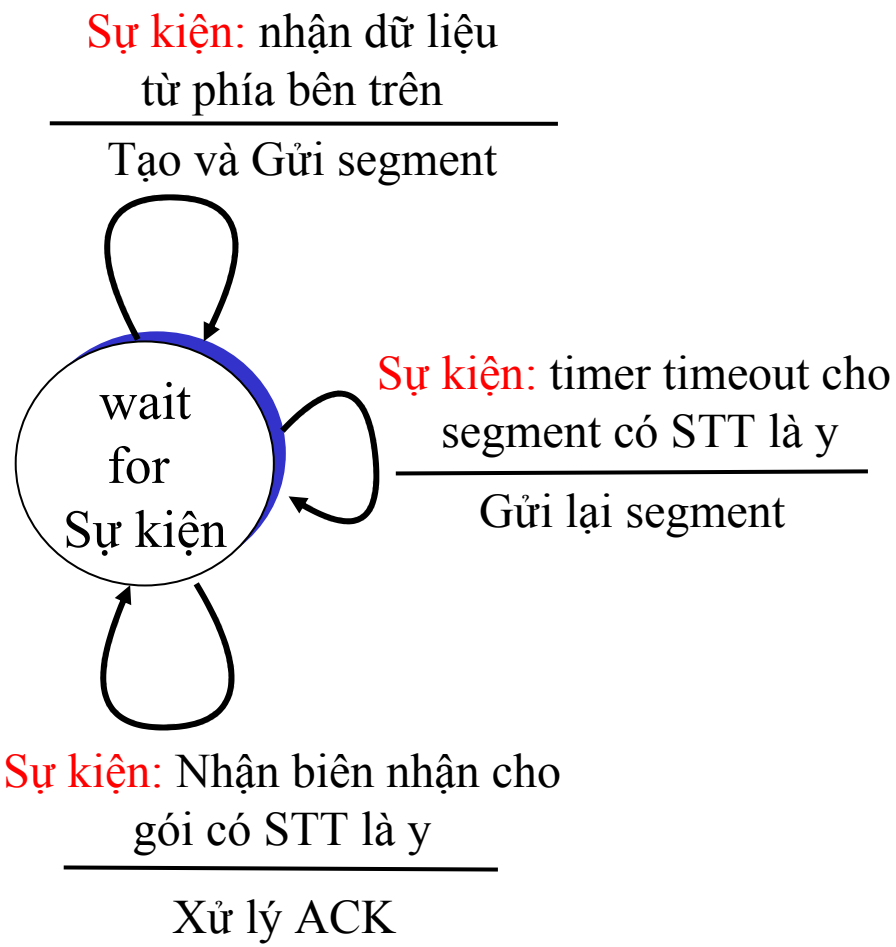
Q? Bên nhận xử lý gói tin không đúng thứ tự ntn ?

- **A**: TCP không quy định. Tùy thuộc vào người cài đặt.



Ứng dụng telnet đơn giản

TCP: Truyền Tin cậy



FSM bên Gửi **đơn giản**, giả định rằng:

- Dữ liệu truyền theo một hướng
- Không kiểm soát tắc nghẽn
- Không điều khiển lưu lượng

Nhanh chóng truyền lại

- Khoảng thời gian Timeout thường tương đối dài:
  - Chậm trễ trong việc gửi lại gói tin bị mất
- Phát hiện mất gói tin qua các ACK trùng lặp
  - Phía gửi thường gửi nhiều gói tin
  - Nếu gói tin bị mất, sẽ có ACK trùng lặp
- Nếu phía gửi nhận được 3 ACK trùng lặp, có thể giả thiết gói tin ngay sau gói tin được biên nhận 3 lần liên tiếp bị mất:
  - Gửi lại kể cả khi gói này chưa timeout

Fast Retransmit:

```
event: ACK received, with ACK field value of y
            if (y > SendBase) {
                …
                SendBase = y
                if (there are currently not-yet-acknowledged segments)
                    start timer
                …
              }
            else {
                increment count of dup ACKs received for y
                if (count of dup ACKs received for y = 3) {
                    resend segment with sequence number y
                 …
```

ACK trùng lặp cho gói tin Đã được biên nhận

Truyền lại nhanh chóng

TCP: Truyền tin cậy

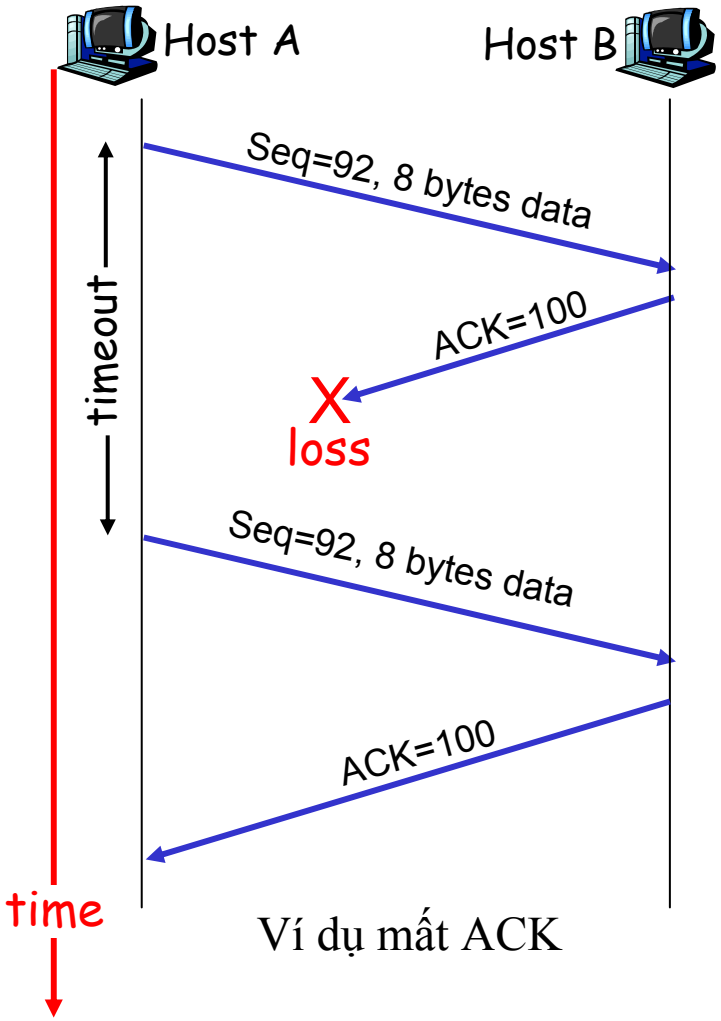
TCP phía Gửi đơn giản

```
00  sendbase = initial_sequence number agreed by TWH
01   nextseqnum = initial_sequence number by TWH
02    loop (forever) {
03      switch(event)
04      event: data received from application above
05                 if (window allow send)
06                    create TCP segment with sequence number nextseqnum
06                    if (no timer) start timer
07                    pass segment to IP
08                    nextseqnum = nextseqnum + length(data)
                   else put packet in buffer
09       event: timer timeout for sendbase
10            retransmit segment
11            compute new timeout interval
12            restart timer
13       event: ACK received, with ACK field value of y
14            if (y > sendbase) { /* cumulative ACK of all data up to y */
15                 cancel the timer for sendbase
16                 sendbase = y
17                 if (no timer and packet pending) start timer for new sendbase
17                 while (there are segments and window allow)
18                      sent a segment;
18            }
19            else { /* y==sendbase, duplicate ACK for already ACKed segment */
20                 increment number of duplicate ACKs received for y
21                 if (number of duplicate ACKS received for y == 3) {
22                      /* TCP fast retransmit */
23                    resend segment with sequence number y
24                    restart timer for segment y
25                 }
26       }  /* end of loop forever */
```

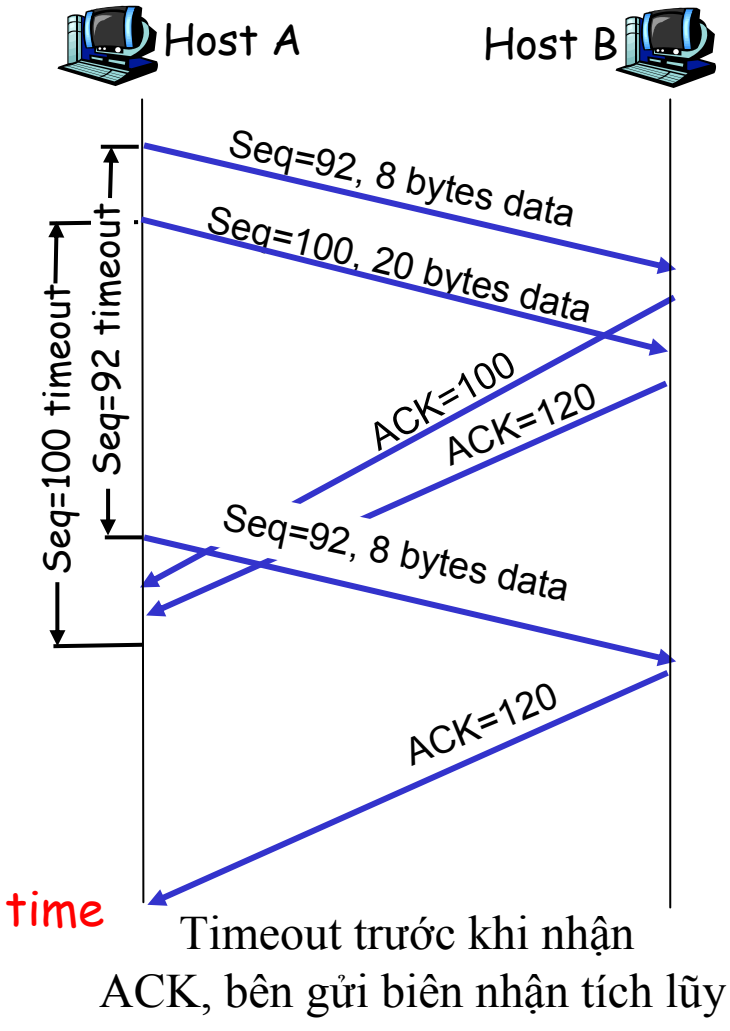
TCP: Chính sách ACK [RFC 1122, RFC 2581]

Sự kiện	Bên nhận (TCP)
Segment theo đúng STT đến, Không thiếu dữ liệu, Không có ACK treo	Trì hoãn ACK. Đợi segment kế tiếp trong 500ms. Nếu không có segment, gửi ACK
Segment theo đúng STT đến, Không thiếu dữ liệu, Có một ACK bị treo	Ngay lập tức gửi một ACK mang giá trị tích lũy
Segment theo đúng STT đến (STT đến lớn hơn số mong đợi). Thiếu dữ liệu	Gửi ACK trùng lặp, chỉ STT của byte dữ liệu mình muốn nhận
Một segment đến điền vào đoạn dữ liệu bị khuyết	Biên nhận STT bên nhận mong muốn nhận

TCP: Ví dụ về Truyền lại



Ví dụ mất ACK



Timeout trước khi nhận ACK, bên gửi biên nhận tích lũy

Điều khiển lưu lượng trong TCP

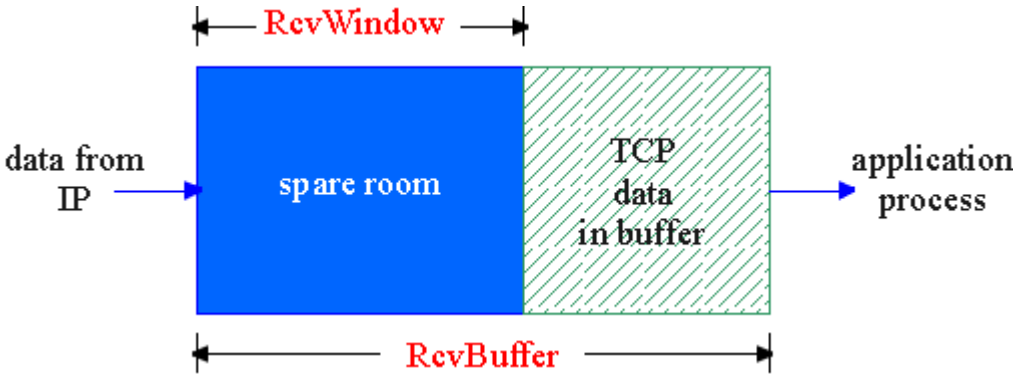
Điều khiển lưu lượng

Không cho bên Gửi gửi quá nhiều, quá nhanh

Phía Nhận: Thông báo rõ ràng cho phía Gửi khả năng nhận dữ liệu của mình (thay đổi thường xuyên)

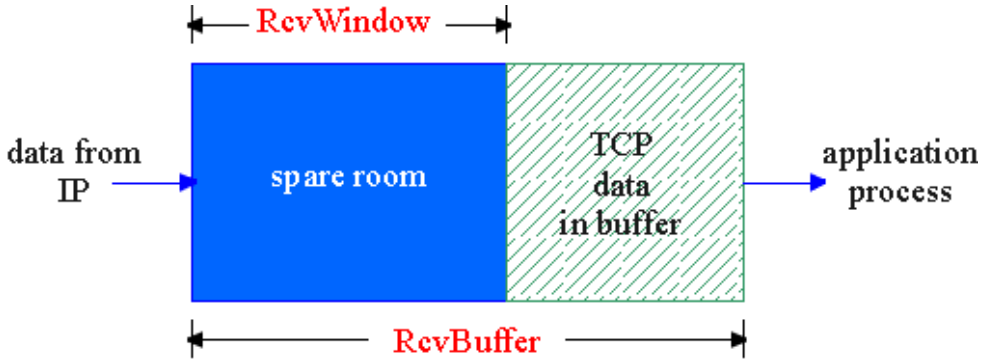
`RcvBuffer` = Kích thước Bộ đệm nhận
`RcvWindow` = Kích thước vùng còn trống trong Bộ đệm

Phía Gửi: Giữ khối lượng dữ liệu gửi đi nhưng chưa được biên nhận nhỏ hơn lượng bên kia chấp nhận được



Bộ đệm phía Nhận

Điều khiển lưu lượng trong TCP



- Chỗ trống trong Bộ đệm

= **RcvWindow**

source port #	dest port #
sequence number	
acknowledgement number	
head len / not used / U A P R S F	rcvr window size
checksum	ptr urgent data
Options (variable length)	
application data (variable length)	

TCP Round Trip Time and Timeout

Q: Thiết lập giá trị timeout ntn ?

- Timeout > RTT
  - Chú ý: RTT thay đổi thường xuyên
- **Quá bé**: timeout ngay
  - Truyền lại không cần thiết
- **Quá lớn**: xử lý việc mất gói tin bị chậm trễ

Q: Làm thế nào để ước lược RTT?

- **`SampleRTT`**: khoảng thời gian từ khi gửi gói tin cho đến khi nhận được biên nhận
  - Bỏ qua truyền lại
- **`SampleRTT`** thay đổi thường xuyên. Chúng ta muốn ước lượng RTT "mịn hơn"
  - Sử dụng nhiều giá trị đo được trong quá khứ, không phải chỉ có một **`SampleRTT`** gần nhất

TCP Round Trip Time và Timeout

EstimatedRTT = (1-x)*EstimatedRTT + x*SampleRTT

- Trọng số sẽ thay đổi giá trị trung bình
- Ảnh hưởng của SampleRTT
- x thường chọn giá trị 0.1

Thiết đặt giá trị timeout

- **EstimtedRTT** cộng thêm một "giá trị an toàn"
- Biến thiên **EstimatedRTT** càng lớn **->** tăng "giá trị an toàn"

Timeout = EstimatedRTT + 4*Deviation

Deviation = (1-x)*Deviation +
x*|SampleRTT-EstimatedRTT|

TCP : Quản lý Kết nối

Chú ý: Trong TCP, phía Gửi và Nhận thiết lập "kết nối" trước khi trao đổi các segment dữ liệu.

- Khởi tạo các biến TCP:
  - Số thứ tự
  - Bộ đệm, Thông tin về lưu lượng (**`RcvWindow`**)
- *client*: Khởi tạo kết nối

```
Socket clientSocket = new
 Socket("hostname","port
 number");
```

- *server*: Đợi kết nối từ client

```
Socket connectionSocket =
 welcomeSocket.accept();
```

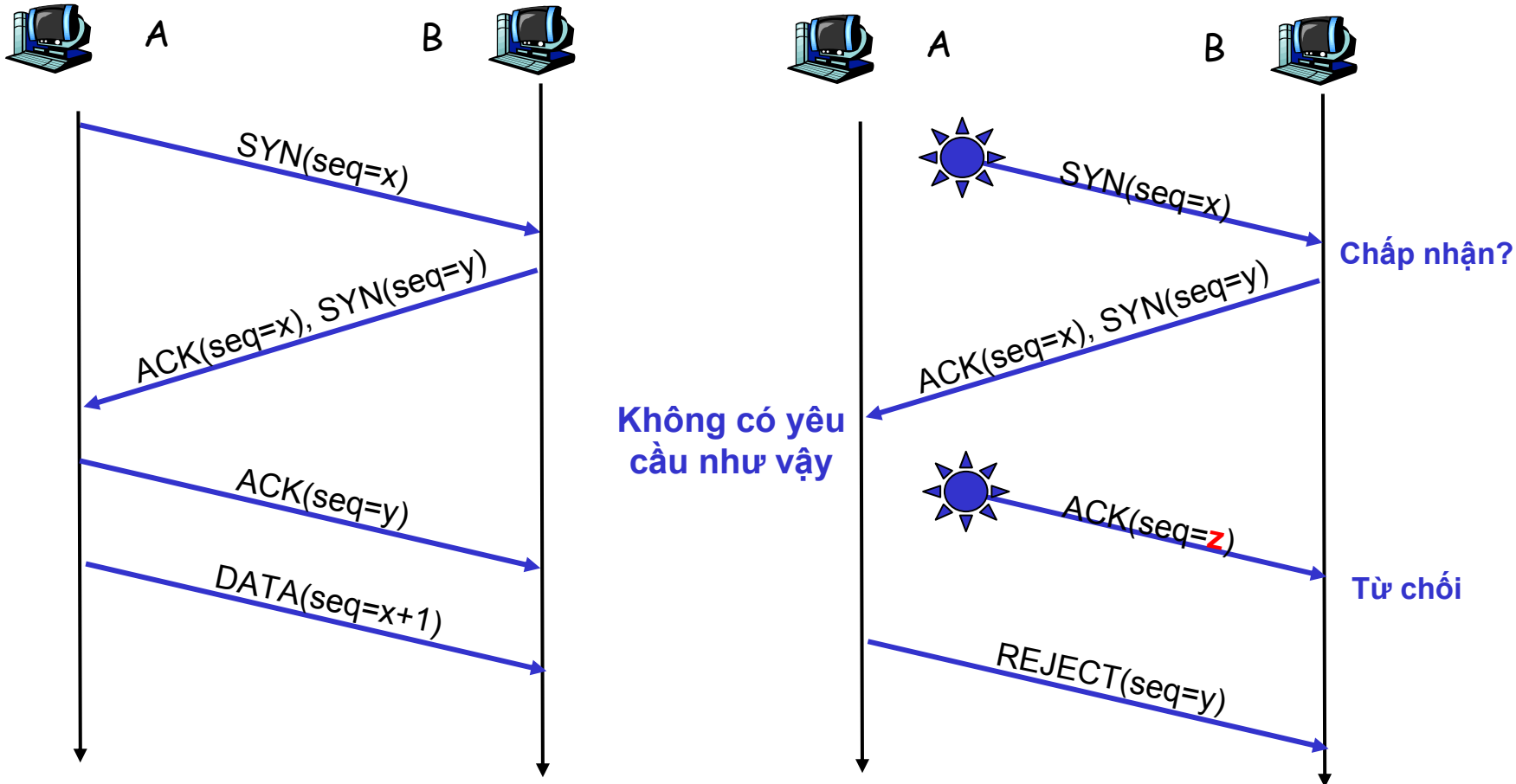
Bắt tay ba bước:

Bước 1: Phía client gửi gói tin điều khiển TCP SYN tới server
- Chứa Số thứ tự khởi đầu

Bước 2: Nhận được gói SYN, nếu chấp nhận kết nối, server gửi trả lời gói tin điều khiển SYNACK
- Biên nhận cho gói SYN vừa nhận
- Cấp phát bộ đệm
- Thông báo về STT khởi đầu của server

Bắt tay ba bước

- Để đảm bảo rằng bên kia thực sự mong muốn thiết lập kết nối



TCP: Quản lý Kết nối (tiếp)

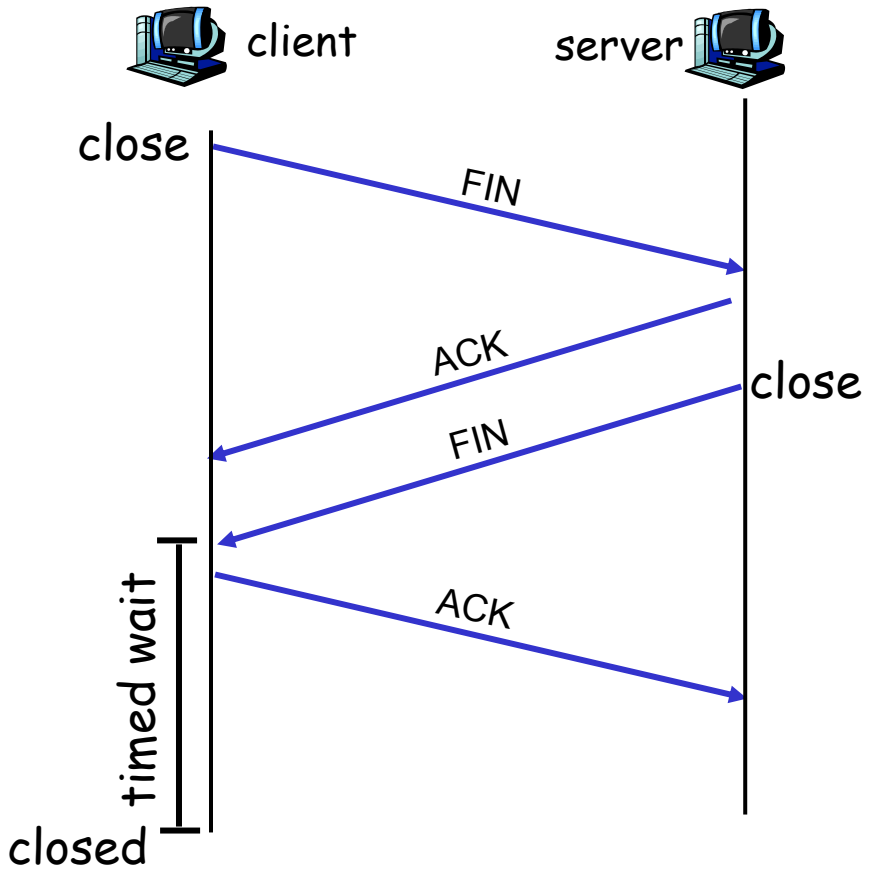
Đóng một kết nối:

client đóng socket:

```
clientSocket.close();
```

Bước 1: client gửi gói điều khiển FIN tới server

Bước 2: server nhận được gói FIN, biên nhận cho gói tin này. Đóng kết nối, gửi gói FIN.



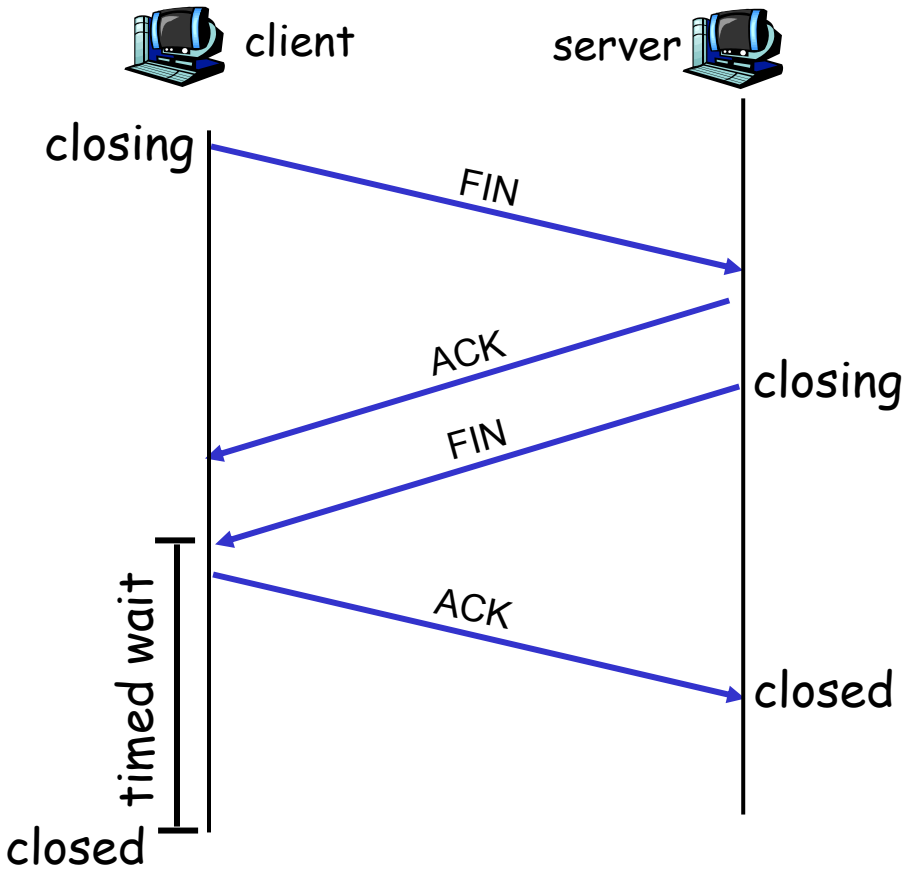
TCP: Quản lý Kết nối (tiếp)

Bước 3: client nhận gói FIN, biên nhận lại ACK.

- Bước vào trạng thái "timed wait" – sẽ biên nhận ACK cho các gói FIN nhận được

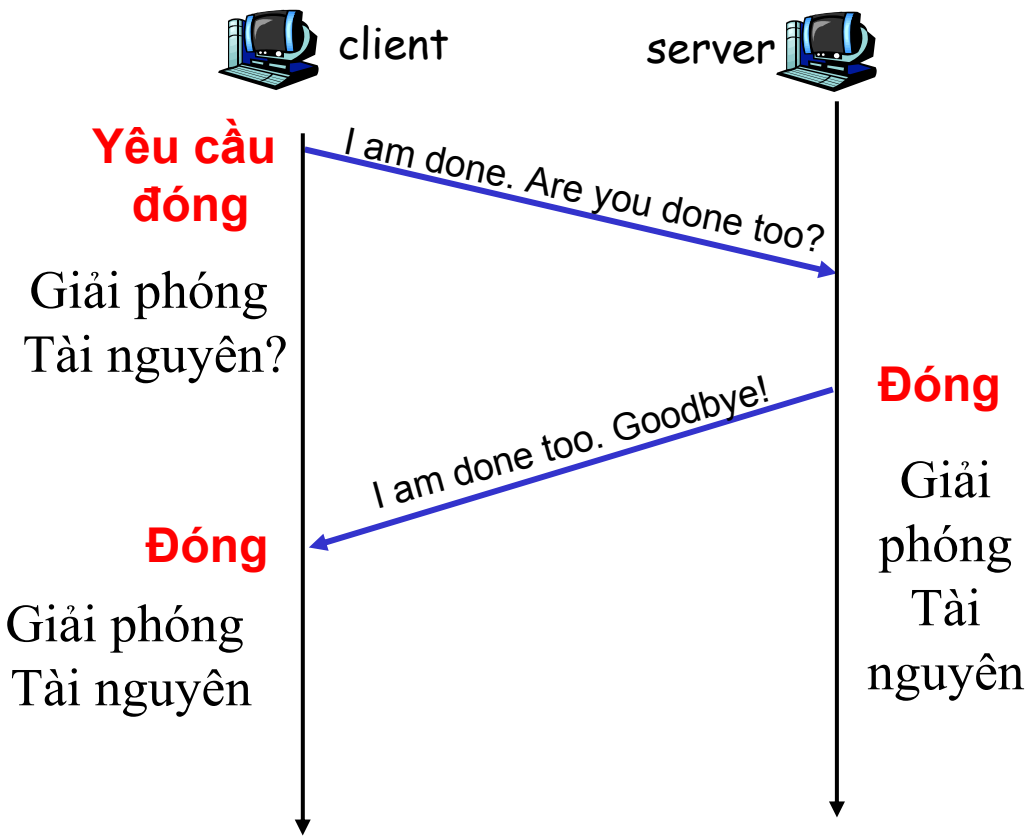
Bước 4: server nhận được ACK, đóng kết nối.

Chú ý: Với vài cải tiến nhỏ, ta có thể xử lý đồng thời nhiều gói FIN.

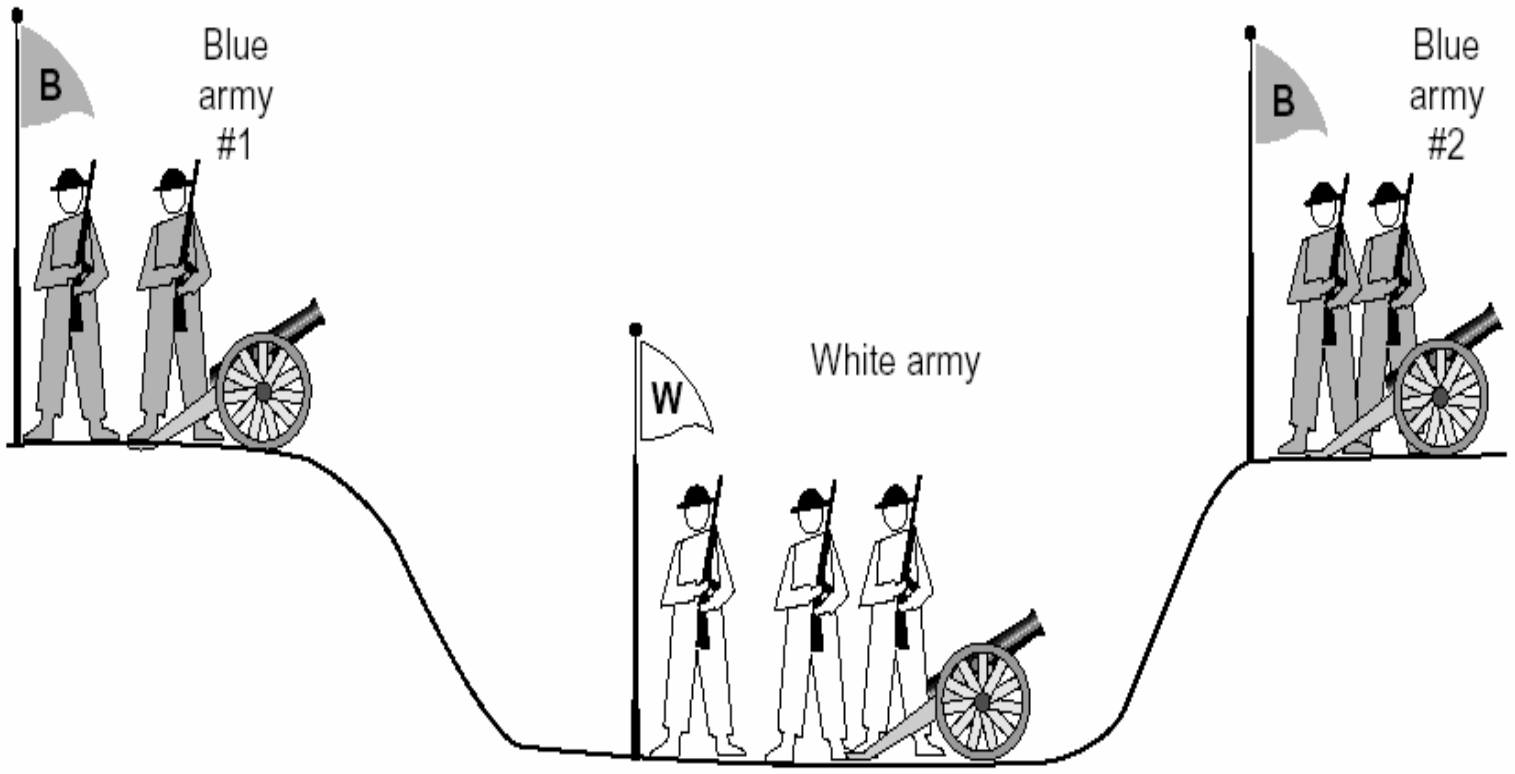


Đóng kết nối

- Mục tiêu:
  - Mỗi phía giải phóng tài nguyên và xóa bỏ trạng thái về kênh truyền

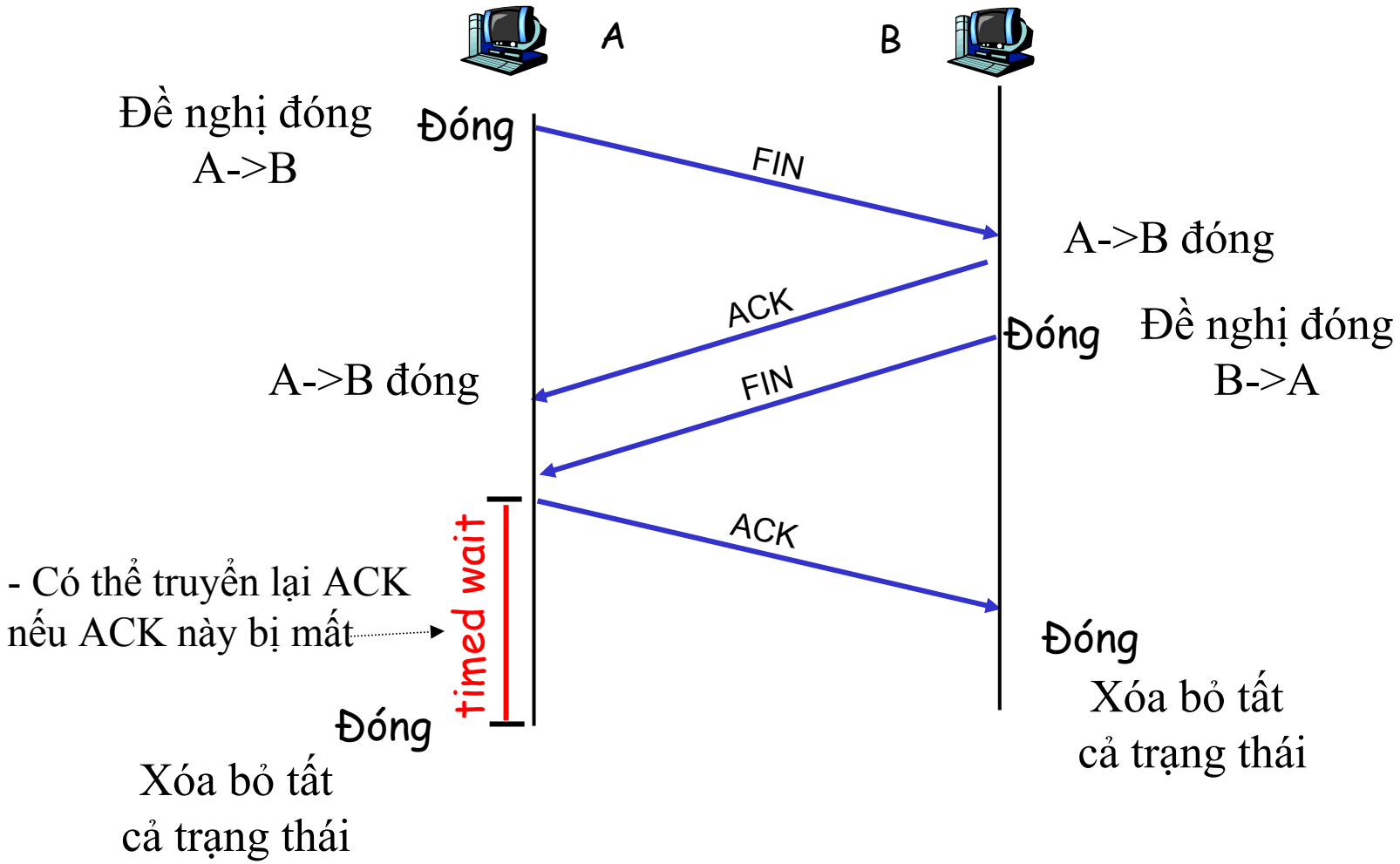


Vấn đề tổng quát: Quân Xanh-Trắng

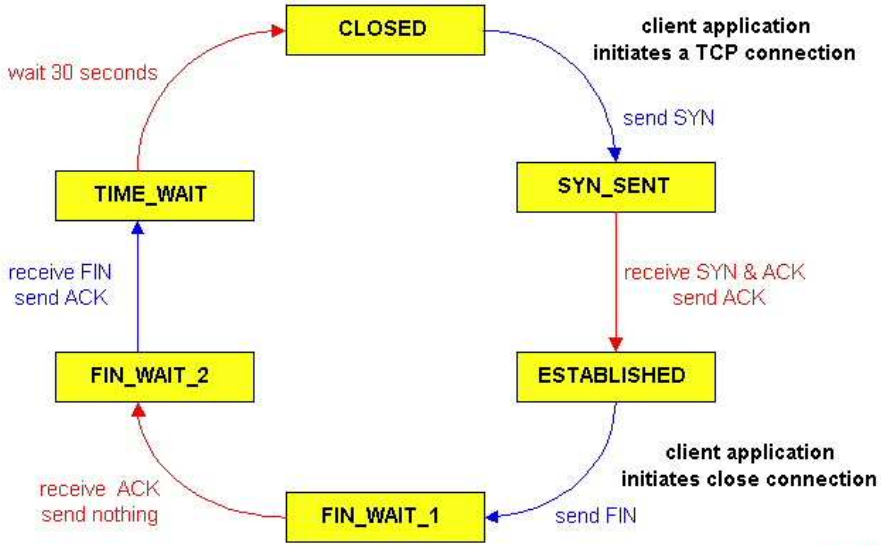


Hai phía quân xanh cần thống nhất thời điểm để cùng tấn công quân trắng. Họ thỏa thuận bằng cách gửi thông điệp cho nhau. Nếu cùng đồng ý : tấn công, còn không sẽ không tấn công. Chú ý rằng người truyền tin có thể bị bắt ! Nếu cùng tấn công, bên xanh thắng, còn nếu tấn công riêng lẻ, bên trắng thắng

Đóng kết nối trong bốn bước

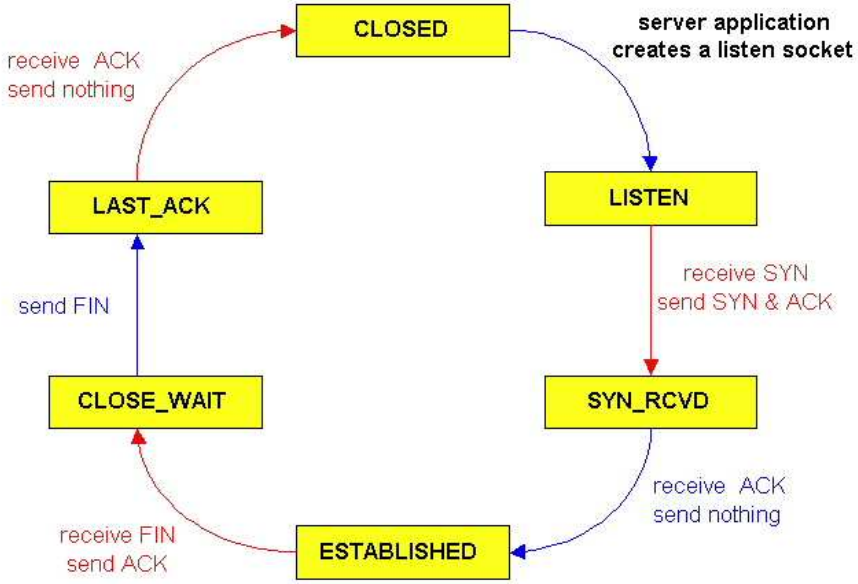


TCP: Quản lý Kết nối (tiếp)

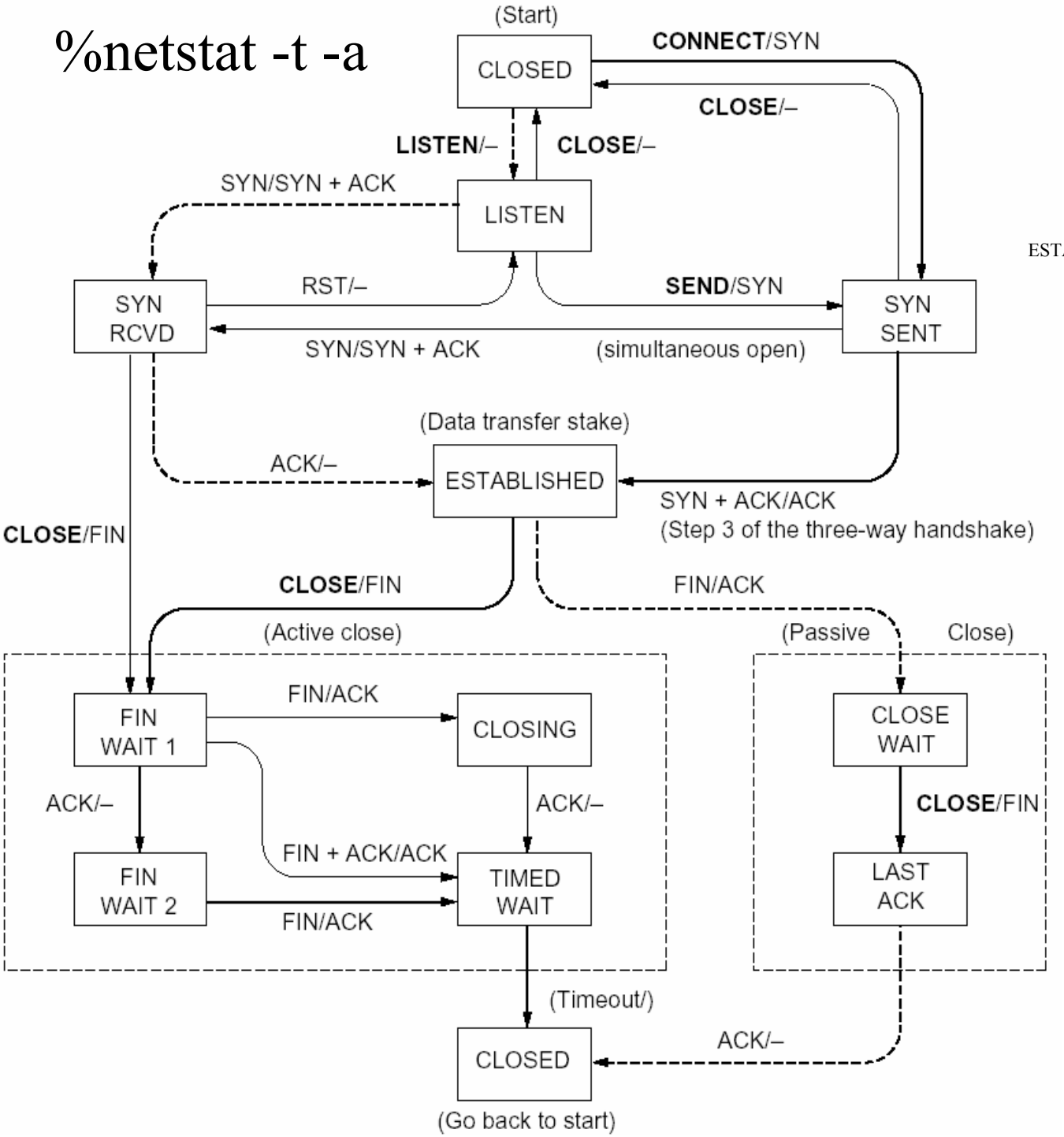


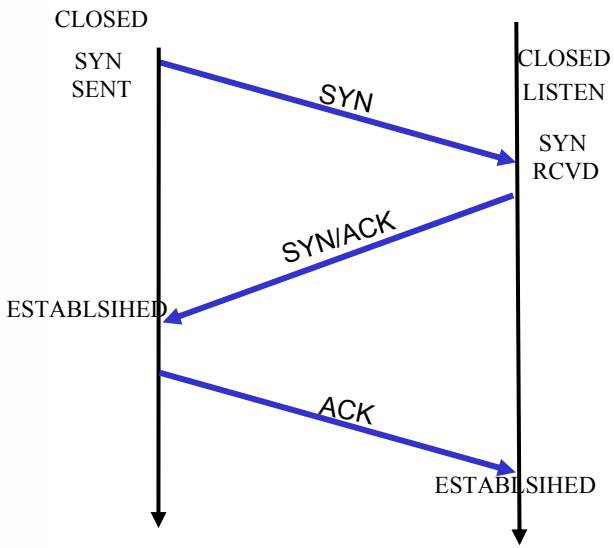
Vòng đời TCP phía Client

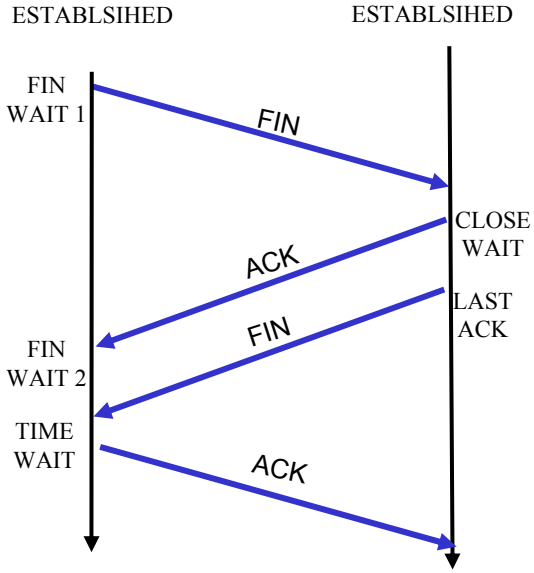
Vòng đời TCP phía Server



%netstat -t -a


(Start)
CLOSED
CONNECT/SYN
CLOSE/–
LISTEN/–
CLOSE/–
SYN/SYN + ACK
LISTEN
SYN RCVD
RST/–
SEND/SYN
SYN SENT
SYN/SYN + ACK
(simultaneous open)
(Data transfer stake)
ACK/–
ESTABLISHED
SYN + ACK/ACK
(Step 3 of the three-way handshake)
CLOSE/FIN
CLOSE/FIN
FIN/ACK
(Active close)
(Passive Close)
FIN WAIT 1
FIN/ACK
CLOSING
ACK/–
ACK/–
FIN + ACK/ACK
FIN WAIT 2
TIMED WAIT
FIN/ACK
CLOSE WAIT
CLOSE/FIN
LAST ACK
(Timeout/)
CLOSED
ACK/–
(Go back to start)


CLOSED
SYN SENT
CLOSED LISTEN
SYN
SYN RCVD
SYN/ACK
ESTABLSIHED
ACK
ESTABLSIHED


ESTABLSIHED
ESTABLSIHED
FIN WAIT 1
FIN
CLOSE WAIT
ACK
LAST ACK
FIN
FIN WAIT 2
TIME WAIT
ACK

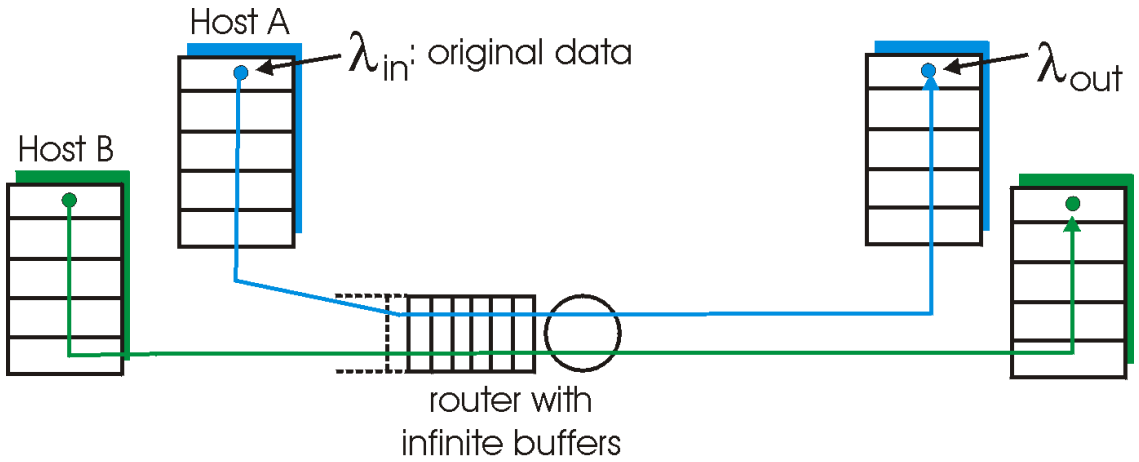
Nguyên tắc Kiểm soát Tắc nghẽn

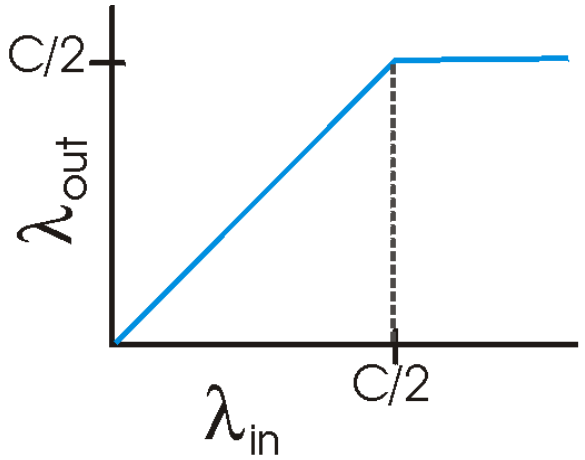
Tắc nghẽn:

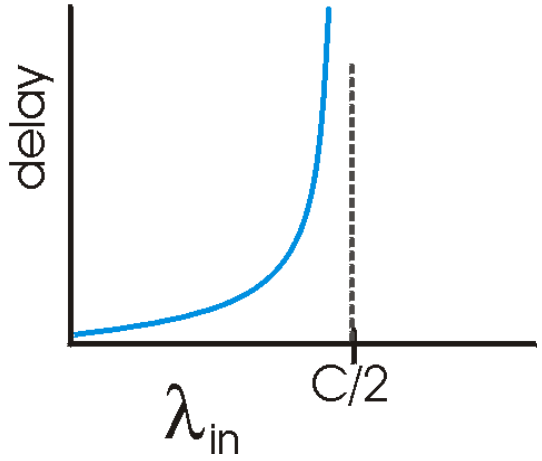
- Mường tượng: "có ***quá nhiều*** nút gửi ***quá nhiều*** dữ liệu với tốc độ ***quá nhanh*** mà mạng không chuyển kịp"
- Khác với ***Điều khiển lưu lượng***!
- Biểu hiện :
  - ***Mất gói tin*** (Tràn bộ đệm tại router)
  - ***Độ trễ lớn*** (Các gói tin phải "xếp hàng" tại router)
- Là một trong **10** vấn đề quan trọng nhất !

Nguyên nhân và Giá tắc nghẽn: Ví dụ 1

- 2 gửi, 2 nhận
- Router với bộ đệm vô hạn
- Không có cơ chế truyền lại



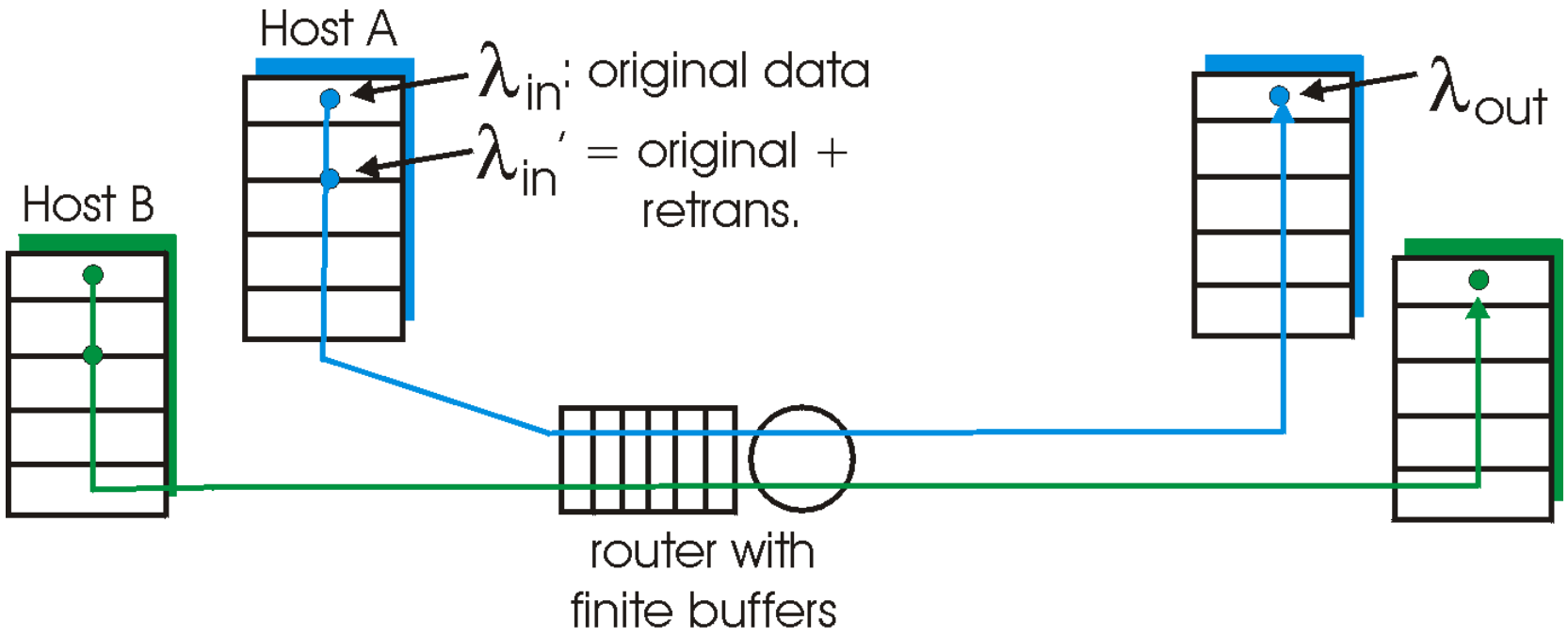




- Độ trễ lớn khi tắc nghẽn
- Thông lượng có thể đạt cực đại

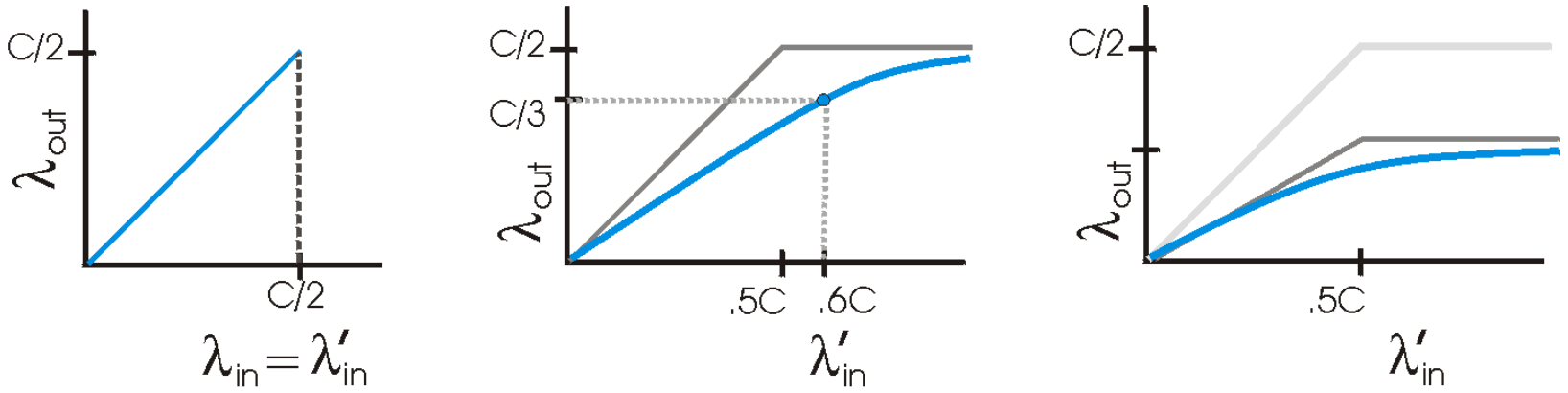
Nguyên nhân và Giá tắc nghẽn: Ví dụ 2

- Một router, bộ đệm *hữu hạn*
- Gửi lại các packet bị mất



Nguyên nhân và Giá tắc nghẽn: Ví dụ 2

- Thông thường: $\lambda_{in} = \lambda_{out}$ (tốt)
- Truyền lại khi mất (lý tưởng): $\lambda'_{in} > \lambda_{out}$
- Truyền lại của các gói tin đến trễ (không bị mất) khiến λ'_{in} lớn hơn (so với trường hợp lý tưởng) λ_{out}



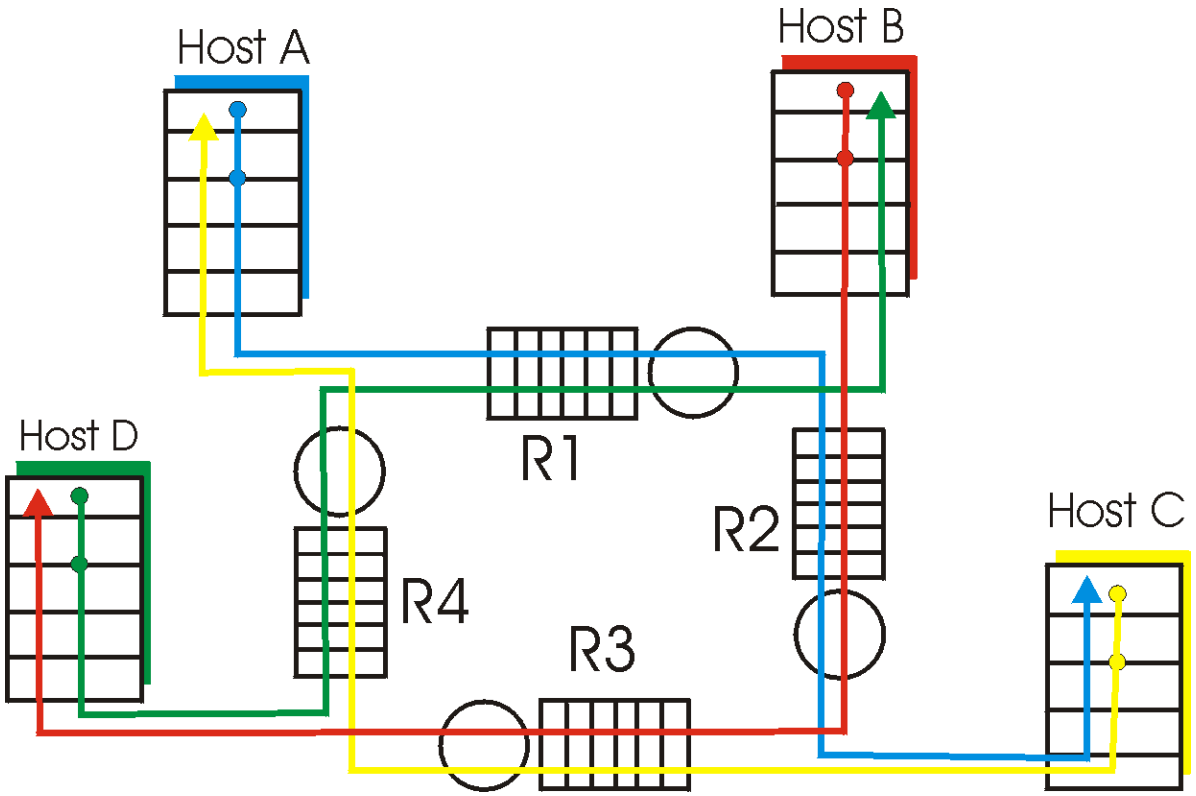
"Giá" của Tắc nghẽn:

- Phải truyền lại nhiều
- Truyền lại không cần thiết: Nhiều bản sao của cùng 1 gói tin có thể nằm trên mạng

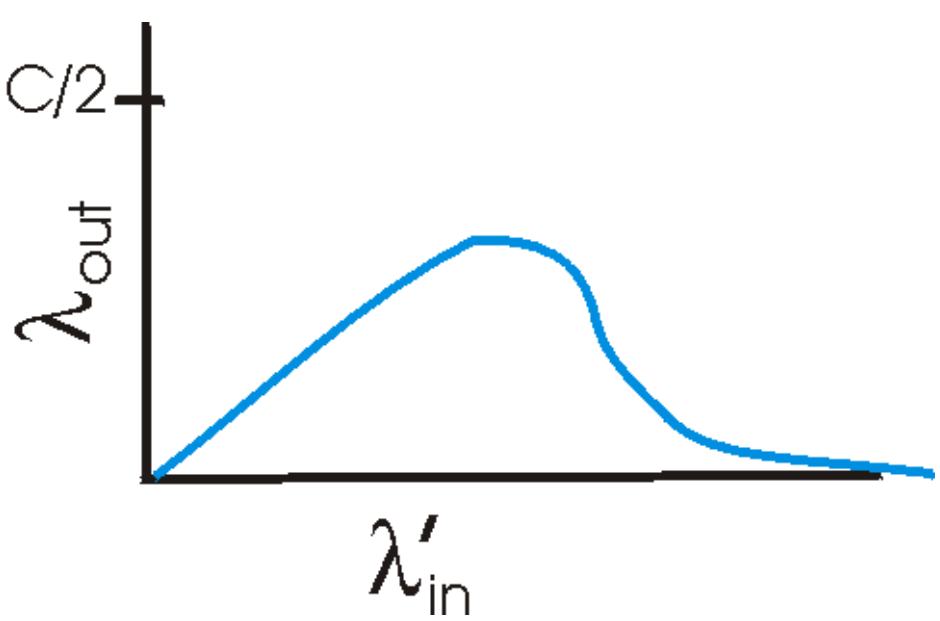
Nguyên nhân và Giá tắc nghẽn: Ví dụ 3

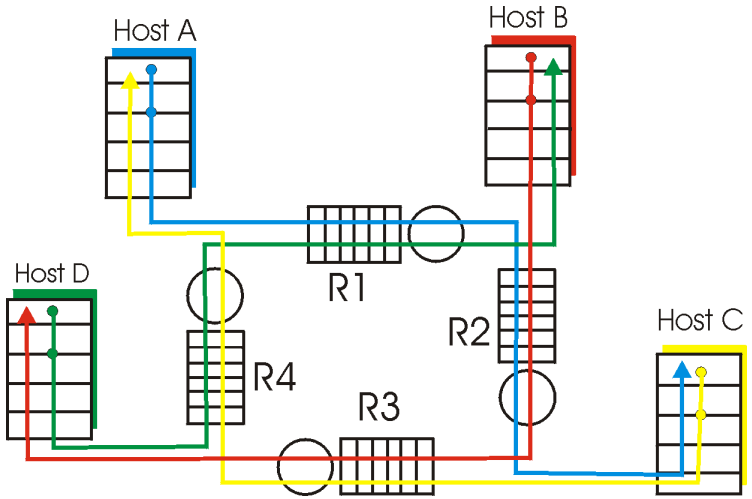
- 4 người gửi
- Nhiều tuyến
- Timeout => Gửi lại

Q: Chuyện gì xảy ra khi λ_{in} và tăng λ'_{in} ?



Nguyên nhân và Giá tắc nghẽn: Ví dụ 3





Một vấn đề khác của tắc nghẽn:

- Khi một packet bị mất, tất cả "công sức" tạo và chuyển gói tin này của các tầng bên trên đều bị mất !

Giải pháp chống Tắc nghẽn

Có hai lớp giải pháp chính:

Giải pháp đầu cuối:

- Tầng mạng (router) không thông báo cho các nút về Tắc nghẽn (nếu có)
- Mất gói tin, Độ trễ lớn: dấu hiệu của Tắc nghẽn
- Là giải pháp được TCP áp dụng

Có sự hỗ trợ từ mạng:

- routers thông báo cho thiết bị đầu cuối
  - Sử dụng một bit thông báo tình trạng tắc nghẽn (SNA, DECbit, TCP/IP ECN, ATM)
  - Thông báo tốc độ gửi tối đa

Ví dụ : Chống tắc nghẽn trong ATM

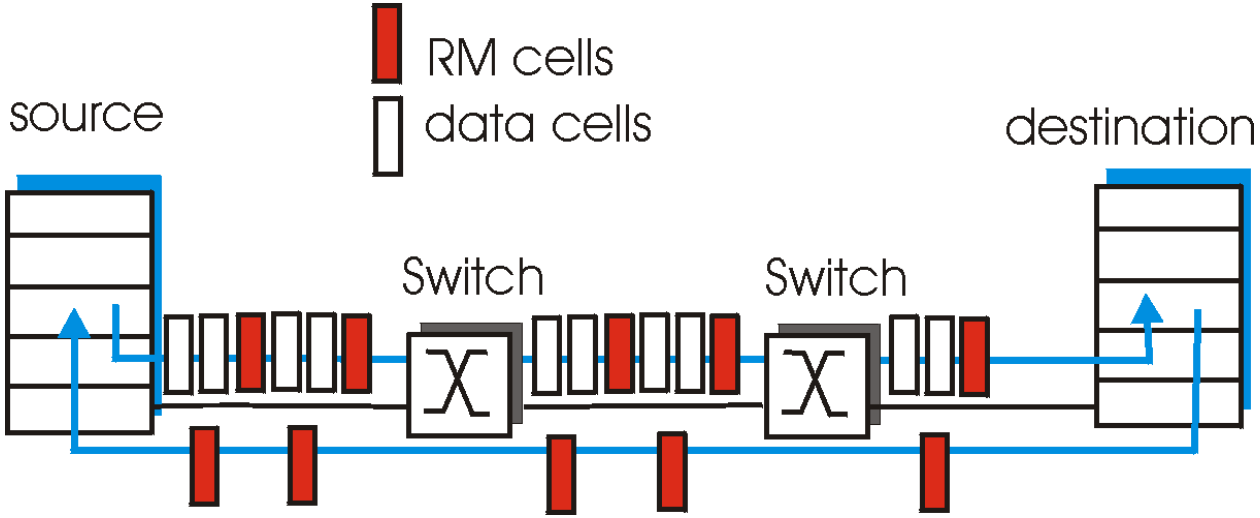
ABR: available bit rate:

- Dịch vụ “co giãn”
- Nếu đường truyền ở phía gửi chưa dùng hết:
  - Phía gửi có thể gửi thêm
- Nếu đường truyền ở phía gửi tắc nghẽn :
  - Phía gửi có thể được đảm bảo một băng thông tối thiểu

Tế bào RM (resource management) :

- Phía Gửi gửi kèm cùng các tế bào dữ liệu
- ATM switch có thể thiết lập một số bit trong tế bào RM (“có sự trợ giúp từ mạng”)
  - NI bit: Không được tăng tốc độ gửi (Tắc nghẽn ít)
  - CI bit: Có tắc nghẽn
- Tế bào RM được phía Nhận gửi trả cho phía Gửi

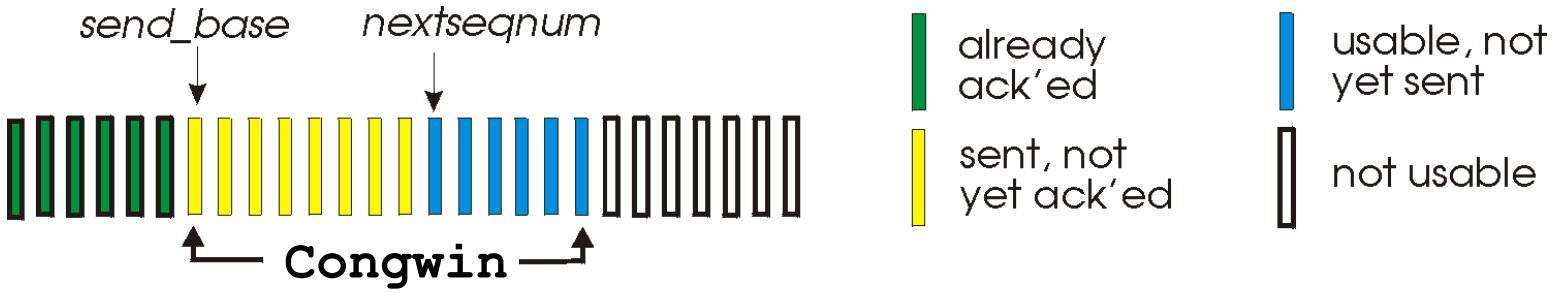
Ví dụ : Chống tắc nghẽn trong ATM



- Trường ER (explicit rate) 2 byte trong tế bào RM
  - switch bị tắc nghẽn có thể giảm ER trong tế bào
  - sender' send rate thus minimum supportable rate on path
- Trường EFCI 1 bit được switch tắc nghẽn thiết lập giá trị 1
  - Nếu tế bào dữ liệu đằng trước tế bào RM có giá trị EFCI =1, phía gửi thiết lập bit CI trong tế bào RM phản hồi

Kiểm soát tắc nghẽn trong TCP

- Kiểm soát Đầu cuối (Mạng không hỗ trợ)
- Tốc độ truyền bị giới hạn bởi cửa sổ kiểm soát tắc nghẽn, **Congwin**, (số lượng segment) :



- w segments, kích thước là MSS byte được gửi đi trong 1 RTT:

$$\text{Thông lượng} = \frac{w * MSS}{RTT} \text{ Bytes/sec}$$

TCP : Kiểm soát tắc nghẽn

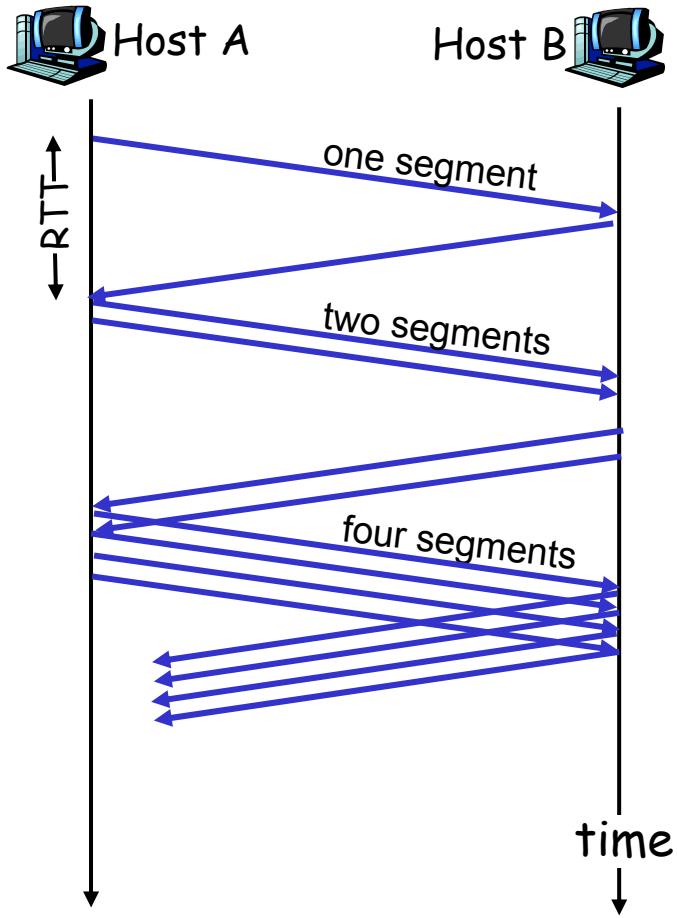
- "**thăm dò**" băng thông của đường truyền:
  - **Lý tưởng**: khi không có tắc nghẽn, truyền nhanh nhất có thể (**`Congwin`** càng lớn càng tốt)
  - ***Tăng*** **`Congwin`** cho đến khi có mất dữ liệu (tắc nghẽn)
  - Mất mát: ***Giảm*** **`Congwin`**, và bắt đầu quá trình thăm dò
- hai "giai đoạn"
  - Khởi đầu chậm
  - Tránh tắc nghẽn
- Một số biến quan trọng:
  - **`Congwin`**
  - **`threshold:`** xác định giá trị ***Ngưỡng*** giữa hai pha

TCP : Khởi đầu chậm

Thuật toán khởi đầu chậm

initialize: Congwin = 1
for (each segment ACKed)
    Congwin++
until (loss Sự kiện OR
    CongWin > threshold)

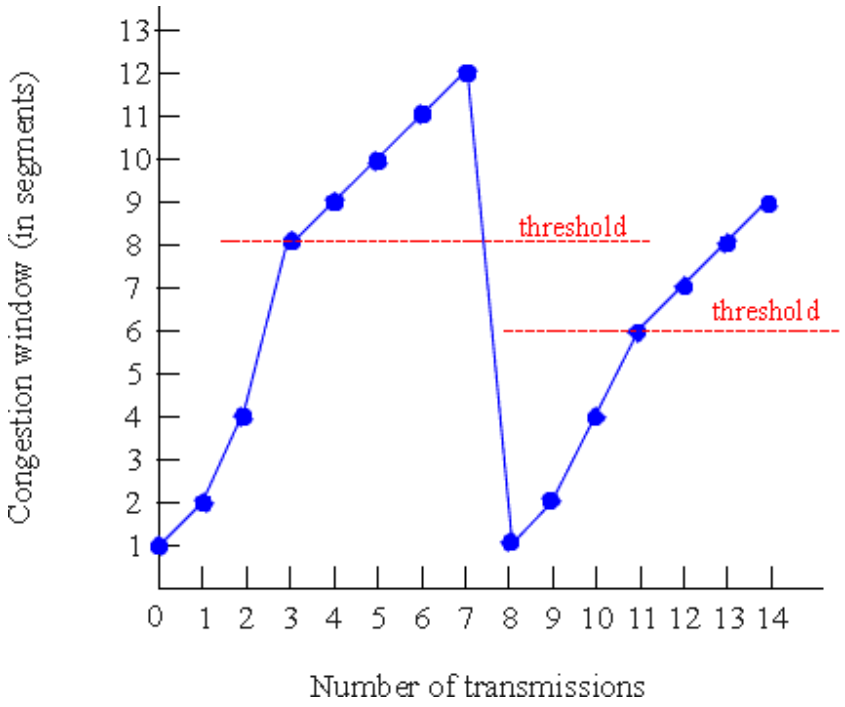
- Kích thước cửa sổ tăng theo hàm số mũ (không quá chậm !)
- Sự kiện loss : timeout (Tahoe TCP) hoặc/và ba lần nhận ACK trùng lặp (Reno TCP)



TCP : Tránh tắc nghẽn

Tránh tắc nghẽn

```
/* slowstart is over        */
/* Congwin > threshold */
Until (loss Sự kiện) {
  every w segments ACKed:
      Congwin++
  }
threshold = Congwin/2
Congwin = 1
perform slowstart[1]
```



1: TCP Reno bỏ qua giai đoạn khởi đầu chậm (khôi phục nhanh) sau khi nhận 3 ACK trùng lặp

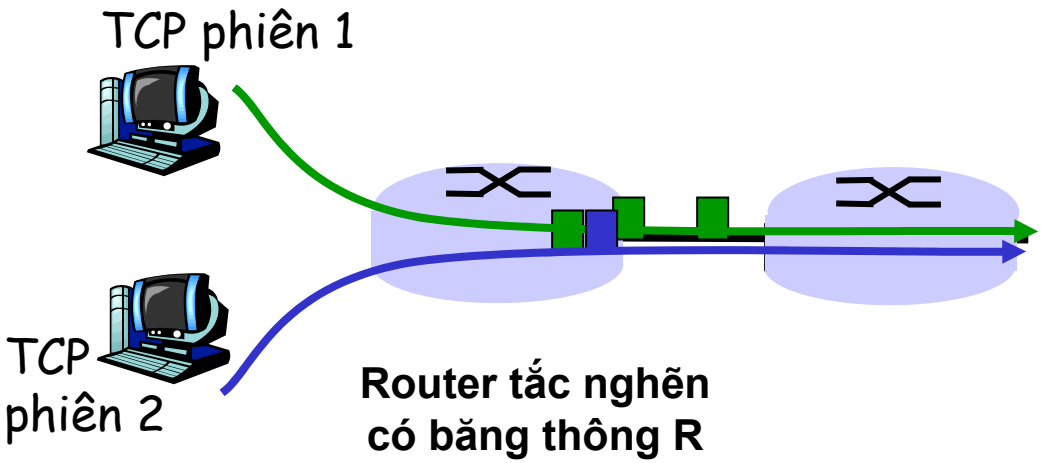
AIMD

Tránh tắc nghẽn trong TCP:

- AIMD: *additive increase, multiplicative decrease*
  - Tăng cửa sổ lên 1 khi nhận được một gói phản hồi
  - Giảm cửa sổ theo số mũ của 2 khi có sự kiện mất gói dữ liệu

Tính công bằng trong TCP

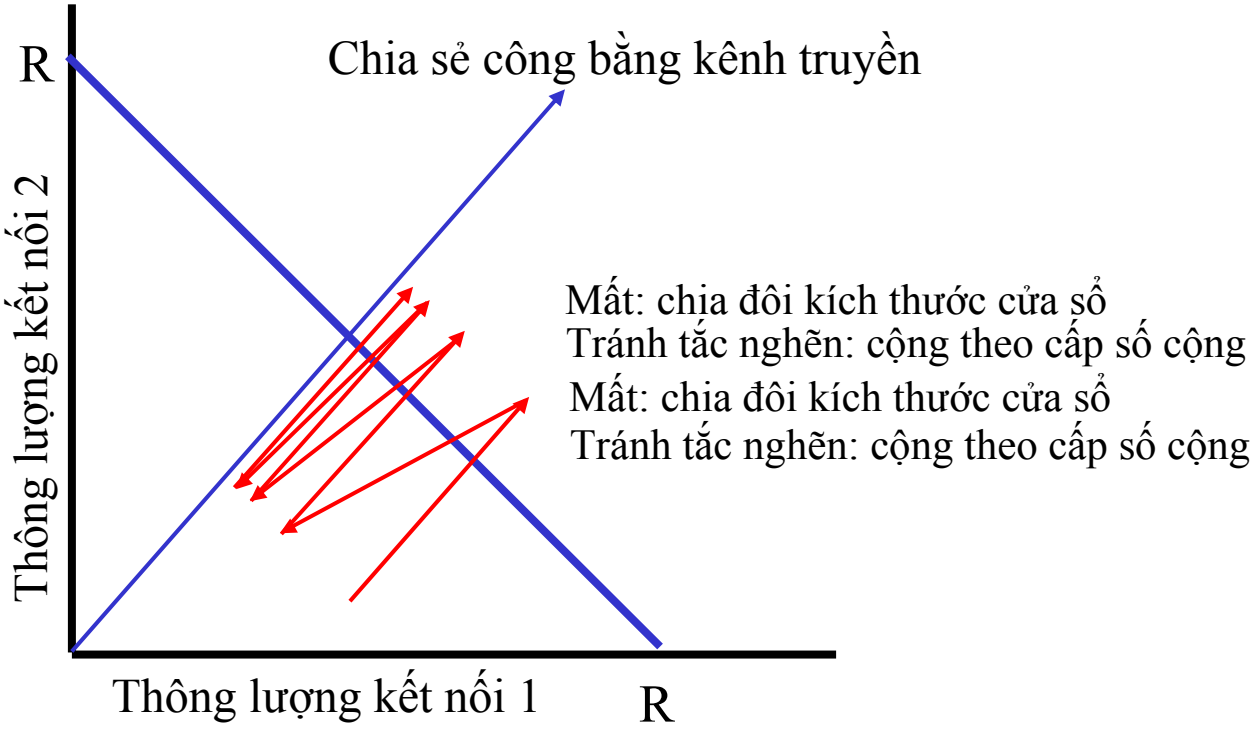
Công bằng: Nếu N phiên TCP cùng nhau chia sẻ một kênh truyền tắc nghẽn, mỗi phiên nhận được 1/N băng thông



Tại sao TCP công bằng?

Hai phiên cạnh tranh nhau sử dụng đường truyền:

- Tăng theo cấp số cộng : băng thông tăng dần dần
- Giảm theo cấp số nhân : giảm đột ngột băng thông



Chapter 3: Tổng kết

- Các dịch vụ của tầng giao vận:
  - Phân kênh/ Dồn kênh
  - Truyền tin cậy
  - Điều khiển lưu lượng
  - Kiểm soát tắc nghẽn
- Cài đặt trên Internet
  - UDP
  - TCP

Tiếp theo:

- Rời khỏi lớp "Rìa" của Mạng
- Tiến vào lớp "Lõi"

Chương 4: LÝ THUYẾT THIẾT KẾ CƠ SỞ DỮ LIỆU QH

I. PHỤ THUỘC HÀM

- Định nghĩa phụ thuộc hàm
- Hệ tiên đề Armstrong
- Bao đóng của tập PTH
- Bao đóng của tập thuộc tính
- Siêu khóa và khóa
- Phủ tối thiểu

II. PHÉP TÁCH LƯỢC ĐỒ QUAN HỆ

- Định nghĩa phép tách
- Phép tách không mất mát thông tin

III. CHUẨN HÓA LƯỢC ĐỒ QUAN HỆ

- Khái niệm chuẩn hóa
- Các dạng chuẩn (1NF, 2NF, 3NF)
- Phép tách lược đồ thành các lược đồ 3NF

I. PHỤ THUỘC HÀM

1.1. Đặt vấn đề

MASV	HOTEN	MAKHOA	TENKHOA	MAMON	TENMON	SOTC	DIEM
01	Lê Văn An	K1	CNTT	M1	PASCAL	4	5
01	??	??	??	M2	CSDL	3	8
02	Nguyễn Văn Bình	K2	TOAN	M1	??	??	9
02	??	??	??	M2	??	??	6
03	Lê Văn An	K2	??	M1	??	??	7
03	??	??	??	M2	??	??	9
01	??	??	??	M1	??	??	??

- **MASV→HOTEN, MAKHOA, TENKHOA**
- **MAMON→TENMON, SOTC**
- **MAKHOA→TENKHOA**
- **MASV, MAMON→ DIEM**

I. PHỤ THUỘC HÀM

1.2. Định nghĩa PTH

Cho quan hệ r(U) với U = {A_1, A_2, . . . , A_n} và các tập thuộc tính X, Y $\subseteq$ U. Phụ thuộc hàm là một phát biểu dạng X$\rightarrow$Y (đọc là X xác định Y hay Y phụ thuộc hàm vào X), nếu với mọi t_1, t_2 $\in$r mà $t_1[X]= t_2[X]$ thì $t_1[Y]= t_2[Y]$. Khi đó ta nói quan hệ r thoả phụ thuộc hàm X $\rightarrow$ Y.

1.3. Hệ tiên đề Armstrong

Cho lược đồ quan hệ R(U) với U={A1, . . . , An}. X, Y, Z $\subseteq$ U. Khi đó ta có các tiên đề sau:

- Tính phản xạ: Nếu Y $\subseteq$ X thì X $\rightarrow$ Y;
- Tính tăng trưởng: Nếu X $\rightarrow$ Y thì XZ $\rightarrow$ YZ;
- Tính bắc cầu: Nếu X $\rightarrow$ Y và Y $\rightarrow$ Z thì X $\rightarrow$ Z.

I. PHỤ THUỘC HÀM

Các luật được suy ra từ hệ tiên đề Armstrong

- Luật hợp: $X \rightarrow Y$ và $X \rightarrow Z$ thì $X \rightarrow YZ$;
- Luật tựa bắc cầu: Nếu $X \rightarrow Y$ và $WY \rightarrow Z$ thì $XW \rightarrow Z$;
- Luật tách: Nếu $X \rightarrow Y$ và $Z \subseteq Y$ thì $X \rightarrow Z$.

1.4. Bao đóng của tập phụ thuộc hàm

Gọi F là tập tất cả các phụ thuộc hàm của lược đồ R(U). $X, Y \subseteq U$. $X \rightarrow Y$ là phụ thuộc hàm. Ta nói rằng phụ thuộc hàm $X \rightarrow Y$ được suy dẫn logic từ F nếu mỗi quan hệ r trên R(U) thoả các phụ thuộc hàm của F thì cũng thoả $X \rightarrow Y$.

Tập F^+={Tập tất cả các phụ thuộc hàm được suy dẫn logic từ F}

⇨ Được gọi là bao đóng của tập phụ thuộc hàm F.

I. PHỤ THUỘC HÀM

Ví dụ 1:

F={A→BC, C→D}

A→D có phải là một suy dẫn logic từ F?

Chứng minh

A→A
A→BC
} => A→ABC
C→D
} => A→ABCD =>A→D □

I. PHỤ THUỘC HÀM

Ví dụ 2:

Cho lược đồ quan hệ R, quan hệ r xác định trên R và tập phụ thuộc hàm

$F = \{AB \rightarrow C, B \rightarrow D, CD \rightarrow E, CE \rightarrow GH, G \rightarrow A\}$

Chứng minh rằng nếu quan hệ r trên lược đồ R thoả F thì r cũng thoả $AB \rightarrow E$, $AB \rightarrow G$.

I. PHỤ THUỘC HÀM

1.5. Bao đóng của tập thuộc tính

Cho tập thuộc tính U, $X \subset U$ và tập phụ thuộc hàm F. Bao đóng của X đối với tập phụ thuộc hàm F là tập

$$X^+ = \{A \in U \mid (X \rightarrow A) \in F^+\}$$

Thuật toán tìm bao đóng tập thuộc tính:

Vào: Lược đồ quan hệ R(U), tập PTH F, tập thuộc tính $X \subset U$

Ra: $X^+ = \{A \in U \mid X \rightarrow A \in F^+\}$

Cách tính:

- Đặt $X^{(0)} = X$
- $X^{(i)} = \begin{cases} X^{(i-1)} \cup A & \text{nếu tồn tại } (Y \rightarrow Z) \in F^+ \text{ mà } A \in Z \text{ và } Y \subseteq X^{(i-1)} \\ X^{(i-1)} & \text{nếu ngược lại} \end{cases}$

Bổ đề: $X \rightarrow Y \Leftrightarrow Y \subseteq X^+$

I. PHỤ THUỘC HÀM

Ví dụ:

Cho lược đồ quan hệ R, quan hệ r xác định trên R và tập phụ thuộc hàm

$F = \{AB \rightarrow C, B \rightarrow D, CD \rightarrow E, CE \rightarrow GH, G \rightarrow A\}$

Tính AB^+ ?

Giải

I. PHỤ THUỘC HÀM

1.6. Định nghĩa siêu khoá, khoá

Cho lược đồ quan hệ R(U). F là tập phụ thuộc hàm của R.

- K được gọi là siêu khoá của lược đồ R nếu $K \rightarrow U$
- K được gọi là khoá của lược đồ R nếu:
  - $K \rightarrow U$
  - Với $\forall K' \subset K$ thì $K' \not\rightarrow U$

I. PHỤ THUỘC HÀM

Thuật toán tìm khoá:

Vào: Lược đồ quan hệ R(U), U={A_1, A_2, ... , A_n } và tập phụ thuộc hàm F.

Ra : K là một khoá của R.

Cách tính:

- Bước 0: Đặt $K^{(0)} = U$
- Bước i: $K^{(i)} = \begin{cases} K^{(i-1)} \setminus \{A_i\} & \text{nếu } K^{(i-1)} \setminus \{A_i\} \rightarrow U \\ K^{(i-1)} & \text{nếu ngược lại} \end{cases}$

I. PHỤ THUỘC HÀM

1.7. Phủ tối thiểu

1.7.1. Tập phụ thuộc hàm tương đương

Cho hai tập phụ thuộc hàm F và G. Ta nói F và G tương đương. Ký hiệu F~ G, nếu $F^+ = G^+$. Khi đó ta còn gọi là G **phủ** F (và F **phủ** G).

Cách kiểm tra F tương đương với G

- Lấy mỗi phụ thuộc hàm $(X \rightarrow Y) \in F$ và kiểm tra $(X \rightarrow Y) \in G^+$.
- Lấy mỗi phụ thuộc hàm $(X \rightarrow Y) \in G$ và kiểm tra $(X \rightarrow Y) \in F^+$.

I. PHỤ THUỘC HÀM

Ví dụ:

Cho tập phụ thuộc hàm

$$F=\{A \rightarrow BCDE,\ C \rightarrow D,\ EG \rightarrow H,\ AE \rightarrow H\}$$

và $G=\{A \rightarrow B,\ A \rightarrow C,\ A \rightarrow E,\ C \rightarrow D,\ EG \rightarrow H,\ A \rightarrow H\}$

Chứng minh rằng F~G ?

I. PHỤ THUỘC HÀM

1.7.2. Tập phụ thuộc hàm tối thiểu

Tập phụ thuộc hàm F được gọi là tối thiểu nếu:

i) Vế phải của mỗi phụ thuộc hàm thuộc F chỉ có một thuộc tính.

(Tính không dư thừa phải).

Ví dụ:

Cho F={**A $\rightarrow$ BCDE**, C $\rightarrow$D, EG $\rightarrow$ H, AE $\rightarrow$ H}

Ta có

F_1={A $\rightarrow$ B, A $\rightarrow$ C, A $\rightarrow$ D, A $\rightarrow$ E, C $\rightarrow$D, EG $\rightarrow$ H, AE $\rightarrow$ H}

I. PHỤ THUỘC HÀM

1.7.2. Tập phụ thuộc hàm tối thiểu

ii) Không tồn tại phụ thuộc hàm $(X \rightarrow A) \in F$ mà $F^+ = (F \setminus \{X \rightarrow A\})^+$.

(Tính không dư thừa phụ thuộc hàm).

Ví dụ:

$$F_1=\{A \rightarrow B,\ A \rightarrow C,\ \mathbf{A \rightarrow D},\ A \rightarrow E,\ C \rightarrow D,\ EG \rightarrow H,\ AE \rightarrow H\}$$

Ta có

$$F_2=\{A \rightarrow B,\ A \rightarrow C,\ A \rightarrow E,\ C \rightarrow D,\ EG \rightarrow H,\ AE \rightarrow H\}$$

I. PHỤ THUỘC HÀM

1.7.2. Tập phụ thuộc hàm tối thiểu

iii) Không tồn tại phụ thuộc hàm $(X \rightarrow A) \in F$ mà

$F^+ = (F \setminus \{X \rightarrow A\} \cup \{Z \rightarrow A\})^+$ với $Z \subset X$

(Tính không dư thừa trái)

Ví dụ:

$F_2 = \{A \rightarrow B, A \rightarrow C, A \rightarrow E, C \rightarrow D, EG \rightarrow H, AE \rightarrow H\}$

Ta có

$F_3 = \{A \rightarrow B, A \rightarrow C, A \rightarrow D, A \rightarrow E, C \rightarrow D, EG \rightarrow H, A \rightarrow H\}$

I. PHỤ THUỘC HÀM

1.7.3. Phủ tối thiểu

Cho tập phụ thuộc hàm F. Tập phụ thuộc hàm F’ là **phủ tối thiểu** của F nếu:

- F’ là tập phụ thuộc hàm tối thiểu
- F’ ~ F.

I. PHỤ THUỘC HÀM

Thuật toán tìm phủ tối thiểu của tập phụ thuộc hàm:

Vào: Lược đồ quan hệ R(U), tập phụ thuộc hàm F = $\{L_i \rightarrow R_i , i=1,..., m\}$

Ra: Tập phụ thuộc hàm F' là phủ tối thiểu của F

Cách thực hiện:

i) Xác định tập phụ thuộc hàm F1 ~ F với F1 không dư thừa phải bằng cách tách các phụ thuộc hàm dạng Li $\rightarrow A_1 ... A_k$ thành các phụ thuộc hàm $L_i \rightarrow A_1, ..., L_i \rightarrow A_k$.

Ví dụ:

Cho F={**A $\rightarrow$ BCDE**, C $\rightarrow$D, EG $\rightarrow$ H, AE $\rightarrow$ H}

Ta có

F_1={**A $\rightarrow$ B, A $\rightarrow$ C, A $\rightarrow$ D, A $\rightarrow$ E**, C $\rightarrow$D, EG $\rightarrow$ H, AE $\rightarrow$ H}

*(Thay PTH **A $\rightarrow$ BCDE** bằng các PTH **A $\rightarrow$ B, A $\rightarrow$ C, A $\rightarrow$ D, A $\rightarrow$ E**)*

I. PHỤ THUỘC HÀM

Thuật toán tìm phủ tối thiểu của tập phụ thuộc hàm:

ii) Xác định tập phụ thuộc hàm $\mathbf{F_2} \sim \mathbf{F_1}$ không dư thừa phụ thuộc hàm theo thuật toán sau:

+ Bước 0: Đặt $F^{(0)} = \mathbf{F_1}$

+ Bước i : Tính $F^{(i)} = \begin{cases} F^{(i-1)} \setminus \{L_i \to R_i\} & \text{nếu } F^{(i-1)} \setminus \{L_i \to R_i\} \sim F^{(i-1)} \\ F^{(i-1)} & \text{nếu ngược lại} \end{cases}$

Ví dụ:

$\mathbf{F_1}$={A $\to$ B, A $\to$ C, **A $\to$ D**, A $\to$ E, C $\to$D, EG $\to$ H, AE $\to$ H}

Ta có

$\mathbf{F_2}$={A $\to$ B, A $\to$ C, A $\to$ E, C $\to$D, EG $\to$ H, AE $\to$ H}

I. PHỤ THUỘC HÀM

Thuật toán tìm phủ tối thiểu của tập phụ thuộc hàm:

iii) Xác định tập phụ thuộc hàm $\mathbf{F_3 \sim F_2}$ không dư thừa trái bằng thuật toán sau:

+ Bước 0: Đặt $F^{(0)}=F_2$

+ Bước i : Tính $F^{(i)} = \begin{cases} F^{(i-1)} \setminus \{L_i \to R_i\} \cup \{Z_i \to R_i\} \text{ nếu tồn tại } Z_i \subset L_i \\ \quad \text{mà } F^{(i-1)} \setminus \{L_i \to R_i\} \cup \{Z_i \to R_i\} \sim F^{(i-1)} \\ F^{(i-1)} \quad \text{nếu ngược lại} \end{cases}$

Ví dụ:

$\mathbf{F_2}=\{A \to B, A \to C, A \to E, C \to D, EG \to H, AE \to H\}$

Ta có

$\mathbf{F_3}=\{A \to B, A \to C, A \to E, C \to D, EG \to H, A \to H\}$

II. PHÉP TÁCH CÁC LƯỢC ĐỒ QUAN HỆ

2.1 Khái niệm phép tách lược đồ quan hệ

Cho lược đồ quan hệ R(U).

Tập các lược đồ $\rho=\{R_1(U_1), R_2(U_2), ...,R_m(U_m)\}$ được gọi là phép tách của R nếu $U_1 \cup U_2 \cup ... \cup U_m=U$.

Nếu ρ là phép tách của R, ta ký hiệu:

$$\rho\ (R_1, R_2, ..., R_m\ )$$

hay $\rho=(R_1, R_2, ..., R_m)$.

II. PHÉP TÁCH CÁC LƯỢC ĐỒ QUAN HỆ

2.2. Phép tách không mất mát thông tin

Cho lược đồ quan hệ R và tập phụ thuộc hàm F. Phép tách lược đồ R thành tập các lược đồ $R_1, \ldots, R_m$ được gọi là không mất mát thông tin đối với F nếu với mỗi quan hệ r trên R thoả F thì

$$\Pi_{U1}(r) * \Pi_{U2}(r) * \ldots * \Pi_{Um}(r) = r$$

III. CHUẨN HOÁ LƯỢC ĐỒ QUAN HỆ

3.1. Các khái niệm

- Chuẩn hoá lược đồ quan hệ là quá trình biến đổi các lược đồ quan hệ thành các dạng thích hợp để tránh các dị thường về dữ liệu.
- *Lược đồ được chuẩn hoá* là lược đồ trong đó miền trị của mỗi thuộc tính chỉ chứa các giá trị nguyên tố (không phân nhỏ được nữa), do đó mỗi giá trị trong các quan hệ cũng là các giá trị nguyên tố.
- Lược đồ có chứa các miền trị không nguyên tố được gọi là *lược đồ không chuẩn hoá*.

III. CHUẨN HOÁ LƯỢC ĐỒ QUAN HỆ

3.2. Một số định nghĩa

Định nghĩa 1:

Cho lược đồ quan hệ R trên tập thuộc tính U = { A_1, A_2,... , A_n}. Thuộc tính $A \in U$ được gọi là *thuộc tính khoá* nếu A là thành phần của một khoá nào đó của R. Ngược lại A được gọi là *thuộc tính không khoá*.

Ký hiệu $\mathbf{F_n}$ là tập *thuộc tính không khóa.*

Ví dụ: Cho lược đồ quan hệ R(U) với U={ A, B, C, D, E}

$$F=\{AB \rightarrow CE,\ B \rightarrow D,\ BC \rightarrow A\}.$$

Các khóa của R là K_1=AB, K_2= BC

☞ **A, B, C** là *thuộc tính khoá*

F_n={D, E}.

III. CHUẨN HOÁ LƯỢC ĐỒ QUAN HỆ

3.2. Một số định nghĩa

Định nghĩa 2:

Cho lược đồ quan hệ R(U). $X \subseteq U$, $A \in U$; A được gọi là *phụ thuộc hàm đầy đủ* vào X nếu A phụ thuộc hàm vào X (nghĩa là $X \rightarrow A$) nhưng A không phụ thuộc hàm vào bất kỳ tập con thực sự nào của X (nghĩa là với $\forall X' \subset X$ thì $X' \not\rightarrow A$).

Ví dụ: Cho lược đồ quan hệ R(U) với U={ A, B, C, D, E}

$F=\{AB \rightarrow CE,\ B \rightarrow D,\ BC \rightarrow A\}$.

☞ Thuộc tính E là *phụ thuộc hàm đầy đủ* vào AB

Thuộc tính D là không *phụ thuộc hàm đầy đủ* vào AB (hay D *phụ thuộc hàm bộ phận* vào AB)

III. CHUẨN HOÁ LƯỢC ĐỒ QUAN HỆ

3.2. Một số định nghĩa

Định nghĩa 3:

Cho lược đồ quan hệ R(U). $X \subseteq U$, $A \in U$. A được gọi là *phụ thuộc bắc cầu* vào X nếu tồn tại $Y \subset U$ sao cho $X \rightarrow Y$, $Y \rightarrow A$ nhưng $Y \not\rightarrow X$ với $A \notin XY$.

Ví dụ: Cho lược đồ quan hệ R(U) với U={ A, B, C, D, E}

F={$A \rightarrow BCE$, $B \rightarrow DC$}.

☞ Thuộc tính D,C là *phụ thuộc hàm bắc cầu* vào A

III. CHUẨN HOÁ LƯỢC ĐỒ QUAN HỆ

3.3. Các dạng chuẩn (Normal Forms)

- Dạng chuẩn 1 (1NF)
- Dạng chuẩn 2 (2NF)
- Dạng chuẩn 3 (3NF)

III. CHUẨN HOÁ LƯỢC ĐỒ QUAN HỆ

- **Dạng chuẩn 1 (1NF)**

  Lược đồ quan hệ R được gọi là ở *dạng chuẩn 1 (1NF)* nếu toàn bộ các miền trị có mặt trong R đều chỉ chứa các giá trị nguyên tố.

Ví dụ:

Canbo

MACB	HOTEN	NGAYSINH	QUEQUAN	NGOAINGU	TRINHDO_NN
01	Lê Văn An	5/5/1977	Nghệ An	Tiếng Anh Tiếng Pháp	C A
02	Nguyễn Văn Bình	10/04/1974	Hà Tĩnh	Tiếng Anh Tiếng Nhật Tiếng Hàn	B B A

Không ở dạng 1NF

Canbo

MACB	HOTEN	NGAYSINH	QUEQUAN	NGOAINGU	TRINHDO_NN
01	Lê Văn An	5/5/1977	Nghệ An	Tiếng Anh	C
01	Lê Văn An	5/5/1977	Nghệ An	Tiếng Pháp	A
02	Nguyễn Văn Bình	10/04/1974	Hà Tĩnh	Tiếng Anh	B
02	Nguyễn Văn Bình	10/04/1974	Hà Tĩnh	Tiếng Nhật	B
02	Nguyễn Văn Bình	10/04/1974	Hà Tĩnh	Tiếng Hàn	A

Ở dạng 1NF

Dư thừa dữ liệu

III. CHUẨN HOÁ LƯỢC ĐỒ QUAN HỆ

- **Dạng chuẩn 2 (2NF)**

  Lược đồ quan hệ R được gọi là *ở dạng chuẩn 2 (2NF)* nếu:

  – R ở dạng chuẩn 1

  – Mọi thuộc tính không khoá của R đều phụ thuộc hàm đầy đủ vào khoá.

Ví dụ:

Cho lược đồ quan hệ R(U) với:

U=ABCDEGH, F={A $\rightarrow$ BC, D$\rightarrow$EG, AD$\rightarrow$H}

R(U) có ở dạng chuẩn 2 không?

III. CHUẨN HOÁ LƯỢC ĐỒ QUAN HỆ

- **Dạng chuẩn 3 (3NF)**

  Lược đồ quan hệ R được gọi là *ở dạng chuẩn 3 (3NF)* nếu:

  - R ở dạng chuẩn 2
  - Mọi thuộc tính không khoá của R đều không phụ thuộc bắc cầu vào khoá của R.

Ví dụ:

Cho lược đồ quan hệ R(U) với:

$$U=ABCDEG,\ F=\{A \rightarrow BCD,\ D \rightarrow EG\}$$

R(U) có ở dạng chuẩn 3 không?

III. CHUẨN HOÁ LƯỢC ĐỒ QUAN HỆ

3.4. Phép tách một lược đồ quan hệ thành các lược đồ 3NF không mất mát thông tin

- **Vào:** Lược đồ quan hệ R(U). Tập phụ thuộc hàm F.
- **Ra:** $\rho = \{R_1(U_1), ..., R_m(U_m) \mid R_i(U_i), i=1, ..., m \text{ là 3NF}\}$

  với phép tách không mất mát thông
- **Phương pháp:**
  - ***Bước 1:*** Tìm phủ tối thiểu F' của F

    (Loại sự dư thừa trong tập phụ thuộc hàm)
  - ***Bước 2: (Tách thành các lược đồ 3NF)***
    1. Đặt $U_0 = X$ với X là tập các thuộc tính không xuất hiện trong bất kỳ một phụ thuộc hàm nào (cả vế trái lẫn vế phải) của F'. Nếu $U_0 \neq \varnothing$ thì lấy $R_0(U_0)$.
    2. Xác định Ri (i>0): Mỗi nhóm PTH có cùng vế trái dạng:

       $L_i \rightarrow A_{i1}, ..., L_i \rightarrow A_{ik}$ thì xác định $R_i(L_i A_{i1} \ldots A_{ik})$

    Ta có $\rho\ (R_i(U_i), R_0(U_0))$ là phép tách thành 3NF

III. CHUẨN HOÁ LƯỢC ĐỒ QUAN HỆ

3.4. Phép tách một lược đồ quan hệ thành các lược đồ 3NF không mất mát thông tin

- ***Bước 3*: (Chuyển thành phép tách không mất mát thông tin)**

1. Tìm một khoá K của R(U)
2. Kiểm tra K
   - i. Nếu tồn tại U_i mà $K \subseteq U_i$ thì $\rho(R_i(U_i), R_0(U_0))$ là phép tách thành 3NF không mất mát thông tin của R(U)
   - ii. Nếu không tồn tại Ui thoả i) nhưng tồn tại U_i mà $K \subseteq U_0 \cup U_i$ thì ta có: $\rho(R_i(U_i), R_0(K)$ với i >0) (thay U_0 =K) là phép tách thành 3NF không mất mát thông tin của R(U)
   - iii. Nếu không tồn tại U_i thoả ii) thì $\rho(R_i(U_i), R_0(U_0), R(K))$ là phép tách thành 3NF không mất mát thông tin của R(U).

Chương 4: Tầng Mạng

Mục tiêu

- Nguyên tắc triển khai dịch vụ của tầng Mạng:
  - Định tuyến (Lựa chọn đường đi)
  - Mở rộng Phạm vi
  - Cách thức router hoạt động
  - Nâng cao: IPv6, multicast
- Cài đặt những dịch vụ này trên Internet như thế nào

Học cái gì:

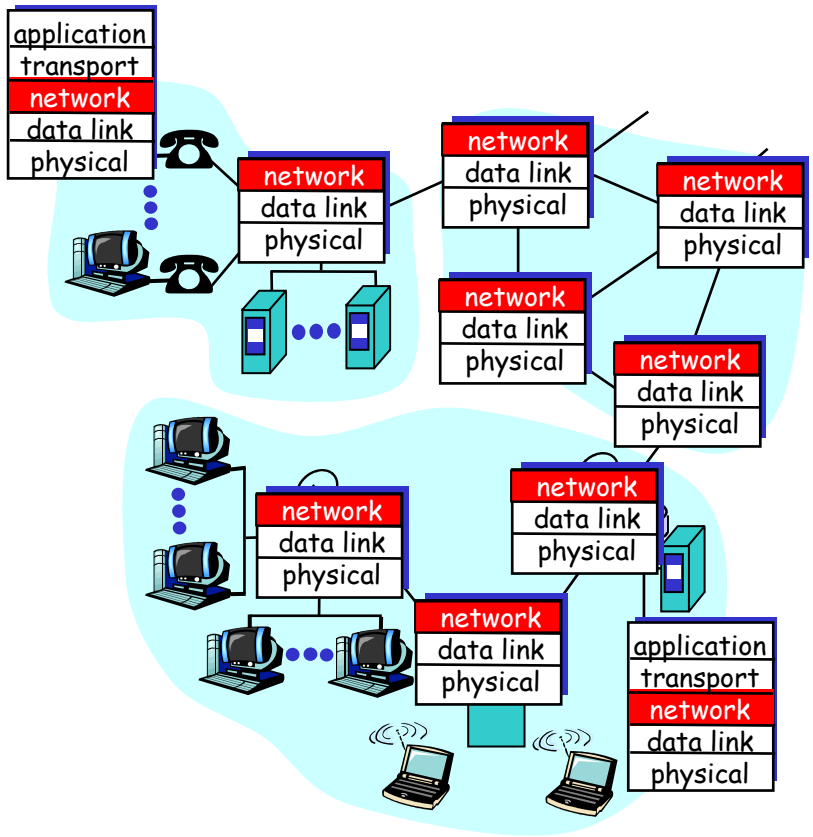
- Các dịch vụ của tầng Mạng
- Nguyên tắc định tuyến: lựa chọn đường đi
- Định tuyến phân cấp
- Giao thức IP
- Giao thức định tuyến trên Internet
  - Nội miền
  - Liên miền
- Kiến trúc Router
- IPv6
- NAT, PAT
- multicast routing

Chức năng của tầng Mạng

- Chuyển gói tin từ máy tính gửi đến máy tính nhận
- Giao thức tầng Mạng có mặt trên ***tất cả*** máy tính, router

Ba chức năng cơ bản:

- ***Lựa chọn tuyến đường:*** Tuyến đường gói tin đi từ nguồn đến đích. *Thuật toán định tuyến*
- ***Chuyển mạch:*** Router chuyển gói tin từ một đầu vào này ra đầu ra thích hợp
- ***Thiết lập tuyến đường:*** một số kiến trúc mạng đòi hỏi tuyến đường gói tin đi qua phải được thiết lập trước



Mô hình Dịch vụ tầng Mạng

?: Chọn *Mô hình dịch vụ* nào của "kênh truyền" để chuyển gói dữ liệu từ nơi gửi đến nơi nhận ?

- Đảm bảo băng thông không?
- Đảm bảo tốc độ nhận đều không (không có hiện tượng jitter)?
- Không mất mát ?
- Đảm bảo thứ tự?
- Có thông báo về tình trạng tắc nghẽn cho bên gửi không ?

Trừu tượng hóa dịch vụ

Vấn đề quan trọng nhất mà tầng Mạng phải trả lời :

Mạch ảo hay Mạch gói?

Chuyển mạch ảo

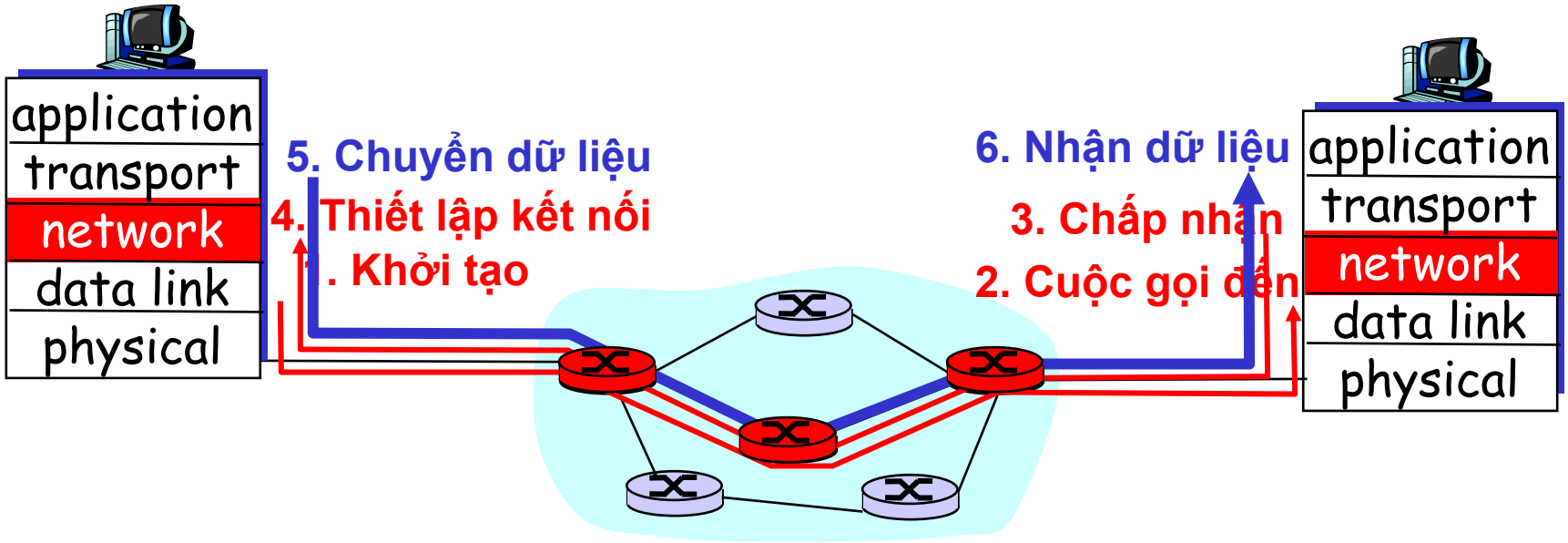
Tuyến đường từ đích đến nguồn giống như mạch trong mạng điện thoại

- Quan trọng : Hiệu suất
- Thiết lập kết nối dọc theo tuyến đường

- Thiết lập và Đóng kết nối *trước* khi truyền dữ liệu
- Mỗi packet mang định danh mạch ảo là VC identifier (Không phải địa chỉ máy đích)
- *Tất cả* router trên tuyến đường duy trì trạng thái của kết nối đi qua
  - Kênh truyền ở tầng giao vận chỉ liên quan đến hai thực thể đầu cuối
- Tài nguyên của đường truyền (Băng thông, Bộ đệm) phải được cấp phát cho mỗi mạch ảo
  - Để đạt được hiệu suất như mạng điện thoại.

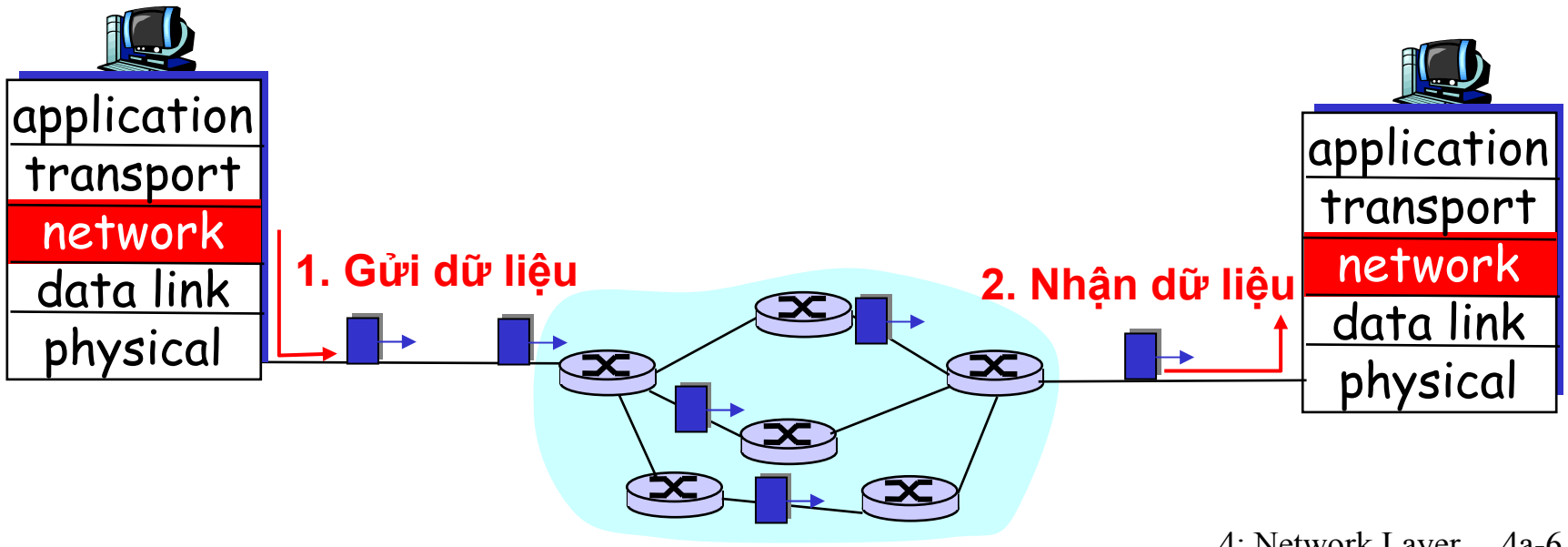
Mạch ảo: Giao thức báo hiệu (Signal)

- Để Thiết lập và Đóng mạch ảo (VC)
- Có trong ATM, Frame-relay, X.25
- Không được sử dụng trên Internet ngày nay



Mạng chuyển mạch gói: Mô hình Internet

- Không cần thiết lập đường truyền ở tầng Mạng
- routers: Không duy trì trạng thái các kết nối đi qua
  - no network-level concept of "connection"
- Gói tin được định tuyến dựa trên địa chỉ gói nhận
  - Các gói tin có thể đi theo những tuyến đường khác nhau



Mô hình Dịch vụ ở tầng Mạng

Kiến trúc Mạng	Mô hình Dịch vụ	Có đảm bảo ?				Phản hồi Tắc nghẽn
		Băng thông	Mất	Thứ tự	Thời gian	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

- Mô hình dịch vụ trên Internet : Intserv, Diffserv
  - Chương 6

Chuyển Mạch hay Chuyển Gói?

Internet

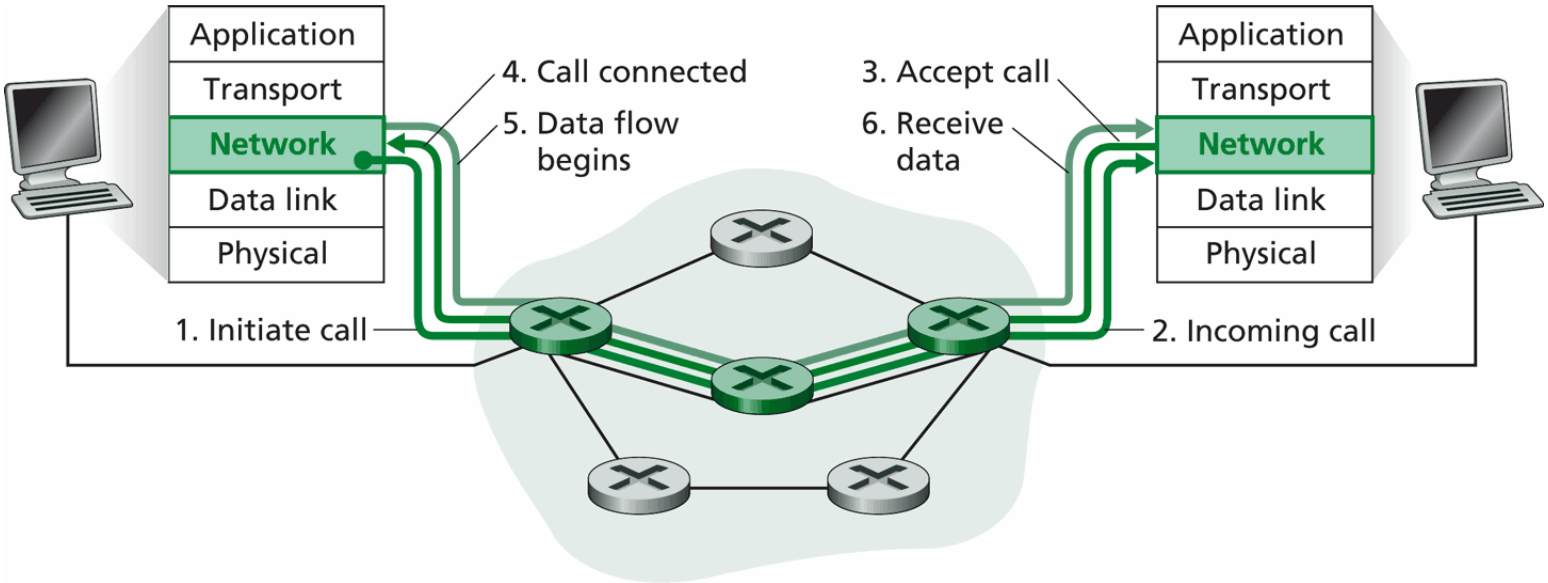
- Dữ liệu trao đổi giữa máy tính
  - Dịch vụ mang tính "co giãn", không đòi hỏi chặt chẽ thời gian.
- Thiết bị đầu cuối "thông minh"
  - Có thể thích nghi, Kiểm soát, Khắc phục lỗi
  - "Lõi" mạng đơn giản, Phức tạp đặt ở "Rìa"
- Nhiều kiểu môi trường truyền dẫn
  - Các đặc điểm khác nhau
  - Không thể có chung chất lượng dịch vụ

ATM

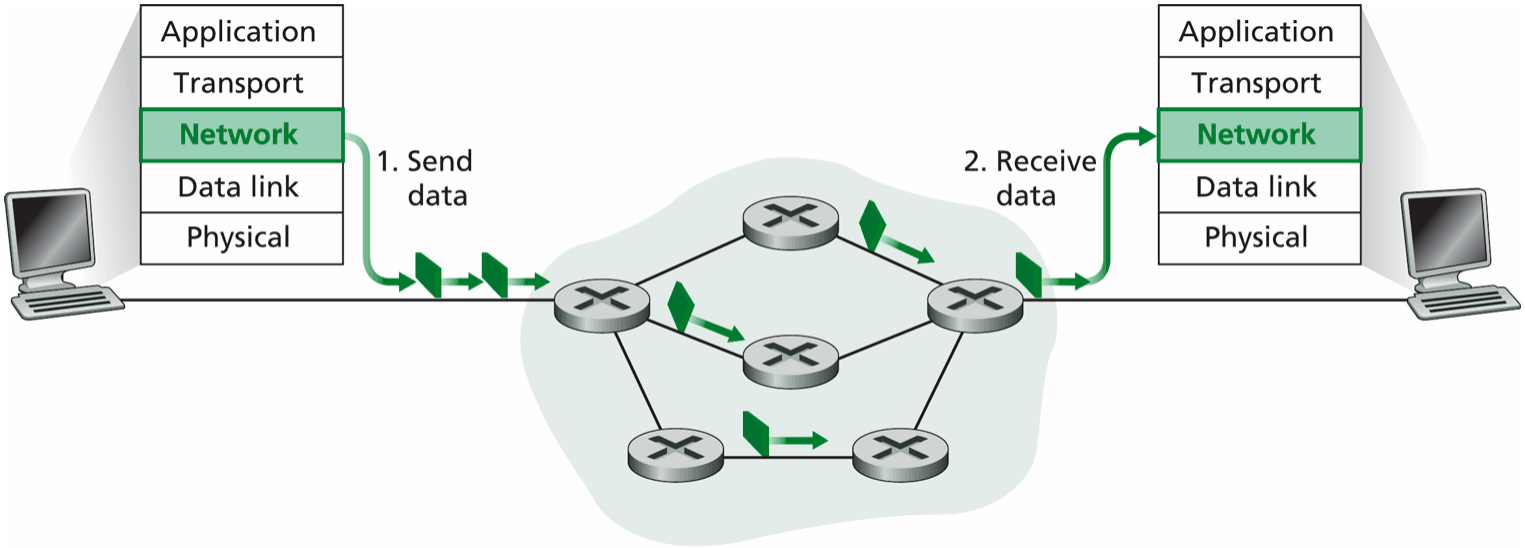
- Phát triển từ Mạng điện thoại
- Hội thoại của con người:
  - Yêu cầu nghiêm ngặt về Thời gian và Độ tin cậy
  - Đảm bảo chất lượng dịch vụ
- Thiết bị đầu cuối đơn giản
  - Máy điện thoại
  - Phức tạp đặt ở bên trong Mạng

Sử dụng Mạch ảo để cài đặt Dịch vụ ở tầng Mạng

- Để cung cấp thêm một số chức năng, có thể lựa chọn công nghệ mạch ảo – ví dụ Virtual Private Network (VPN)



Chuyển mạch gói : Cài đặt các chức năng Mạng cơ bản nhất

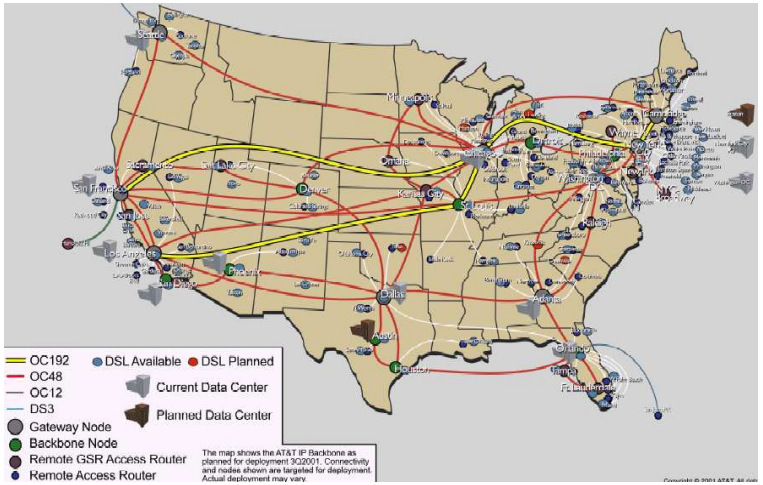


- Thường chỉ cung cấp dịch vụ cơ bản nhất: chuyển gói tin đi từ nơi Gửi đến nơi Nhận.
- Tập trung vào mô hình Internet theo kiểu "cố gắng tối đa" này

Định tuyến

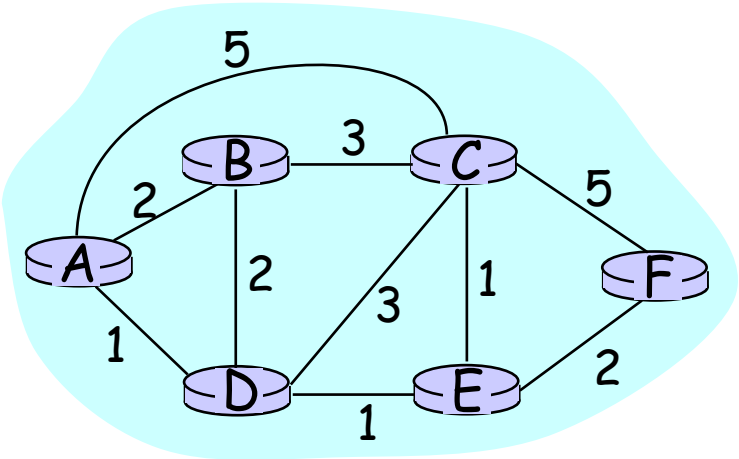
Định tuyến

Mục tiêu: xác định tuyến đường "tốt" (dãy các router) trên mạng từ nút gửi đến nút nhận.

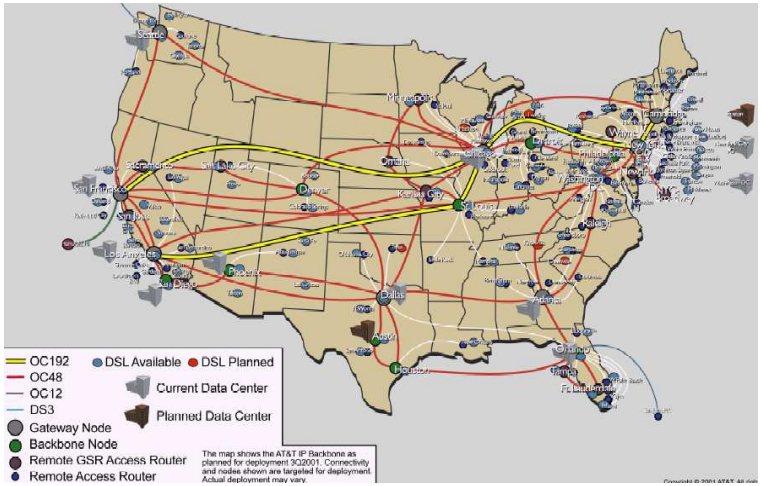


Đồ thị để xây dựng thuật toán định tuyến:

- Đỉnh là Router
- Cạnh là các đường kết nối trực tiếp
  - Giá của cạnh: Độ trễ, Chi phí, hay Mức độ Tắc nghẽn



Định tuyến : Các Yêu cầu khác

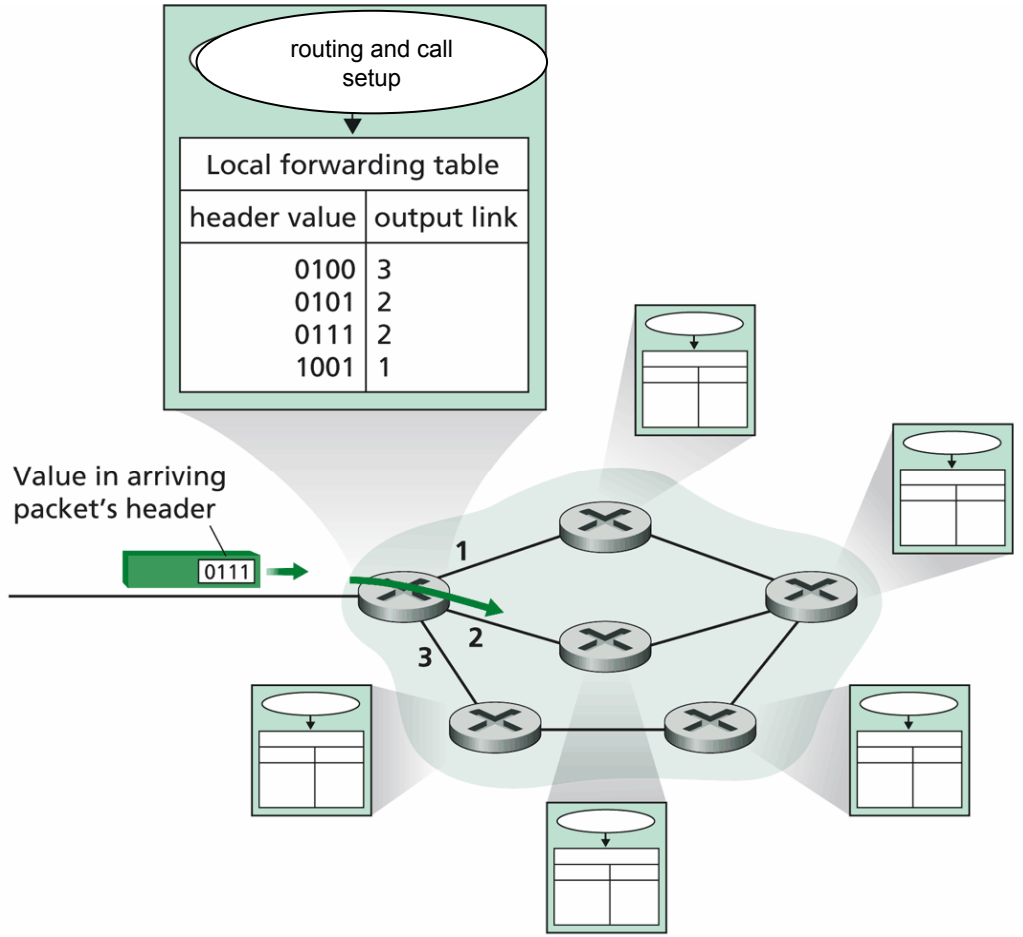


- Một vài Yêu cầu khác
  - Mạnh và Tin cậy
  - Tối ưu và có Hiệu quả
  - Phân tán, Tự quản, Không có điểm hỏng duy nhất
  - Đơn giản
  - Công bằng

Định tuyến

Mục tiêu: xác định tuyến đường "tốt" (dãy các router) trên mạng từ nút gửi đến nút nhận.

Minh họa Chức năng Tầng Mạng



Không gian Định tuyến

- Định tuyến
  - Ai quyết định tuyến đường?
    - Nút gửi quyết định
    - Mạng quyết định
  - Có bao nhiêu tuyến đường từ nút Gửi s đến nút Nhận d?
    - Định tuyến nhiều chặng
    - Định tuyến nhiều chặng
  - Tuyến đường có thay đổi theo tải của mạng không?
    - Định tuyến thích nghi
    - Định tuyến tĩnh
  - …

Phân loại Thuật toán Định tuyến

Thông tin *Toàn cục* hay *Phân tán* ?

Toàn cục (Global):

- Mọi router đều biết về toàn bộ topo của đồ thị
- Thuật toán "link state"

Phân tán (Decentralized):

- Mỗi router chỉ biết về router và giá đến các router hàng xóm
- Thực hiện quá trình trao đổi thông tin với các "hàng xóm"
- Thuật toán "distance vector"

Động hay Tĩnh?

Tĩnh (Static):

- Router rất ít thay đổi

Động (Dynamic):

- Router thay đổi thường xuyên
  - Cập nhật định kỳ
  - Chạy lại thuật toán khi có 1 giá đường đi thay đổi

Thuật toán Định tuyến Link-State

Thuật toán Dijkstra

- Tất cả các nút đều biết được về topo của toàn mạng
  - Biết được do các thông điệp quảng cáo được gửi quảng bá
  - Tất cả các nút có thông tin giống nhau
- Tính toán đường đi tốt nhất đến tất cả các nút khác
  - Tạo ra **Bảng Định tuyến**
- Sau k bước tính toán, xác định được đường ngắn nhất tới k đích

Ký hiệu:

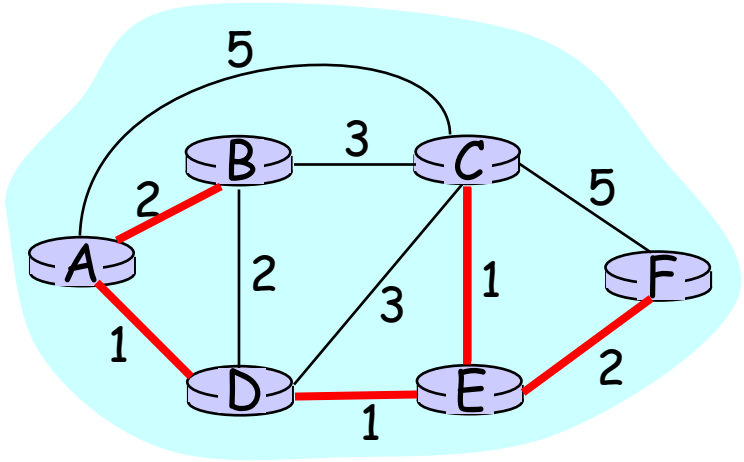
- c(i,j): Giá đường đi từ i tới j. Có giá trị vô cùng nếu i và j không có đường trực tiếp
- D(v): Giá hiện tại của đường đi tới đích V
- p(v): Nút kề trước V trong tuyến đường đến V
- N: Tập hợp các đỉnh đã xác định được đường đi ngắn nhất

Thuật toán Dijsktra

```
1  Khởi tạo:
2    N = {A}
3    Với tất cả các nút v
4      Nếu v kề với A
5        thì D(v) = c(A,v)
6        nếu không D(v) = infty
7
8  Lặp
9      Tìm w không trong N sao cho D(w) nhỏ nhất
10     Bổ sung w vào N
11     Cập nhật D(v) cho tất cả v kề với w và không trong N:
12       D(v) = min( D(v), D(w) + c(w,v) )
13     /* Giá mới tới v hoặc là giá cũ đến v hoặc đường đi ngắn nhất
14     đến w cộng thêm khoảng cách từ w tới v */
15 Cho đến khi tất cả các nút đều nằm trong N
```

Ví dụ về Thuật toán Dijsktra

Bước	Tập N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
→ 0	A	2,A	5,A	1,A	vô cùng	vô cùng
→ 1	AD	2,A	4,D		2,D	vô cùng
→ 2	ADE	2,A	3,E			4,E
→ 3	ADEB		3,E			4,E
→ 4	ADEBC					4,E
5	ADEBCF					



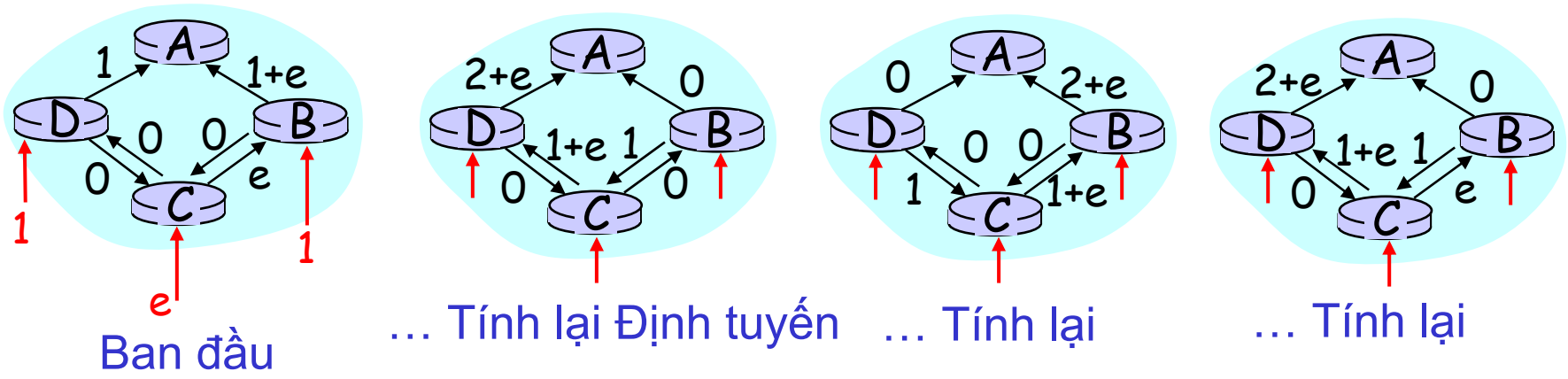
Thảo luận về Thuật toán Dijsktra

Độ phức tạp Thuật toán: n nút

- Mỗi vòng lặp: Cần kiểm tra tất cả các nút w không ở trong N
- n*(n+1)/2 : O(n**2)
- Bằng thuật toán "mịn" hơn: O(nlogn)

Tình huống Dao động:

- Ví dụ khi Chi phí kênh truyền = khối lượng dữ liệu truyền qua



Thuật toán Định tuyến Distance Vector

Lặp:

- Liên tục diễn ra cho đến khi không còn thông điệp trao đổi.
- *Tự kết thúc*: Không có "tín hiệu" để dừng lại

Không đồng bộ:

- Các nút sau khi gửi/ nhận thông điệp không bị phong tỏa!

Phân tán:

- Mỗi nút chỉ truyền thông với hàng xóm

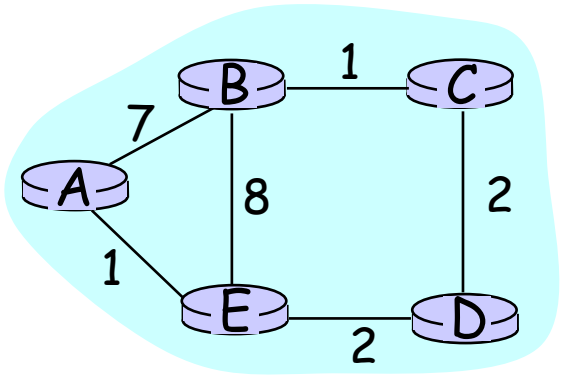
Cấu trúc Dữ liệu Bảng Distance

- Mỗi nút có một Bảng riêng
- Mỗi hàng ứng với một đích cụ thể
- Mỗi cột ứng với một hàng xóm có đường kết nối trực tiếp
- Ví dụ: trong nút X, với đích Y qua hàng xóm Z:

$$D^{X}(Y,Z) = \text{Khoảng cách } \textit{từ } X \textit{ tới } Y, \textit{ qua } Z \text{ là chặng tiếp}$$

$$= c(X,Z) + \min_{w}\{D^{Z}(Y,w)\}$$

Ví dụ Bảng Distance



$$D^{E}(C,D) = c(E,D) + \min_{w}\{D^{D}(C,w)\}$$
$$= 2+2 = 4$$

$$D^{E}(A,D) = c(E,D) + \min_{w}\{D^{D}(A,w)\}$$
$$= 2+3 = 5$$ **Lặp !**

$$D^{E}(A,B) = c(E,B) + \min_{w}\{D^{B}(A,w)\}$$
$$= 8+6 = 14$$ **Lặp !**

Chi phí đến các đích thông qua

Đích	$D^{E}()$ A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

Bảng Distance tạo ra Bảng định tuyến

Chí phí đến đích qua

$D^E()$

Đích	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

Đường ra được dùng + Giá

Đích	
A	A,1
B	D,5
C	D,4
D	D,4

Bảng Distance → Bảng định tuyến

Tổng quan Định tuyến Distance Vector

Lặp, Không đồng bộ: mỗi lần tính toán cục bộ là do:

- Chi phí một đường thay đổi
- Thông điệp từ hàng xóm: Đường đi ngắn nhất của hàng xóm cũng thay đổi

Phân tán:

- Mỗi nốt chỉ thông báo với hàng xóm *khi* có một giá đường đi nào đó thay đổi
  - Chỉ thông báo trong trường hợp cần phải thông báo

Mỗi nút

Thuật toán Distance Vector

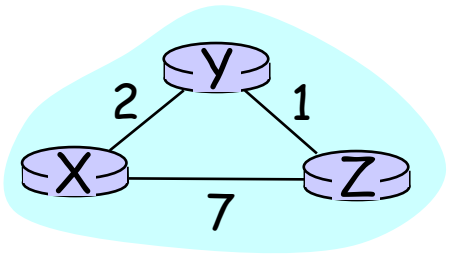
At all nodes, X:

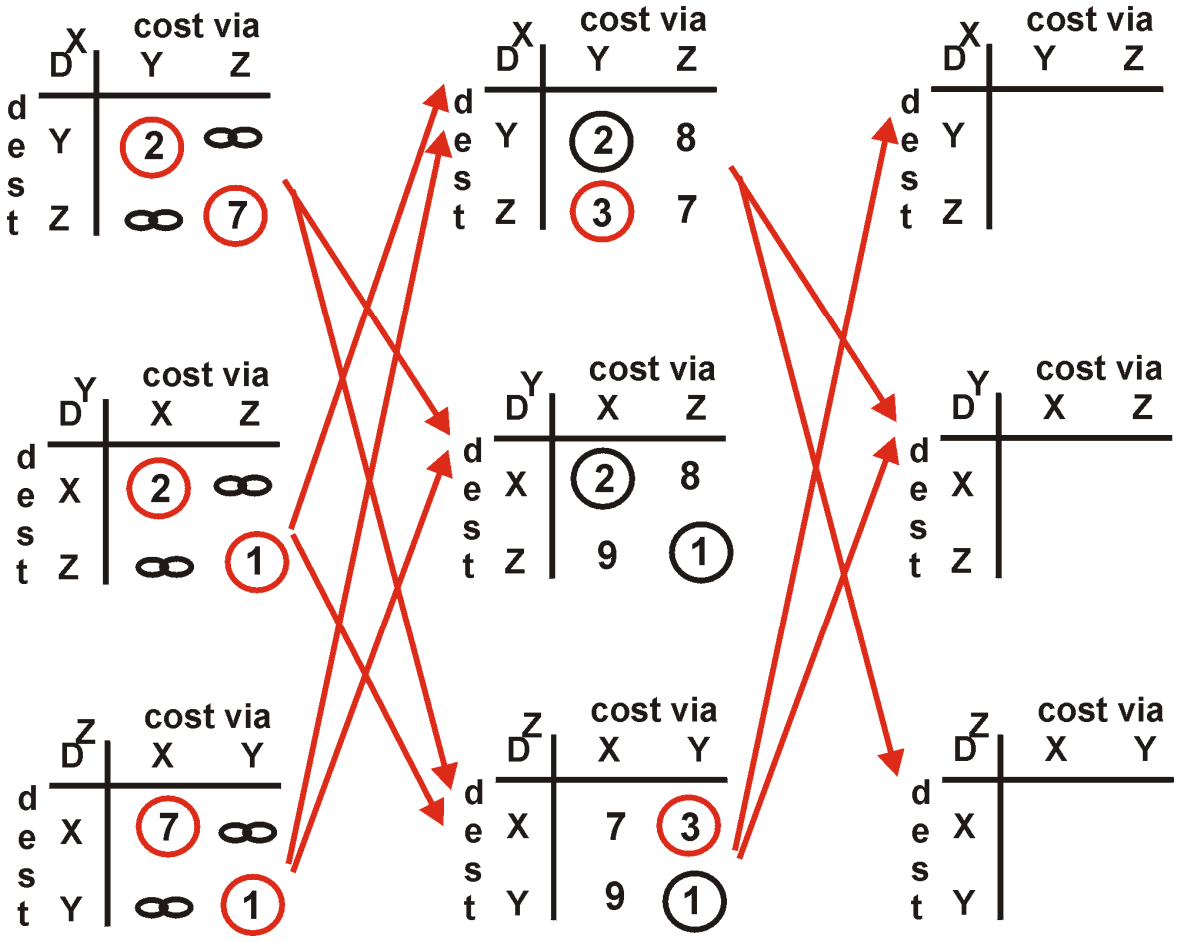
1 Initialization:
2 for all adjacent nodes v:
3 $D^X(*,v) = infty$ /* the * operator means "for all rows" */
4 $D^X(v,v) = c(X,v)$
5 for all destinations, y
6 send $\min_w D^X(y,w)$ to each neighbor /* w over all X's neighbors */

Thuật toán Distance Vector:

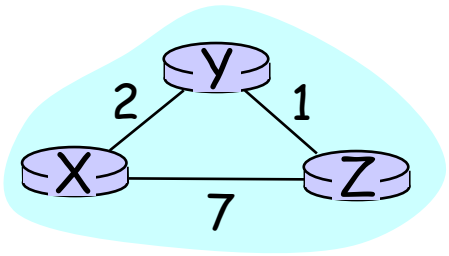
```
8  loop
9    wait (until I see a link cost change to neighbor V
10         or until I receive update from neighbor V)
11
12   if (c(X,V) changes by d)
13     /* change cost to all dest's via neighbor v by d */
14     /* note: d could be positive or negative */
15     for all destinations y:  D^X(y,V) =  D^X(y,V) + d
16
17   else if (update received from V wrt destination Y)
18     /* shortest path from V to some Y has changed  */
19     /* V has sent a new value for its  min_w DV(Y,w) */
20     /* call this received new value is "newval"     */
21     for the single destination y: D^X(Y,V) = c(X,V) + newval
22
23   if we have a new min_w D^X(Y,w)for any destination Y
24      send new value of min_w D^X(Y,w) to all neighbors
25
26 forever
```

Ví dụ Thuật toán Distance Vector





Ví dụ Thuật toán Distance Vector



D^X

dest	via Y	via Z
Y	2	∞
Z	∞	7

D^Y

dest	via X	via Z
X	2	∞
Z	∞	1

D^Z

dest	via X	via Y
X	7	∞
Y	∞	1

D^X

dest	via Y	via Z
Y	2	8
Z	3	7

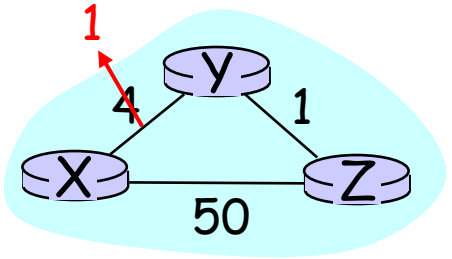
$$D^X(Y,Z) = c(X,Z) + \min_w\{D^Z(Y,w)\}$$
$$= 7+1 = 8$$

$$D^X(Z,Y) = c(X,Y) + \min_w\{D^Y(Z,w)\}$$
$$= 2+1 = 3$$

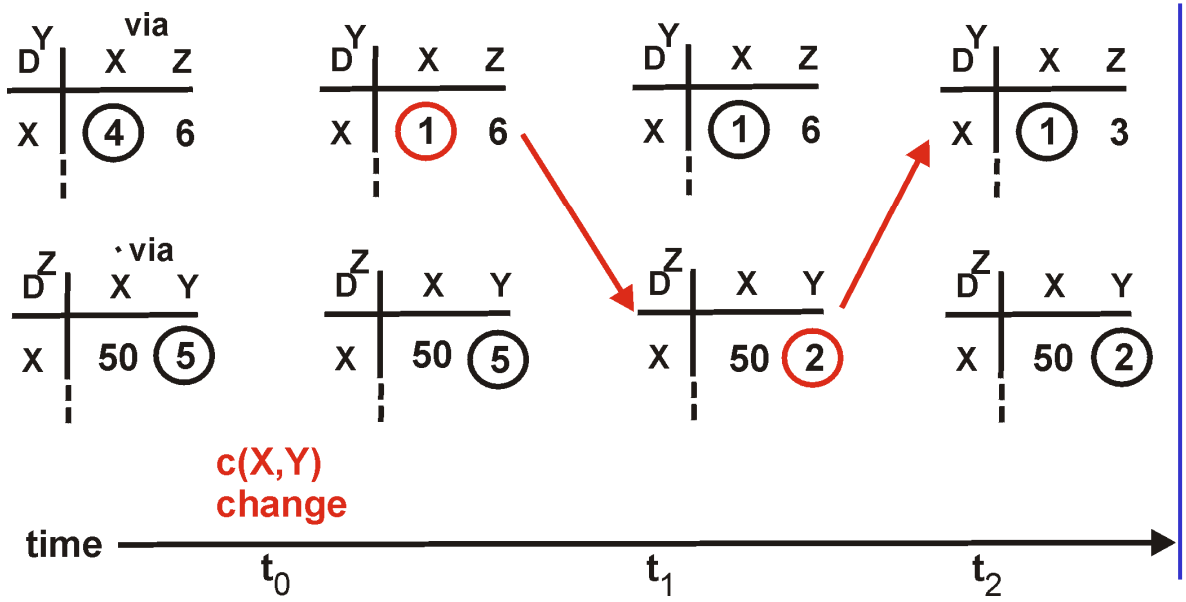
Distance Vector: Thay đổi giá đường đi

Thay đổi giá đường đi:

- Nút phát hiện giá đường đi thay đổi
- Cập nhật Bảng Distance (line 15)
- Nếu có một giá nào đó thay đổi, thông báo với các hàng xóm (dòng 23,24)



"Tin tốt lan nhanh"

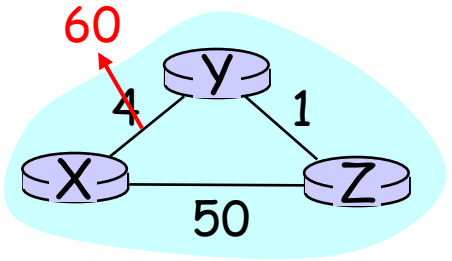


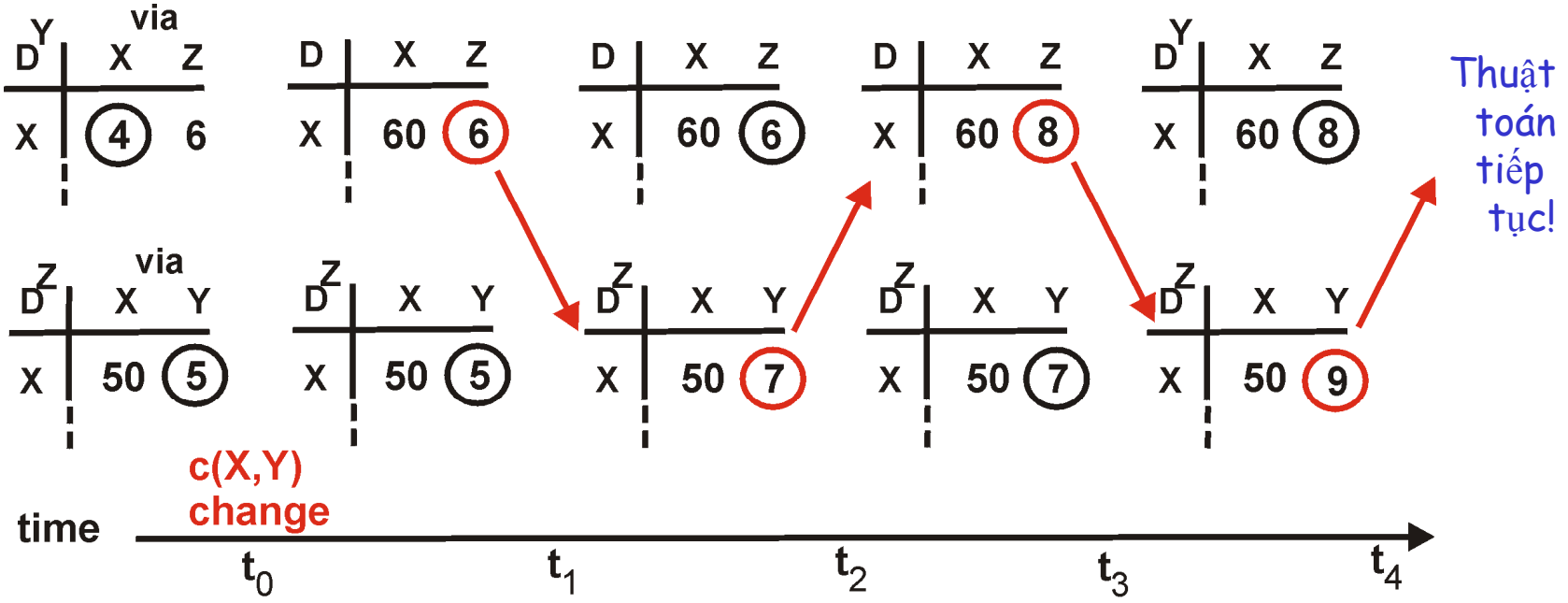
Thuật toán Kết thúc

Distance Vector: Thay đổi giá đường đi

Thay đổi giá đường đi:

- Tin tốt lan nhanh
- Tin xấu lan lâu !

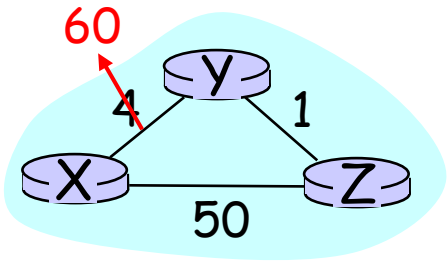


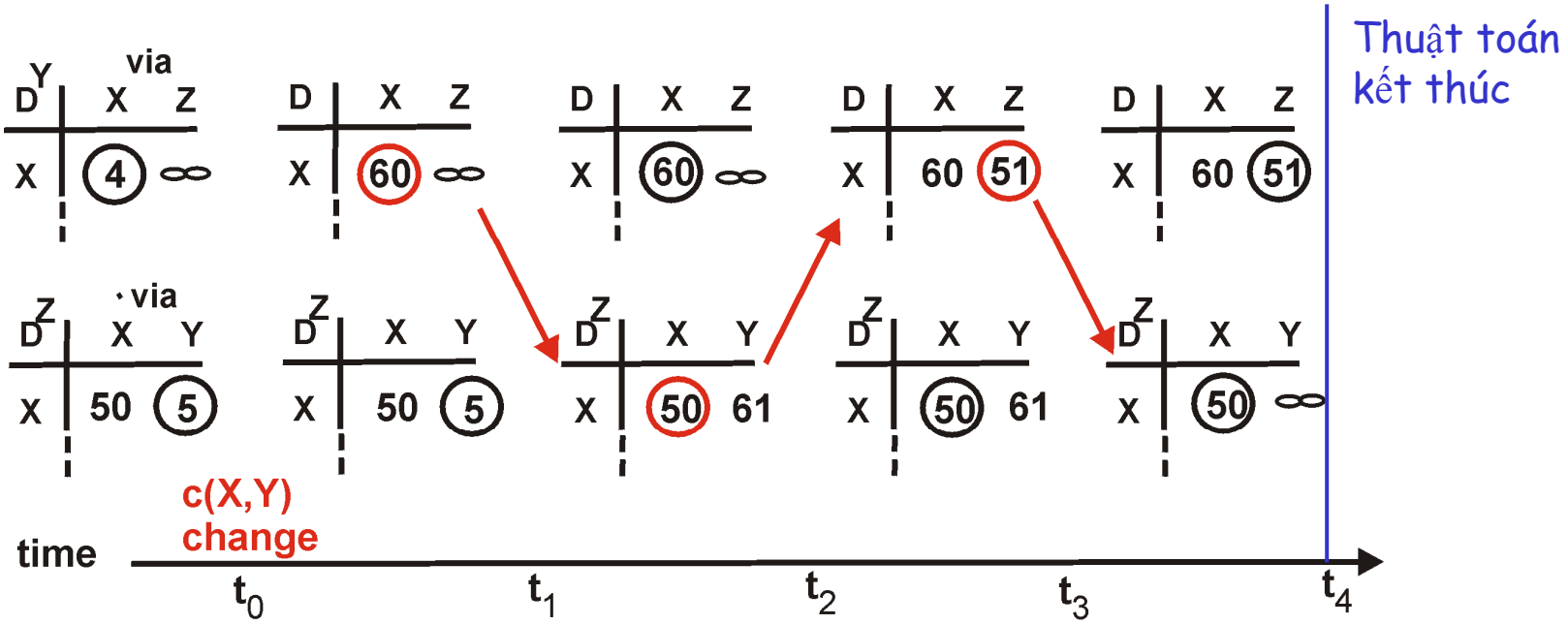


Distance Vector: Đảo ngược

If Z định tuyến tới X qua Y:

- Z thông báo với Y khoảng cách từ Z tới X là vô cùng (để Y không chuyển gói tin đến X qua Z)
- Có thể khắc phục vấn đề được không?





So sánh hai Thuật toán : LS và DV

Độ phức tạp của Thông điệp

- LS: Với n nút, E links, O(nE) thông điệp
- DV: Chỉ trao đổi thông điệp với các Hàng xóm
  - Thời gian Hội tụ biến thiên

Tốc độ Hội tụ

- LS: Độ phức tạp O(n**2) cần O(nE) thông điệp
  - Có thể bị dao động
- DV: Thời gian Hội tụ biến thiên
  - Bị vòng lặp
  - Vấn đề count-to-infinity

Sức mạnh: Chuyện gì xảy ra nếu router hoạt động không chính xác ?

LS:

- Nút quảng cáo *giá* sai
- Mỗi nút chỉ tính riêng bảng của mình

DV:

- Nút có thể quảng cáo tuyến đường sai
- Bảng của nút có thể được các nút khác sử dụng
  - Lỗi lan đi trên toàn mạng

Định tuyến Phân cấp

Từ trước đến nay định tuyến lý tưởng

- Tất cả routers giống nhau
- Mạng "phẳng"

... *Không có* trên thực tế

Phạm vi: 50 triệu đích:

- Không thể lưu trữ 50 triệu địa chỉ trong bảng định tuyến!
- Khối lượng trao đổi quá lớn!

Mục tiêu quản trị

- internet = Mạng các Mạng
- Quản trị viên trong mỗi Mạng muốn giám sát thông tin lưu chuyển trong Mạng của mình

Định tuyến Phân cấp

- Sắp xếp router theo vùng, "Miền tự trị" (AS)
- router trong cùng AS chạy cùng thuật toán định tuyến
  - Giao thức Định tuyến "nội miền" routing
  - Router trong các AS khác nhau có thể chạy các thuật toán định tuyến nội miền khác nhau

gateway router

- Đóng vai trò đặc biệt trong AS
- Chạy giao thức nội miền để "trò chuyện" với các router khác trong miền
- *Cũng* chịu trách nhiệm định tuyến cho các gói tin mà địa chỉ đích ở bên ngoài AS
  - Chạy *thuật toán Định tuyến Liên miền* với các gateway routers khác

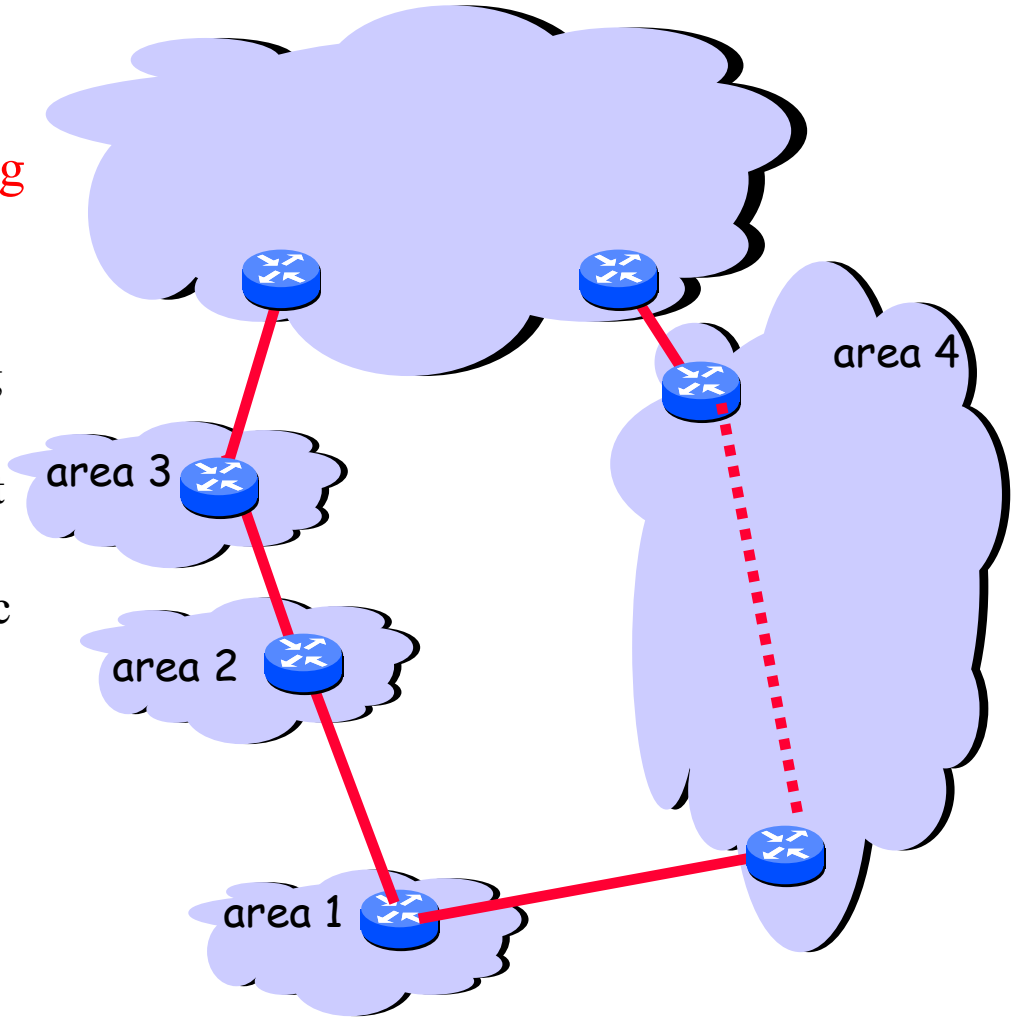
Tại sao Phải phân loại Nội miền – Liên miền?

Do phân cấp Định tuyến

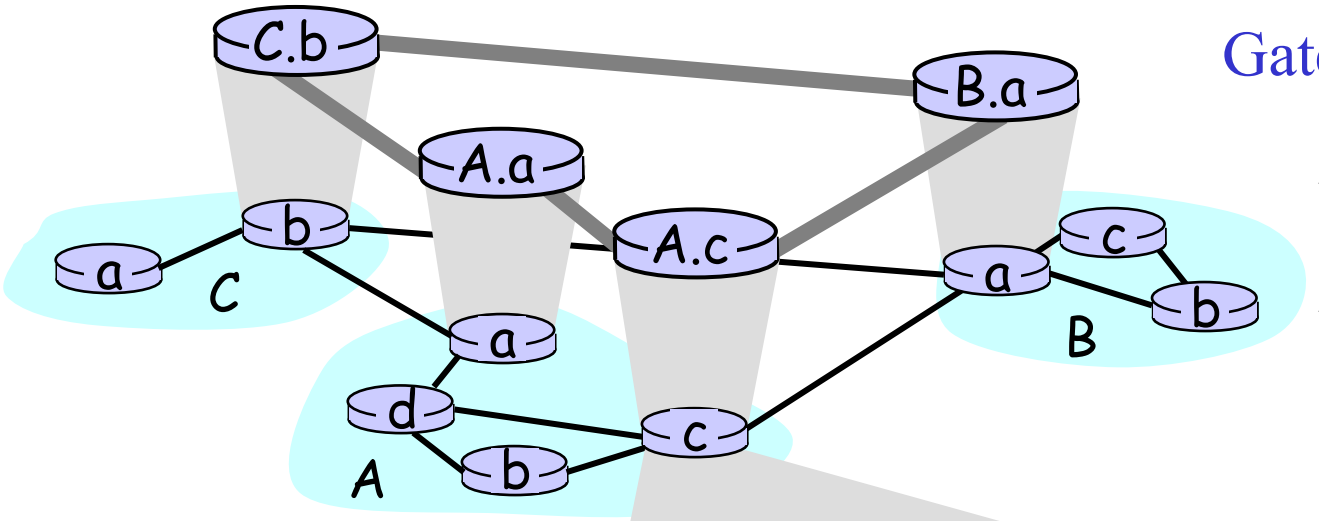
- So sánh với định tuyến phẳng

Sử dụng Định tuyến phân cấp để tăng cường Khả năng mở rộng:

- Với hàng triệu đích đến
  - Không thể lưu trữ tất cả địa chỉ trong Bảng
  - Trao đổi Bảng định tuyến "ngốn" hết băng thông Mạng
- Định tuyến phân cấp giảm kích thước bảng định tuyến và giảm khối lượng trao đổi trên Mạng
  - Vấn đề Chất lượng của Tuyến đường



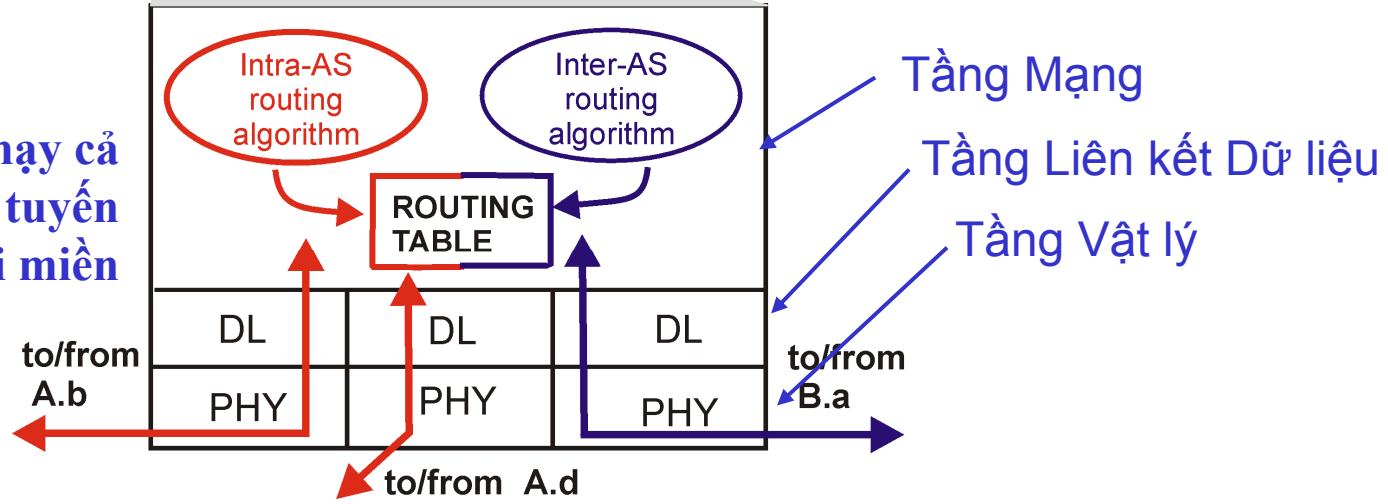
Định tuyến Nội miền – Liên miền



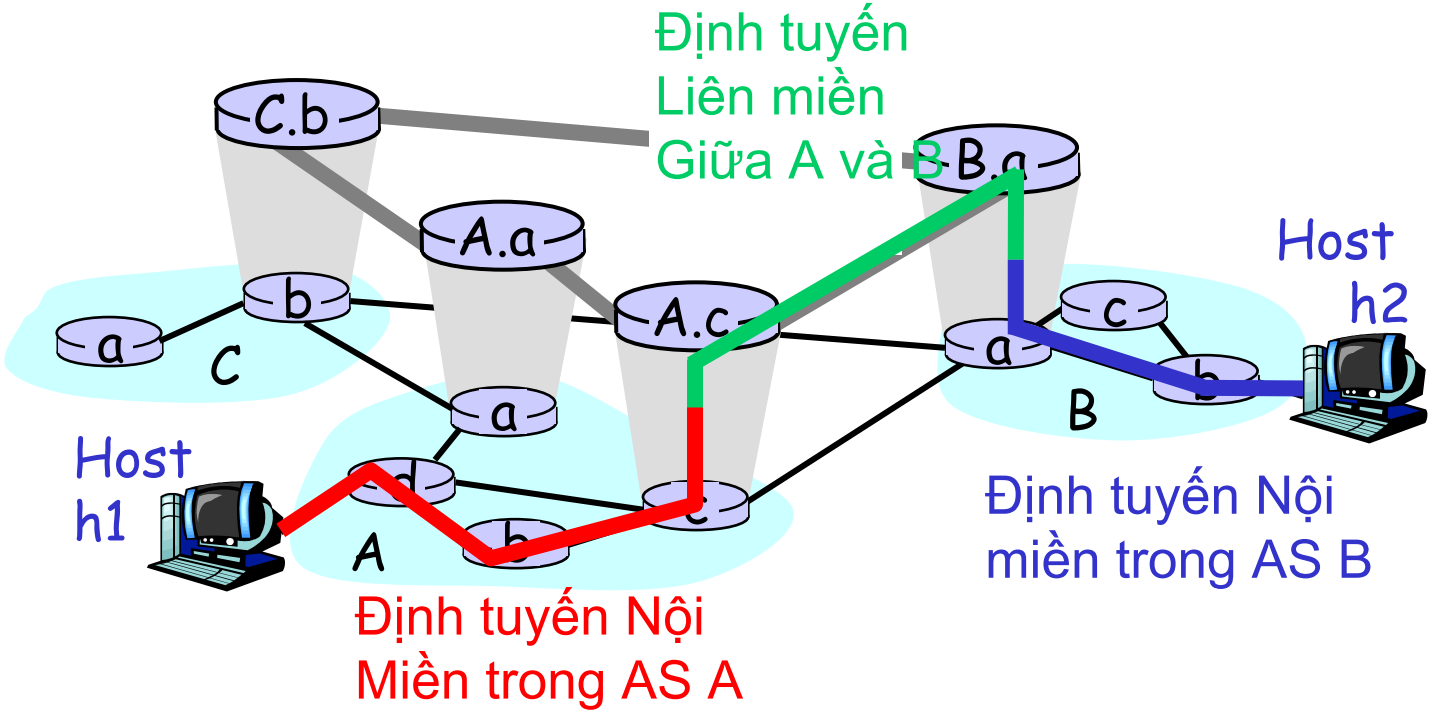
Gateway:

- Thực hiện định tuyến liên miền
- Thực hiện định tuyến nội miền giữa các AS

gateway A.c chạy cả Giao thức Định tuyến Liên miền và Nội miền

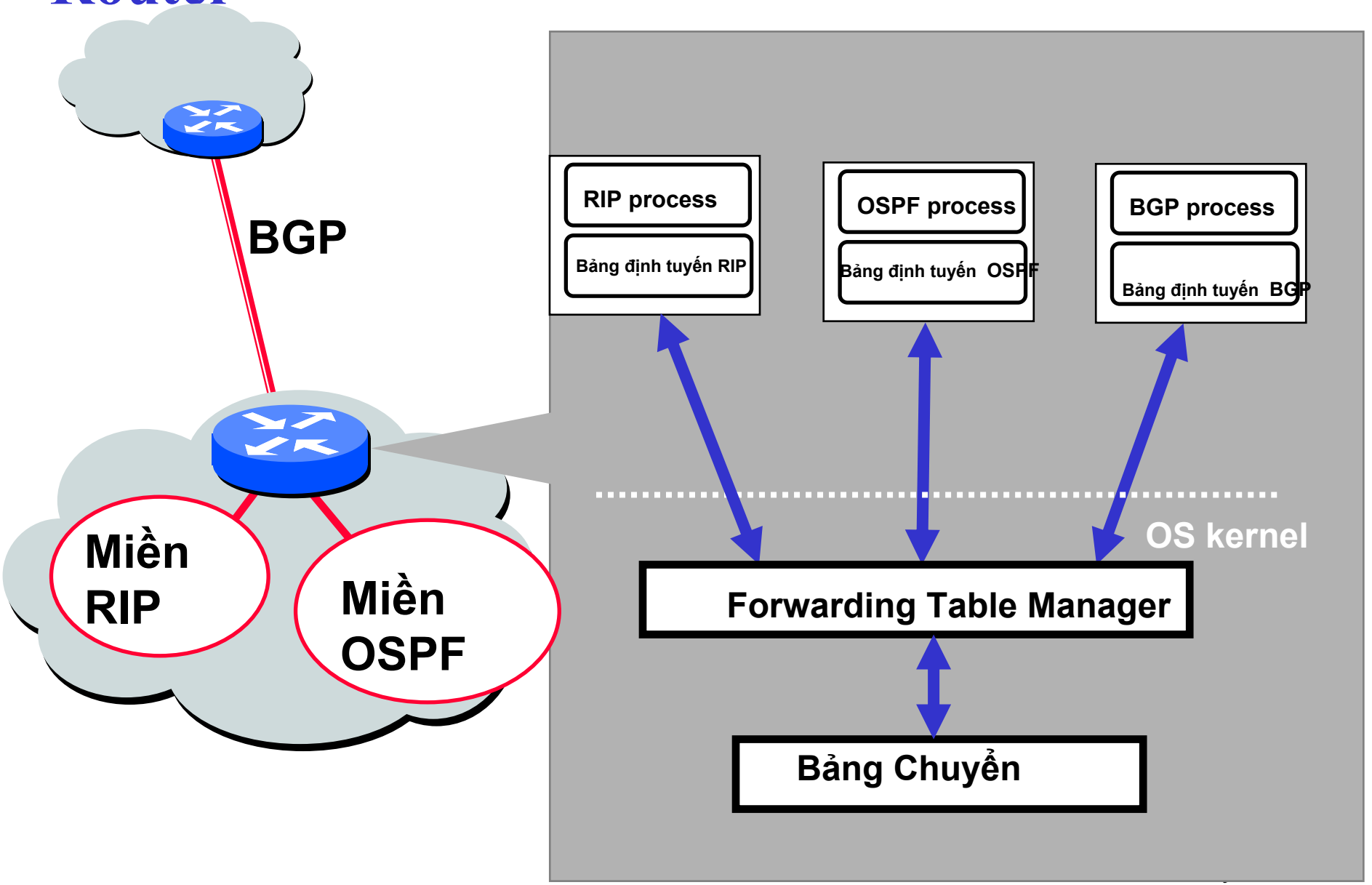


Định tuyến Nội miền – Liên miền

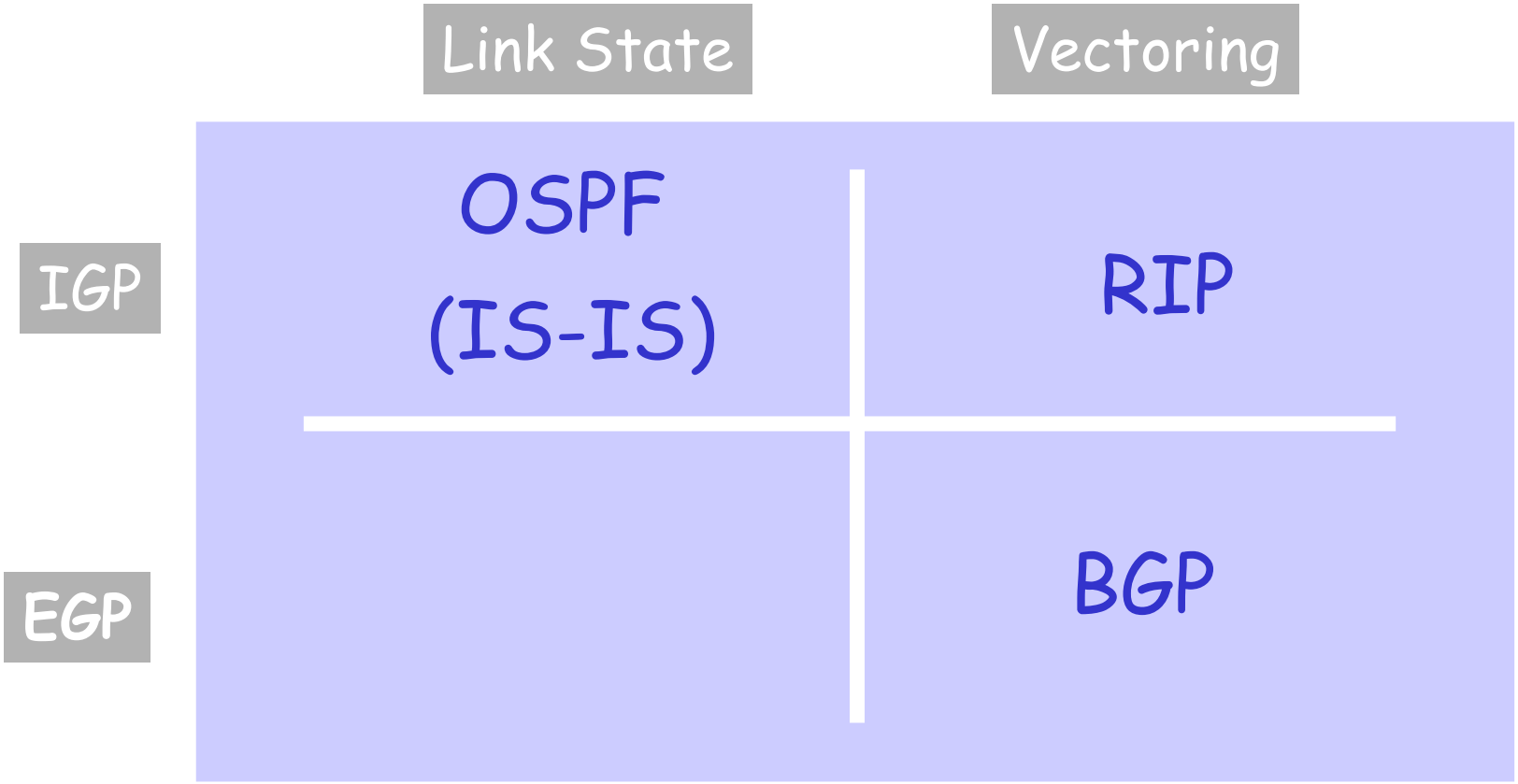


- Chi tiết về các giao thức định tuyến Nội miền và Liên miền sẽ được trình bày trong các phần sau

Nhiều Tiến trình Định tuyến chạy trên Một Router

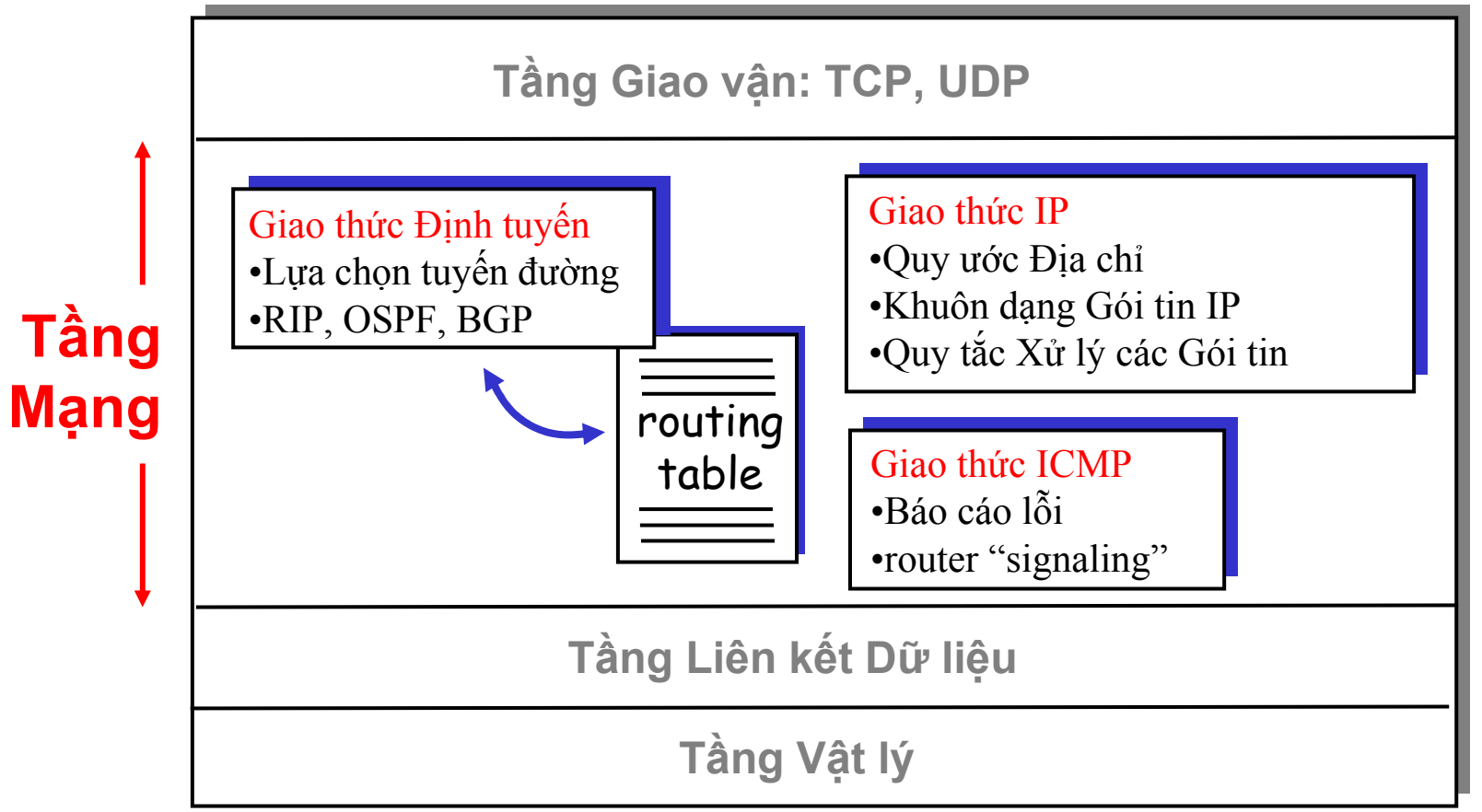


Bè lũ Bốn Tên ☺



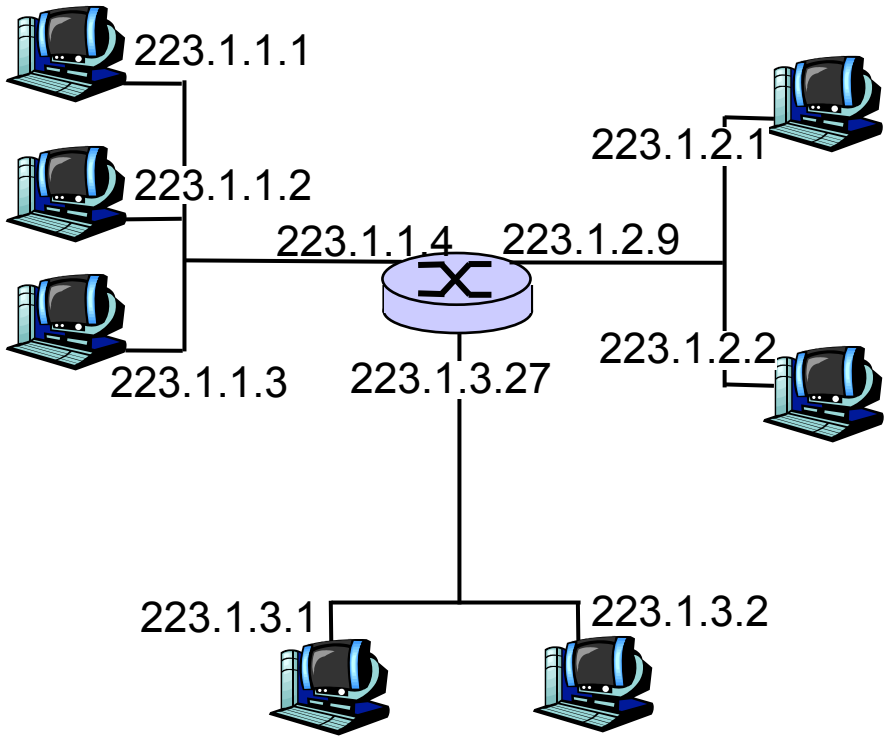
Tầng Mạng trên Internet

Chức năng của tầng Mạng trên các Máy tính, Router:



Địa chỉ IP: Giới thiệu

- Địa chỉ IP : Định danh 32-bit cho *Giao diện* của Máy tính, Router
- *Giao diện:* Kết nối giữa máy tính, router với kênh truyền Vật lý
  - Router thường có nhiều Giao diện
  - Máy tính có thể có nhiều Giao diện
  - Địa chỉ IP gắn với giao diện chứ không phải với máy tính hay router

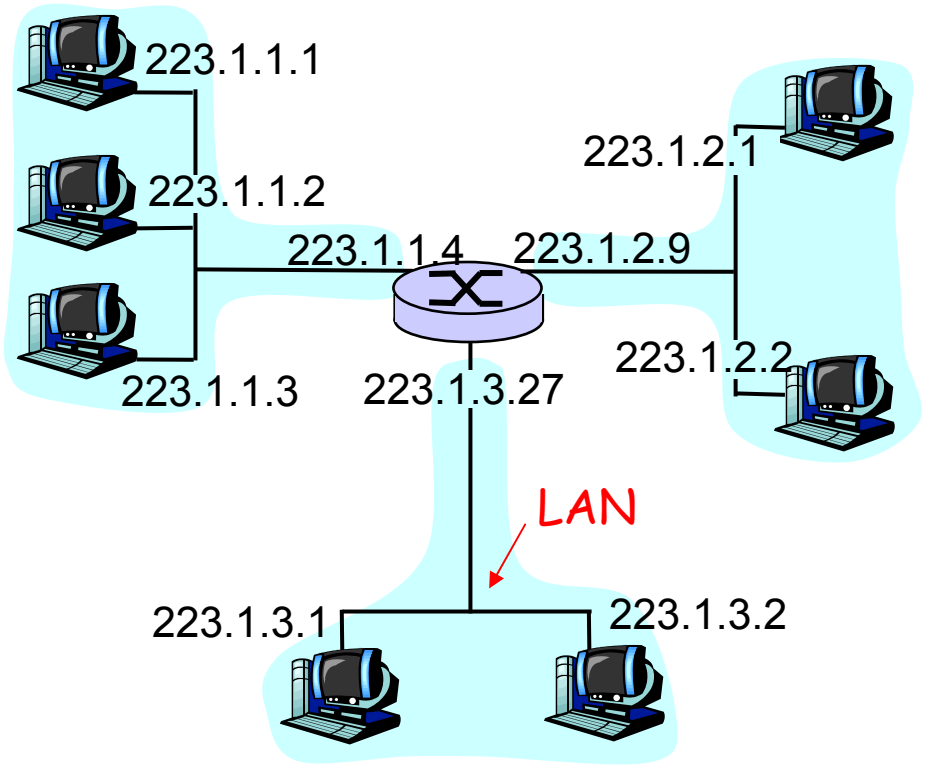


223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

Địa chỉ IP

- Địa chỉ IP:
  - Phần network (các bit cao)
  - Phần host (các bit thấp)
- *Thế nào là Mạng ?* (Theo quan điểm Địa chỉ IP)
  - Giao diện của các thiết bị mà phần network trong địa chỉ IP giống nhau
  - Có thể trao đổi dữ liệu với nhau mà không cần qua router

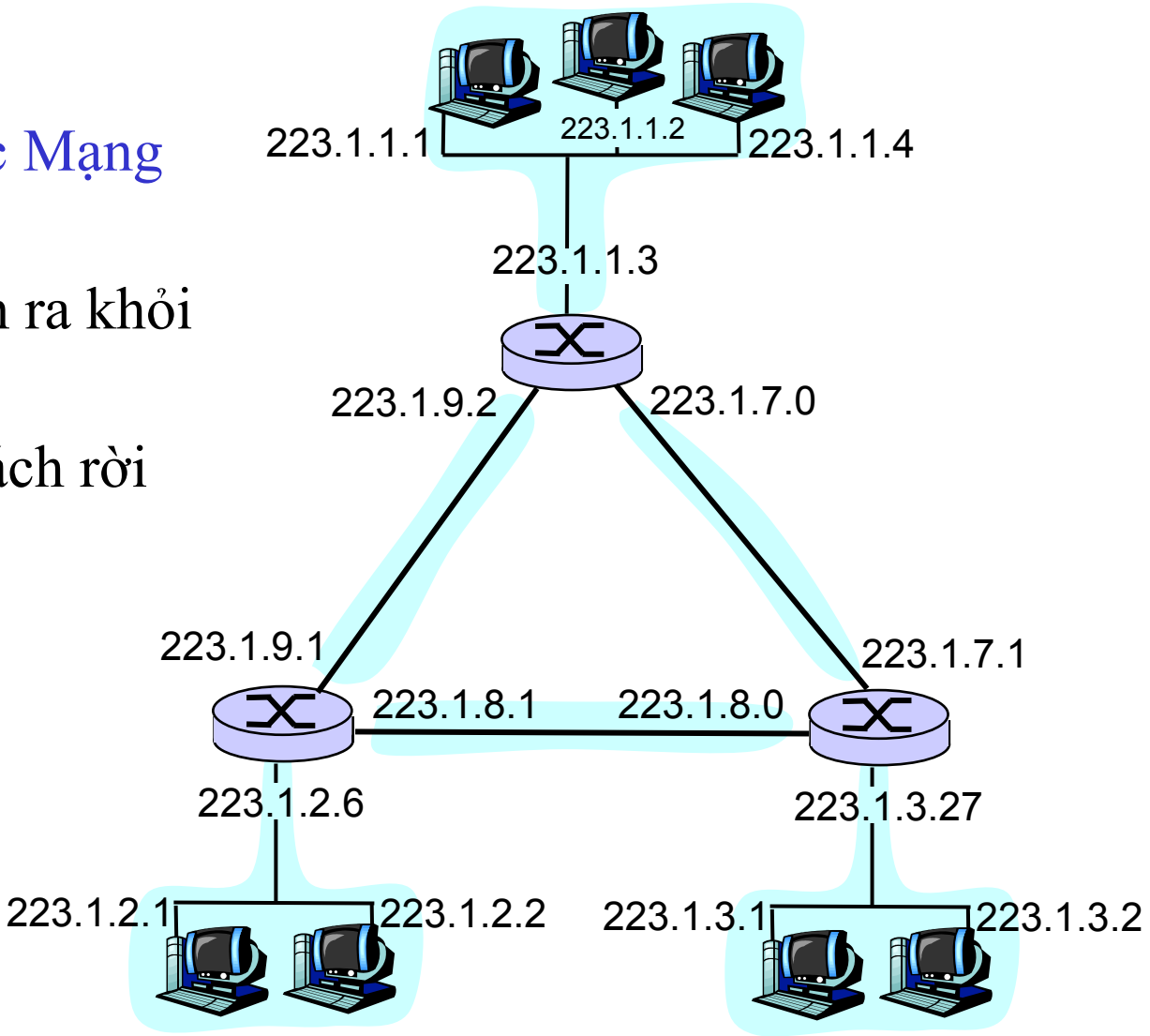


Mạng với 03 Mạng địa chỉ IP
(Với địa chỉ IP bắt đầu bằng 223, 24 bit đầu là phần network)

Địa chỉ IP

Làm sao tìm được các Mạng IP?

- Tách các Giao diện ra khỏi Máy tính, router
- Tạo ra các Mạng tách rời nhau

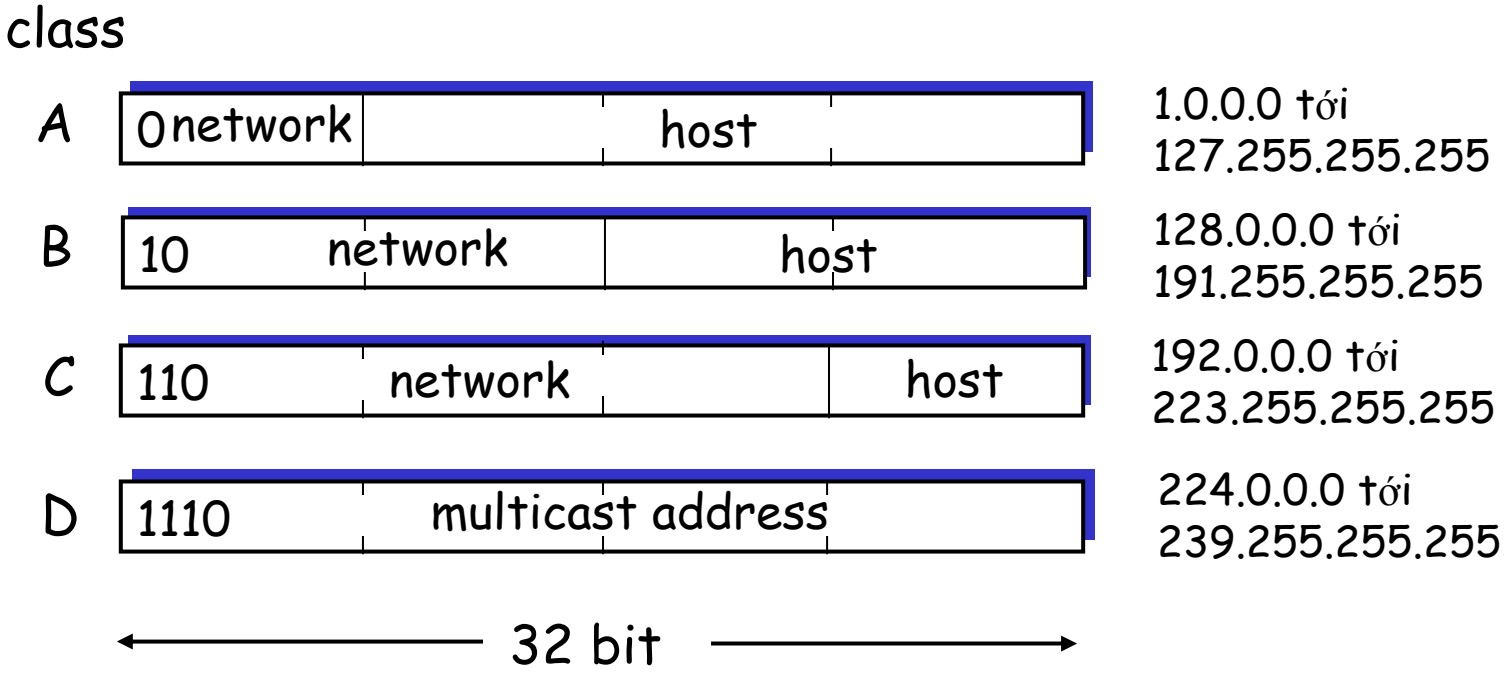


Hệ thống kết nối có 6 mạng

Địa chỉ IP

Với định nghĩa mới về "Mạng", xét lại Địa chỉ IP:

Địa chỉ "phân lớp" :



Địa chỉ IP : CIDR

- Địa chỉ phân lớp:
  - Không gian Địa chỉ bị sử dụng lãng phí và nhanh chóng cạn kiệt
  - Ví dụ một lớp B có thể cấp phát tới 65K máy tính, kể cả khi mạng chỉ có 2K máy tính
- CIDR: Classless InterDomain Routing
  - Phần network của địa chỉ có kích thước tùy ý
  - Khuôn dạng địa chỉ: a.b.c.d/x, trong đó x là số bit trong phần network của địa chỉ

200.23.16.0/23

Địa chỉ IP : Làm sao có được?

Các máy tính (Phần host):

- Được người quản trị Hệ thống cấu hình cứng và ghi vào file
  - wintel: control-panel->network->configuration->tcp/ip->properties
  - unix:
    %/sbin/ifconfig eth0 inet 192.168.0.10 netmask 255.255.255.0

- DHCP: Dynamic Host Configuration Protocol: tự động xin cấp phát địa chỉ theo kiểu "plug-and-play"
  - Máy tính quảng bá thông điệp "DHCP discover"
  - DHCP server trả lời với thông điệp "DHCP offer"
  - Máy tính yêu cầu địa chỉ IP bằng thông điệp "DHCP request"
  - DHCP server gửi địa chỉ qua thông điệp "DHCP ack"

Sử dụng
%whois –h whois.arin.net "n <org>"
Để kiểm tra các địa chỉ cấp phát cho <org>

Địa chỉ IP : Làm sao có được?

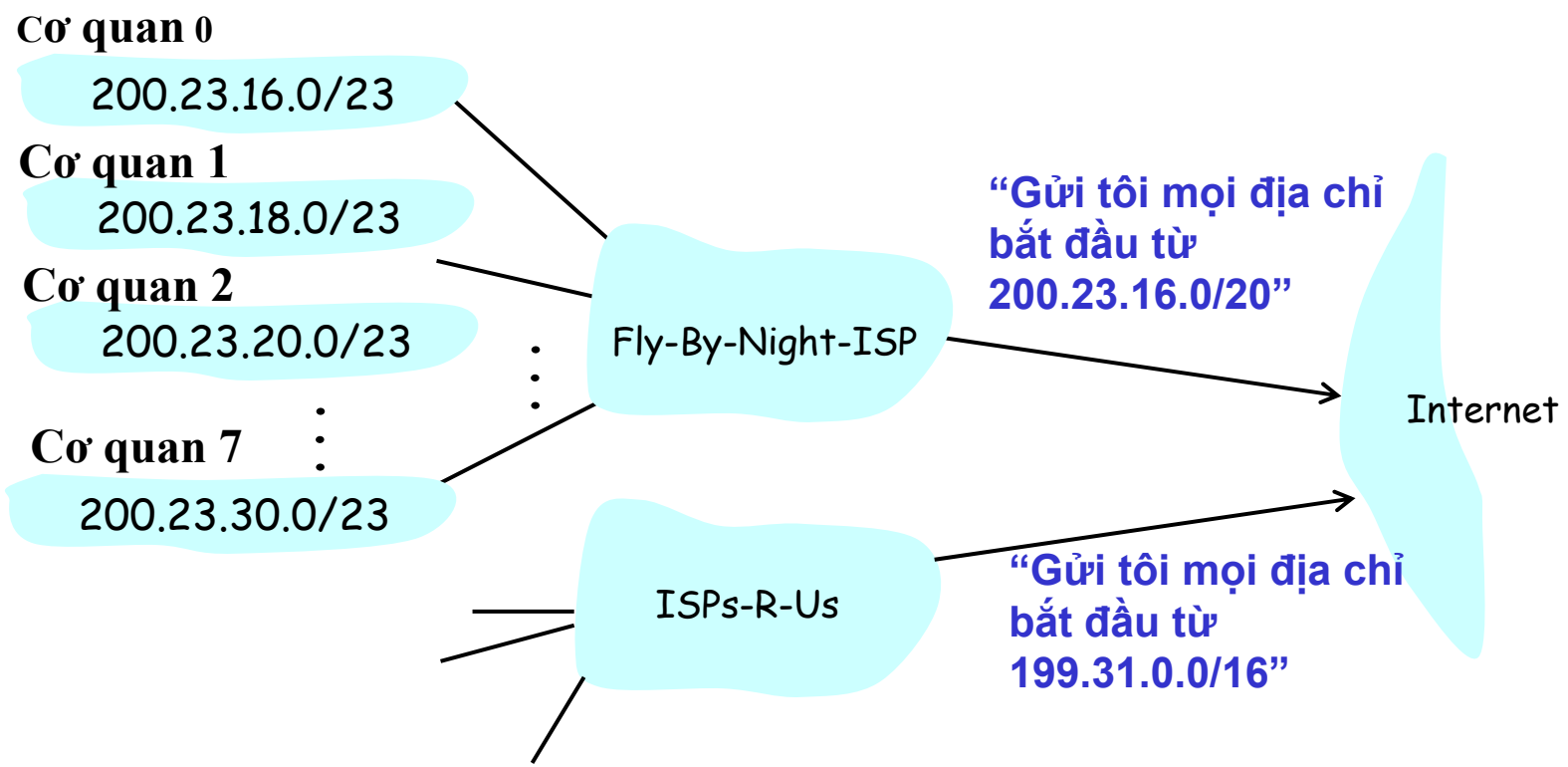
Địa chỉ Mạng (Phần network):

- Được cấp phát một phần từ không gian địa chỉ ISP:

ISP		11001000 00010111 00010000	00000000	200.23.16.0/20
Cơ quan	0	11001000 00010111 00010000	00000000	200.23.16.0/23
Cơ quan	1	11001000 00010111 00010010	00000000	200.23.18.0/23
Cơ quan	2	11001000 00010111 00010100	00000000	200.23.20.0/23
...		.....	....	....
Cơ quan	7	11001000 00010111 00011110	00000000	200.23.30.0/23

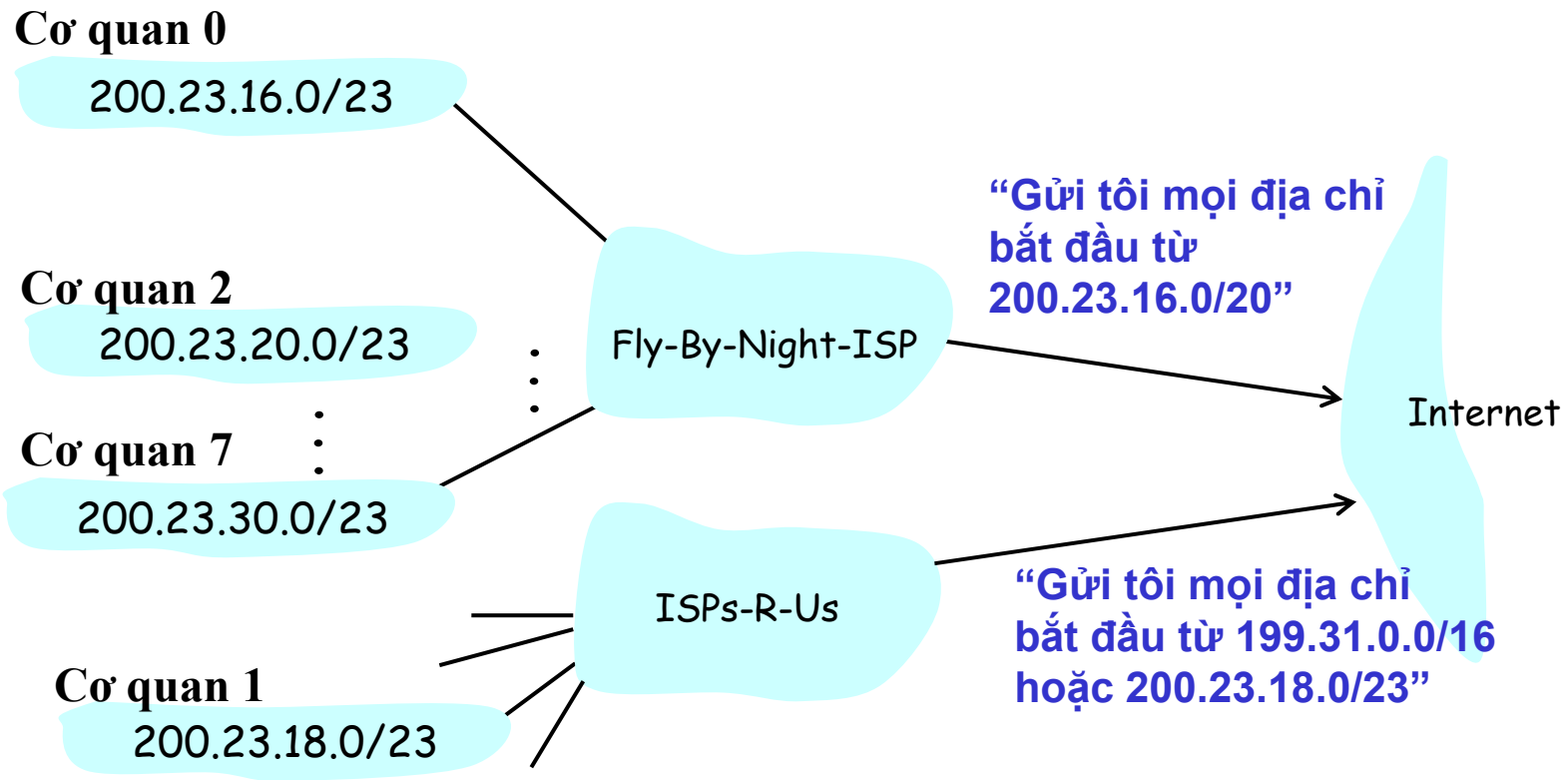
Địa chỉ Phân cấp: Kết nối các tuyến đường

Định tuyến phân cấp cho phép quảng cáo các thông tin định tuyến một cách hiệu quả:



Địa chỉ phân cấp: Các tuyến cụ thể hơn

ISPs-R-Us có tuyến đường cụ thể tới Cơ quan 1 hơn



Địa chỉ IP : cuối cùng...

Q: Làm sao ISP có được một dải địa chỉ?

A: ICANN: Internet Corporation for Assigned Names and Numbers

- Cấp phát Địa chỉ
- Quản lý DNS
- Gán tên miền, quản lý tranh chấp

Để gửi datagram từ Nguồn đến Đích

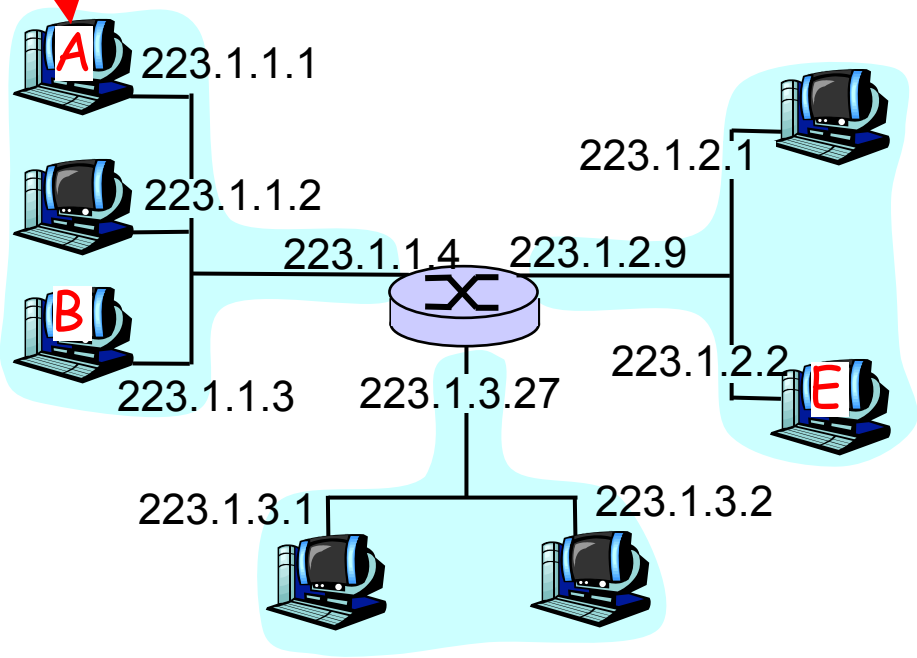
IP datagram:

misc fields	source IP addr	dest IP addr	data

- datagram gần như không thay đổi trên tuyến đường từ đích đến nguồn
- Chỉ quan tâm tới địa chỉ nhận

Bảng Định tuyến ở A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



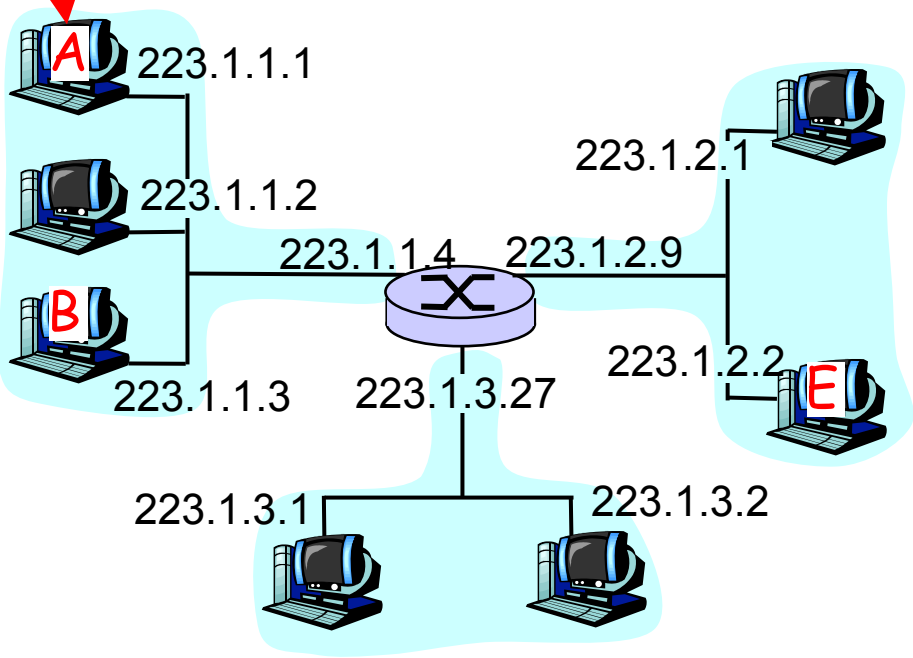
Để gửi datagram từ Nguồn đến Đích

misc fields	223.1.1.1	223.1.1.3	data

Bắt đầu từ A, với gói tin IP có địa chỉ đích là B:

- Tìm kiếm địa chỉ B trong bảng
- Thấy rằng B nằm trong cùng Mạng với A
- Tầng liên kết dữ liệu chịu trách nhiệm chuyển IP datagram trực tiếp tới B bên trong frame của tầng liên kết
  - B và A trực tiếp kết nối với nhau

Mạng đích	Router kế	Số chặng
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



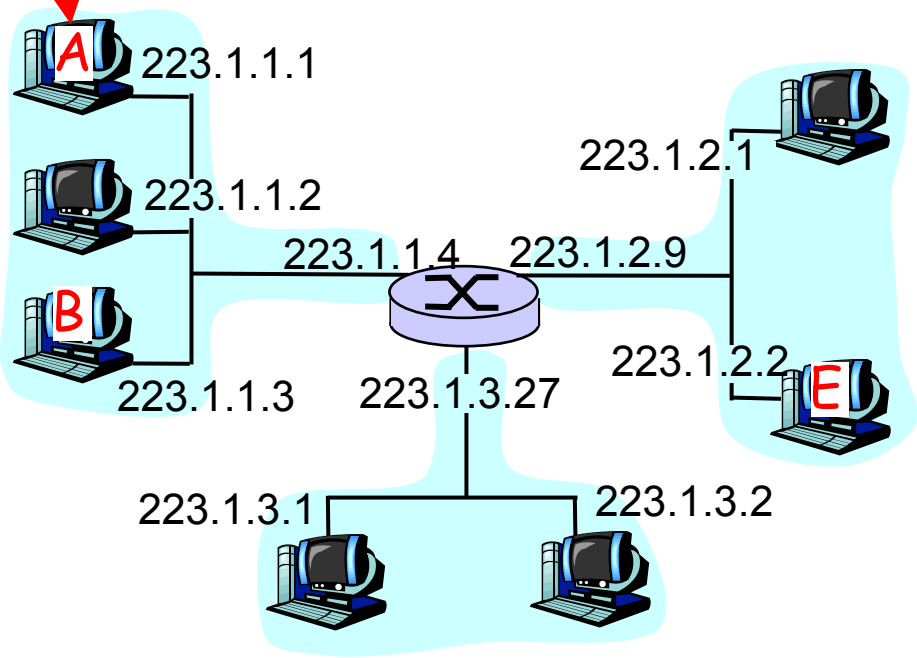
Để gửi datagram từ Nguồn đến Đích

misc fields	223.1.1.1	223.1.2.3	data

Bắt đầu từ A, đích là E:

- Tìm kiếm địa chỉ E trong mạng
- E ở mạng *khác*
  - A, E không có kết nối trực tiếp
- Trong bảng định tuyến: router kế tiếp tới E là 223.1.1.4
- Tầng liên kết dữ liệu gửi datagram tới router 223.1.1.4 bên trong frame của tầng liên kết
- datagram đến 223.1.1.4
- ***Tiếp tục…..***

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



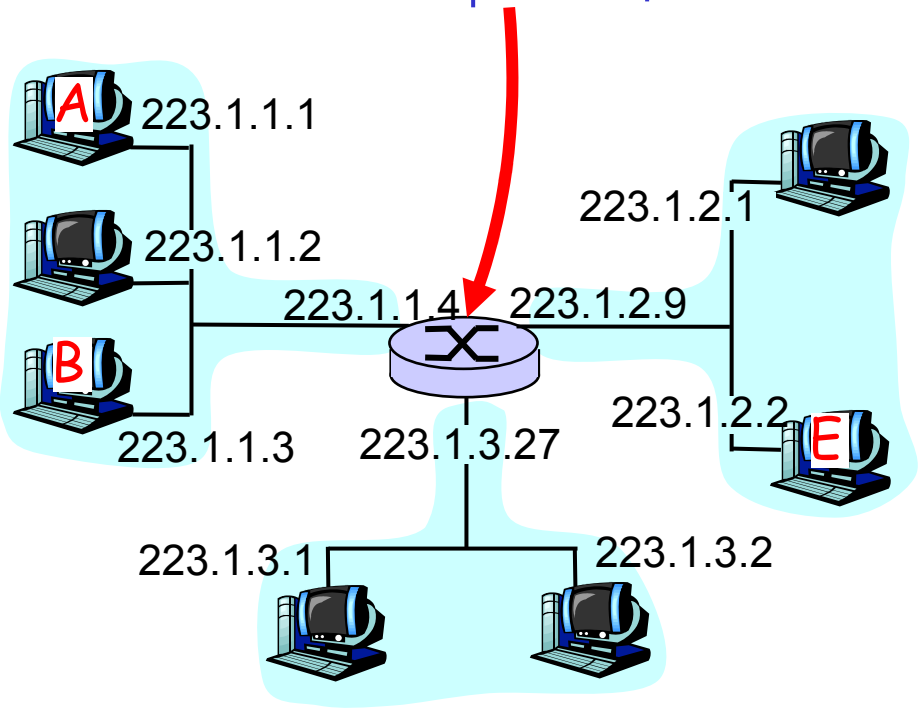
Để gửi datagram từ Nguồn đến Đích

misc fields	223.1.1.1	223.1.2.3	data

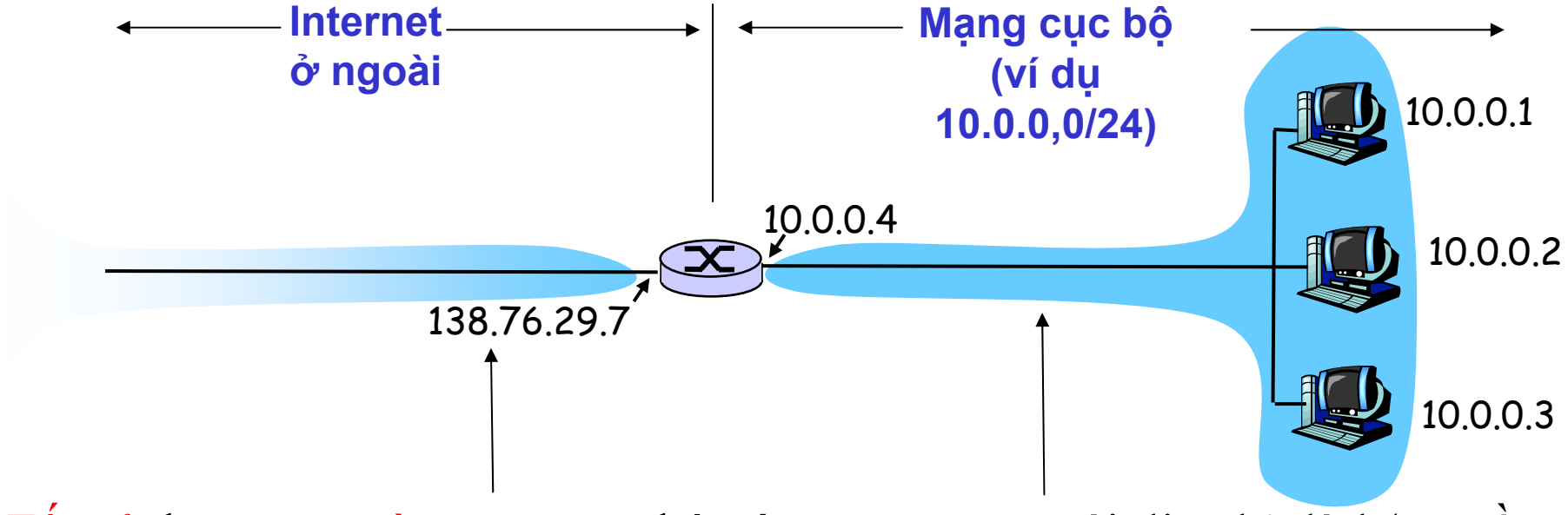
Đến 223.1.4, đích là 223.1.2.2

- Xác định địa chỉ đích là E
- E trên *cùng* mạng với giao diện 223.1.2.9 của router
  - router, E có kết nối trực tiếp
- Tầng liên kết dữ liệu gửi datagram tới 223.1.2.2 bên trong frame tầng liên kết dữ liệu qua giao diện 223.1.2.9
- Gói tin đã đến được 223.1.2.2!!! (***Chúc mừng!)***

Mạng đích	router kế tiếp	Số chặng	Giao diện
223.1.1	-	1	223.1.1.4
223.1.2	-	1	223.1.2.9
223.1.3	-	1	223.1.3.27



NAT: Network Address Translation



Tất cả datagram *rời* mạng cục bộ có *cùng* địa chỉ IP đích (đã bị biến đổi): 138.76.29.7,
Tuy nhiên cổng nhận giá trị khác nhau

Datagram với địa chỉ đích/ nguồn ở trong mạng IP (địa chỉ 10.0.0/24) sẽ giữ nguyên địa chỉ (như bình thường)

NAT: Network Address Translation

- **Động lực**: Mạng cục bộ chỉ có một địa chỉ IP để kết nối với bên ngoài:
  - Không cần thiết phải có 1 khoảng địa chỉ IP từ ISP: - một địa chỉ IP được sử dụng chung cho tất cả các thiết bị
  - Bên ngoài không nhìn thấy sự thay đổi địa chỉ bên trong
  - Có thể thay đổi ISP mà không thay đổi IP của các máy tính bên trong
  - Bên ngoài không thể nhìn thấy địa chỉ tường minh của các thiết bị bên trong mạng cục bộ.

NAT: Network Address Translation

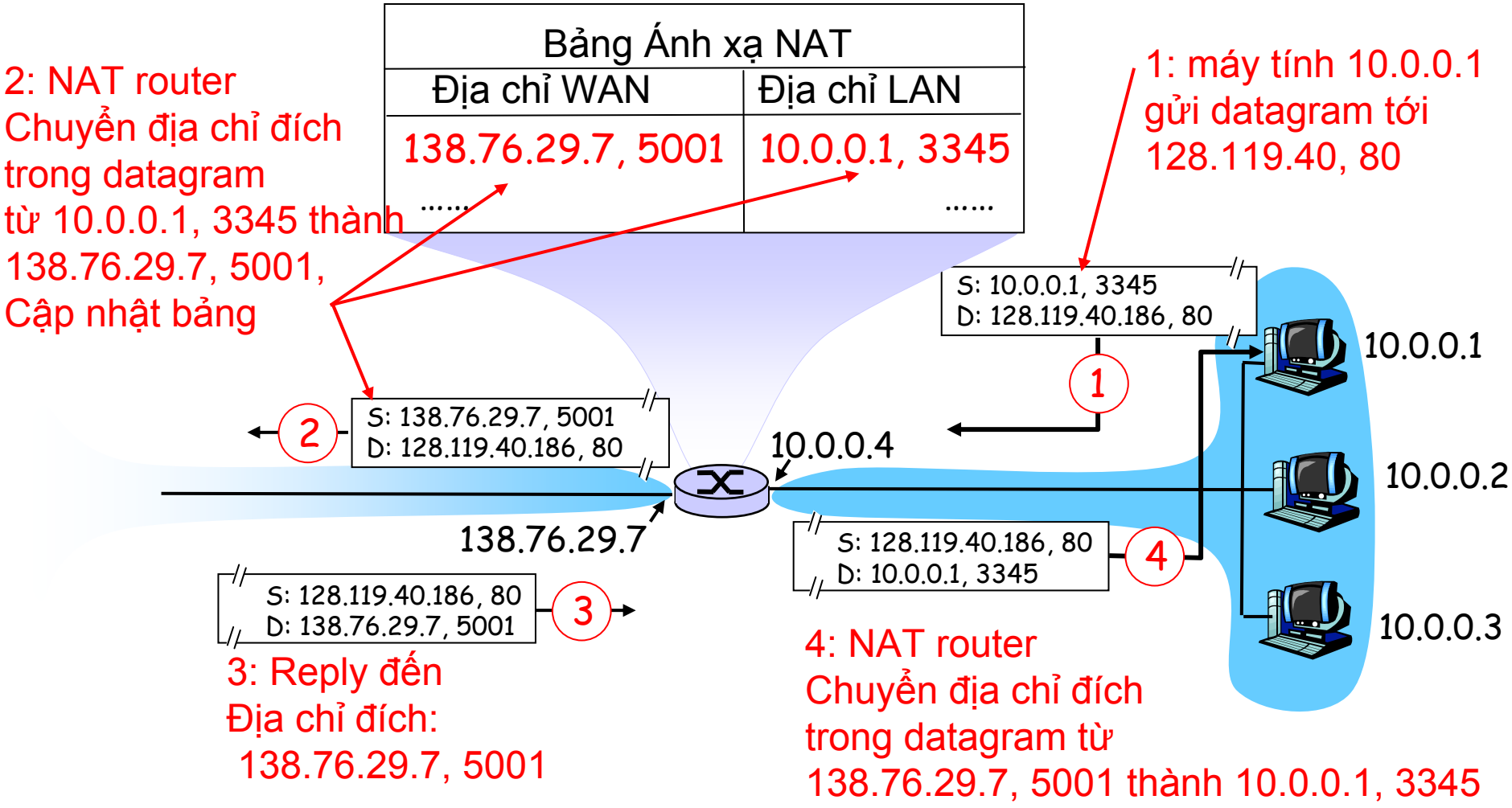
Cài đặt: NAT router phải:

- ***Datagram chuyển ra ngoài: thay thế*** (IP gửi, port #) tất cả datagram chuyển ra ngoài thành (địa chỉ IP được chuyển, cổng # mới)
  . . . Tương tác clients/servers sử dụng (địa chỉ IP được chuyển, cổng # mới) làm địa chỉ đích.

- ***Ghi nhớ (trong bảng biến đổi địa chỉ)*** tất cả các cặp (IP đích, port #) thành (địa chỉ IP được chuyển, port # mới) phục vụ cho mục đích chuyển đổi

- ***Datagram đến: thay thế*** (địa chỉ IP được chuyển, port # mới) trong các trường địa chỉ đích của mỗi datagram đến với ánh xạ tương ứng (IP gửi, port #) lưu trong bảng NAT

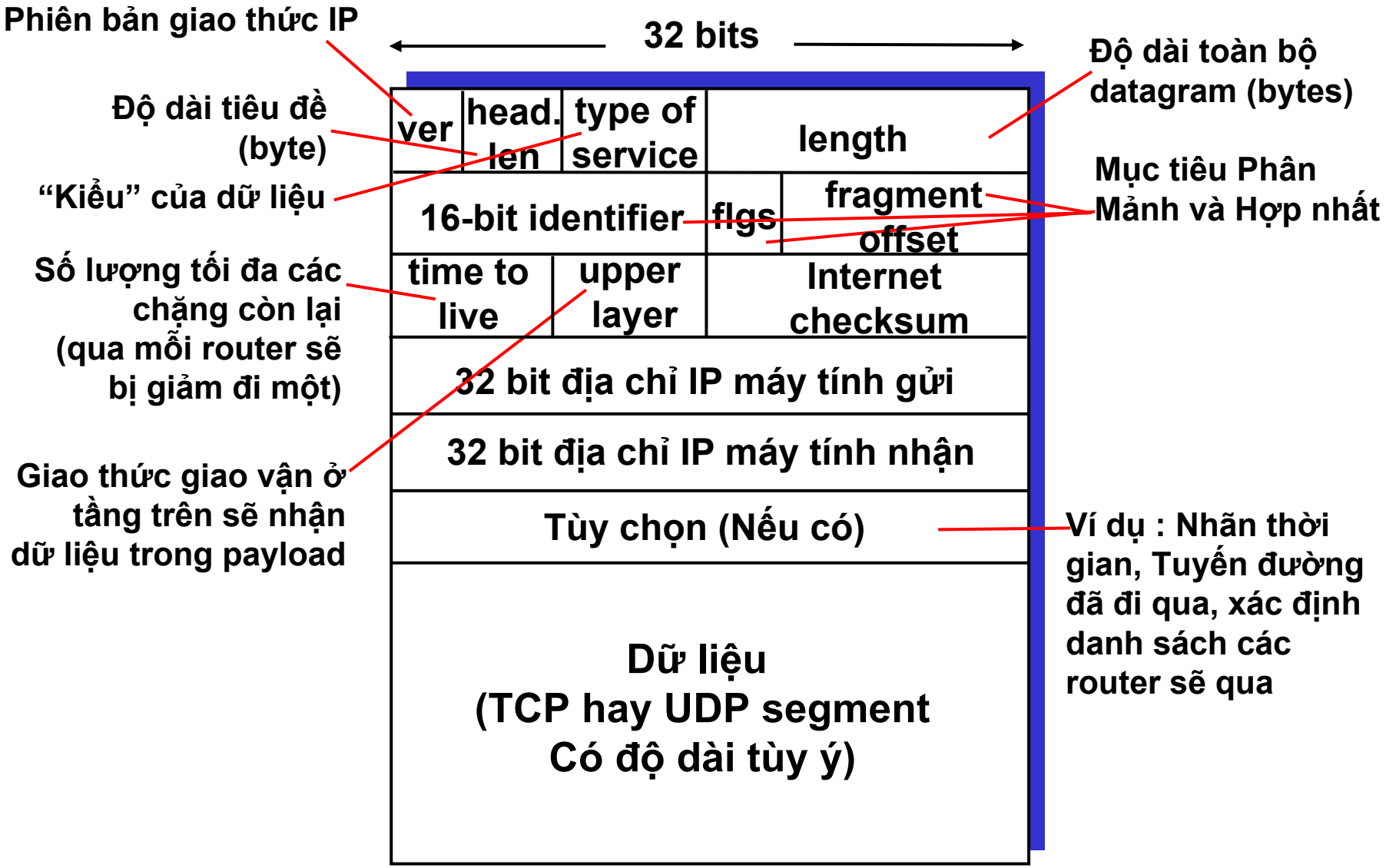
NAT: Network Address Translation



NAT: Network Address Translation

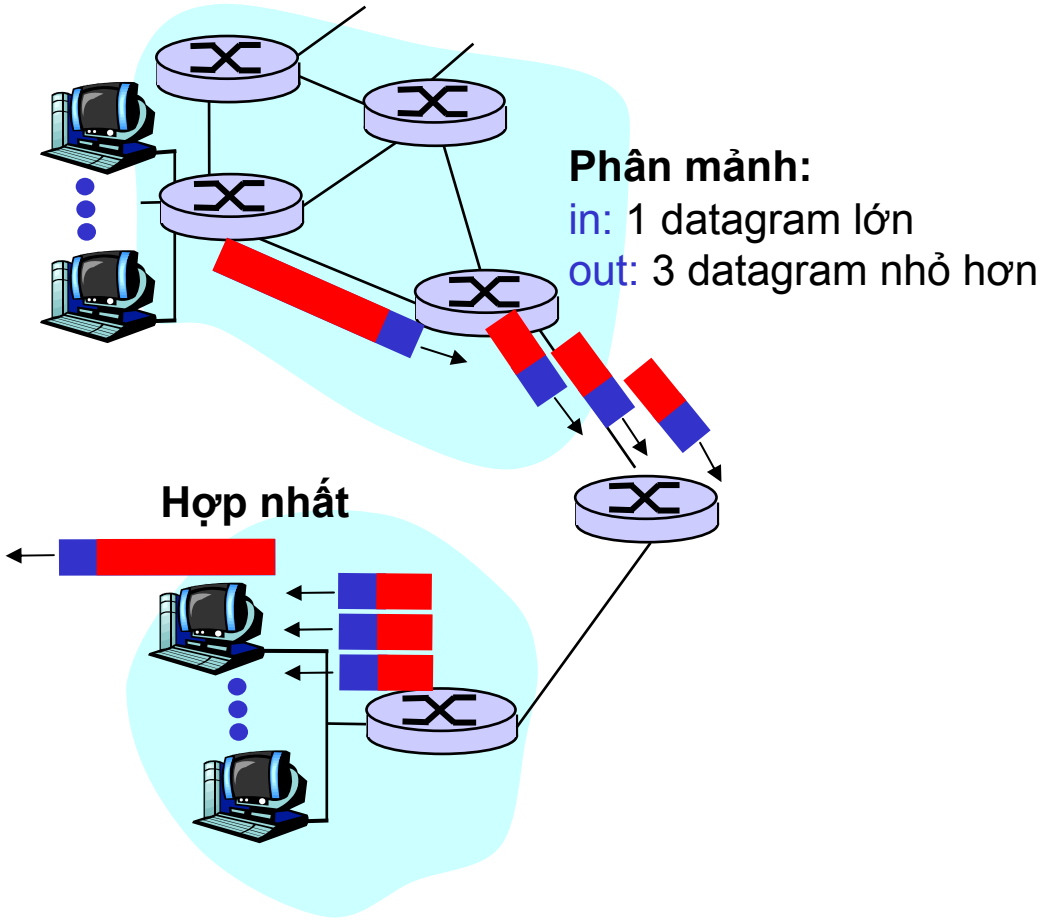
- 16-bit địa chỉ cổng:
  - 60,000 kết nối đồng thời trên cùng một địa chỉ mạng LAN duy nhất !
- NAT có vấn đề:
  - Router chỉ nên xử lý ở tầng 3
  - Vi phạm nguyên tắc đầu cuối
    - Các nhà phát triển ứng dụng, đặc biệt các ứng dụng P2P phải tính toán đến việc sử dụng NAT
  - Có thể việc khan hiếm địa chỉ sẽ hết khi sử dụng IPv6

Khuôn dạng Gói tin IP

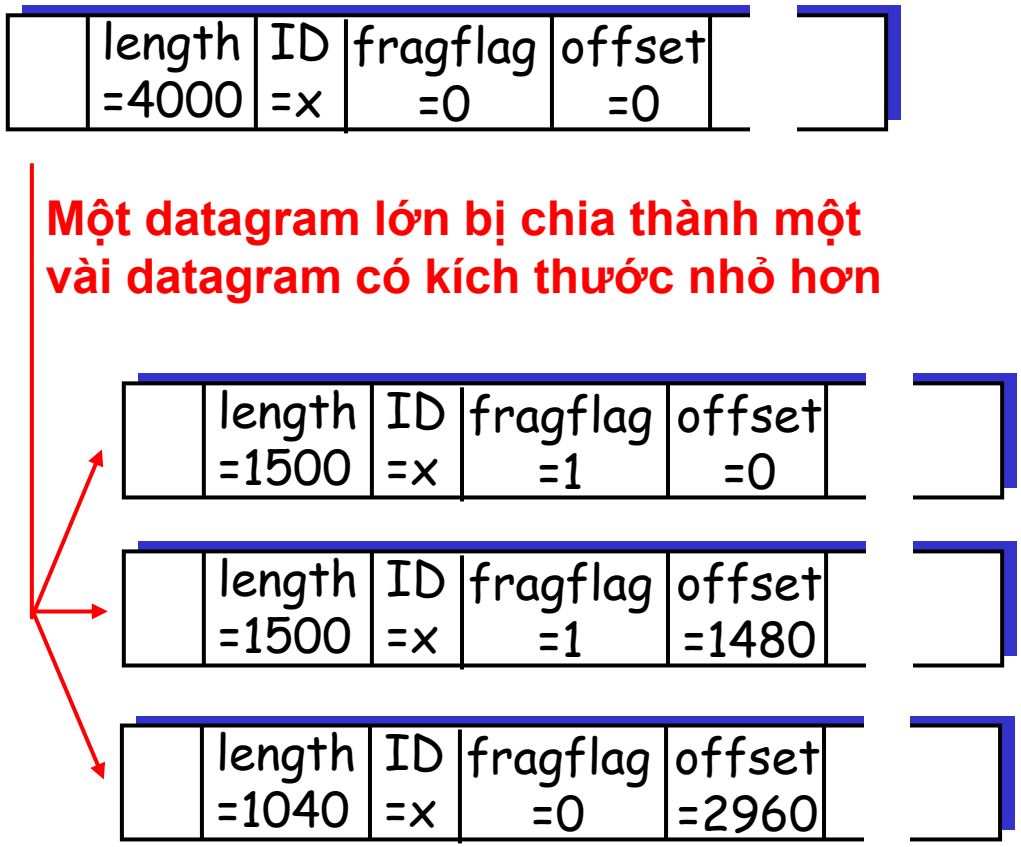


Phân mảnh và Hợp nhất Gói tin IP

- Kênh truyền có MTU (max.transfer size) – frame lớn nhất mà tầng liên kết dữ liệu có thể gửi được.
  - Các công nghệ truyền có giá trị MTU có thể khác nhau
- IP datagram "lớn" có thể bị chia nhỏ trong mạng (Phân mảnh)
  - Một datagram bị chia thành nhiều datagram
  - Hợp nhất được thực hiện tại đích
  - Các bit trong trường tiêu đề của gói tin IP được sử dụng để xác định và sắp xếp đúng thứ tự các "mảnh"



Ví dụ về Phân mảnh và Hợp nhất



ICMP: Internet Control Message Protocol

- Được các Máy tính, Router, Gateway sử dụng để trao đổi các thông tin về tầng Mạng
  - Báo lỗi: unreachable host, network, port, protocol
  - Hiển thị request/reply (Lệnh ping)
- Trong hệ thống "nằm trên" IP:
  - Thông điệp ICMP được đặt trong IP datagram
- **Thông điệp ICMP** : Kiểu (Type), Mã (code) cùng với 8 byte đầu tiên của IP datagram gây lỗi

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Chuyển tiếp Dữ liệu: Các bước

- Nếu không có lỗi, tìm kiếm địa chỉ đích của gói tin trong bảng Chuyển tiếp:
  - Nếu nút đích nằm trên mạng mà router có kết nối trực tiếp: công việc tiếp theo của tầng Liên kết dữ liệu
  - Ngược lại,
    - Tìm kiếm: xác định *router chặng kế tiếp* và giao diện ra tương ứng
    - Nếu cần thiết, phân mảnh gói tin
    - Chuyển tiếp gói tin ra giao diện của cổng ra tương ứng (là router "hàng xóm")

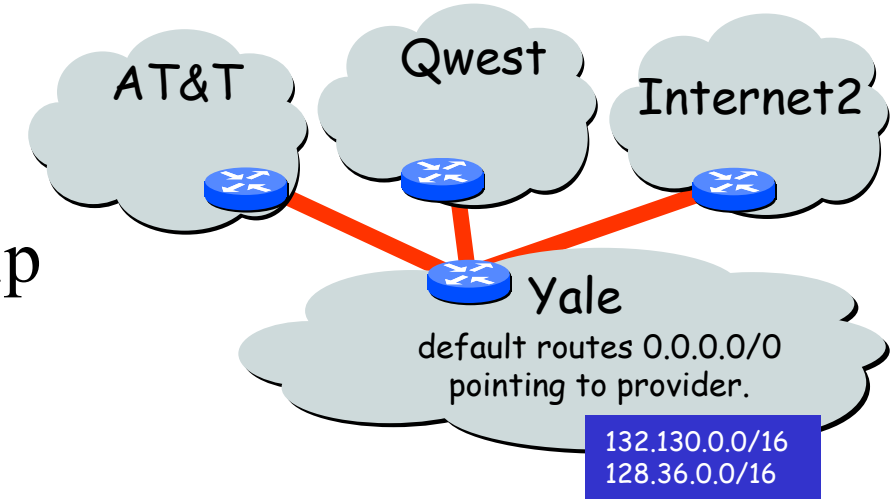
Thử %netstat –rn đê xem Bảng chuyển tiếp

Định tuyến trên Internet

- Mạng toàn cầu Internet bao gồm các Miền tự trị (Autonomous Systems - AS) kết nối với nhau:
  - Mỗi AS có một định danh riêng (ASN – AS Number)

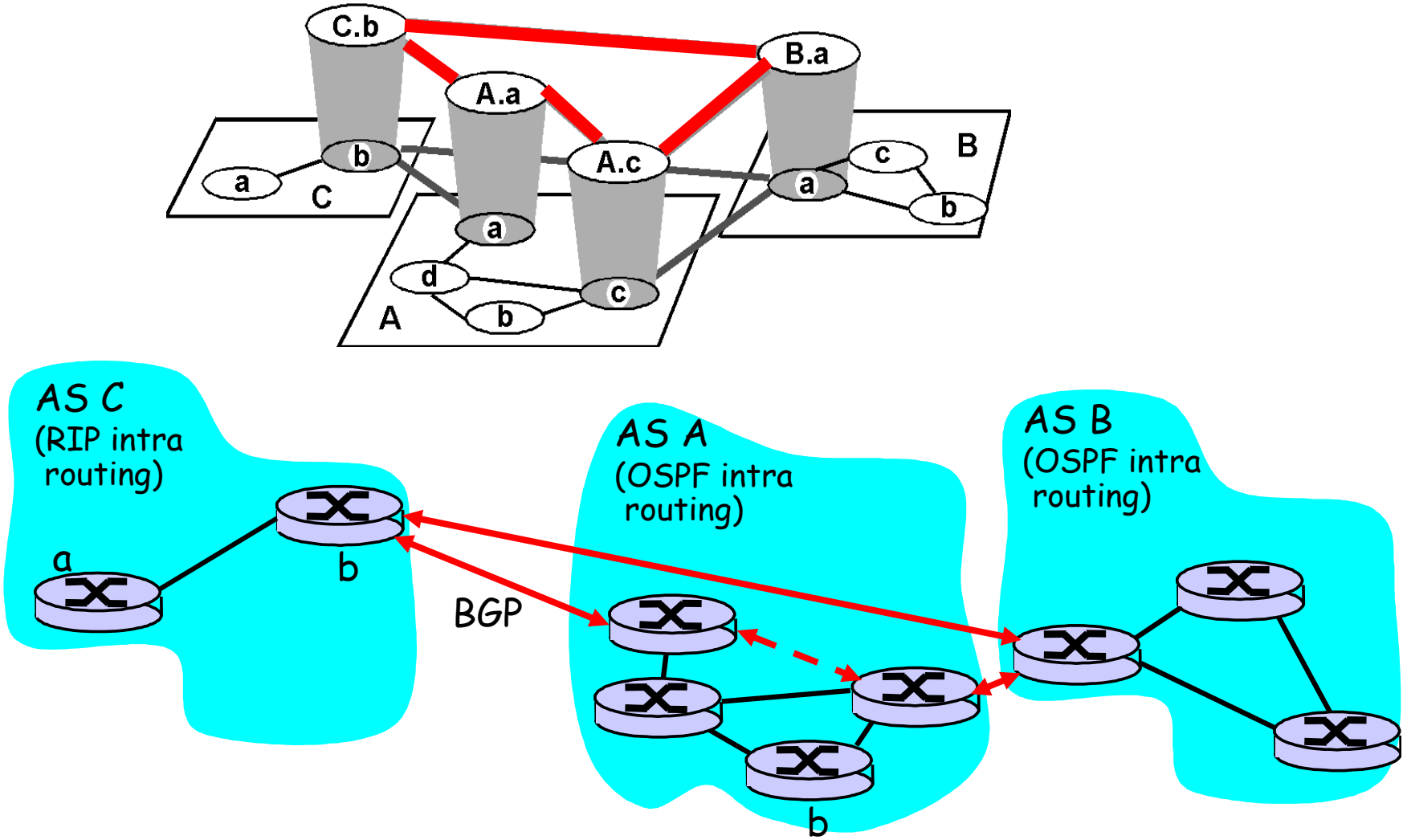
- Định tuyến hai mức:
  - **Intra-AS (Nội miền):** Người quản trị chịu trách nhiệm lựa chọn
  - **Inter-AS (Liên miền):** Chuẩn thống nhất chung

Các kiểu AS khác nhau

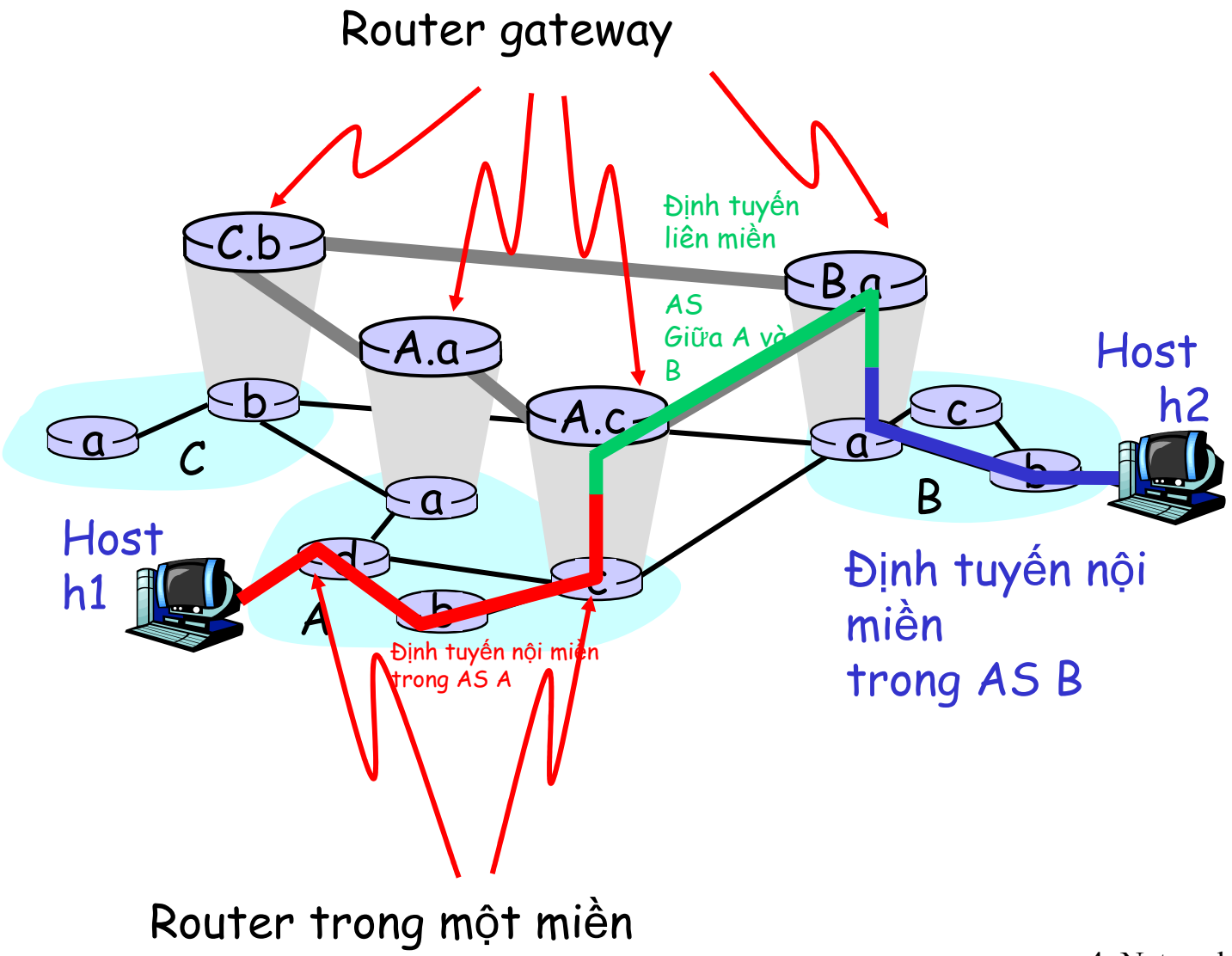


- Transit AS: Các nhà cung cấp
- Non-Transit (stub) AS
  - Phạm vi nhỏ (công ty)
- multihomed AS: Công ty lớn (Không chuyển tiếp)

Ví dụ Định tuyến trên Internet



Định tuyến Liên miền và Nội miền



Định tuyến Nội miền (Intra-AS)

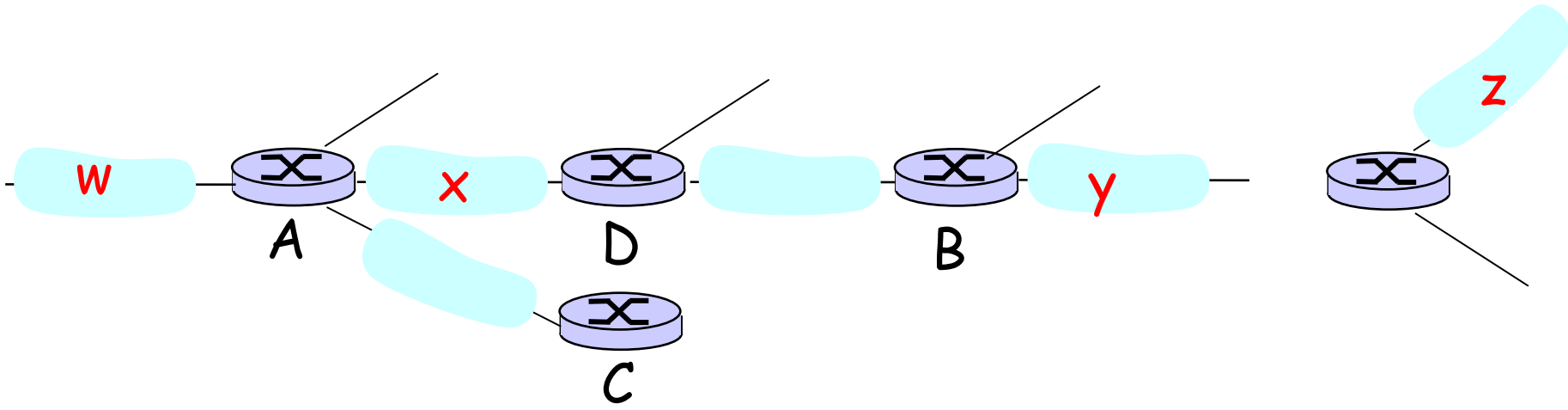
- Còn gọi là Interior Gateway Protocols (IGP)
- Một số IGP phổ biến
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (độc quyền của Cisco)

RIP (Routing Information Protocol)

- Thuật toán Distance vector
- Tích hợp trong HĐH BSD-UNIX vào năm 1982
- Đo khoảng cách: Số chặng (cực đại = 15 chặng)
  - *Tại sao như vậy?*

- Distance vectors : 30s trao đổi thông tin một lần thông qua Response Message (cũng còn gọi là **quảng cáo - advertisement**)
- Mỗi quảng cáo: có thể chuyển tới 25 trạm khác

RIP (Routing Information Protocol)



Mạng Đích	Router Kế tiếp	Số lượng các chặng đến đích.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

Bảng định tuyến trong D

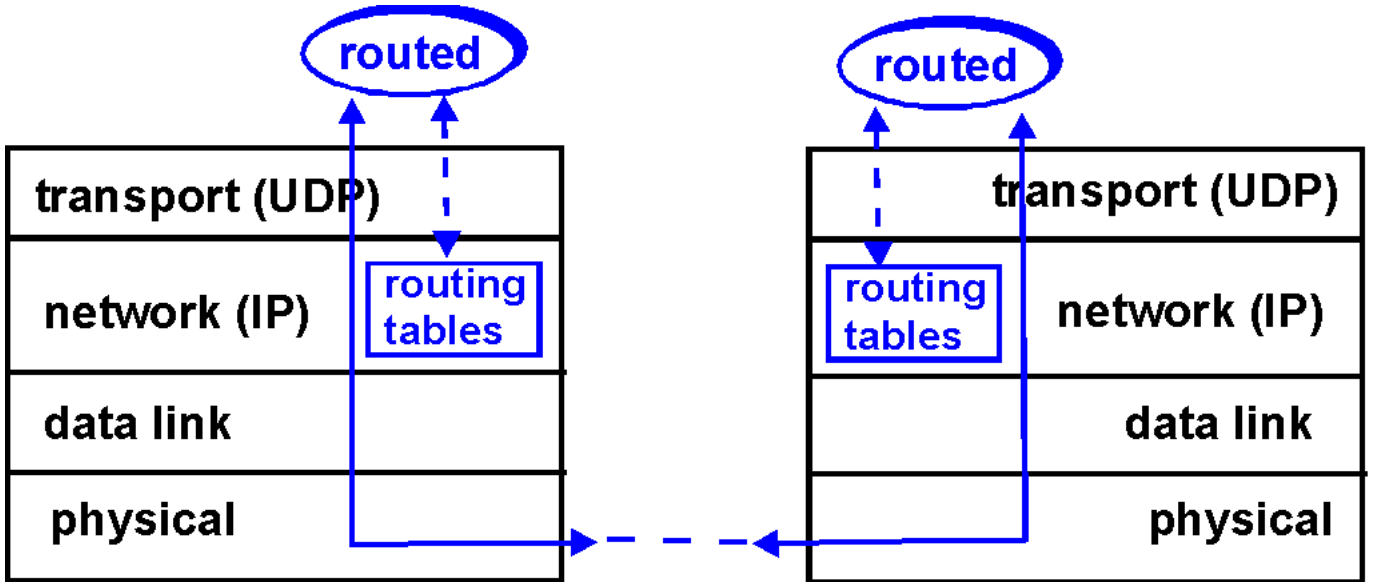
RIP: Đường truyền bị Hỏng và Khôi phục lại

Nếu không nhận được quảng cáo nào trong 180s --> Đường truyền đến hàng xóm coi như bị cắt đứt

- Tuyến đường đến hàng xóm coi như không hợp lệ
- Gửi quảng cáo đến các hàng xóm khác
- Đến lượt mình hàng xóm cũng gửi quảng cáo mới (nếu bảng định tuyến thay đổi)
- Việc một đường truyền bị hỏng nhanh chóng được các router khác biết
- poison reverse được sử dụng để ngăn chặn lặp vô hạn

Xử lý Bảng định tuyến trong RIP

- Bảng định tuyến trong RIP được tiến trình ở **tầng ứng dụng** quản lý (route-d daemon)
- Các quảng cáo được gửi định kỳ trong gói tin UDP



Ví dụ về Bảng định tuyến trong RIP

Router: *giroflee.eurocom.fr*

Destination	Gateway	Flags	Ref	Use	Interface
127.0.0.1	127.0.0.1	UH	0	26492	lo0
192.168.2.	192.168.2.5	U	2	13	fa0
193.55.114.	193.55.114.6	U	3	58503	le0
192.168.3.	192.168.3.5	U	2	25	qaa0
224.0.0.0	193.55.114.6	U	3	0	le0
default	193.55.114.129	UG	0	143454	

- Có nối với Ba mạng LAN lớp C
- Router chỉ biết các tuyến đường nối tới các LAN đó
- Router ngầm định được sử dụng để chuyển đi chỗ khác
- Địa chỉ multicast: 224.0.0.0
- Giao diện Loopback (để debug)

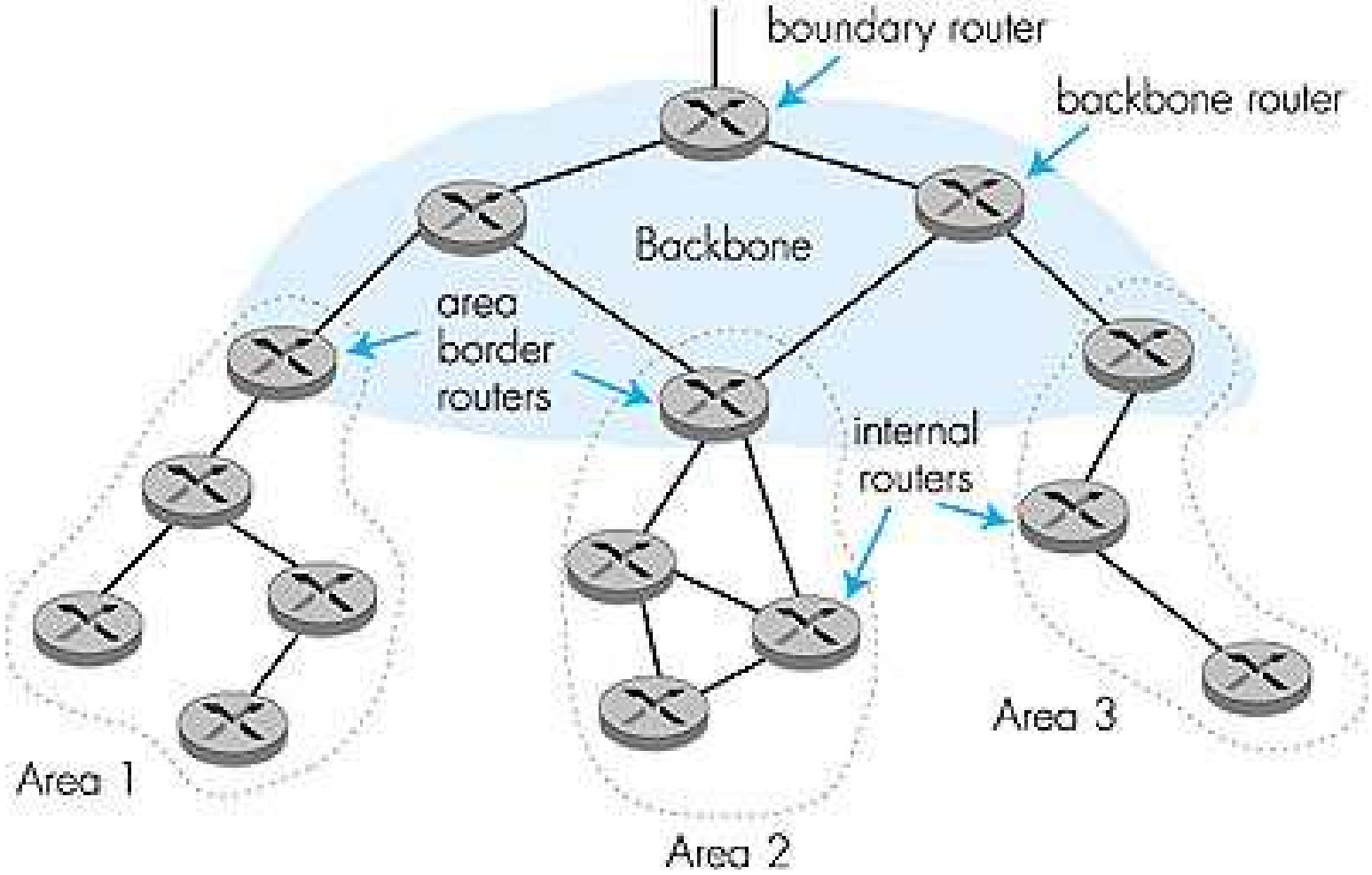
OSPF (Open Shortest Path First)

- "open": chuẩn mở
- Sử dụng thuật toán Link State
  - Gói LS được gửi
  - Nút biết về toàn bộ topo mạng
  - Tuyến đường được xác định nhờ thuật toán Dijkstra
- Thông điệp quảng cáo trong OSPF : Mỗi mục ứng với một router hàng xóm
- Các quảng cáo được gửi trên toàn bộ AS (gửi tràn ngập)

Các đặc điểm "ưu việt" của OSPF (so với RIP)

- **An ninh**: Có thể kiểm chứng các thông điệp OSPF (để ngăn ngừa phá hoại); Sử dụng kết nối TCP
- Cho phép các tuyến đường có cùng một giá (không có trong RIP)
- Trên mỗi đường truyền, có nhiều giá khác nhau cho các **TOS** khác nhau (ví dụ đường truyền vệ tinh có giá "thấp" cho dịch vụ cố gắng tối đa; "cao" cho dịch vụ thời gian thực)
- Hỗ trợ gửi một đích và gửi nhiều đích (multicast):
  - Multicast OSPF (MOSPF) giống OSPF
- **Phân nhỏ** OSPF trong các miền lớn.

Phân cấp OSPF



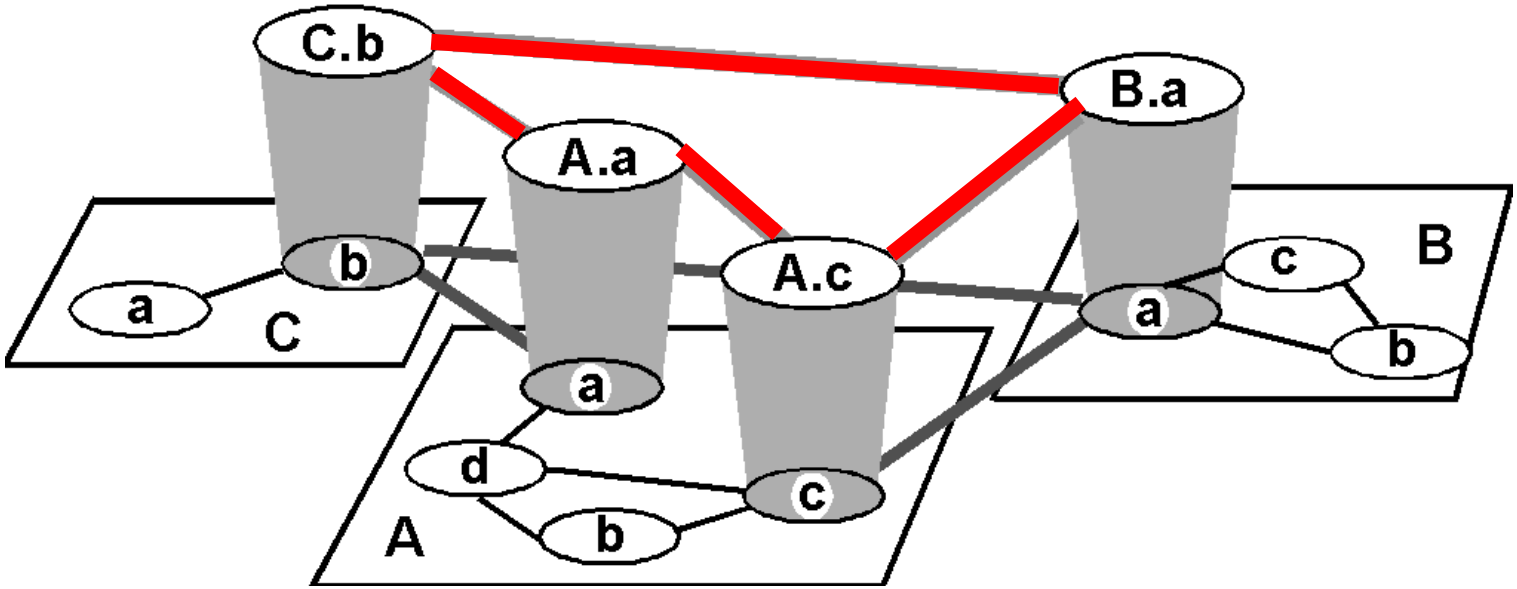
Phân cấp OSPF

- Phân cấp hai mức: cục bộ, trục chính.
  - Quảng cáo Link-state trong một vùng cục bộ
  - Mỗi nút chỉ biết topo trong một vùng; chỉ biết hướng (đường đi tốt nhất) tới các vùng khác.
- **Area border router:** "tổng hợp" khoảng cách đến các nút trong vùng, quảng cáo đến các Area Border router khác.
- **Backbone router:** chạy thuật toán định tuyến OSPF trên trục chính.
- **Boundary router:** Kết nối tới các AS khác.

IGRP (Interior Gateway Routing Protocol)

- Độc quyền của công ty CISCO; thay thế RIP (giữa 1980)
- Distance Vector, giống RIP
- Đo bằng nhiều tiêu chí khác nhau (Độ trễ, Băng thông, Độ tin cậy, Tải…)
- Sử dụng TCP để cập nhật thông tin định tuyến
- Định tuyến không bị lặp thông qua Distributed Updating Alg. (DUAL)

Định tuyến Liên miền (Inter-AS)



BGP - Định tuyến Liên miền trên Internet

- BGP (Border Gateway Protocol): *chuẩn* de facto
- Giao thức **Path Vector** :
  - Tương tự Distance Vector
  - Mỗi Border Gateway quảng bá đến các hàng xóm toàn bộ tuyến đường (là dãy các AS) tới đích
  - Ví dụ Gateway X có thể gửi đường dẫn tới đích Z:

Path (X,Z) = X,Y1,Y2,Y3,…,Z

BGP - Định tuyến Liên miền trên Internet

Giả sử: gateway X gửi tuyến đường đến gateway W

- W có thể hoặc không lựa chọn tuyến đường đi qua X
  - Chi phí, Chính sách (Không chuyển qua AS của công ty thù địch).
- Nếu W lựa chọn tuyến đường do X quảng cáo thì:

$$\text{Path (W,Z)} = w, \text{Path (X,Z)}$$

- Chú ý: X có thể kiểm soát luồng dữ liệu chuyển qua nó bằng cách kiểm soát nội dung quảng cáo gửi đến các hàng xóm:
  - Ví dụ không muốn chuyển tiếp gói tin đến Z -> không quảng cáo tuyến đường nào đến Z

BGP - Định tuyến Liên miền trên Internet

- Thông điệp BGP được đặt trong gói tin TCP.
- Thông điệp BGP :
  - OPEN: Mở kết nối TCP tới nút đối tác, kiểm chứng
  - UPDATE: Quảng cáo tuyến đường mới (hoặc xóa tuyến đường cũ)
  - KEEPALIVE Giữ tuyến đường ở trạng thái kết nối khi không gửi UPDATE; Biên nhận cho yêu cầu OPEN
  - NOTIFICATION: Báo lỗi trong thông điệp trước; cũng còn được sử dụng để đóng kết nối

Phân biệt Định tuyến Liên miền – Nội miền ?

Chính sách:

- Inter-AS: Người quản trị mong muốn kiểm soát thông tin truyền qua mạng của mình.
- Intra-AS: Một người quản trị duy nhất, do vậy không cần đến chính sách quản lý

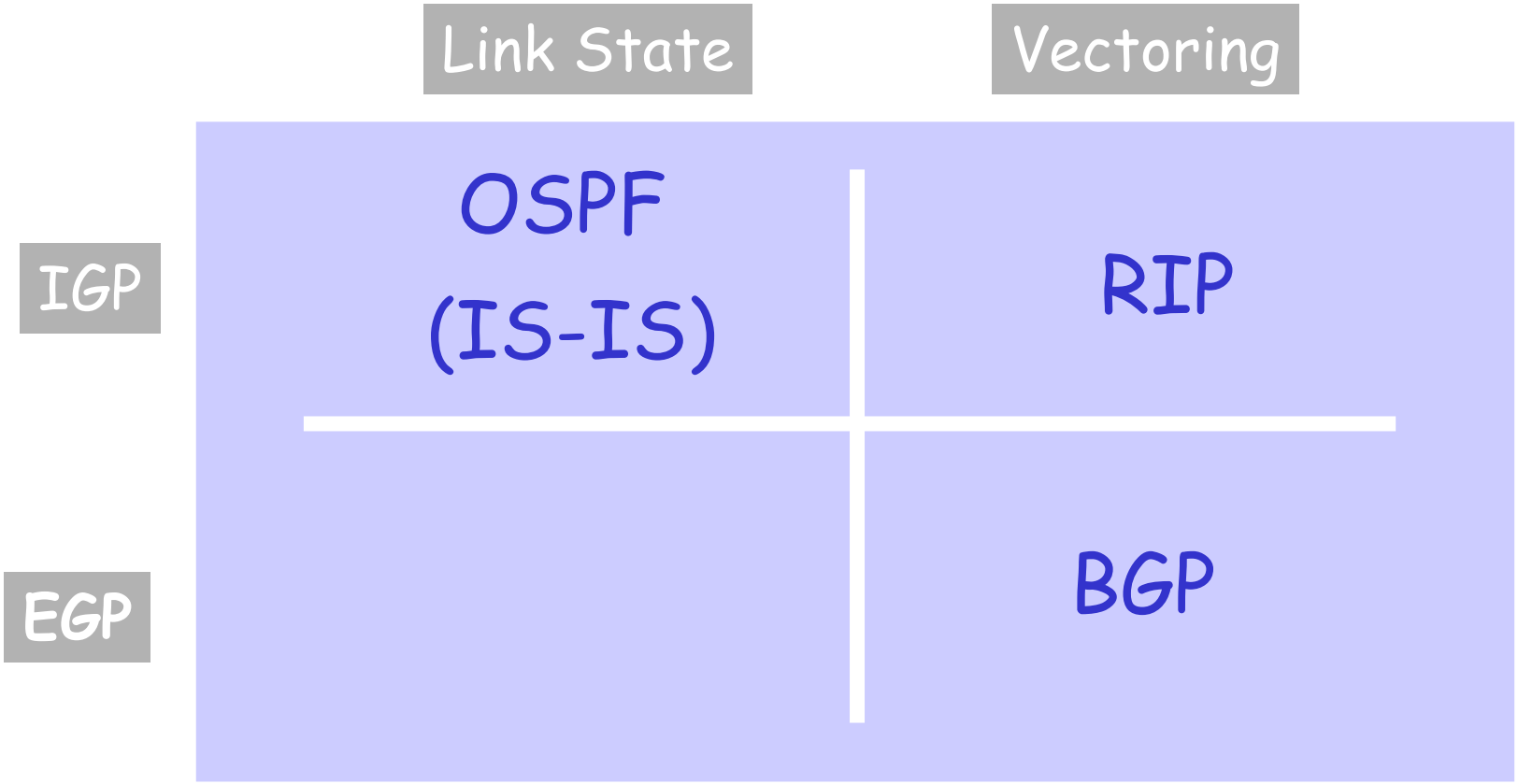
Mở rộng:

- Phân cấp giúp giảm kích thước bảng và giảm khối lượng thông tin cập nhật

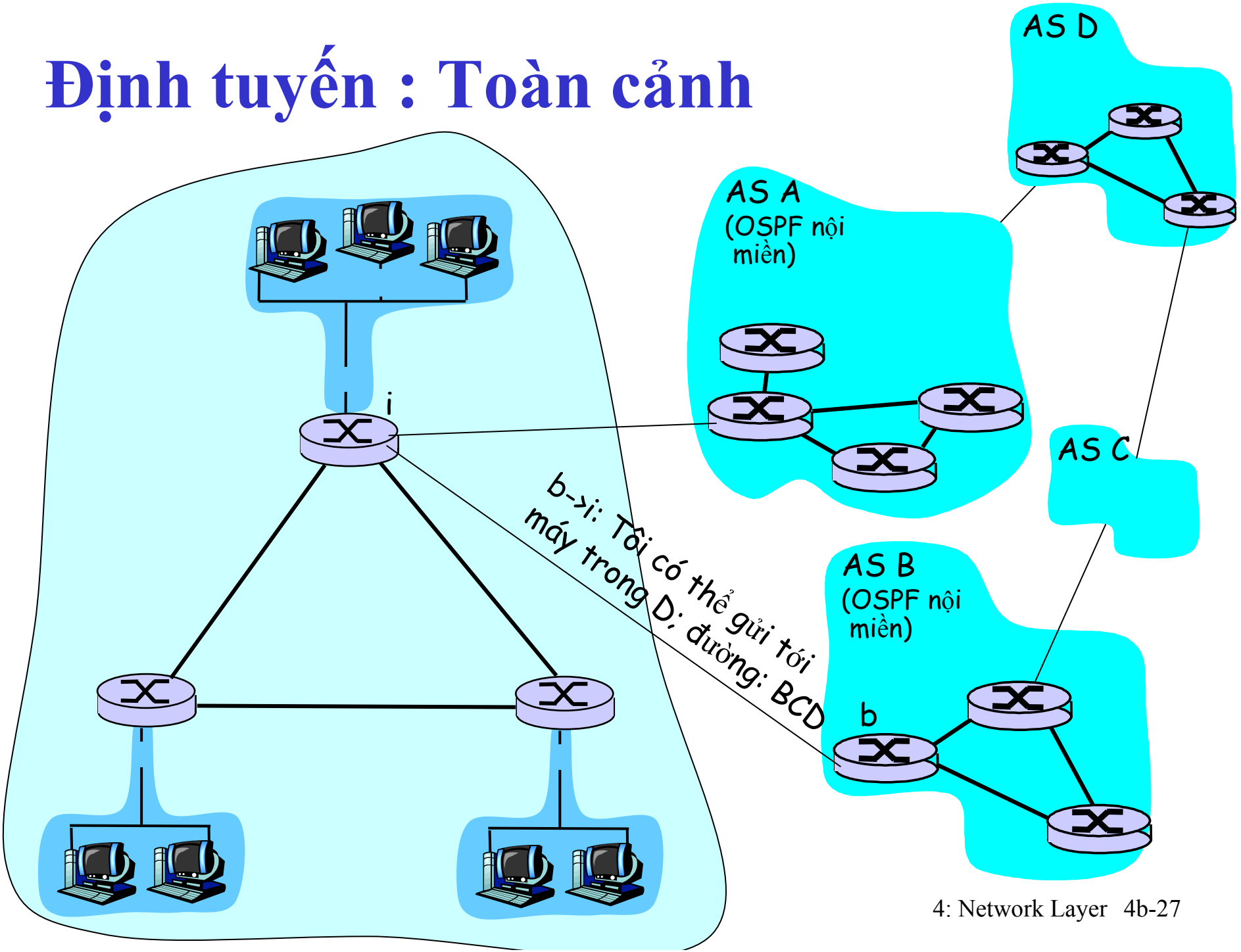
Hiệu suất:

- Intra-AS: Tập trung vào Khía cạnh Hiệu suất
- Inter-AS: Chính sách có thể được ưu tiên hơn Hiệu suất

Summary: The Gang of Four

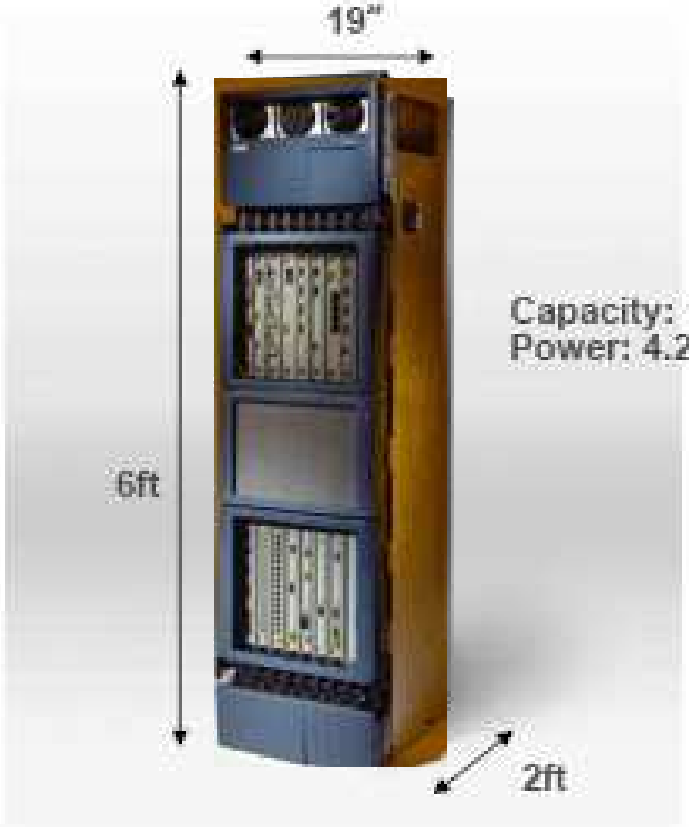


Định tuyến : Toàn cảnh



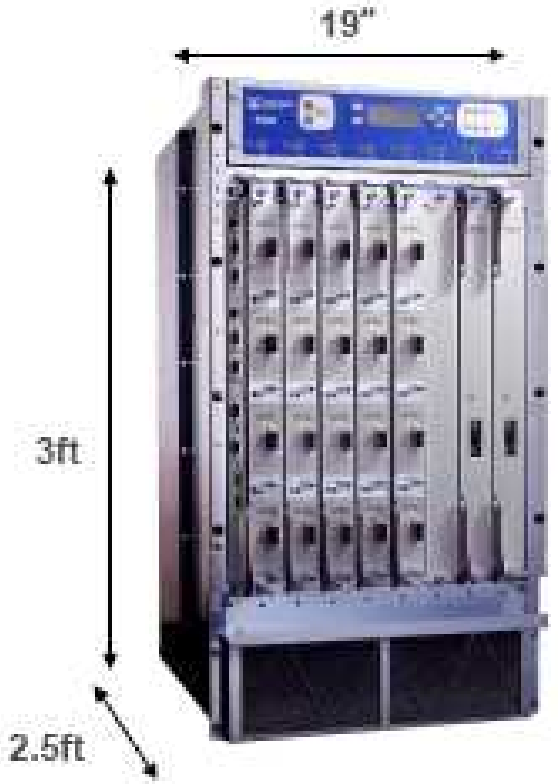
Router trông như thế nào ?

Cisco GSR 12416



Capacity: 160Gb/s
Power: 4.2kW

Juniper M160

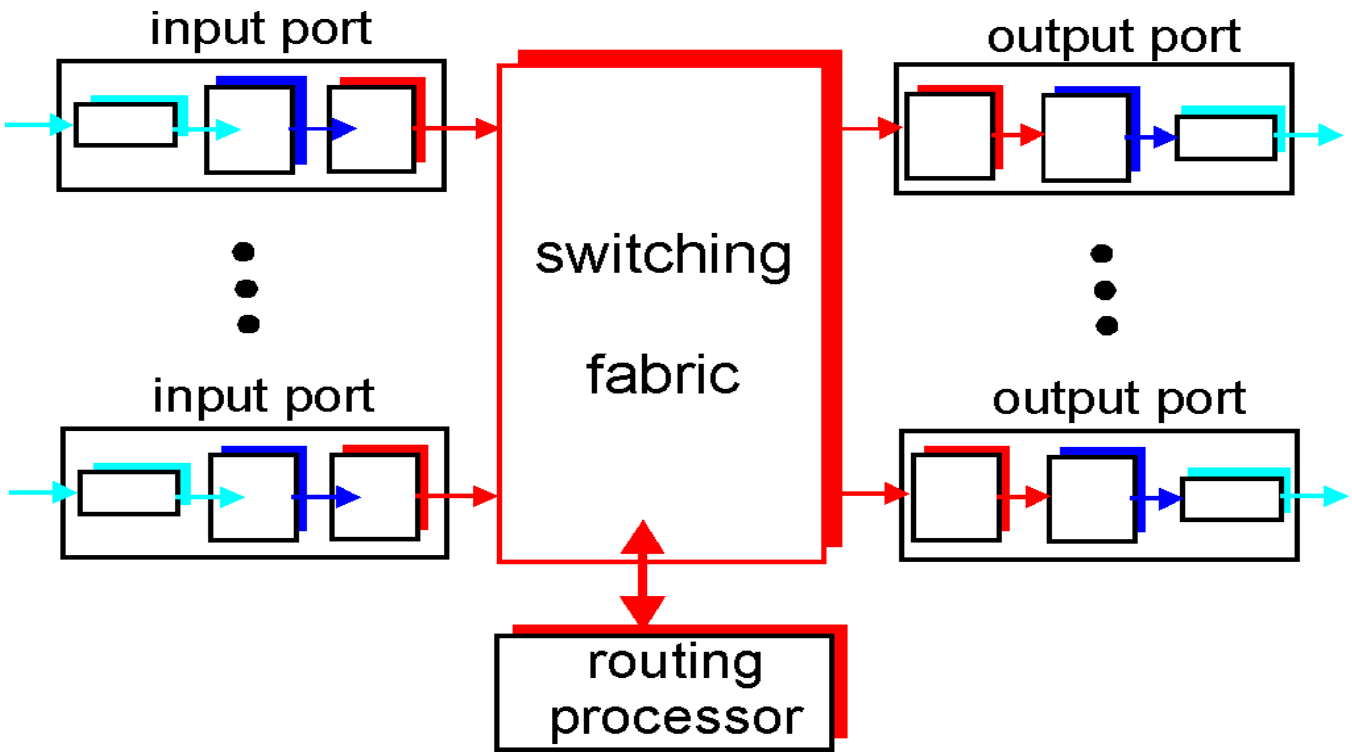


Capacity: 80Gb/s
Power: 2.6kW

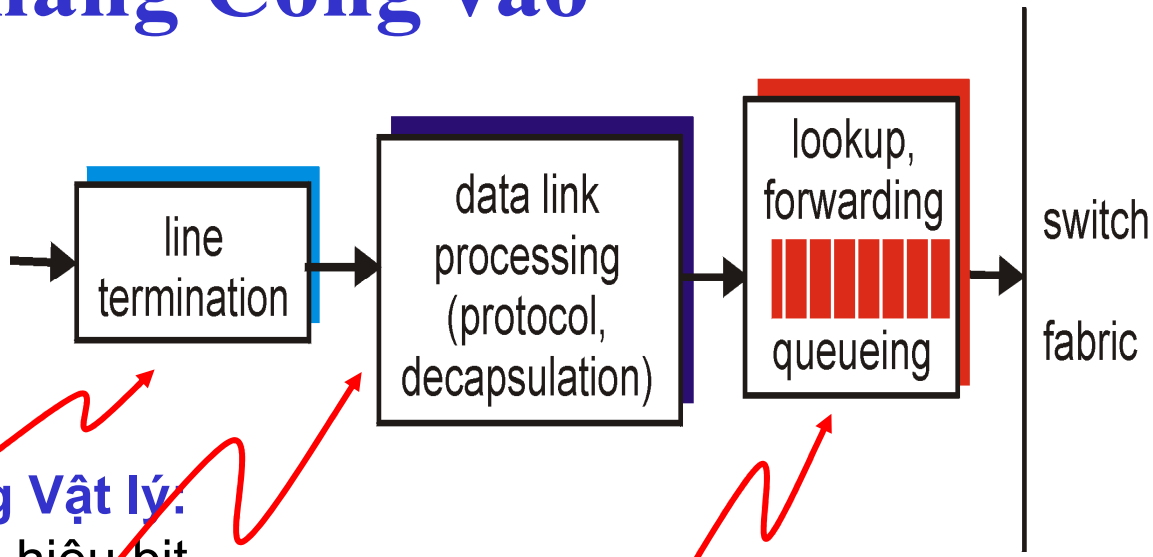
Tổng quan Kiến trúc của Router

Hai chức năng chính:

- Chạy các thuật toán, giao thức định tuyến (RIP, OSPF, BGP)
- *Chuyển* datagram từ cổng vào đến cổng ra thích hợp



Chức năng Cổng vào



Tầng Vật lý:
Nhận tín hiệu bit

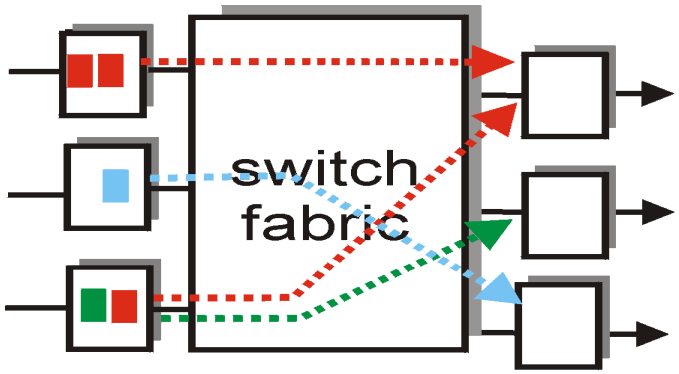
Liên kết Dữ liệu
Ví dụ Ethernet (chương 5)

Chuyển Không tập trung

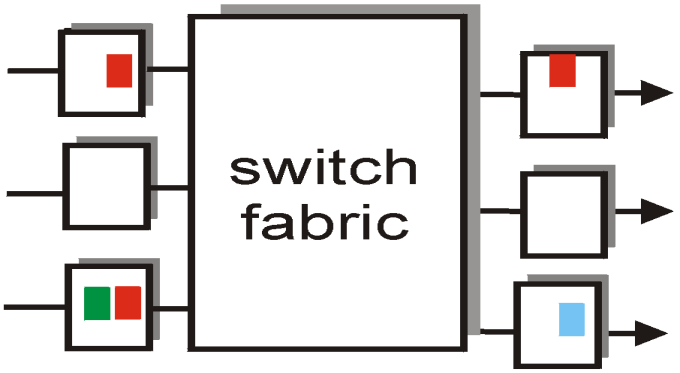
- Với một địa chỉ đích, tìm kiếm trên bảng định tuyến để xác định cổng ra phù hợp
- Mục tiêu: Thực hiện xử lý tại cổng vào ở tốc độ đến của gói tin
- Hàng đợi: Xuất hiện khi tốc độ đến của gói tin nhanh hơn khả năng chuyển đi của Kết cấu chuyển

Hàng đợi tại Cổng vào

- Kết cấu chuyển chậm hơn tổng lượng đến từ các cổng vào -> Xuất hiện Hàng đợi ở một số cổng
- Phong tỏa Đầu Hàng đợi (Head-of-the-Line HOL) : datagram ở đầu hàng đợi bị phong tỏa ngăn chặn các datagram sau
- *Độ trễ hay Mất do xếp hàng Vì tràn bộ đệm tại Cổng vào!*

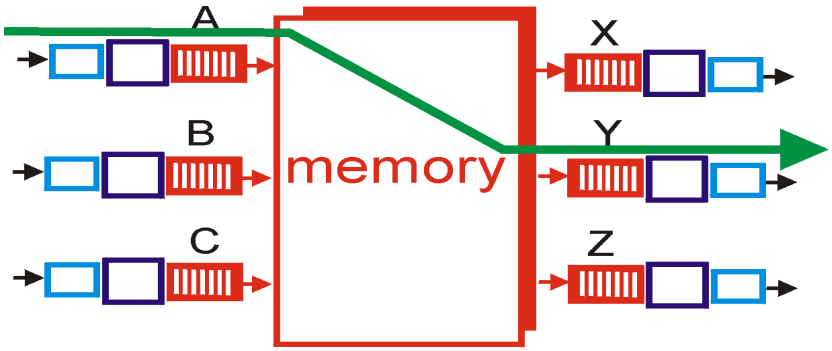


output port contention
at time t - only one red
packet can be transferred

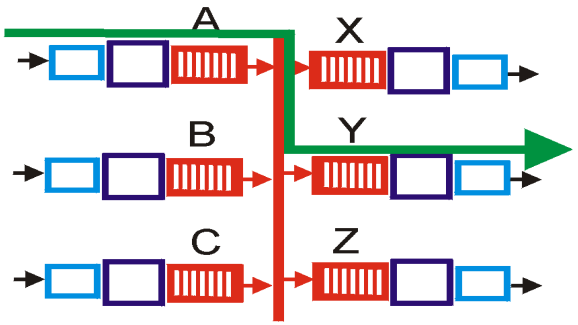


green packet
experiences HOL blocking

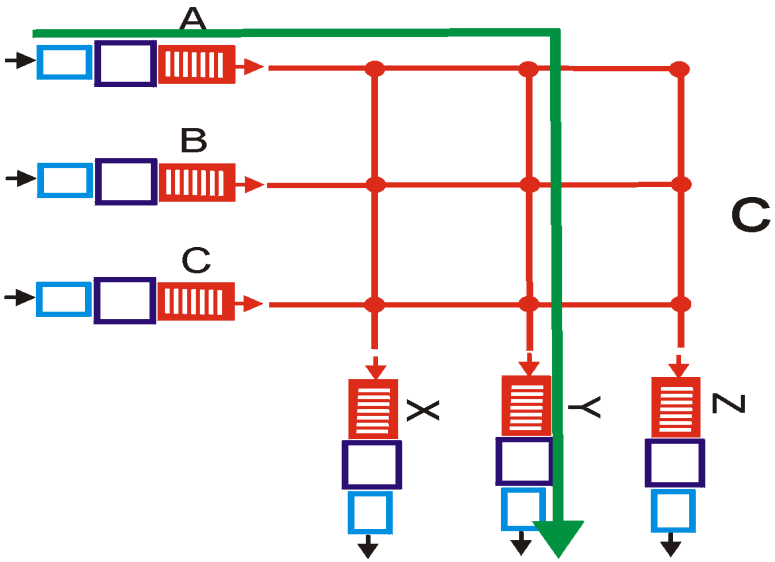
Ba kiểu Kết cấu chuyển



memory



bus

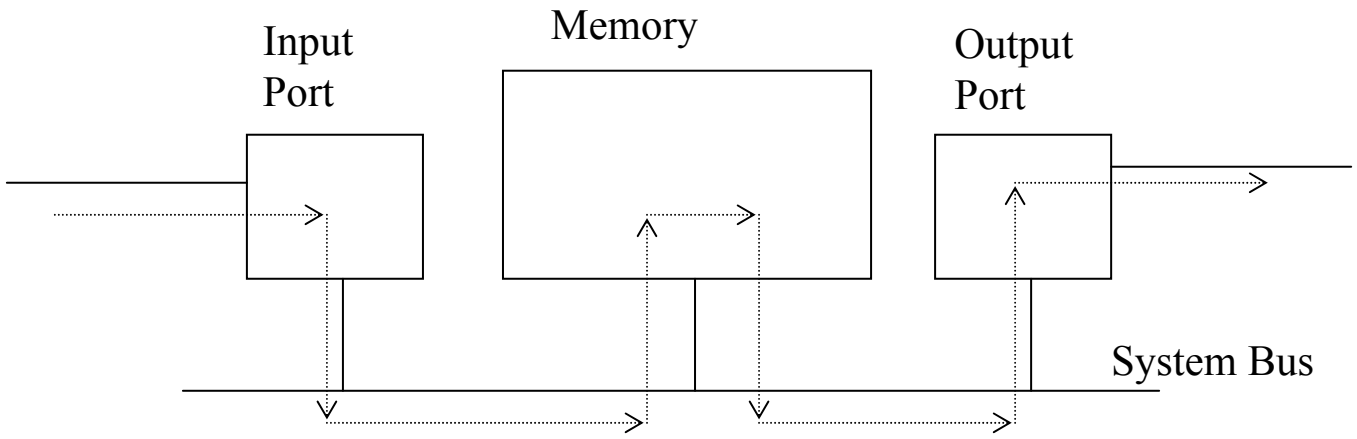


crossbar

Chuyển qua Bộ nhớ

Các Router thế hệ đầu tiên:

- CPU (duy nhất) thực hiện sao chép các packet
- Tốc độ bị giới hạn bởi băng thông bộ nhớ (mỗi datagram cần 2 lần sử dụng bus)



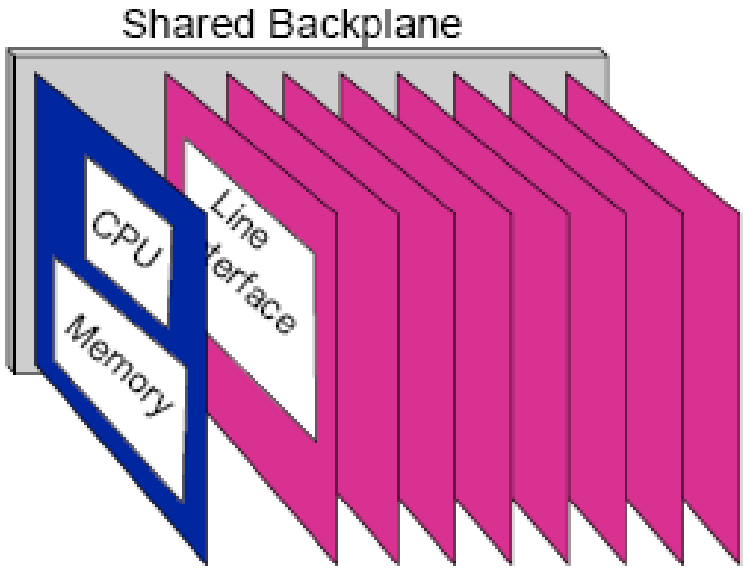
Router hiện đại:

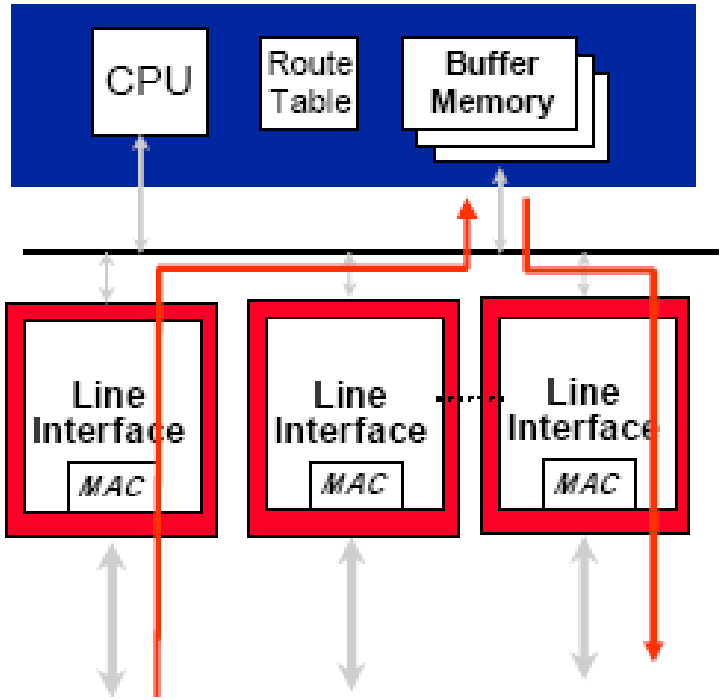
- CPU tại cổng vào thực hiện tìm kiếm và chuyển vào bộ nhớ
- Cisco Catalyst 8500

Chuyển qua Bộ nhớ

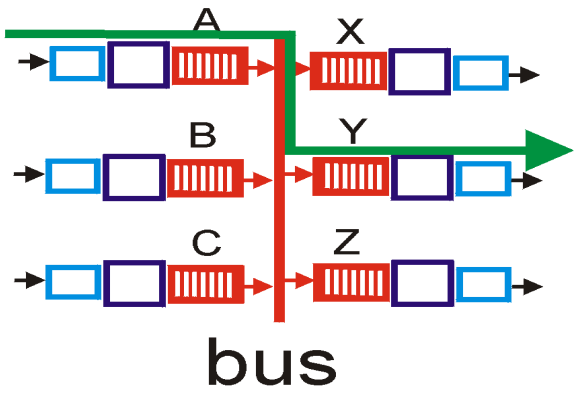
Thế hệ router đầu tiên: CPU duy nhất của máy tính di chuyển các packet

- Nút cổ chai: Bộ nhớ dùng chung; datagram chuyển qua bus 2 lần





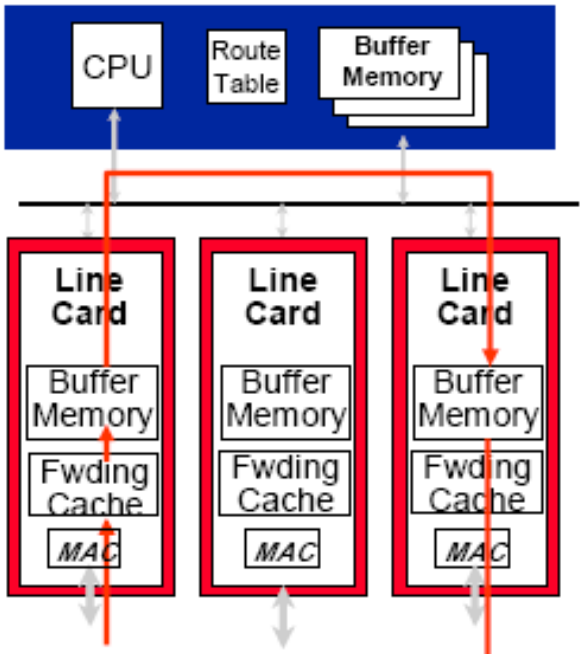
Chuyển qua Bus



- datagram chuyển từ Bộ nhớ Cổng vào đến Bộ nhớ Cổng ra qua bus dùng chung
- Tranh chấp bus: Tốc độ chuyển mạch bị giới hạn bởi băng thông của bus
- 1 Gbps bus, Cisco 1900: Tốc độ đủ cao cho các router của công ty

Chuyển qua Bus

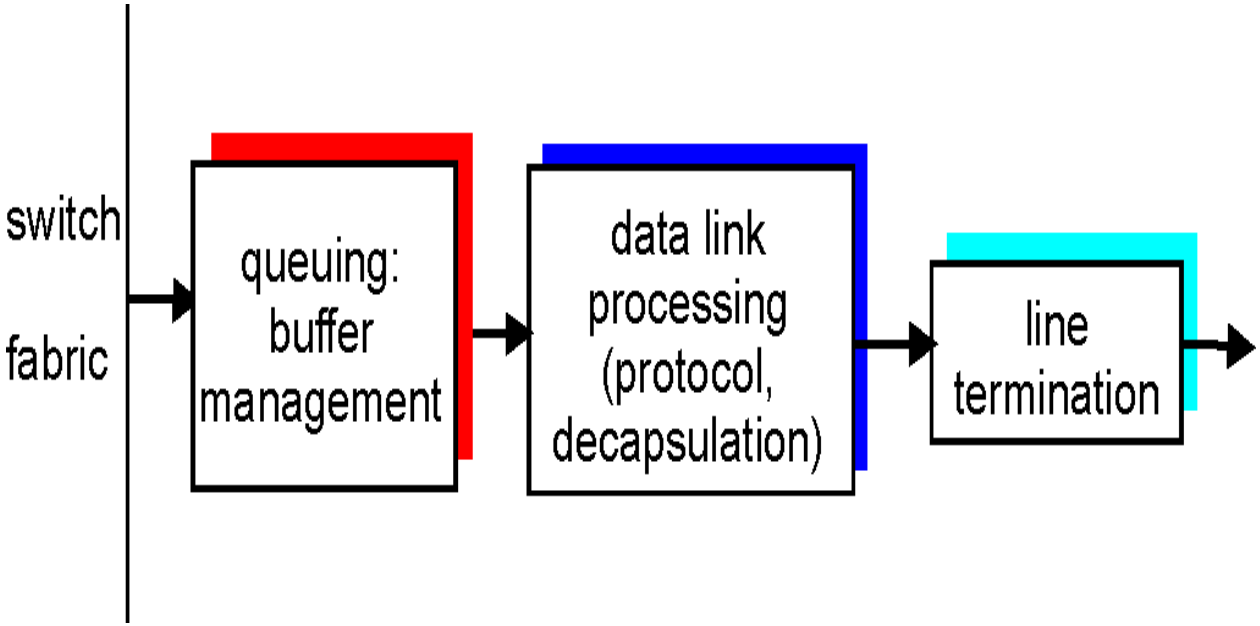
- Datagram từ Bộ nhớ cổng vào tới Bộ nhớ cổng ra qua bus dùng chung
- Nút cổ chai: tranh chấp bus
  - < 5Gbps, ví dụ 1 Gbps bus, Cisco 1900: tốc độ truy cập vừa đủ cho router của công ty (không phải router trên các trục chính)



Chuyển qua Mạng liên hợp

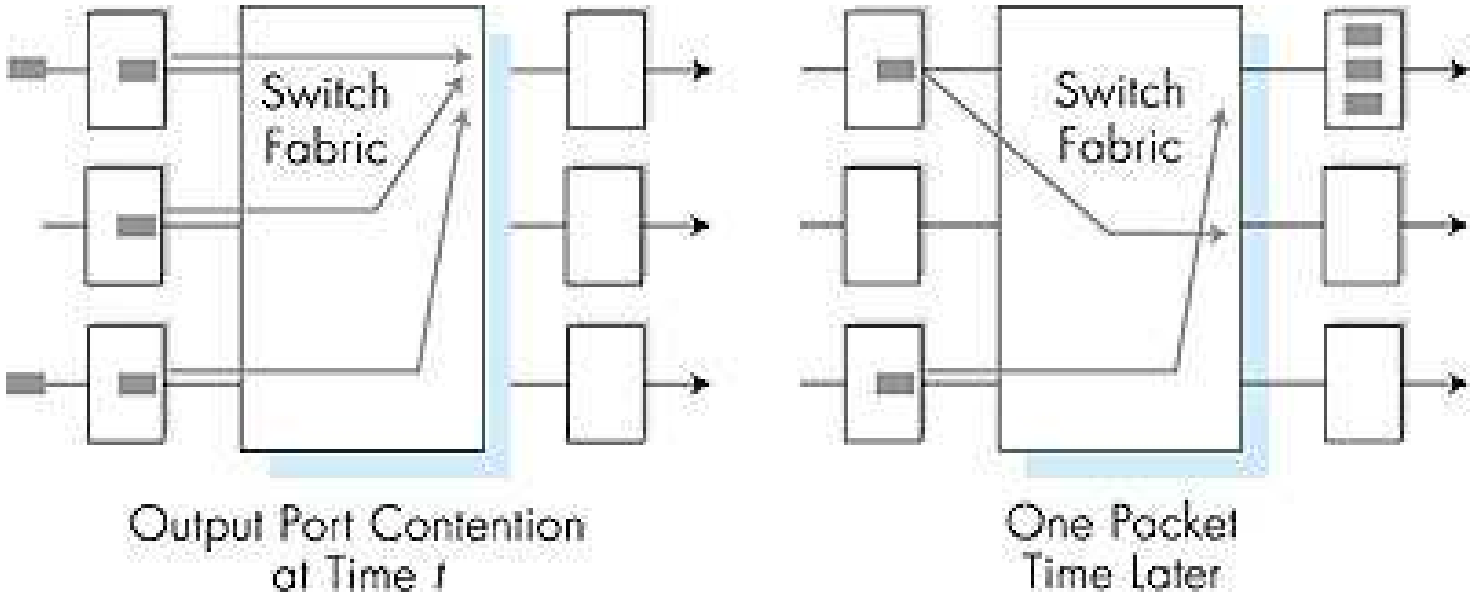
- Khắc phục hạn chế Băng thông của Bus
- Mạng Banyan, Khởi đầu để kết nối các CPU trong hệ thống có nhiều CPU
- Thiết kế cao cấp: chia datagram thành các "tế bào" có kích thước cố định và chuyển các "tế bào" qua kết cấu chuyển.
- Cisco 12000: Tốc độ Gbps qua mạng kết cấu chuyển

Cổng Ra



- ***Bộ đệm*** : Cần thiết trong trường hợp khi gói tin đến từ Kết cấu chuyển nhanh hơn Tốc độ gửi của Kênh truyền
- ***Chiến lược điều phối*** lựa chọn datagram để chuyển trong các datagram nằm ở Bộ đệm

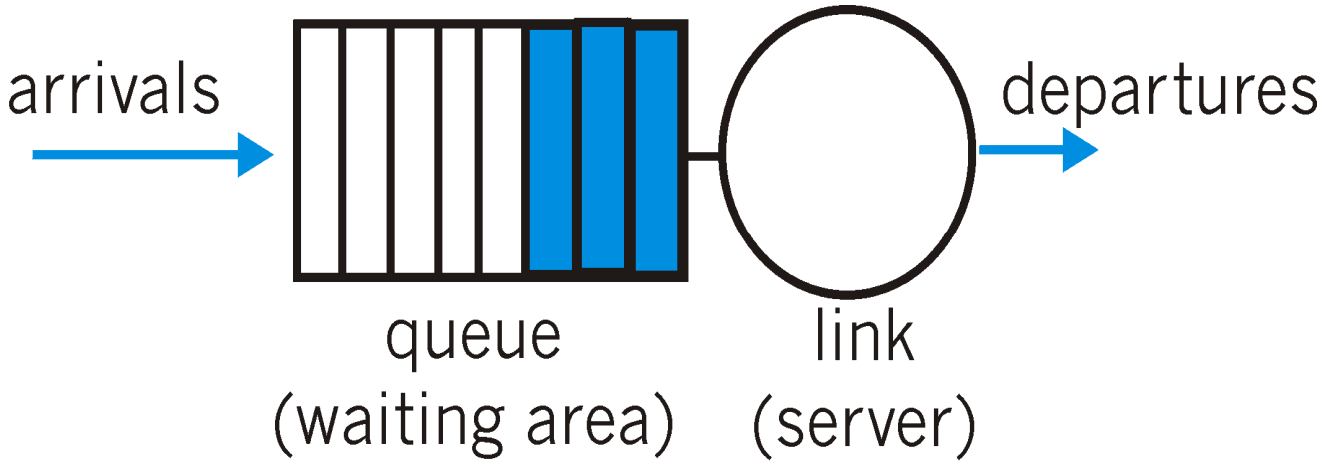
Hàng đợi tại Cổng ra



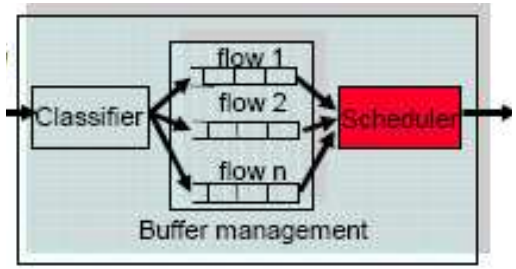
- Tốc độ gửi chậm hơn tốc độ kết cấu chuyển
- *Xếp hàng (Trễ) và Mất dữ liệu do **tràn Bộ đệm** ở Cổng ra!*

Cơ chế Điều phối

- Điều phối: Chọn packet kế tiếp để chuyển đi trên đường truyền
- Điều phối FIFO (first in first out) : Gửi theo thứ tự đến Hàng đợi

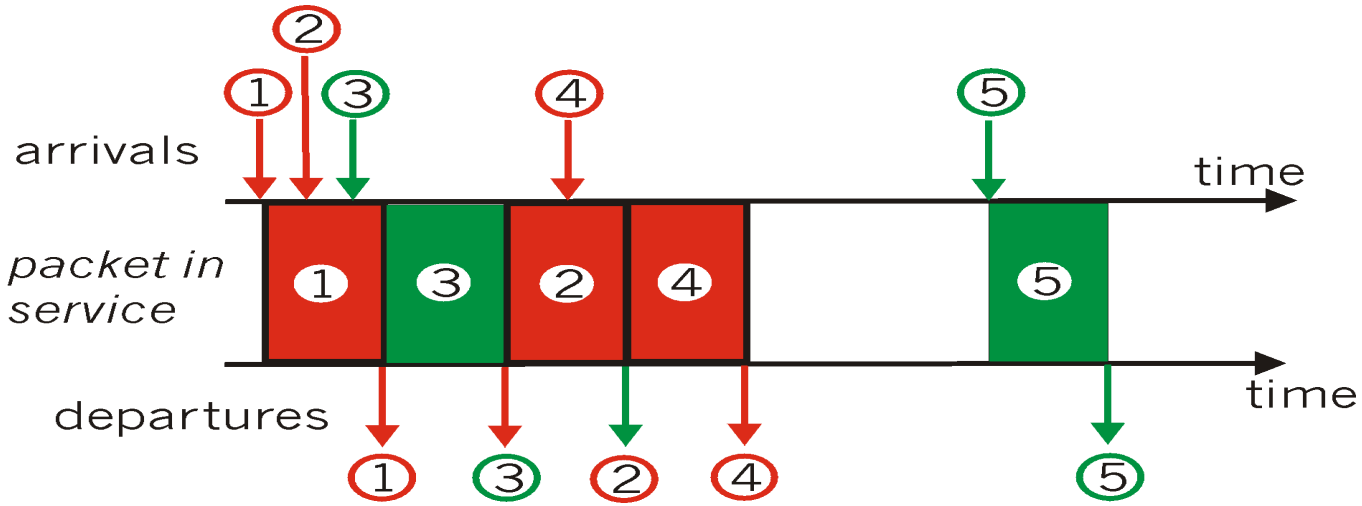


Các chính sách điều phối khác



Điều phối xoay vòng (Round Robin):

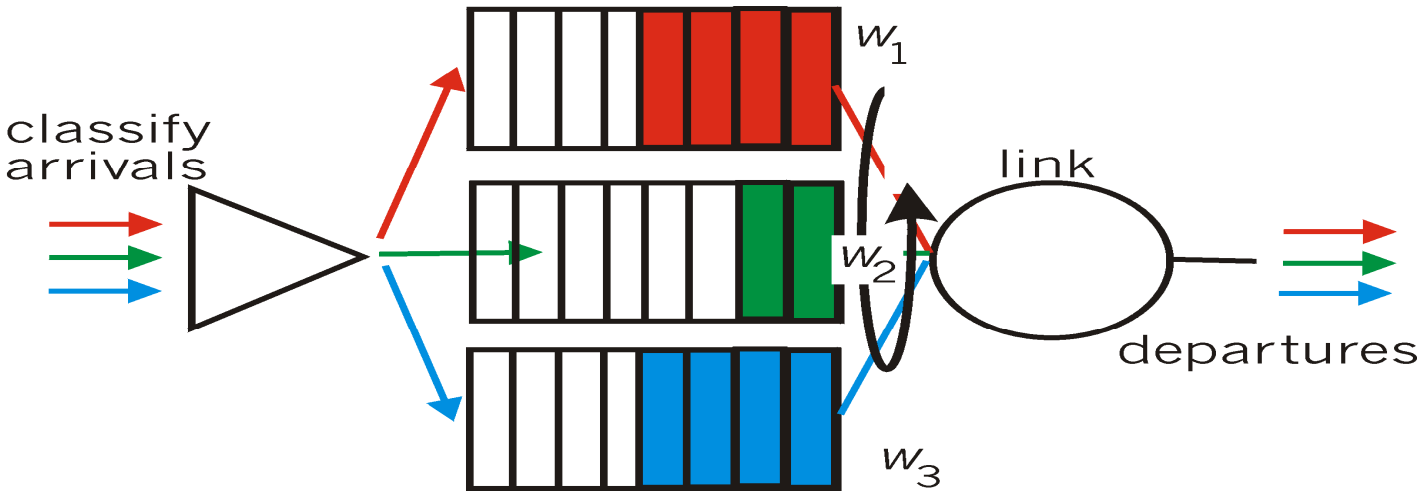
- Chia ra nhiều lớp
- Lần lượt và xoay vòng phục vụ từng hàng đợi



Các chính sách điều phối khác

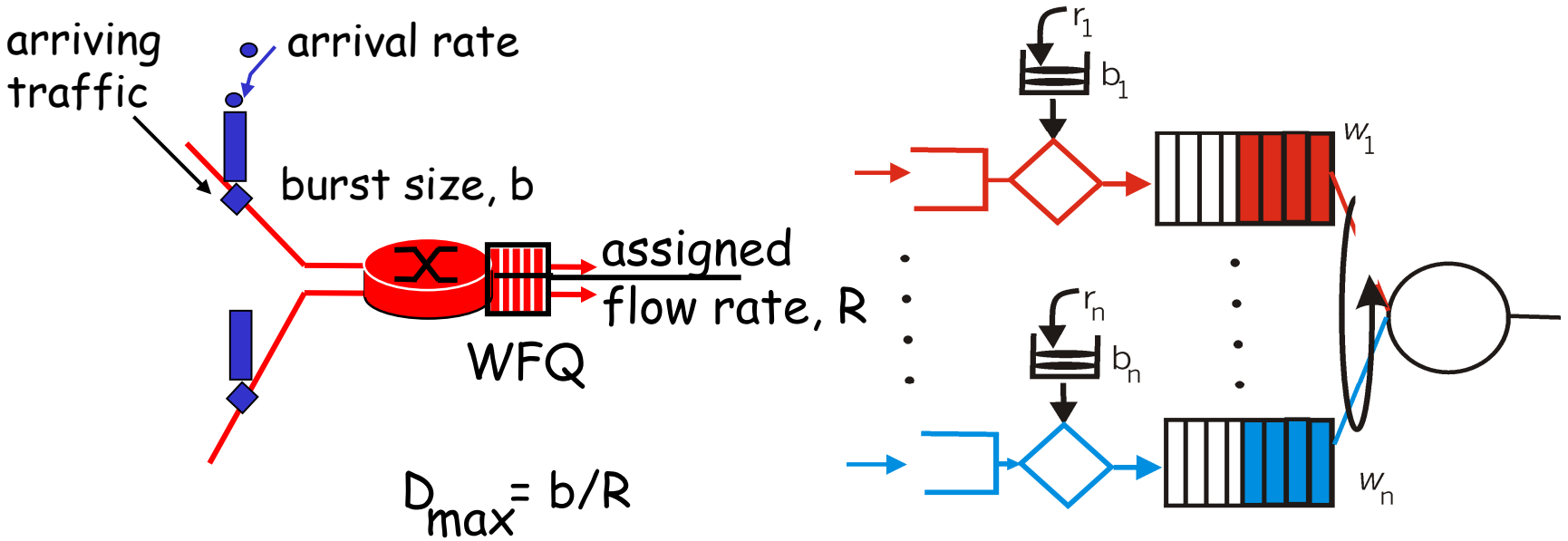
Theo trọng số (Weighted Fair Queuing):

- Tổng quát hóa Round Robin
- Mỗi lớp có trọng số phục vụ riêng



Đảm bảo giới hạn Độ trễ

- WFQ đảm bảo cận trên của độ trễ tức là vấn đề *Đảm bảo Chất lượng Dịch vụ (QoS guarantee)*!



IPv6

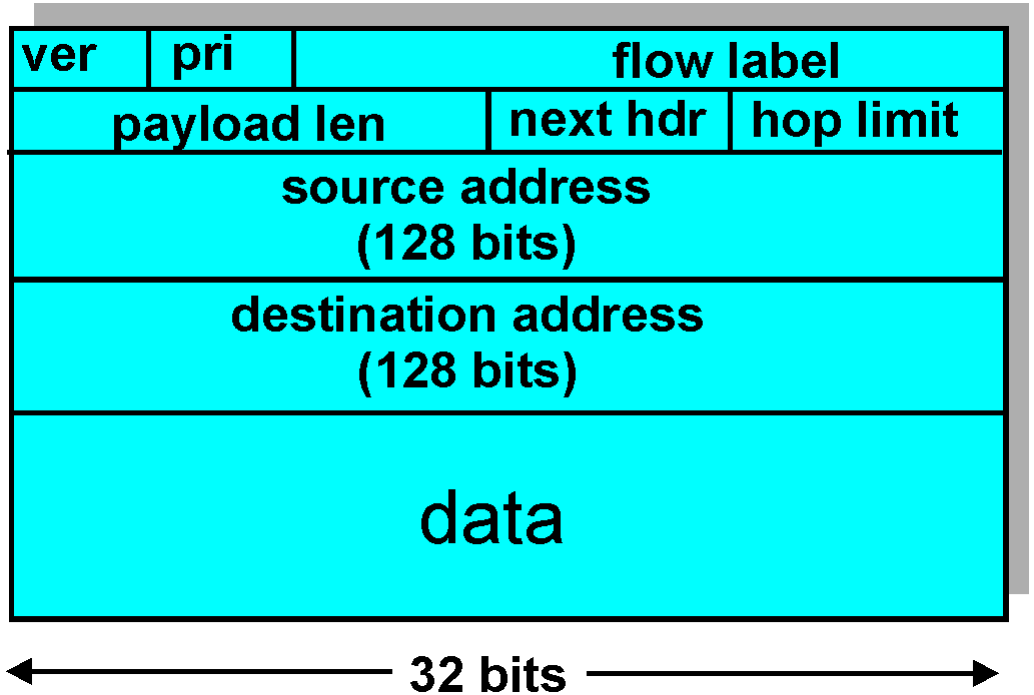
- **Động lực đầu tiên**: Không gian địa chỉ 32-bit sẽ cạn kiệt vào 2008.
- Các động lực khác:
  - Khuôn dạng của tiêu đề ảnh hưởng đến tốc độ Xử lý và Chuyển tiếp gói tin
  - Thay đổi tiêu đề để đáp ứng Chất lượng Dịch vụ (QoS)
  - Địa chỉ kiểu "anycast" : tuyến đường "tốt nhất" trong một vài server nhân bản
- **Khuôn dạng gói IPv6 :**
  - Tiêu đề có độ dài cố định 40 byte
  - Không cho phép Phân mảnh

Tiêu đề của IPv6

Priority: Xác định độ ưu tiên trong luồng dữ liệu

Flow Label: xác định các datagrams trong cùng một "luồng" (khái niệm "luồng" chưa rõ ràng).

Next header: Xác định giao thức giao vận nhận dữ liệu



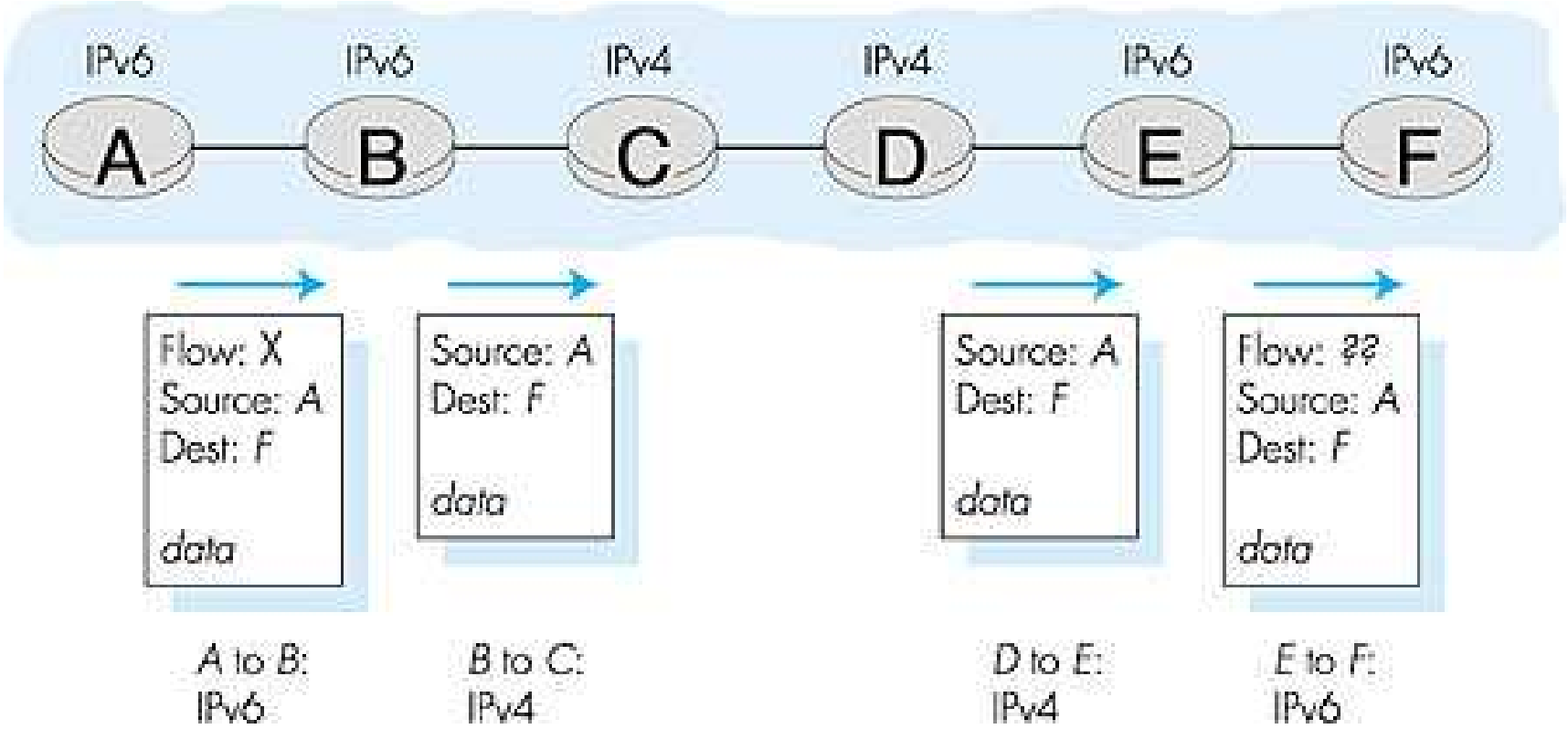
Khác biệt so với IPv4

- *Checksum*: Bị loại bỏ để tăng tốc độ xử lý gói tin tại mỗi router
- *Options:* cho phép, nhưng nằm ngoài tiêu đề, được xác định qua trường "Next Header"
- *ICMPv6:* Phiên bản mới của ICMP
  - Các kiểu thông điệp mới, ví dụ "Packet Too Big"
  - Chức năng quản lý nhóm Multicast

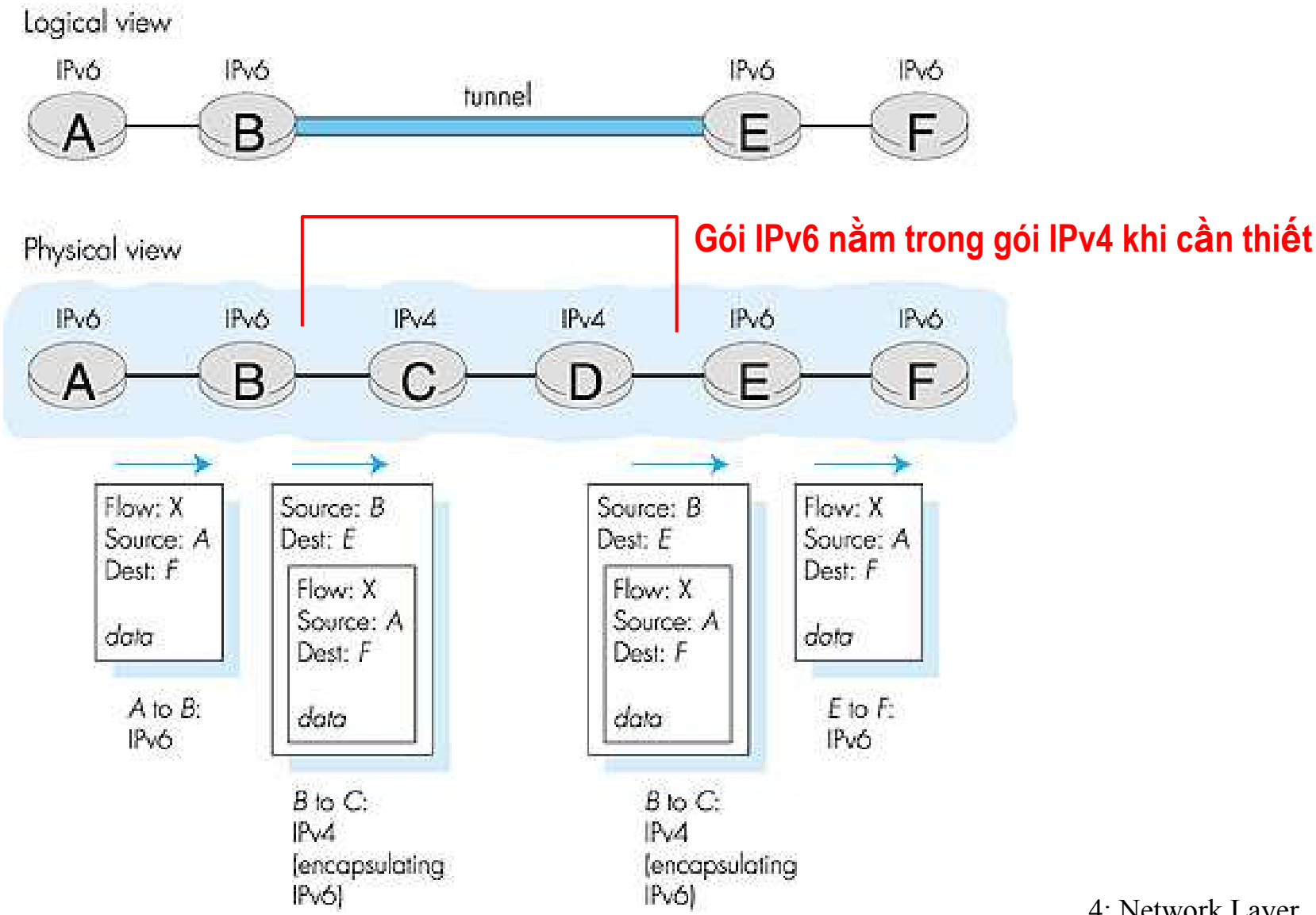
Chuyển từ IPv4 sang IPv6

- Không thể đồng thời nâng cấp tất cả router
  - Không chọn được ngày
  - Làm thế nào để tồn tại cả hai Hệ thống IPv4 và IPv6?
- Hai giải pháp được đề xuất:
  - *Dual Stack*: Một số router hỗ trợ cả 2 bộ giao thức (v6, v4) để "biên dịch" giữa hai khuôn dạng
  - *Tunneling:* Gói tin IPv6 được đặt trong trường dữ liệu của gói tin IPv4 khi truyền giữa các router IPv4

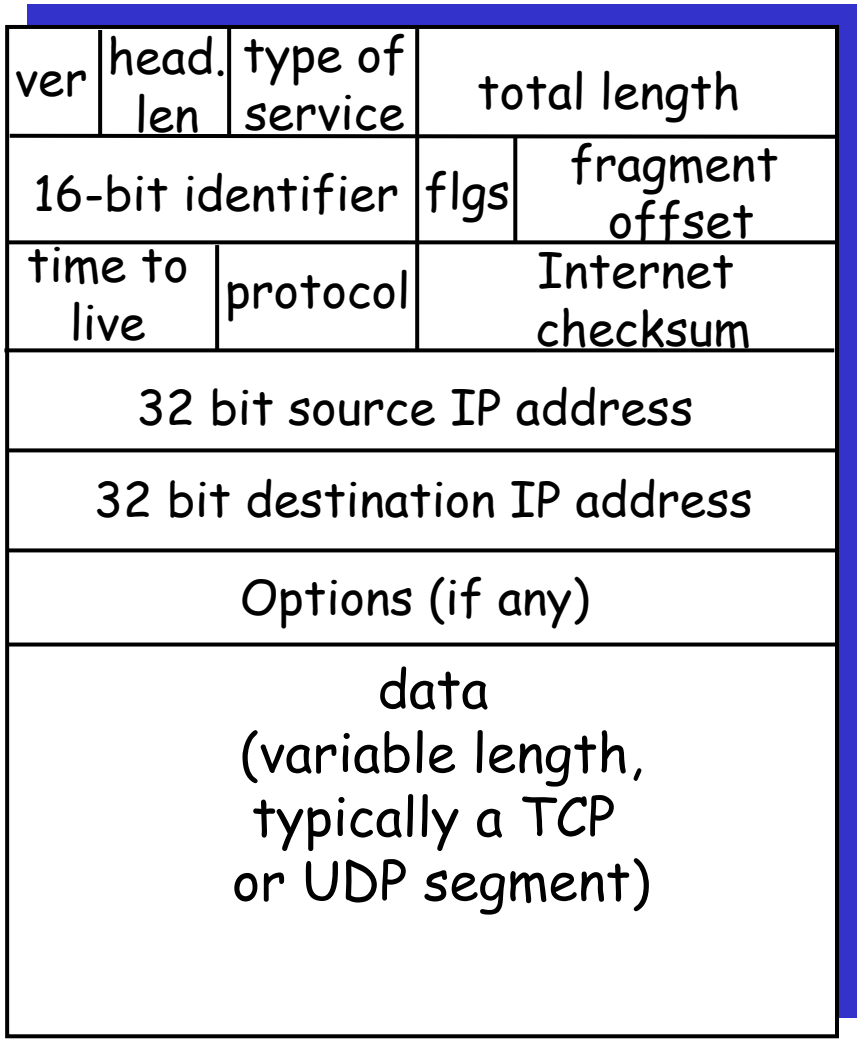
Giải pháp Dual Stack

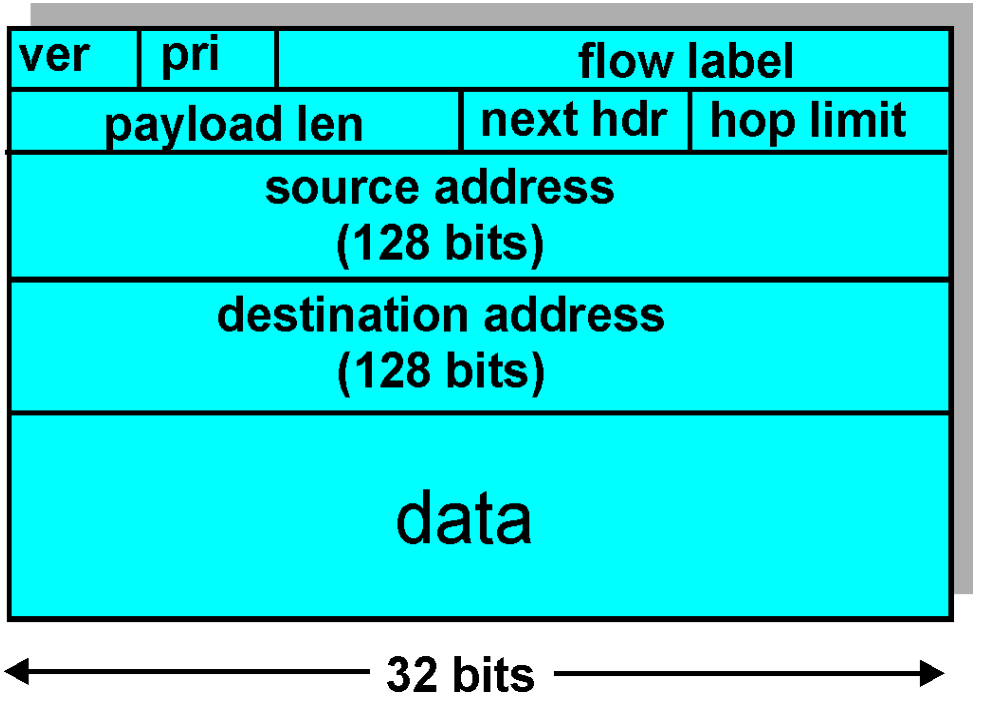


Đường ống (Tunneling)



IPv4 vs. IPv6


ver
head. len
type of service
total length
16-bit identifier
flgs
fragment offset
time to live
protocol
Internet checksum
32 bit source IP address
32 bit destination IP address
Options (if any)
data
(variable length,
typically a TCP
or UDP segment)


ver
pri
flow label
payload len
next hdr
hop limit
source address
(128 bits)
destination address
(128 bits)
data
32 bits

Chương 5: Tầng Liên kết dữ liệu

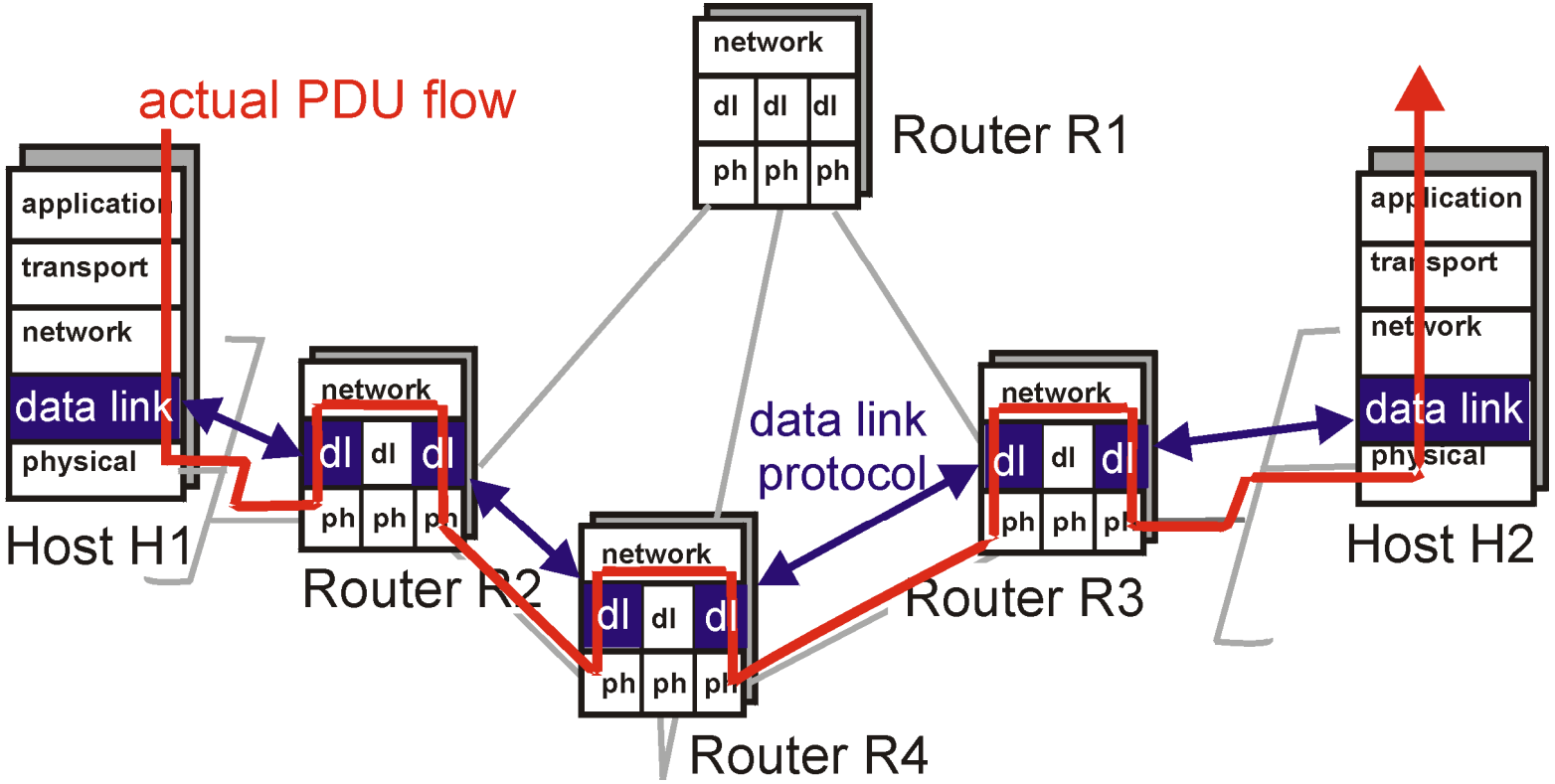
Mục tiêu:

- Các nguyên lý cung cấp dịch vụ của tầng Liên kết dữ liệu:
  - Phát hiện và Sửa lỗi
  - Chia sẻ kênh truyền dùng chung: đa truy cập
  - Địa chỉ tầng link
  - Truyền tin cậy, Điều khiển lưu lượng: ***đã học !***
- Cài đặt trên các công nghệ Liên kết dữ liệu khác nhau (rất nhiều)

Sẽ học gì:

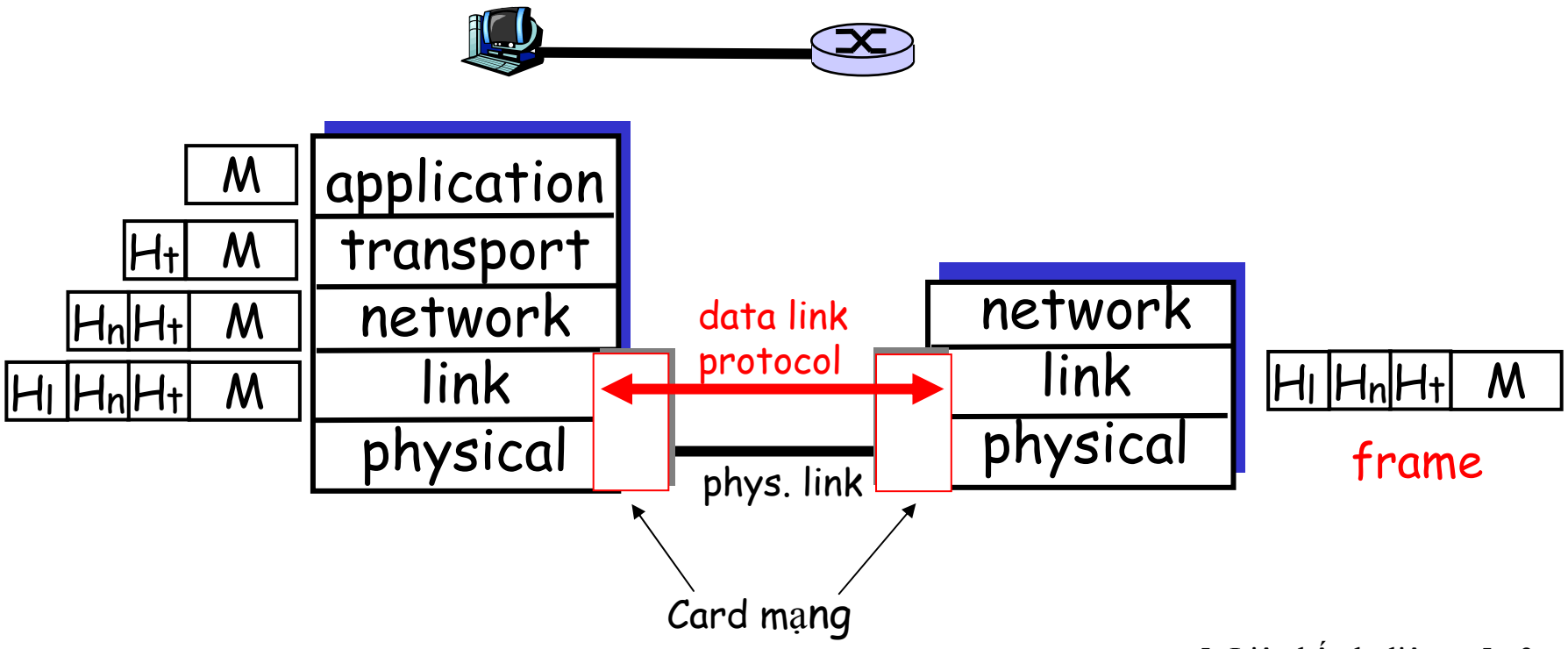
- Các dịch vụ ở tầng Liên kết dữ liệu
- Phát hiện, Sửa lỗi
- Đa truy cập và mạng LAN
- Địa chỉ ở tầng Liên kết dữ liệu và ARP
- Một vài công nghệ Liên kết dữ liệu cụ thể:
  - Ethernet
  - hubs, bridges, switches
  - IEEE 802.11 LANs
  - PPP

Liên kết dữ liệu: Vị trí trong Mô hình



Liên kết dữ liệu: Bối cảnh

- Hai thiết bị có **kết nối về mặt Vật lý**:
  - host-router, router-router, host-host
- Đơn vị trao đổi dữ liệu : ***frame***



Liên kết dữ liệu : Dịch vụ

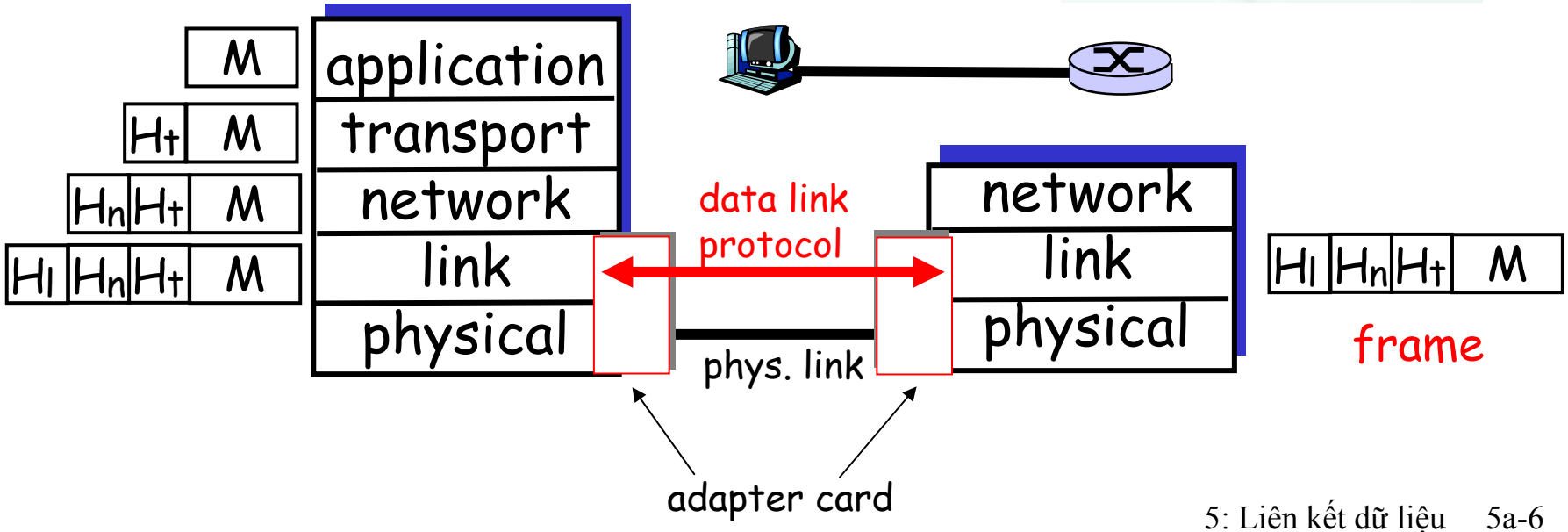
- **Tạo frame, Truy cập môi trường:**
  - Đặt các datagram trong các frame, bổ sung thêm header, trailer
  - Nếu môi trường truyền dùng chung, cài đặt chức năng đa truy cập
  - "địa chỉ Vật lý" trong tiêu đề của frame xác định Địa chỉ Gửi/ Nhận
    - Khác địa chỉ IP !
- **Truyền tin cậy giữa hai thiết bị có kết nối Vật lý trực tiếp:**
  - Nguyên lý đã được giải quyết (chương 3)!
  - Ít khi được dùng trên kênh truyền có tỷ lệ lỗi thấp (cáp quang, một số cáp đồng trục)
  - Đường truyền không dây: Tỷ lệ lỗi cao
    - Vấn đề: Tại sao đặt Tính tin cậy ở cả hai tầng ?

Liên kết dữ liệu : Dịch vụ (tiếp)

- **Điều khiển lưu lượng:**
  - Phù hợp Tốc độ Gửi và Nhận
- **Phát hiện lỗi:**
  - Lỗi do nhiễu.
  - Phía Nhận xác định được có lỗi:
    - Yêu cầu bên Gửi truyền lại hoặc loại bỏ Frame
- **Sửa lỗi:**
  - Phía Nhận xác định *và sửa* được các bit bị lỗi mà không yêu cầu truyền lại

Liên kết dữ liệu: Cài đặt ở đâu

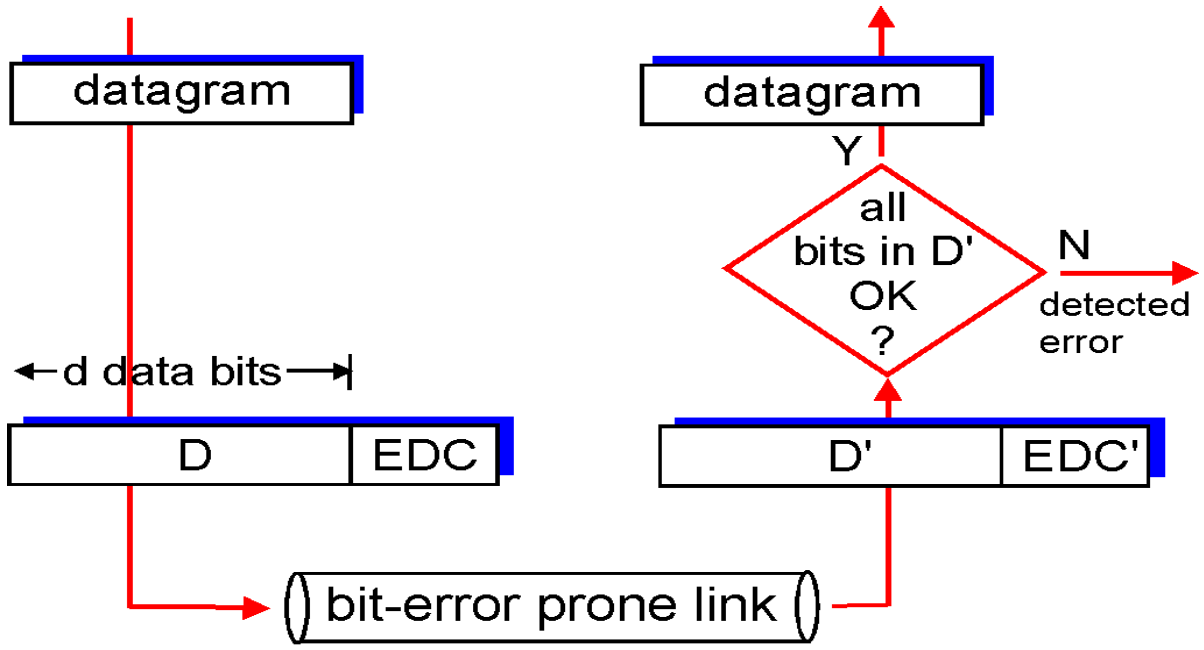
- Cài đặt trên các "adapter"
  - Ví dụ PCMCIA card, Ethernet card
  - Thường có: RAM, DSP chips, host bus interface, và link interface



Phát hiện Lỗi

EDC = Error Detection and Correction bit (Dư thừa)
D = Dữ liệu cần được bảo vệ (có thể thêm phần Tiêu đề)

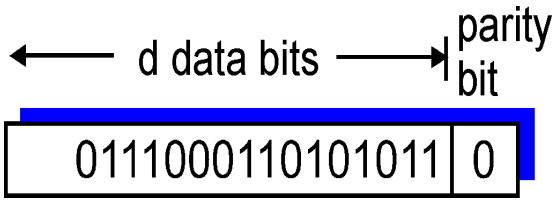
- Phát hiện lỗi Không hoàn toàn đáng tin cậy !
  - Giao thức có thể để "lọt" một số lỗi (hiếm khi)
  - Trường EDC lớn giúp Phát hiện và Sửa lỗi tốt hơn



Kiểm tra Chẵn lẻ

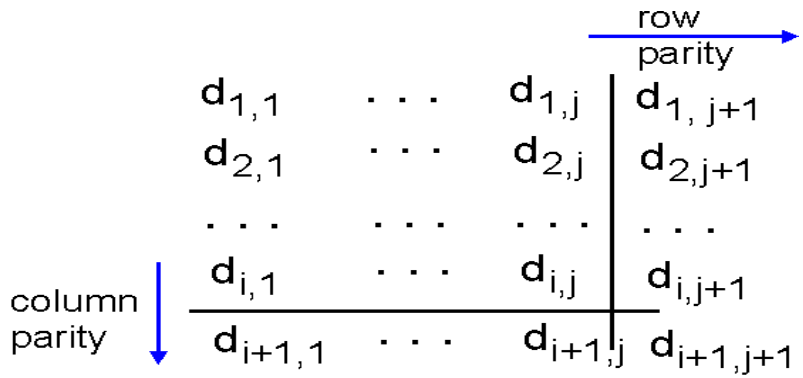
Một Bit Chẵn Lẻ:

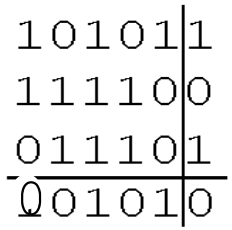
Phát hiện Một lỗi



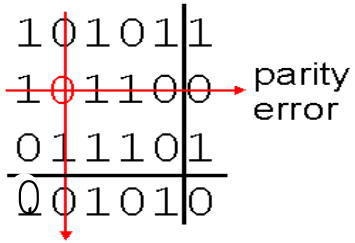
Bit Chẵn Lẻ hai chiều:

Phát hiện và Sửa được một Lỗi





no errors



correctable single bit error

Internet checksum

Mục tiêu: phát hiện "lỗi" (bit bị đổi) trong segment được truyền (chú ý: *chỉ* được sử dụng ở Tầng Giao vận)

Phía Gửi:

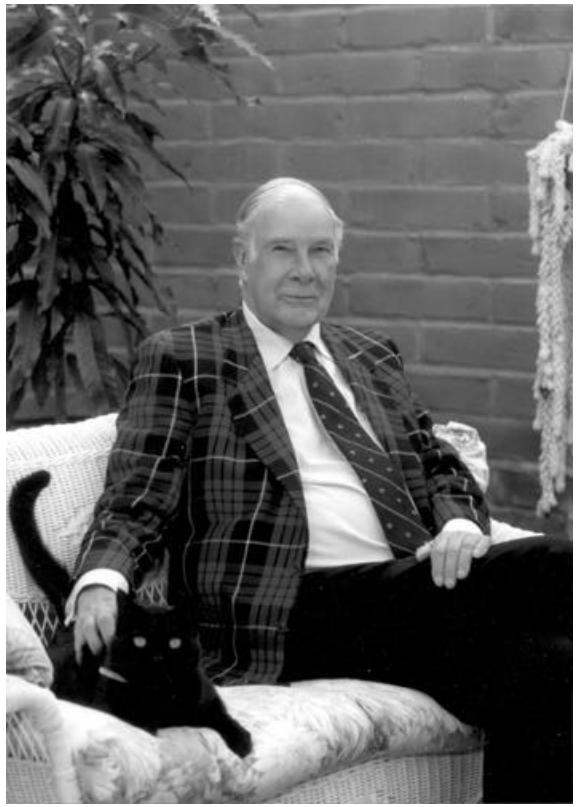
- Xem nội dung segment như các chuỗi số nguyên 16 bit.
- checksum: Tổng bù 1 của tất cả các từ
- Phía Gửi đặt giá trị checksum trong trường checksum của UDP

Phía Nhận:

- Tính checksum của segment nhận được
- Kiểm tra checksum vừa tính được với giá trị Trường checksum:
  - KHÔNG TRÙNG – Phát hiện có Lỗi
  - TRÙNG – Không phát hiện được Lỗi. ***Nhưng vẫn có thể có Lỗi?***

Mã sửa lỗi Hamming

- **Động lực** : Muốn có mã sửa lỗi cần ít dư thừa hơn kiểu mã ma trận chẵn lẻ hai chiều
- Mã Hamming : với log(M) bit dư thừa
  - Sửa tất cả lỗi một bit
  - Phát hiện các lỗi hai bit
- Đặt các bit chẵn lẻ kiểm tra xen kẽ

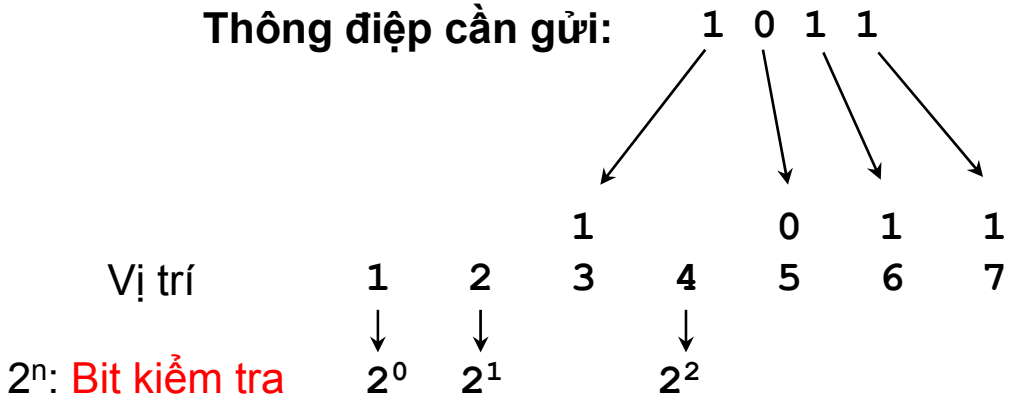


Richard W. Hamming

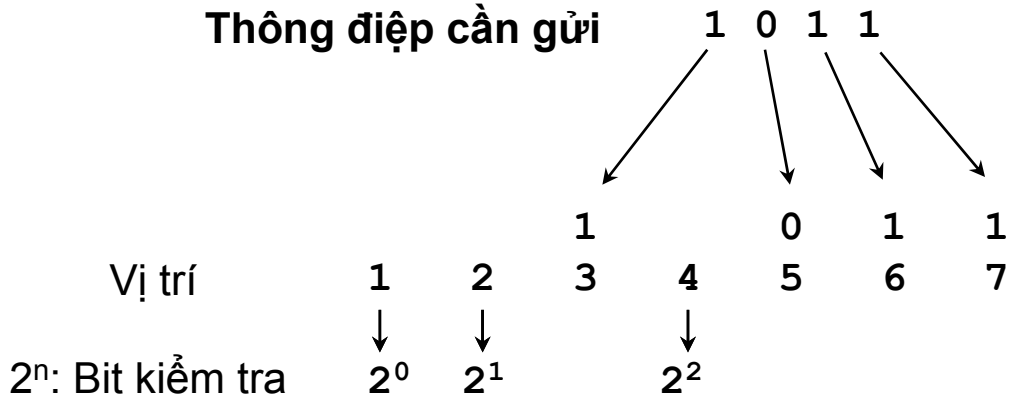
Xác định mã Hamming

- Thủ tục:
  - Đặt các bit dữ liệu thực sự (thông điệp) tại các vị trí Không phải là lũy thừa của hai.
  - Xây dựng bảng liệt kê biểu diễn nhị phân cho mỗi vị trí của bit dữ liệu
  - Tính giá trị các bit kiểm tra

Ví dụ về mã Hamming



Ví dụ về mã Hamming



Tính toán các bit kiểm tra:

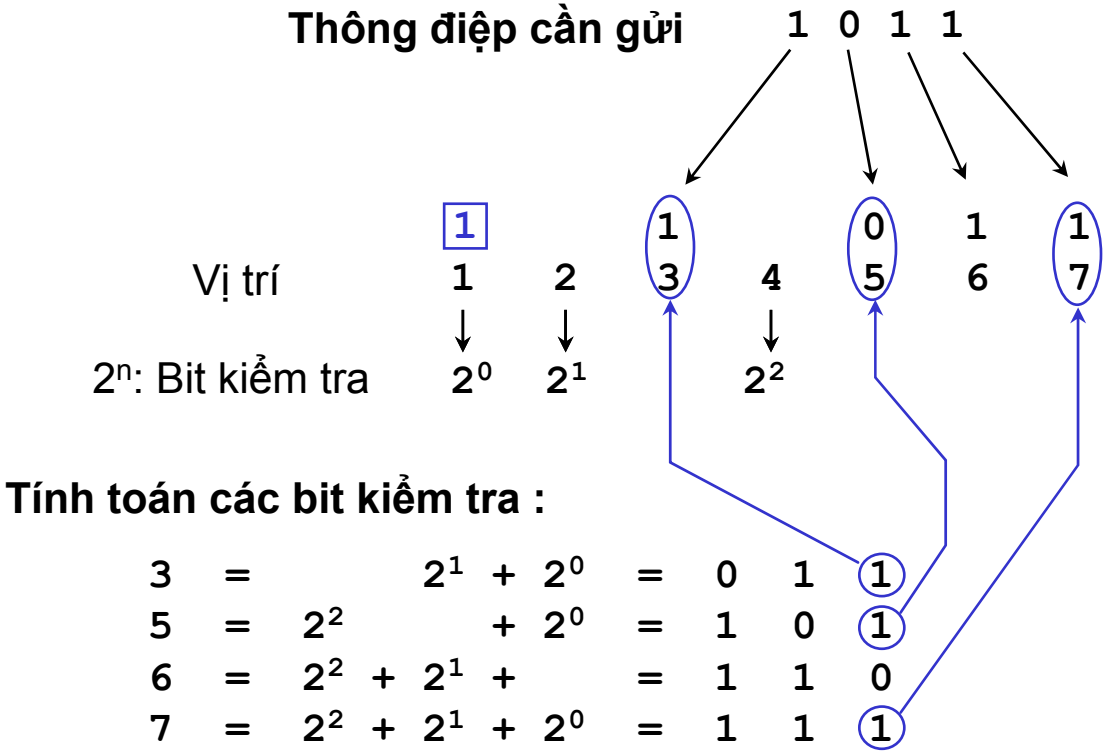
$3 = 2^1 + 2^0 = 0\ 1\ 1$

$5 = 2^2 + 2^0 = 1\ 0\ 1$

$6 = 2^2 + 2^1 = 1\ 1\ 0$

$7 = 2^2 + 2^1 + 2^0 = 1\ 1\ 1$

Ví dụ về mã Hamming



Bắt đầu từ vị trí 2^0 :

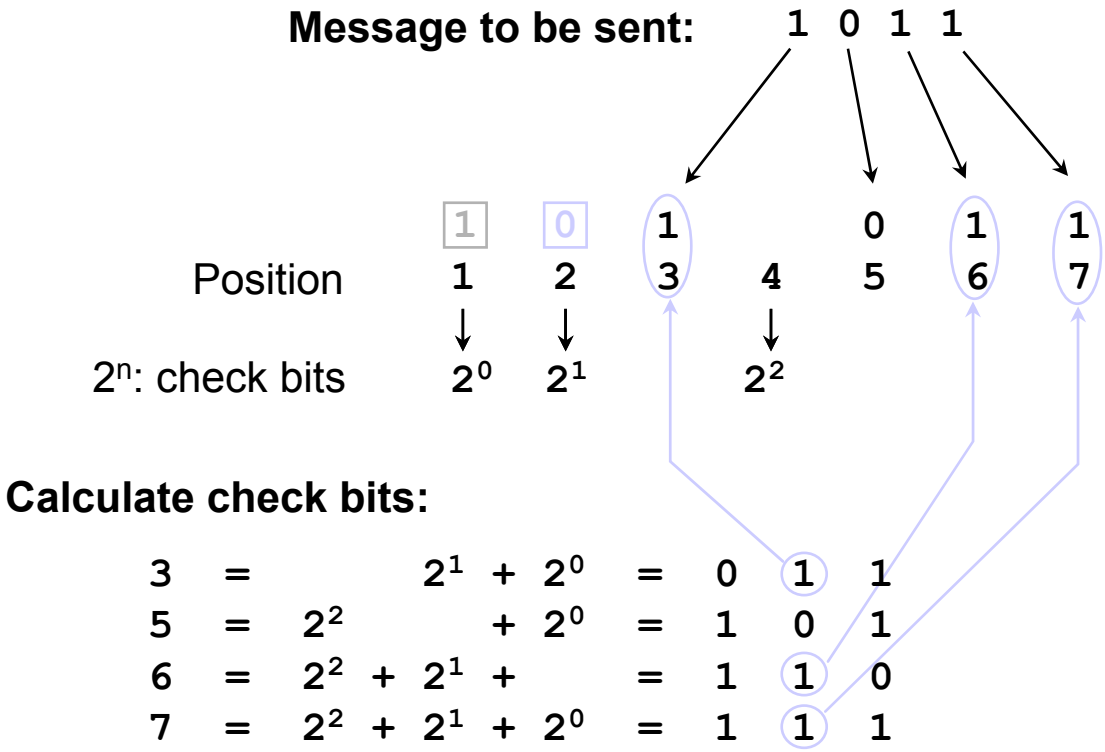
Kiểm tra tất cả các vị trí có giá trị 1 tại vị trí 2^0

Đếm số lượng số 1 trong các bit thông điệp tương ứng

Nếu CHẴN, đặt 1 ở vị trí bit kiểm tra 2^0 (Sử dụng bit chẵn lẻ lẻ)

Nếu LẺ, đặt số 0

Ví dụ về mã Hamming



Calculate check bits:

$3 = 2^1 + 2^0 = 0\ 1\ 1$

$5 = 2^2 + 2^0 = 1\ 0\ 1$

$6 = 2^2 + 2^1 = 1\ 1\ 0$

$7 = 2^2 + 2^1 + 2^0 = 1\ 1\ 1$

Bắt đầu từ vị trí 2^1 :

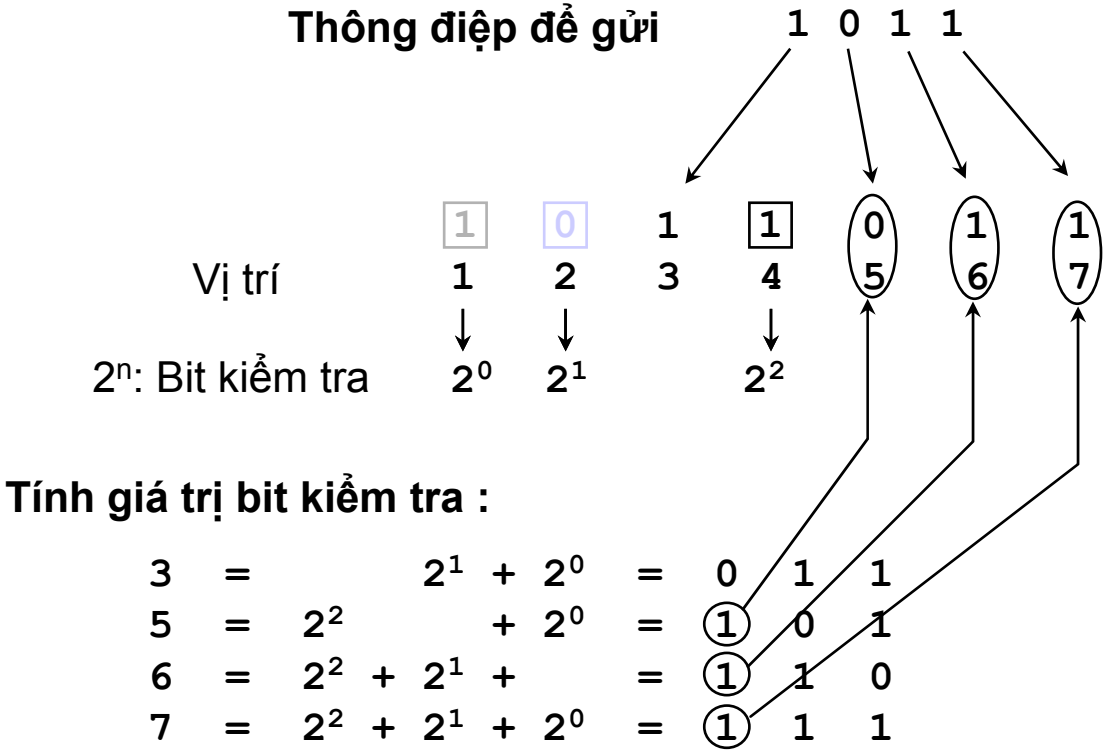
Kiểm tra tất cả các vị trí có giá trị 1 tại vị trí 2^1

Đếm số lượng số 1 trong các bit thông điệp tương ứng

Nếu CHẴN, đặt 1 ở vị trí bit kiểm tra 2^1 (Sử dụng bit chẵn lẻ lẻ)

Nếu LẺ, đặt số 0

Ví dụ về mã Hamming



Bắt đầu từ vị trí 2^2 :

Kiểm tra tất cả các vị trí có giá trị 1 tại vị trí 2^2

Đếm số lượng số 1 trong các bit thông điệp tương ứng

Nếu CHẴN, đặt 1 ở vị trí bit kiểm tra 2^2 (Sử dụng bit chẵn lẻ lẻ)

Nếu LẺ, đặt số 0

Ví dụ về mã Hamming

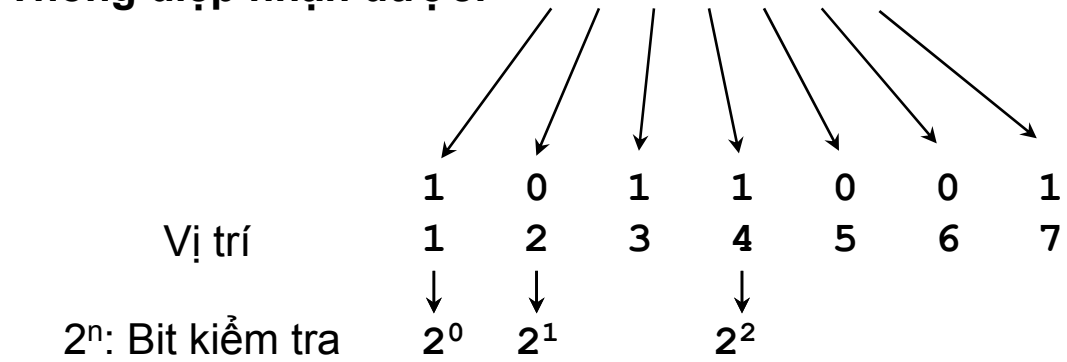
Thông điệp ban đầu = 1011

Thông điệp gửi đi = 1011011

Vậy làm thế nào để có thể xác định vị trí bit bị lỗi trong thông điệp bằng cách sử dụng mã Hamming ?

Sử dụng mã Hamming để sửa lỗi 1 bit

Thông điệp nhận được: 1 0 1 1 0 0 1



Tính các bit kiểm tra:

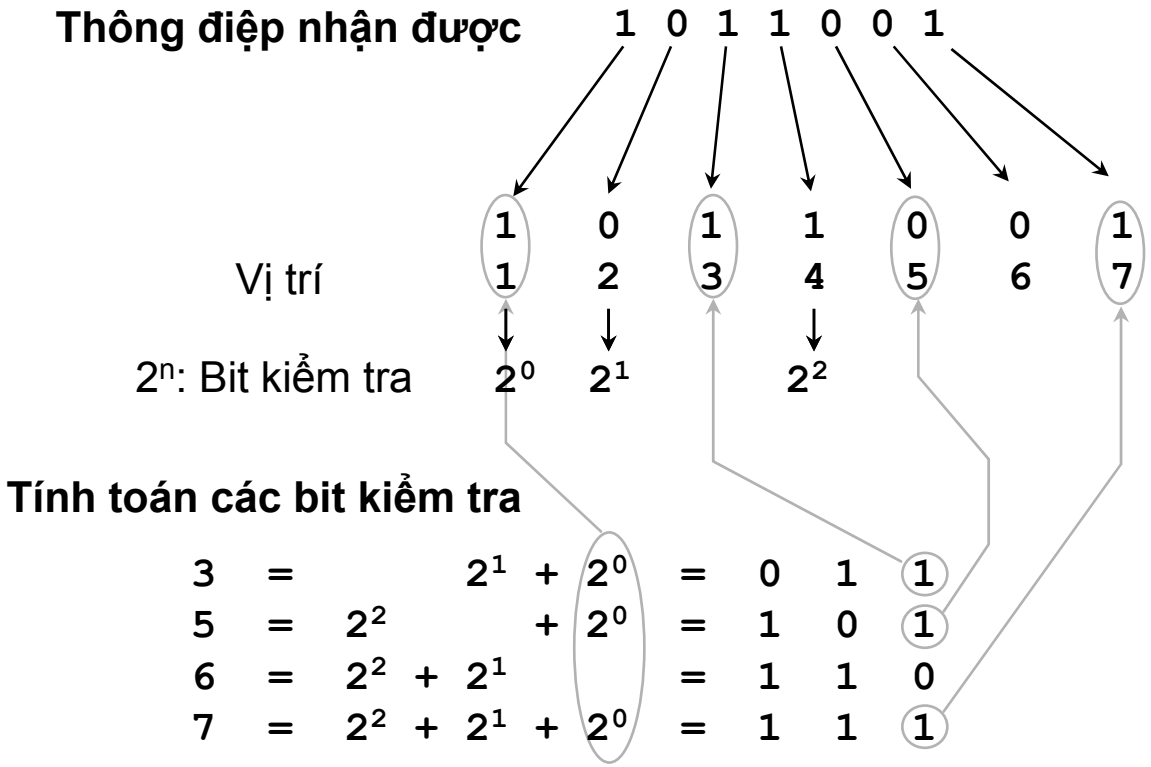
$3 = 2^1 + 2^0 = 0\ 1\ 1$

$5 = 2^2 + 2^0 = 1\ 0\ 1$

$6 = 2^2 + 2^1 = 1\ 1\ 0$

$7 = 2^2 + 2^1 + 2^0 = 1\ 1\ 1$

Sử dụng mã Hamming để Sửa lỗi 1 bit



Chẵn lẻ lẻ: Không có lỗi ở các bit 1, 3, 5, 7

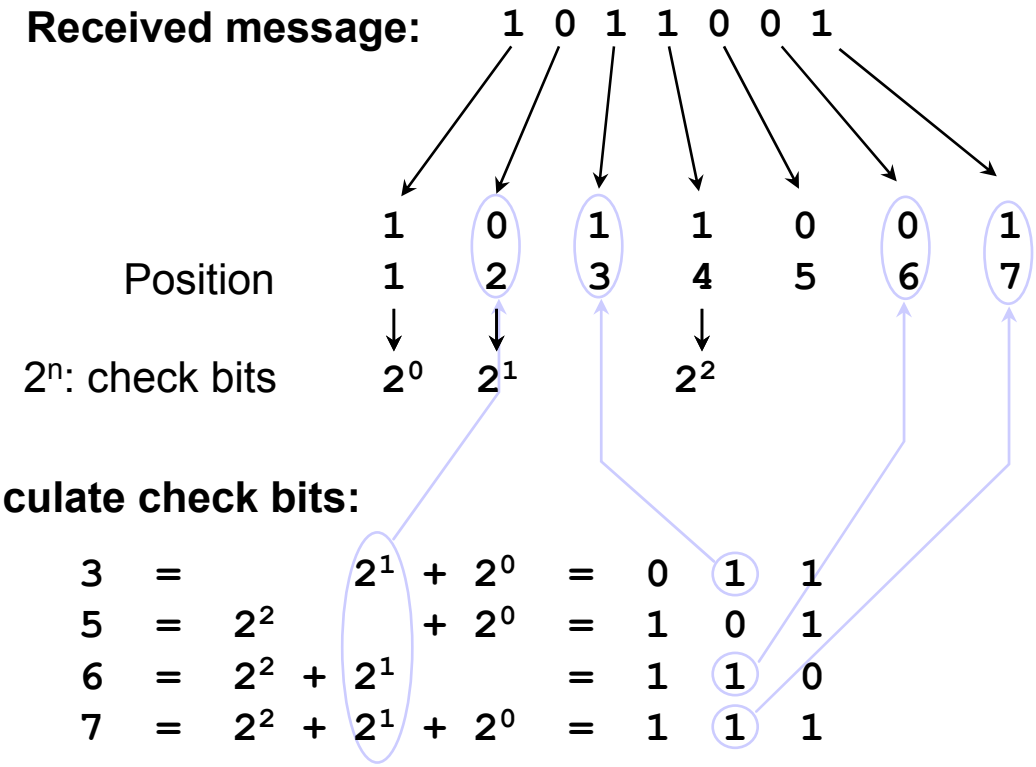
Bắt đầu từ vị trí 2^0:

Kiểm tra tất cả các vị trí có giá trị 1 tại 2^0

Đếm số số 1 trong tất cả các bit tương ứng của thông điệp và vị trí 2^0 và Giá trị chẵn lẻ của số đếm này.

Nếu số đếm này là một số chẵn, chắc chắn có một trong bốn bit được kiểm tra bị lỗi.

Sử dụng mã Hamming để Sửa lỗi 1 bit



Calculate check bits:

3	=		2^1	+	2^0	=	0	1	1
5	=	2^2		+	2^0	=	1	0	1
6	=	2^2	+ 2^1			=	1	1	0
7	=	2^2	+ 2^1	+	2^0	=	1	1	1

Chẵn lẻ Chẵn: LỖI trong các bit 2, 3, 6 hoặc 7!

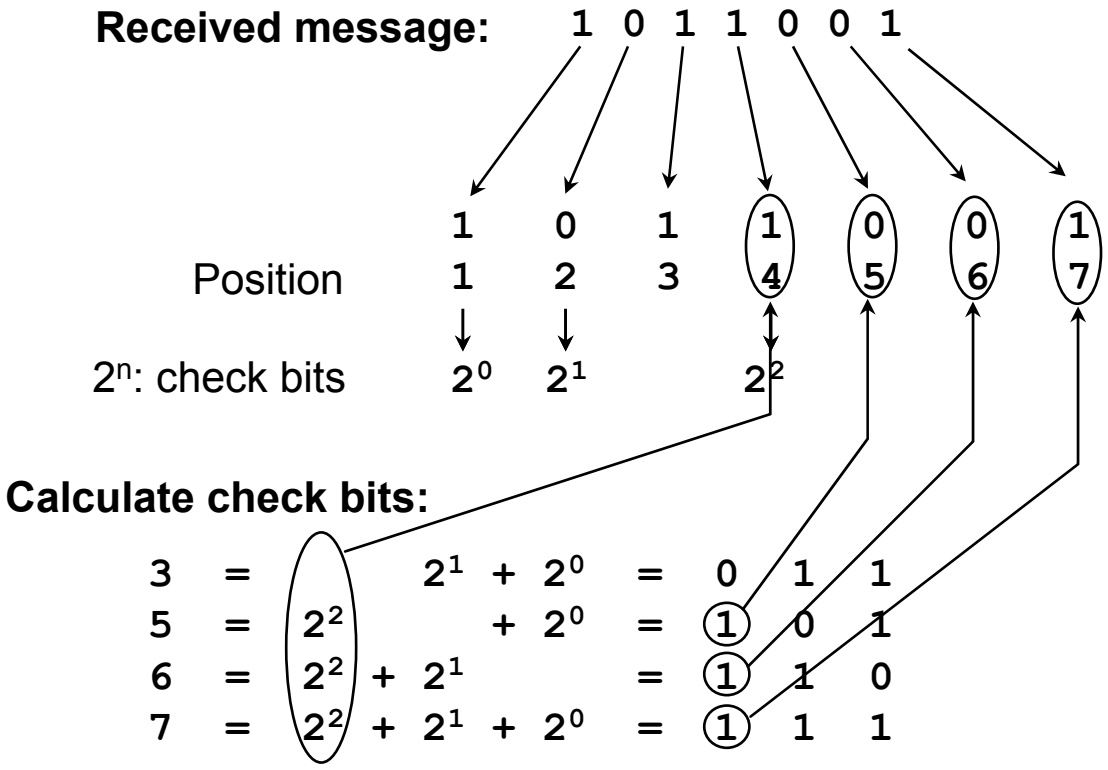
Tiếp tục với vị trí 2^1 :

Kiểm tra tất cả các vị trí có giá trị 1

Đếm số số 1 trong tất cả các bit tương ứng của thông điệp và vị trí 2^1 và Giá trị chẵn lẻ của số đếm này.

Nếu số đếm này là một số chẵn, chắc chắn có một trong bốn bit được kiểm tra bị lỗi.

Sử dụng mã Hamming để Sửa lỗi 1 bit



Chẵn lẻ Chẵn: LỖI trong bit 4, 5, 6 hoặc 7!

Tiếp tục với vị trí 2^2 :

Kiểm tra tất cả các vị trí có giá trị 1

Đếm số số 1 trong tất cả các bit tương ứng của thông điệp và vị trí 2^2 và Giá trị chẵn lẻ của số đếm này.

Nếu số đếm này là một số chẵn, chắc chắn có một trong bốn bit được kiểm tra bị lỗi.

Xác định Vị trí Lỗi

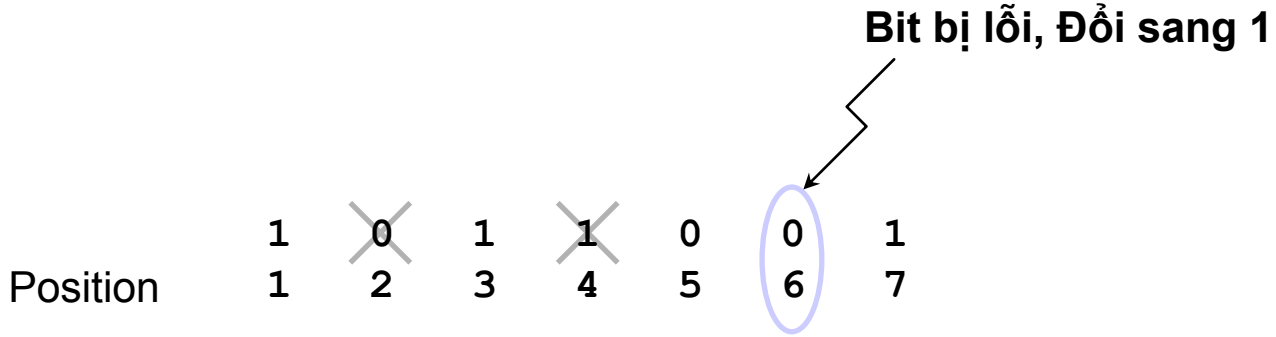
	1	0	1	1	0	0	1
Vị trí	1	2	3	4	5	6	7

Xác định Vị trí Lỗi

	1	0	1	1	0	0	1
Position	1	2	3	4	5	6	7

Không có lỗi tại các vị trí 1, 3, 5, 7

Xác định Vị trí Lỗi



LỖI ở các bit 2, 3, 6 hoặc 7

LỖI ở các bit 4, 5, 6 hoặc 7

Lỗi phải ở trong bit 6 vì các bit 3, 5, 7 đều đúng và tất cả các thông tin còn lại đều khẳng định bit 6 bị lỗi

Xác định Vị trí Lỗi

Giải pháp đơn giản hơn so với slide trước

3	=		2^1	+	2^0	=	0	1	1
5	=	2^2		+	2^0	=	1	0	1
6	=	2^2	+ 2^1			=	1	1	0
7	=	2^2	+ 2^1	+	2^0	=	1	1	1

E E NE

↓ ↓ ↓

1 1 0 = 6

E = Cột bị lỗi

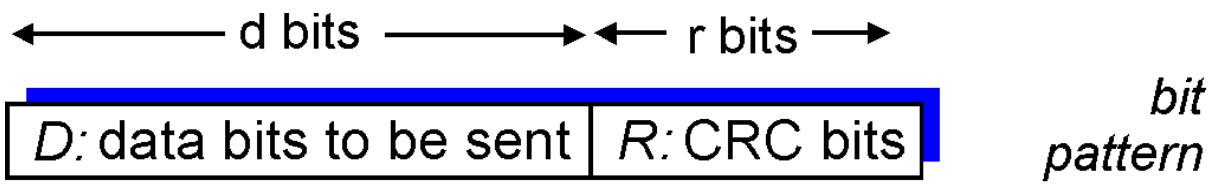
NE = Cột không có lỗi

Mã Hamming – Kết thúc

- Mã Hamming có thể Phát hiện và Sửa được Lỗi một bit
- Nếu có nhiều hơn một bit bị lỗi, mã Hamming không thể sửa được
- Giống bit chẵn lẻ, mã Hamming chỉ có hiệu quả khi thông điệp ngắn

Tổng Kiểm tra: Cyclic Redundancy Check

- Xem các bit dữ liệu (**D**), như một số nhị phân
- Thống nhất một nhóm r+1 bit mẫu **G** (bộ sinh)
- Mục tiêu: Chọn r bit CRC là **R**, sao cho
  - <D,R> chia hết cho G (modulo 2)
  - Phía Nhận cũng xác định được G, chia <D,R> cho G. Nếu dư khác 0 : Phát hiện được Lỗi!
  - Phát hiện được các cụm Lỗi lớn hơn (r+1) bit
- Được sử dụng nhiều trong (ATM, HDCL)



$D * 2^r$ XOR R

mathematical formula

Ví dụ về CRC

Muốn:

$D \cdot 2^r \text{ XOR } R = nG$

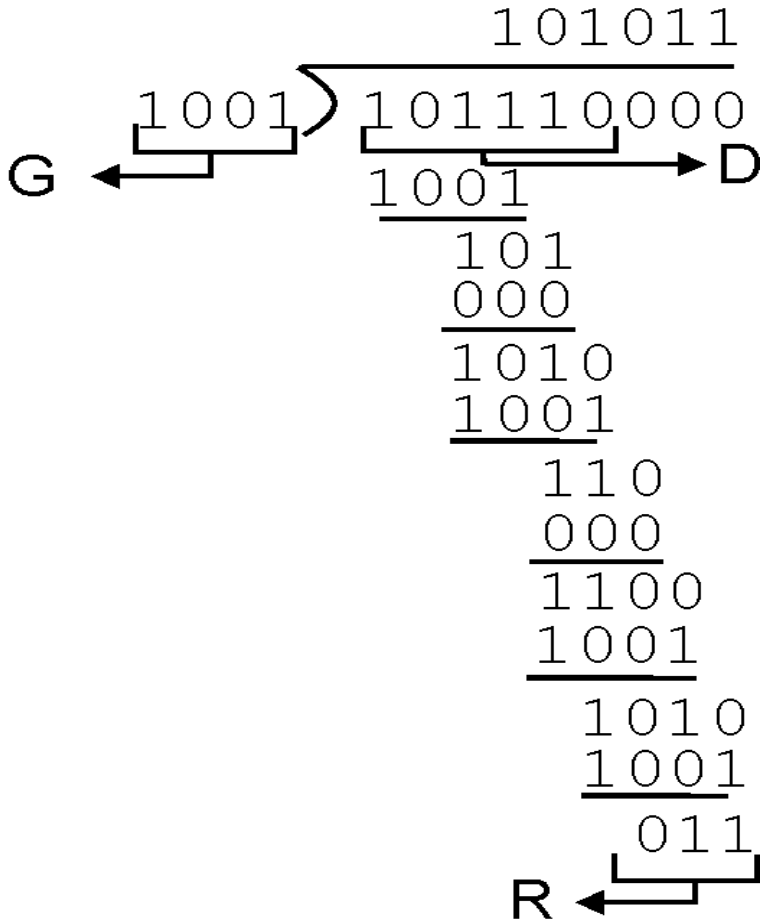
Tương đương với:

$D \cdot 2^r = nG \text{ XOR } R$

Tương đương với:

nếu chia $D \cdot 2^r$ cho G, muốn dư là R

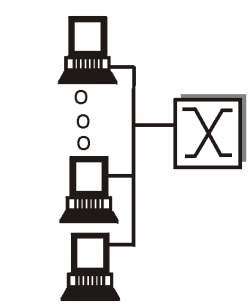
$$R = \text{Dư } \left[\frac{D \cdot 2^r}{G}\right]$$



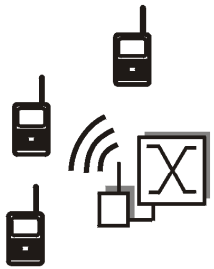
Đa truy cập đường truyền và các Giao thức

Ba kiểu "đường truyền":

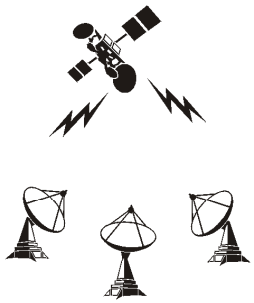
- Điểm nối Điểm (trên một dây truyền, ví dụ PPP, SLIP)
- Quảng bá (dây hay môi trường dùng chung; ví dụ Ethernet, Wavelan,...)



shared wire
(e.g. Ethernet)



shared wireless
(e.g. Wavelan)



satellite

cocktail party

- Chuyển (ví dụ switched Ethernet, ATM)

Giao thức Đa truy cập

- Chia sẻ kênh truyền duy nhất dùng chung
- Nếu có hai cuộc truyền diễn ra đồng thời: xung đột
  - Tại một thời điểm chỉ có một nút có thể truyền Thành công
- *Giao thức Đa truy cập:*
  - Thuật toán phân tán xác định cách thức các trạm chia sẻ kênh truyền, tức là được truyền khi nào.
  - Thông tin về Kênh truyền được lấy từ chính Kênh truyền !
  - Đối với giao thức Đa truy cập:
    - Đồng bộ hay Không đồng bộ
    - Thông tin cần thiết về các trạm khác
    - Khả năng (ví dụ mức độ có lỗi của kênh truyền)
    - Hiệu quả sử dụng

Các Giao thức Đa truy cập

- Chú ý: Loài người thường xuyên sử dụng Giao thức Đa truy cập
- Các bạn có thể đưa ra các giao thức Đa truy cập trong xã hội loài người
  - Giao thức 1:
  - Giao thức 2:
  - Giao thức 3:
  - Giao thức 4:

Đa truy cập : Phân loại

Chia ra ba nhóm chính :

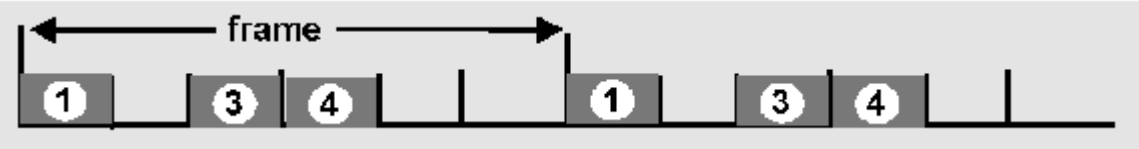
- Phân chia Kênh truyền
  - Chia kênh truyền thành các "phần" nhỏ hơn (theo thời gian, tần số)
  - Mỗi phần được cấp phát riêng cho một nút
- Truy cập Ngẫu nhiên
  - Cho phép xung đột
  - "khắc phục" từ xung đột
- **"Lấy lượt"**
  - Phối hợp chặt chẽ với nhau để đảm bảo không có xung đột

Mục tiêu: Hiệu quả, Công bằng, Đơn giản, Phân tán

Giao thức phân chia kênh truyền: TDMA

TDMA: Time Division Multiple Access

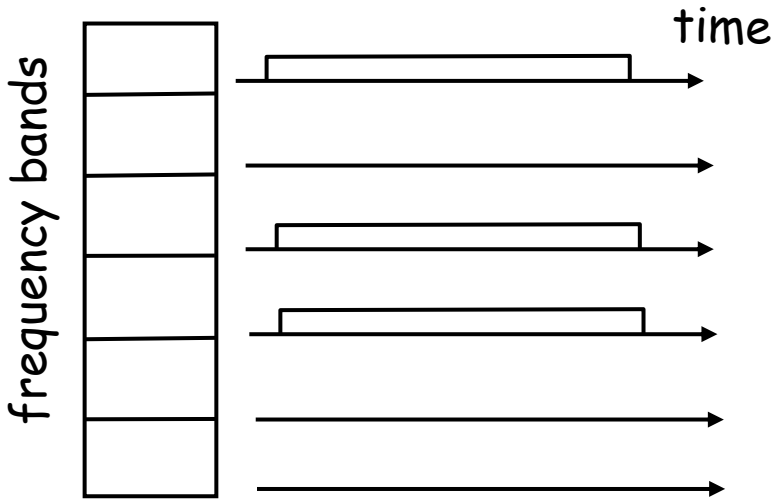
- Truy cập kênh truyền theo từng “vòng"
- Trong mỗi vòng, mỗi trạm sẽ được cấp phát một slot có kích thước cố định (đủ để truyền đi một gói tin)
- Các slot rỗi không được sử dụng
- Ví dụ: 6 trạm LAN là 1,3,4 có dữ liệu, slots 2,5,6 rỗi



Giao thức phân chia kênh truyền: FDMA

FDMA: Frequency Division Multiple Access

- Phổ của kênh truyền được chia thành các dải tần số
- Mỗi trạm được cấp phát một dải băng tần cố định
- Các băng tần rỗi không được sử dụng
- Ví dụ: 6 trạm LAN là 1,3,4 có dữ liệu, slots 2,5,6 rỗi

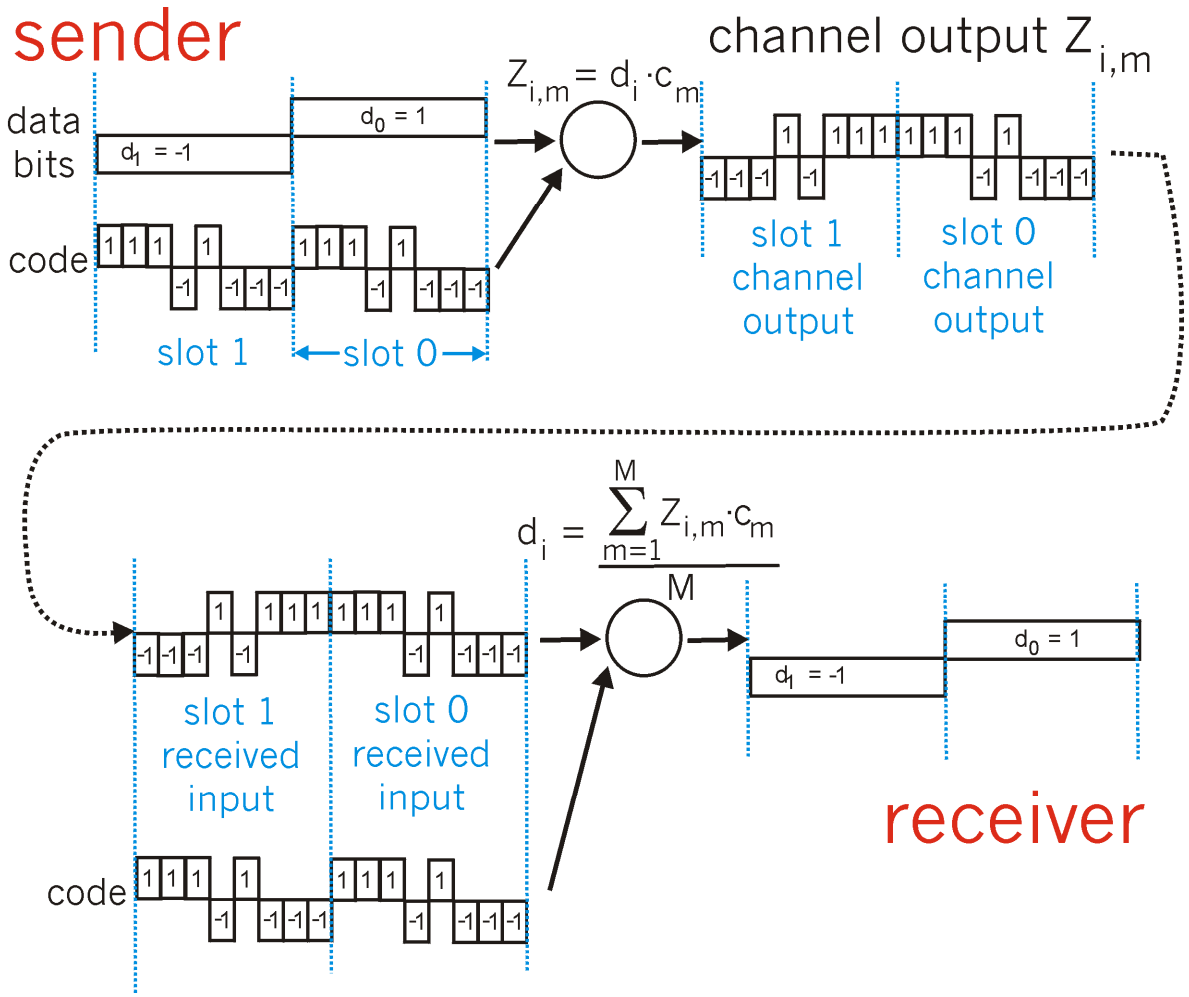


Giao thức phân chia kênh truyền: CDMA

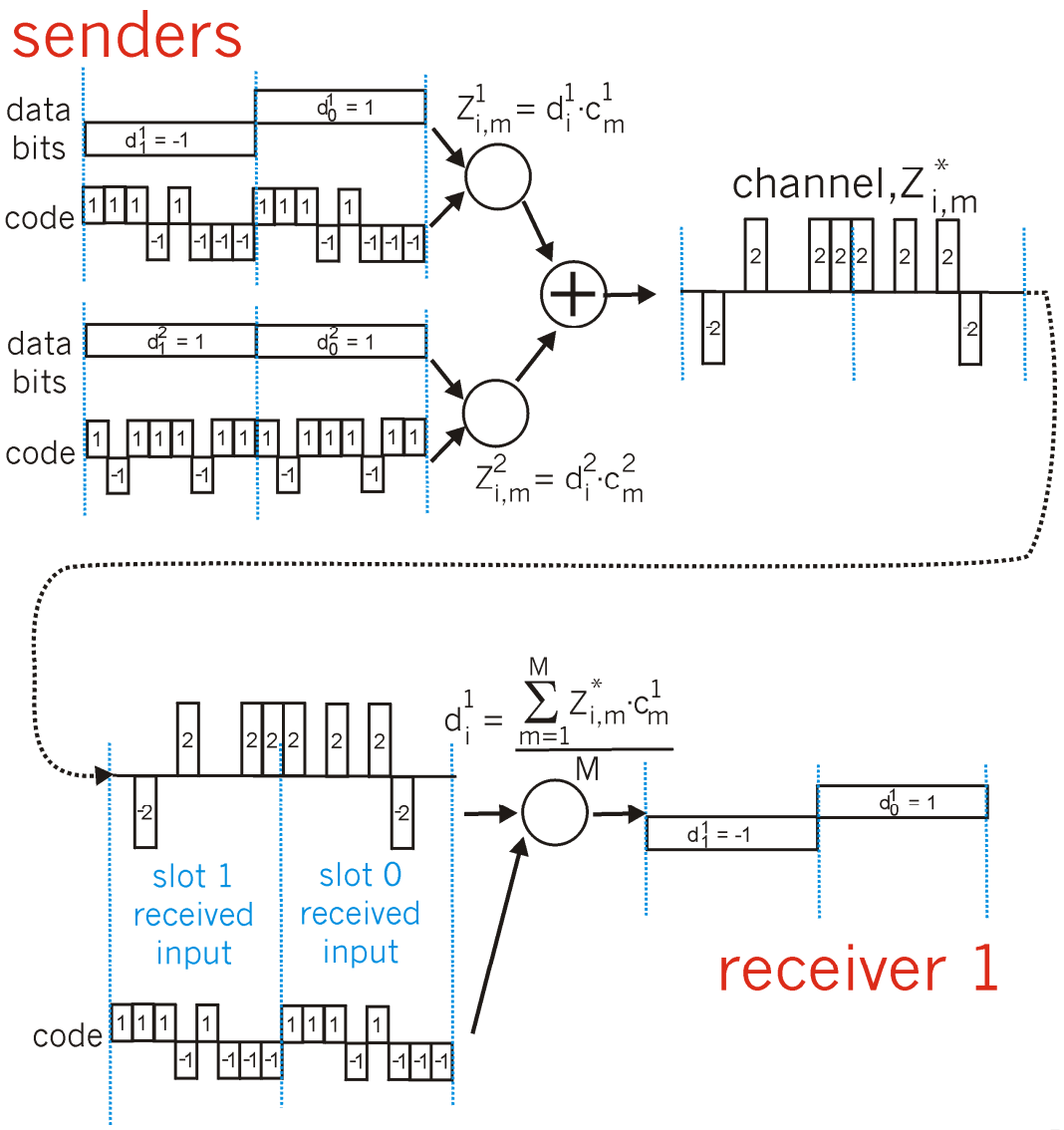
CDMA (Code Division Multiple Access)

- Một "mã" duy nhất được gán cho mỗi người sử dụng (phân chia theo mã)
- Chủ yếu sử dụng trong kênh truyền Quảng bá không dây (cellular, satellite,etc)
- Tất cả người sử dụng dùng chung một tần số, nhưng sử dụng các "mã" riêng để mã hóa dữ liệu
- *Tín hiệu được mã hóa* = (Dữ liệu gốc) X (mã)
- *Giải mã:* Tích của Dữ liệu gốc với Mã
- Cho phép nhiều người dùng có thể truyền đồng thời

CDMA : Mã hóa và Giải mã



CDMA: Hai phía Gửi xen kẽ nhau

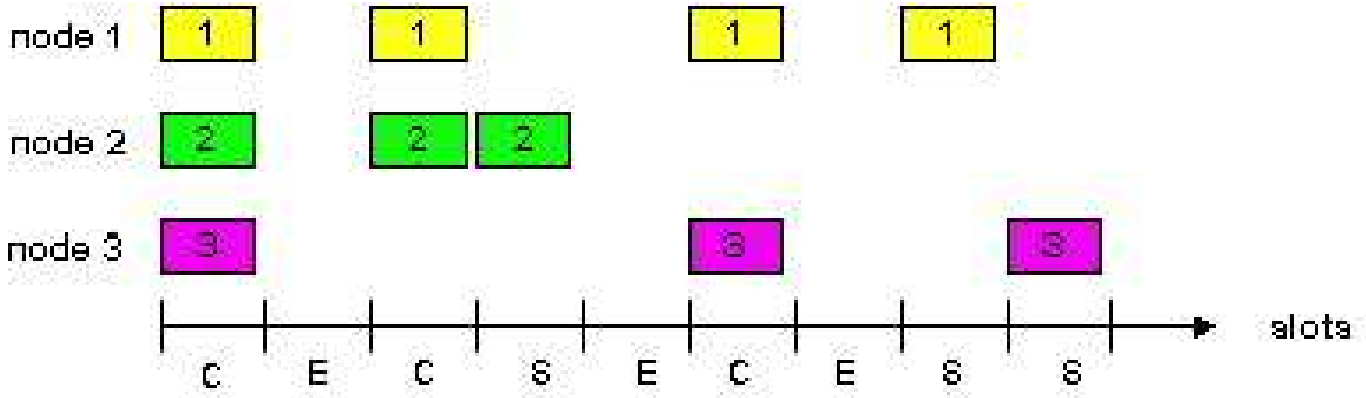


Giao thức Truy cập Ngẫu nhiên

- Khi nút có dữ liệu để truyền đi
  - Truyền với tốc độ tối đa R.
  - Không có sự phối hợp trước với các nút
- Nhiều hơn hai nút cần truyền -> "xung đột",
- Giao thức đa truy cập ngẫu nhiên cần phải:
  - Làm thế nào để xác định có xung đột
  - Khắc phục xung đột như thế nào (ví dụ truyền lại sau một khoảng thời gian)
- Ví dụ các Giao thức Đa truy cập ngẫu nhiên:
  - ALOHA chia khe
  - ALOHA
  - CSMA và CSMA/CD

ALOHA chia khe

- Thời gian được chia thành các slot có kích thước bằng nhau (thời gian đủ để truyền đi một gói tin)
- Nếu có dữ liệu cần gửi: Nút truyền tại đầu slot
- Nếu xung đột: truyền lại gói tin trong slot sau với xác suất truyền là p.



Thành công (S), Xung đột (C), Rỗi (E)

Hiệu suất của ALOHA chia khe

Q: Tỷ lệ truyền thành công của slot là bao nhiêu?

A: Giả sử có N trạm có Dữ liệu cần truyền

- Mỗi trạm truyền tại đầu slot với xác suất *p*
- Xác suất truyền thành công *S* được xác định:

Một nút truyền thành công: $S = p\,(1-p)^{(N-1)}$

Bởi N nút

S = *Xác suất (chỉ một nút truyền)*

$= N\,p\,(1-p)^{(N-1)}$

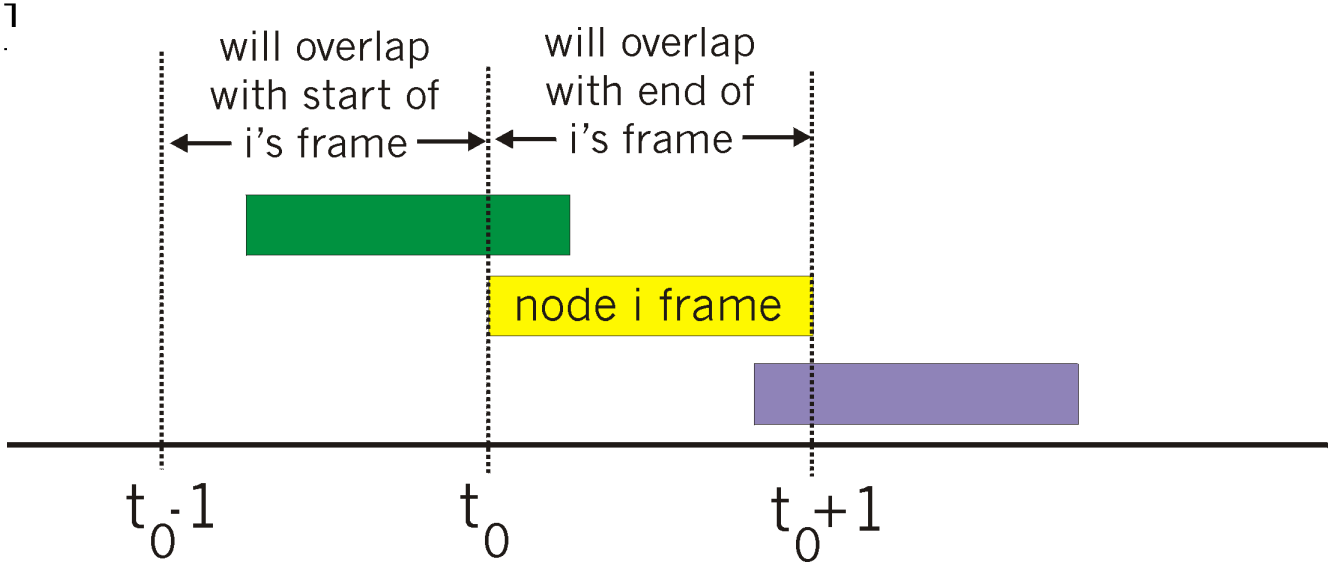
…

$= 1/e = .37$ *khi N tiến ra vô cùng*

Tốt nhất: **Kênh chỉ được sử dụng hữu ích trong 37% Thời gian**

ALOHA thuần túy (Không chia khe)

- Đơn giản hơn, Không cần đồng bộ
- Cũng phải truyền lại gói tin:
  - Gửi không cần phải tại đầu slot
- Xác suất xung đột tăng:
  - Gói tin gửi lúc t0 xung đột với gói tin gửi trong khoảng [t0-1, t0+1]



ALOHA thuần túy (tiếp)

P(một nút truyền thành công) = P(nút truyền) .

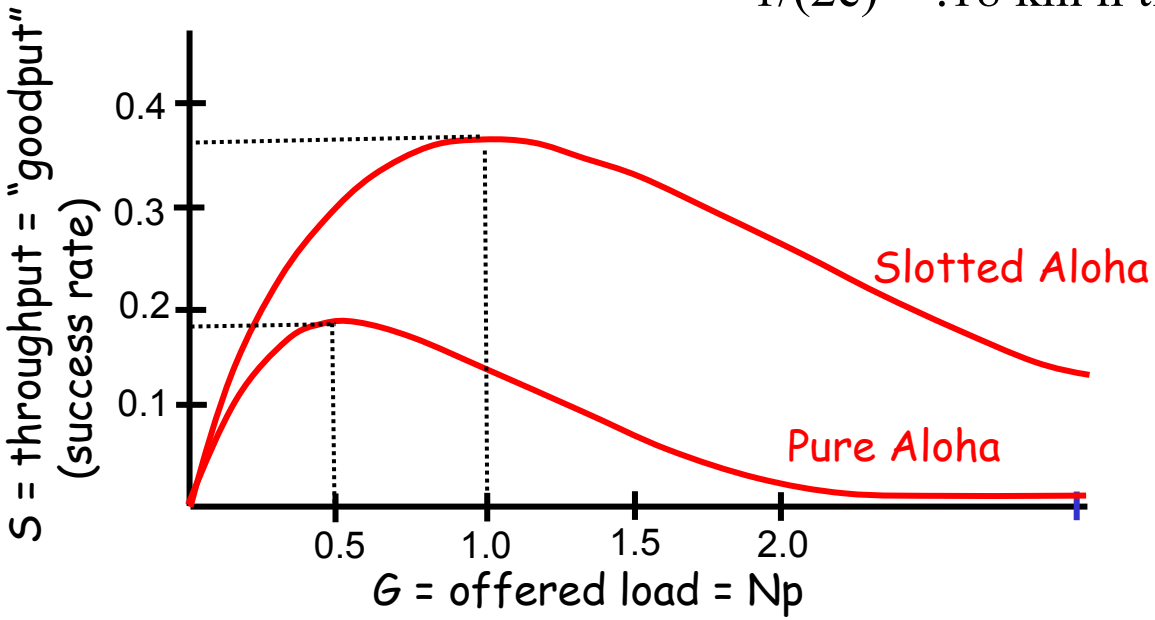
P(không có nút nào truyền trong [p0-1,p0] .

P(không có nút nào truyền trong [p0-1,p0]

= p · (1-p) · (1-p)

P(Bất kỳ nút nào trong N nút truyền thành công) = N p . (1-p) . (1-p)

= 1/(2e) = .18 khi n tiến ra vô cùng



Giao thức hạn chế đáng kể thông lượng Kênh truyền!

CSMA: Carrier Sense Multiple Access

CSMA: Nghe trước khi nói:

- Nếu kênh truyền rỗi : truyền toàn bộ gói tin
- Nếu kênh truyền bận : trì hoãn việc truyền
  - CSMA Kiên trì: thử lại ngay lập tức việc lắng nghe kênh truyền với xác suất p (có thể diễn ra ngay lập tức)
  - CSMA không kiên trì: Thử lại sau một khoảng thời gian ngẫu nhiên
- Xã hội loài người : Không "nhảy vào miệng người khác" !

CSMA : Xung đột

Vẫn có thể có Xung đột :

Độ trễ lan toả nghĩa là hai nút có thể không nghe được cuộc truyền của bên kia

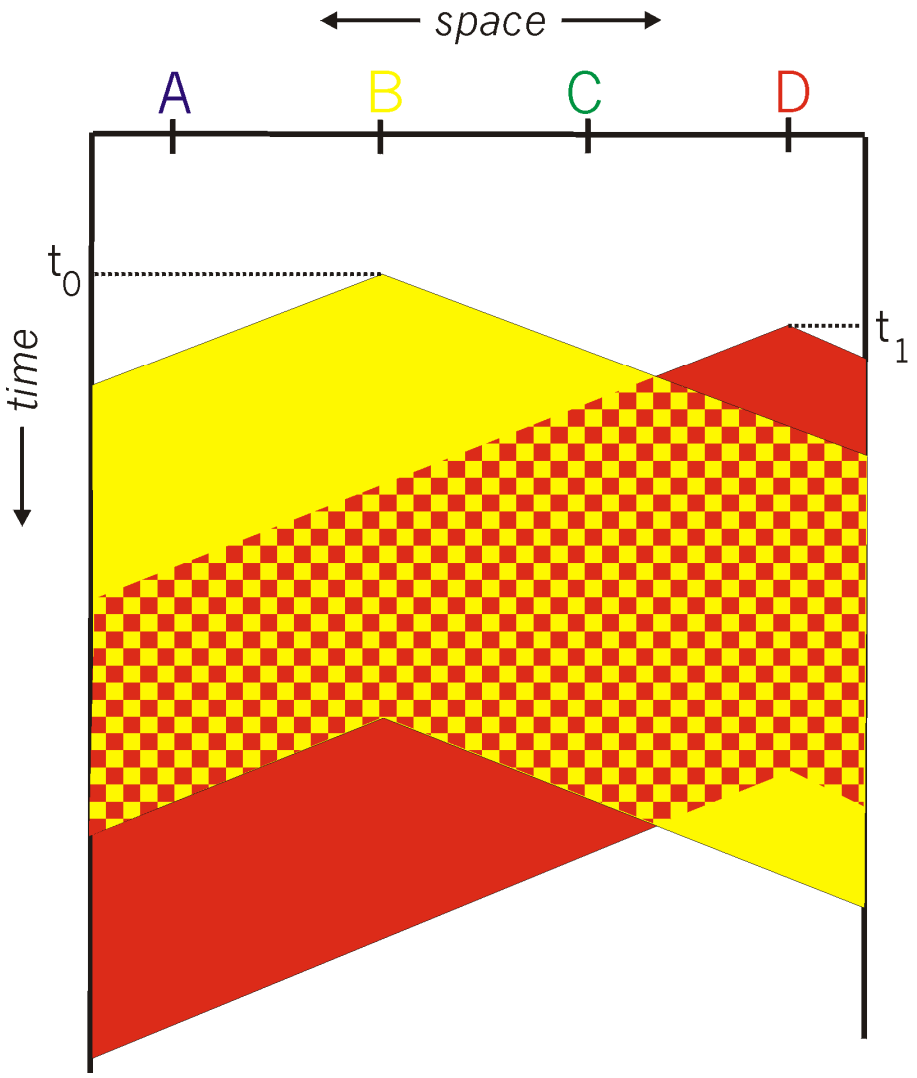
Xung đột:

Thời gian truyền gói tin bị lãng phí

Chú ý:

Khoảng cách cũng như Vận tốc lan tỏa ảnh hưởng lớn đến xác suất xảy ra xung đột.

Cần tính Khoảng cách giữa các nút



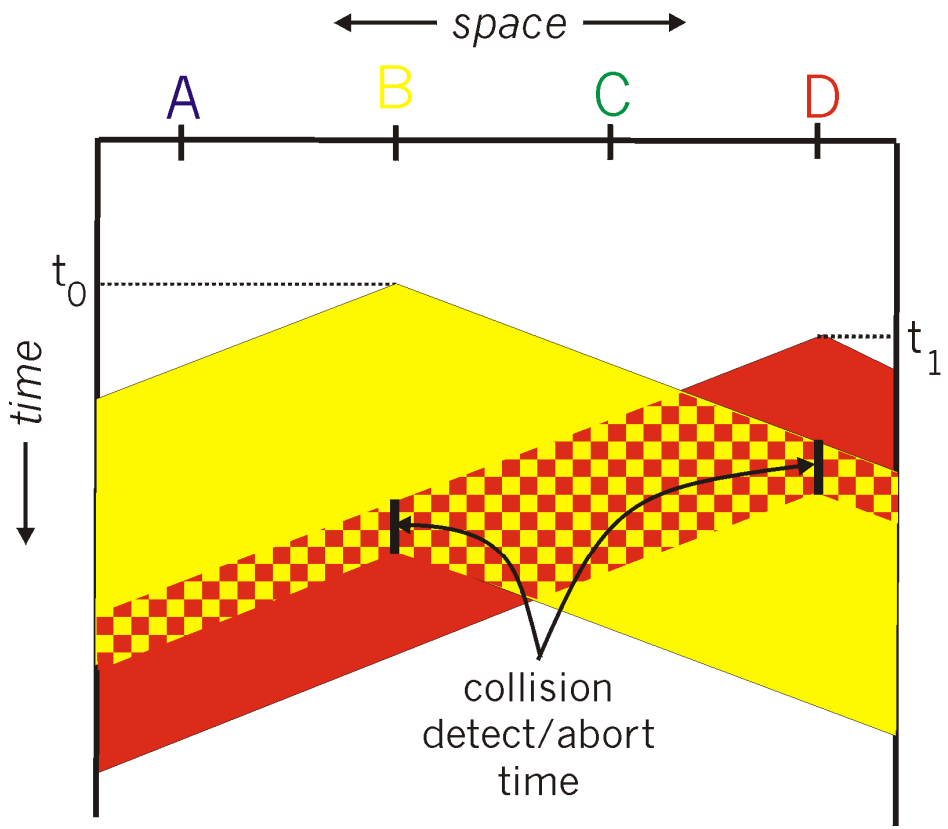
CSMA/CD (Collision Detection)

CSMA/CD: cảm nhận sóng mang, giống như trong CSMA

- Xung đột *bị phát hiện* trong thời gian ngắn
- Loại bỏ các cuộc truyền xung đột => Không lãng phí
- Truyền lại : Kiên trì hoặc Không kiên trì

- Phát hiện Xung đột:
  - Dễ trong môi trường Hữu tuyến: đo bước sóng, so sánh tín hiệu truyền đi và tín hiệu nhận được
  - Khó trong môi trường Vô tuyến: Phía nhận tự dưng bị treo
- Con người: Hội thoại một cách Lịch sự

CSMA/CD collision detection



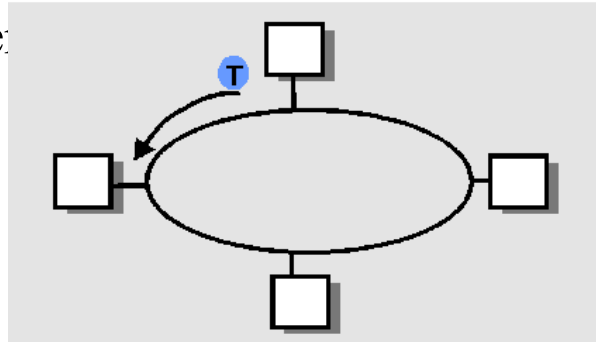
Các Giao thức "Lần lượt"

Hỏi vòng:

- Nút master "mời" các nút slave truyền theo lượt
- Các thông điệp Request to Send, Clear to Send
- Vấn đề:
  - Chi phí phụ trội
  - Độ trễ
  - Sụp đổ khi nút master lỗi

Chuyển Thẻ bài :

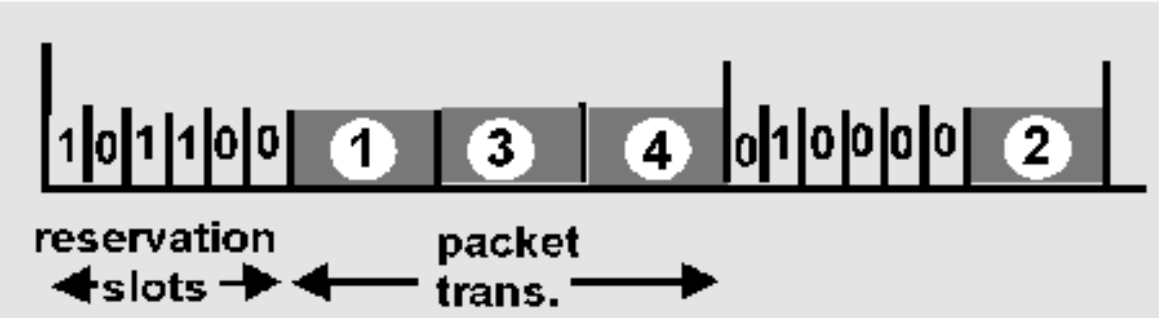
- **Thẻ bài** (thông điệp điều khiển) chuyển từ nút này sang nút kế tiếp.
- Thông điệp Thẻ bài
- Vấn đề:
  - Chi phí phụ trội
  - Độ trễ
  - Sụp đổ khi mất thẻ bài



Các Giao thức dựa trên việc đặt chỗ trước

Thăm dò Phân tán:

- Thời gian được chia thành các slot
- Bắt đầu bằng N slot đặt chỗ
  - Thời gian mỗi slot đặt chỗ bằng độ trễ lan tỏa
  - Trạm cần truyền dữ liệu : đặt chỗ
  - Tất cả các trạm đều quan sát được slot đặt chỗ
- Sau các slot đặt chỗ, các nút truyền theo thứ tự



Giao thức Đếm nhị phân

- Trong thời gian có tranh chấp,
  - Mỗi máy tính gửi Quảng bá địa chỉ cả mình, lần lượt từng bit một, bắt đầu từ bit cao nhất
  - Các bit được truyền đồng thời sẽ được OR với nhau

- Quy tắc Trọng tài tranh chấp:
  - Nếu máy tính gửi đi bit 0 nhưng kết quả phép toán OR là 1 thì sẽ tự động dừng lại và không tiếp tục gửi các bit địa chỉ nữa
  - Nút nào có thể truyền đi tất cả các bit địa chỉ sẽ được quyền truy cập kênh truyền

Giao thức Đếm nhị phân

Địa chỉ Host	Lần lượt từng bit			
	0	1	2	3
0 0 1 0	0	-	-	-
0 1 0 1	0	-	-	-
1 0 0 1	1	0	0	-
1 0 1 0	1	0	1	0

Giao thức Đếm nhị phân : Tính công bằng

- Trạm có địa chỉ lớn nhất luôn "chiến thắng".
- Ưu điểm : Khi muốn cài đặt dựa trên Độ ưu tiên
- Nhược điểm : Khi muốn phân chia kênh truyền công bằng
- Giải pháp:
  - Thay đổi địa chỉ của nút sau mỗi lần truyền thành công
  - Các nút có địa chỉ nhỏ hơn A được cộng thêm 1
  - A nhận địa chỉ 0

Các Giao thức "Lần lượt"

Phân chia Kênh truyền:

- Chỉ Hiệu quả cao khi tải lớn
- Không hiệu quả khi tải thấp: có độ trễ, băng thông chỉ là 1/N kể cả khi chỉ có một nút truyền!

Truy cập Ngẫu nhiên

- Hiệu quả khi tải thấp: Nút duy nhất có thể tận dụng toàn bộ kênh truyền
- Tải cao: Quá nhiều Xung đột

Truy cập Lần lượt

Có vẻ tốt hơn cả hai!

Tổng kết các Giao thức Đa truy cập

- Phải làm gì với Môi trường dùng chung ?
  - Phân chia Kênh truyền (theo thời gian, tần số hay mã)
    - Time Division,Code Division, Frequency Division
  - Phân chia Ngẫu nhiên (động),
    - ALOHA, S-ALOHA, CSMA, CSMA/CD
    - Cảm nhận sóng mang: Hữu tuyến đơn giản, Vô tuyến phức tạp
    - CSMA/CD được sử dụng trong Ethernet
  - Lần lượt
    - Thăm dò từ một trạm trung tâm,
    - Chuyển Thẻ bài

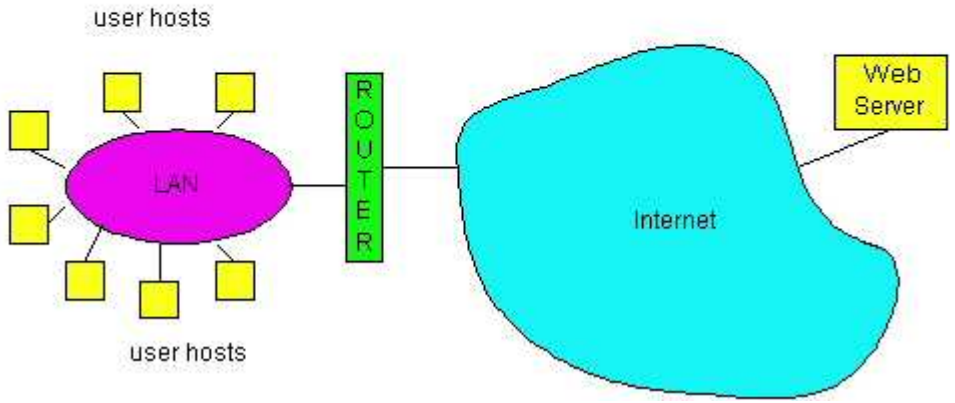
Công Nghệ Mạng cục bộ (LAN)

Về tầng Liên kết Dữ liệu :

- Dịch vụ, Phát hiện/ Sửa Lỗi, Đa truy cập

Tiếp theo: Công nghệ LAN

- Địa chỉ
- Ethernet
- hubs, bridges, switches
- 802.11
- PPP



Địa chỉ LAN và Giao thức ARP

32-bit Địa chỉ IP :

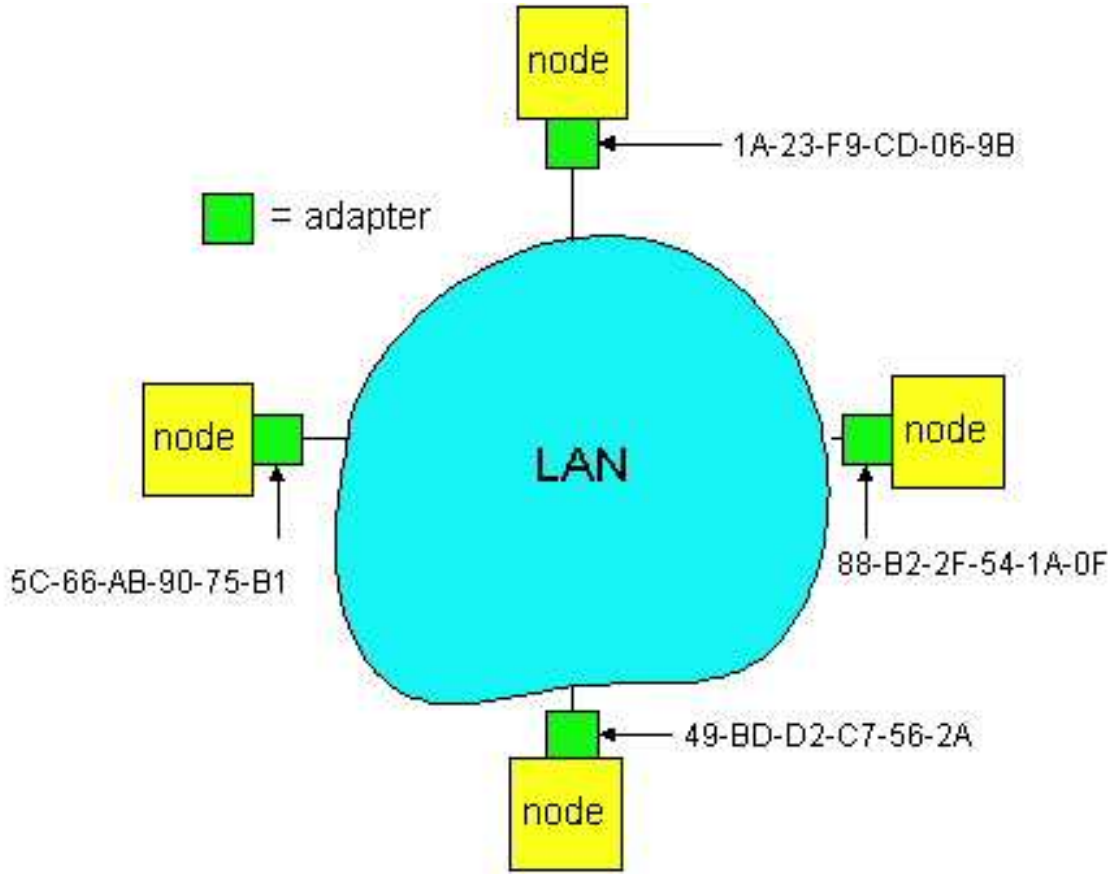
- Địa chỉ tầng Mạng
- Được sử dụng để chuyển datagram tới máy nhận (nhớ lại định nghĩa mạng IP)

Địa chỉ LAN (hay địa chỉ MAC, Vật lý):

- Được sử dụng để chuyển datagram từ interface này sang interface khác (2 interface trên cùng một mạng)
- Địa chỉ MAC 48 bit được ghi trên ROM

Địa chỉ LAN và Giao thức ARP

Mỗi card mạng có một địa chỉ LAN duy nhất



Địa chỉ LAN (tiếp)

- Không gian địa chỉ MAC được IEEE quản trị
- Các nhà sản xuất phải mua một phần không gian địa chỉ (để đảm bảo tính duy nhất)
- Tương tự :

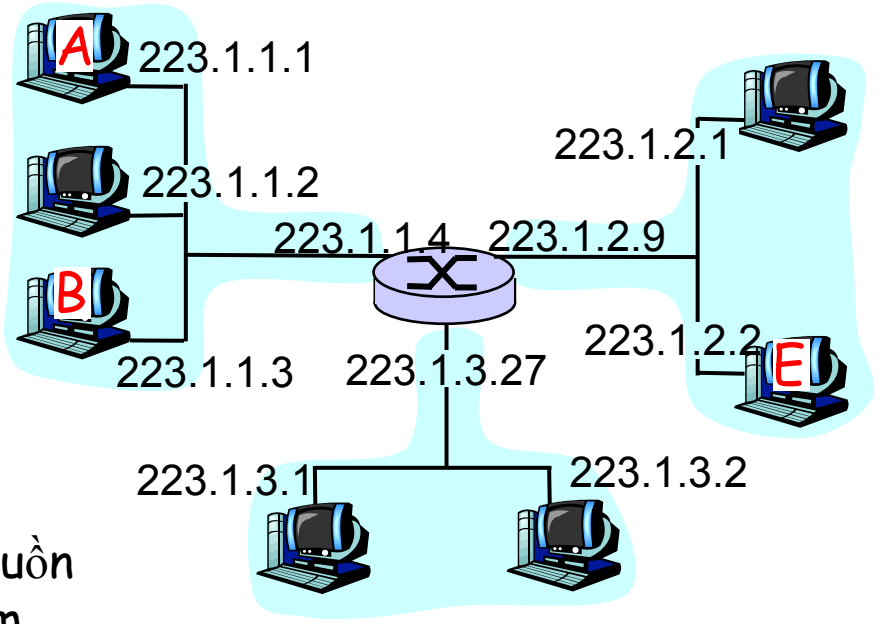
  (a) Địa chỉ MAC : Giống số CMT Nhân dân

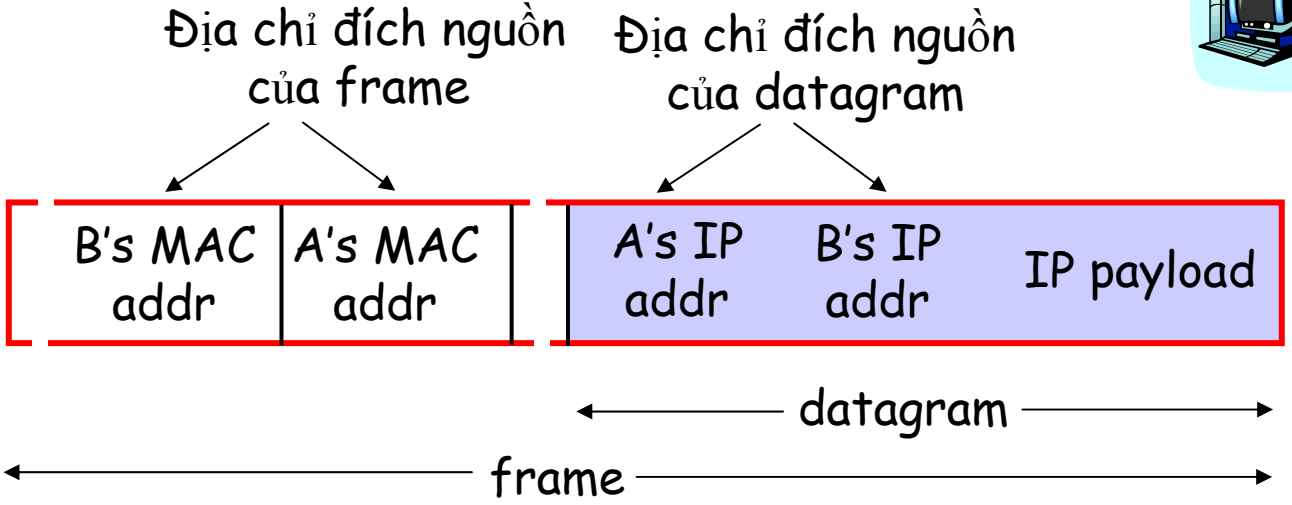
  (b) Địa chỉ IP : Giống Địa chỉ nhà riêng
- Không gian địa chỉ MAC phẳng => khả chuyển
  - Có thể di chuyển card mạng giữa các LAN
- Địa chỉ IP phân cấp => không khả chuyển
  - Phụ thuộc vào mạng IP kết nối tới

Nhớ lại Vấn đề Định tuyến

Khởi đầu từ A, với địa chỉ đích là IP của B:

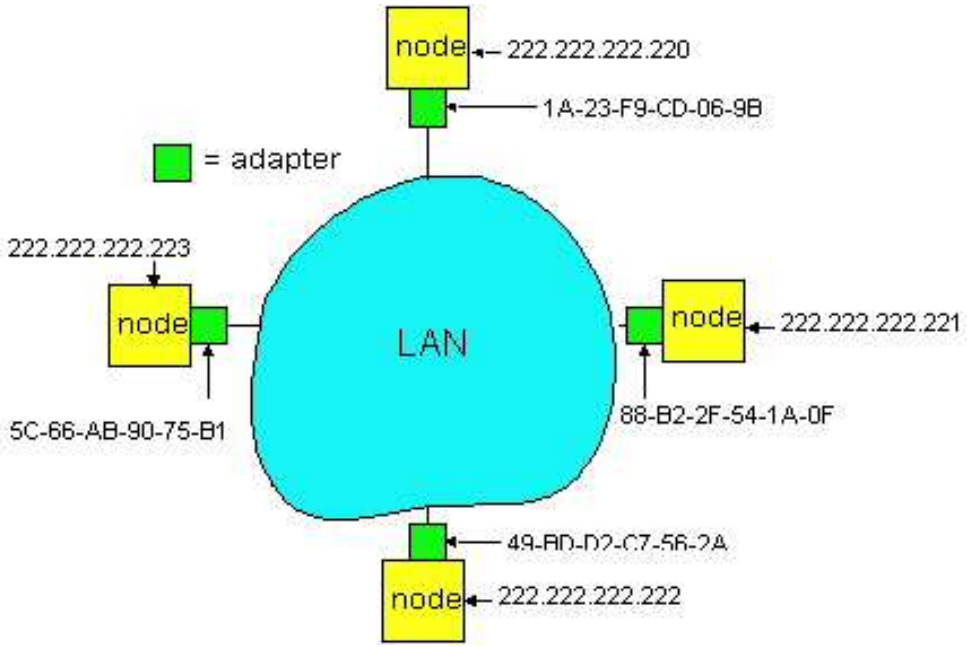
- Kiểm tra địa chỉ của B, thấy rằng B nằm trên cùng một mạng với A
- Tầng liên kết dữ liệu gửi datagram tới B bên trong frame của tầng liên kết dữ liệu





ARP: Giao thức Giải mã Địa chỉ

Vấn đề: Làm sao biết được Địa chỉ MAC của B khi biết Địa chỉ IP của B ?



- Mỗi IP nút (Host, Router) trên mạng LAN có module ARP
- Bảng ARP Table: IP/MAC address mappings for some LAN nodes

< IP address; MAC address; TTL>

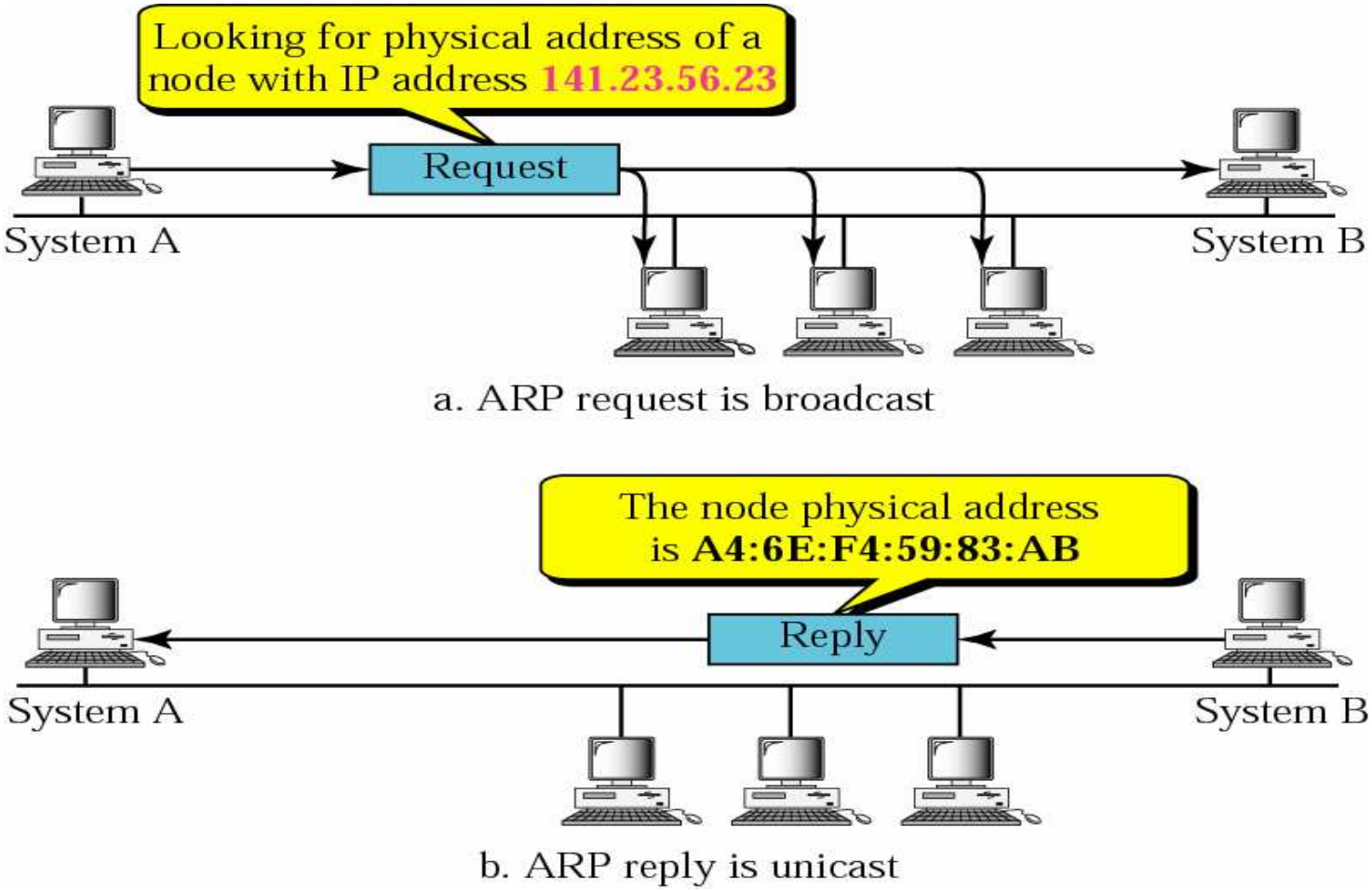
< …….. >

  - TTL (Time To Live): Thời gian để xoá đi một ánh xạ (thường là 20 phút)

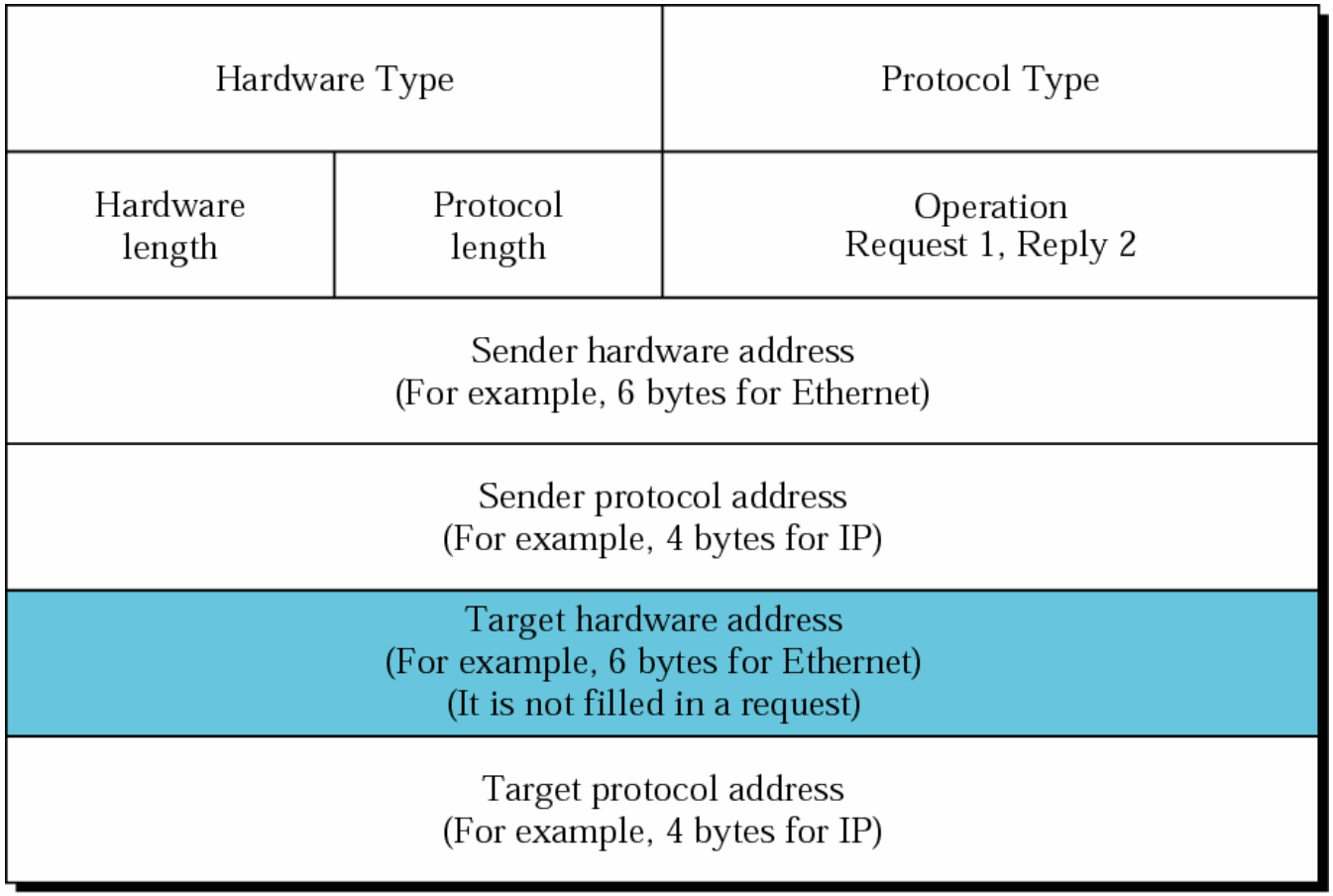
Giao thức ARP

- A biết Địa chỉ IP của B, A muốn biết Địa chỉ Vật lý của B
- A Quảng bá thông điệp truy vấn ARP, chứa địa chỉ IP của B
  - Tất cả máy tính trên mạng LAN nhận được truy vấn này
- B nhận được truy vấn, sẽ trả lời A địa chỉ Vật lý của mình
- A ghi tạm ánh xạ Địa chỉ IP – Địa chỉ Vật lý trong một khoảng thời gian
  - Tại sao ?

Ví dụ ARP


Looking for physical address of a node with IP address 141.23.56.23
Request
System A
System B
a. ARP request is broadcast
The node physical address is A4:6E:F4:59:83:AB
Reply
System A
System B
b. ARP reply is unicast

Khuôn dạng gói tin ARP



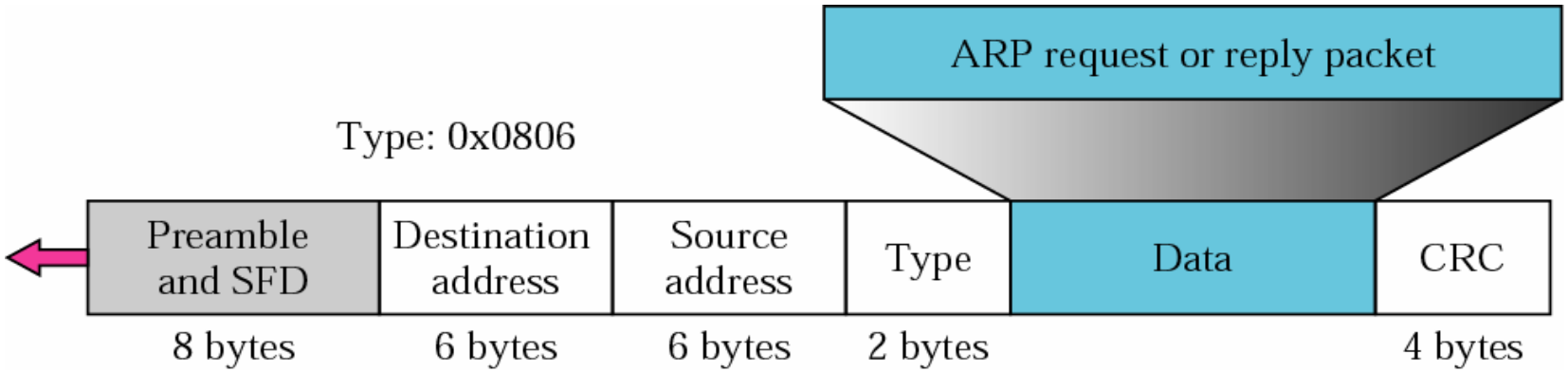
Hardware Type - Ethernet is type 1

Protocol Type- IPv4=x0800

Hardware Length:length of Ethernet Address (6)

Protocol Length:length of IPv4 address (4)

Bao bọc Gói tin ARP

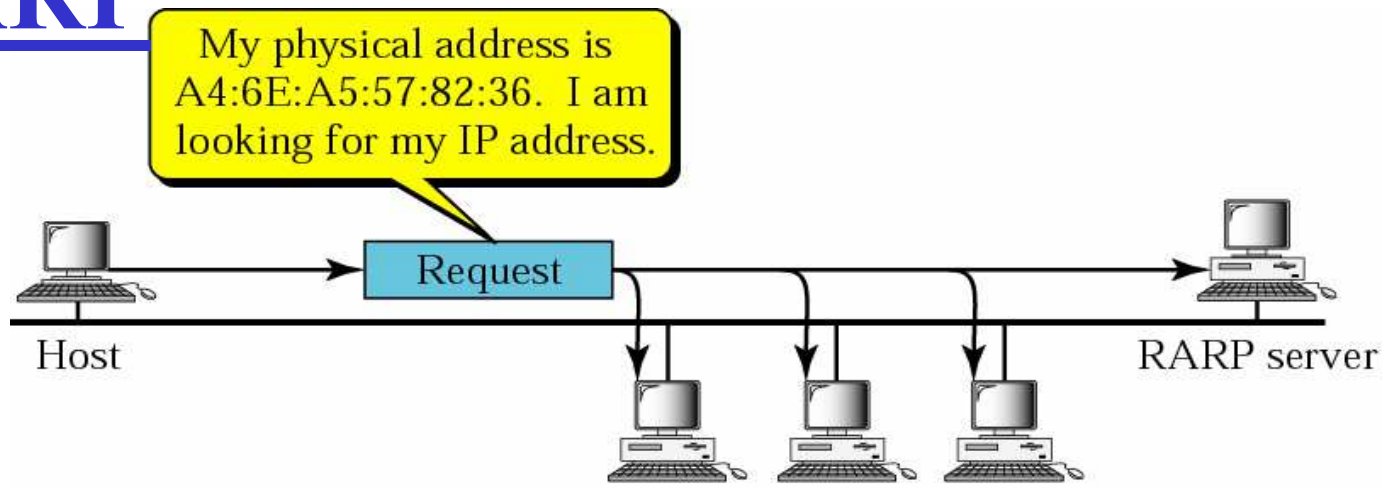


Thông điệp ARP được đặt trong frame Ethernet

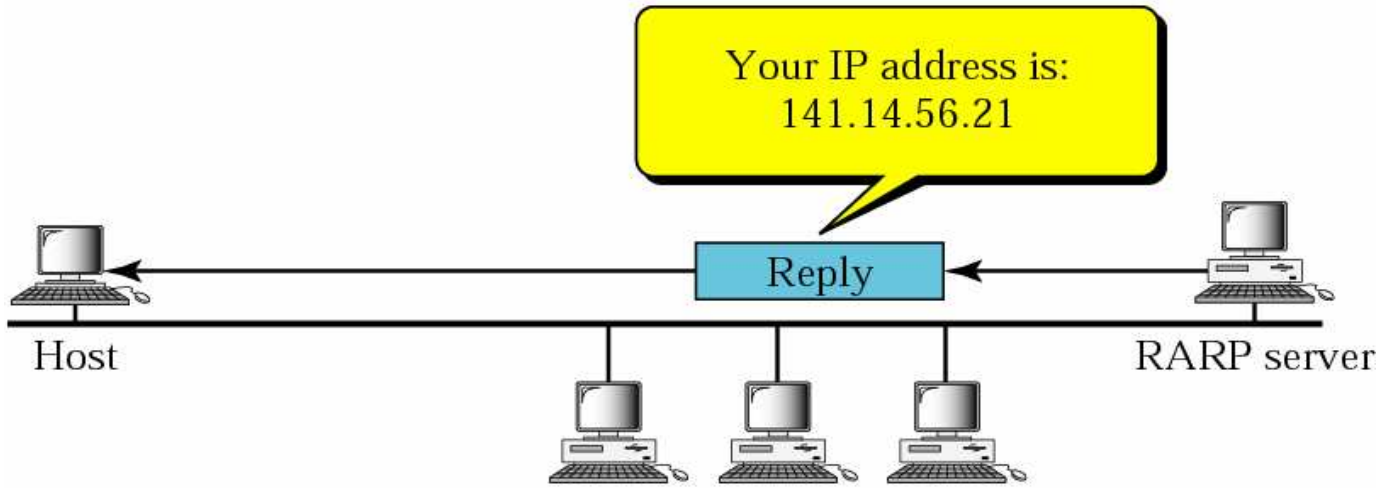
RARP

- B nhận được truy vấn, sẽ trả lời A địa chỉ Vật lý của mình
- RARP được sử dụng để xác định Địa chỉ Logic từ Địa chỉ Vật lý .
- Thường gặp trên các Hệ thống thin-client. Máy tính không có ổ đĩa cứng. Khi khởi động, máy tính cần biết địa chỉ IP (người ta không muốn ghi IP vào ROM)
- Thông điệp yêu cầu RARP được gửi Quảng bá, Thông điệp trả lời RARP được gửi tới một đích.
- Cần phải biết thêm về subnet mask, Địa chỉ router, DNS address, ... : DHCP thay thế RARP.

RARP

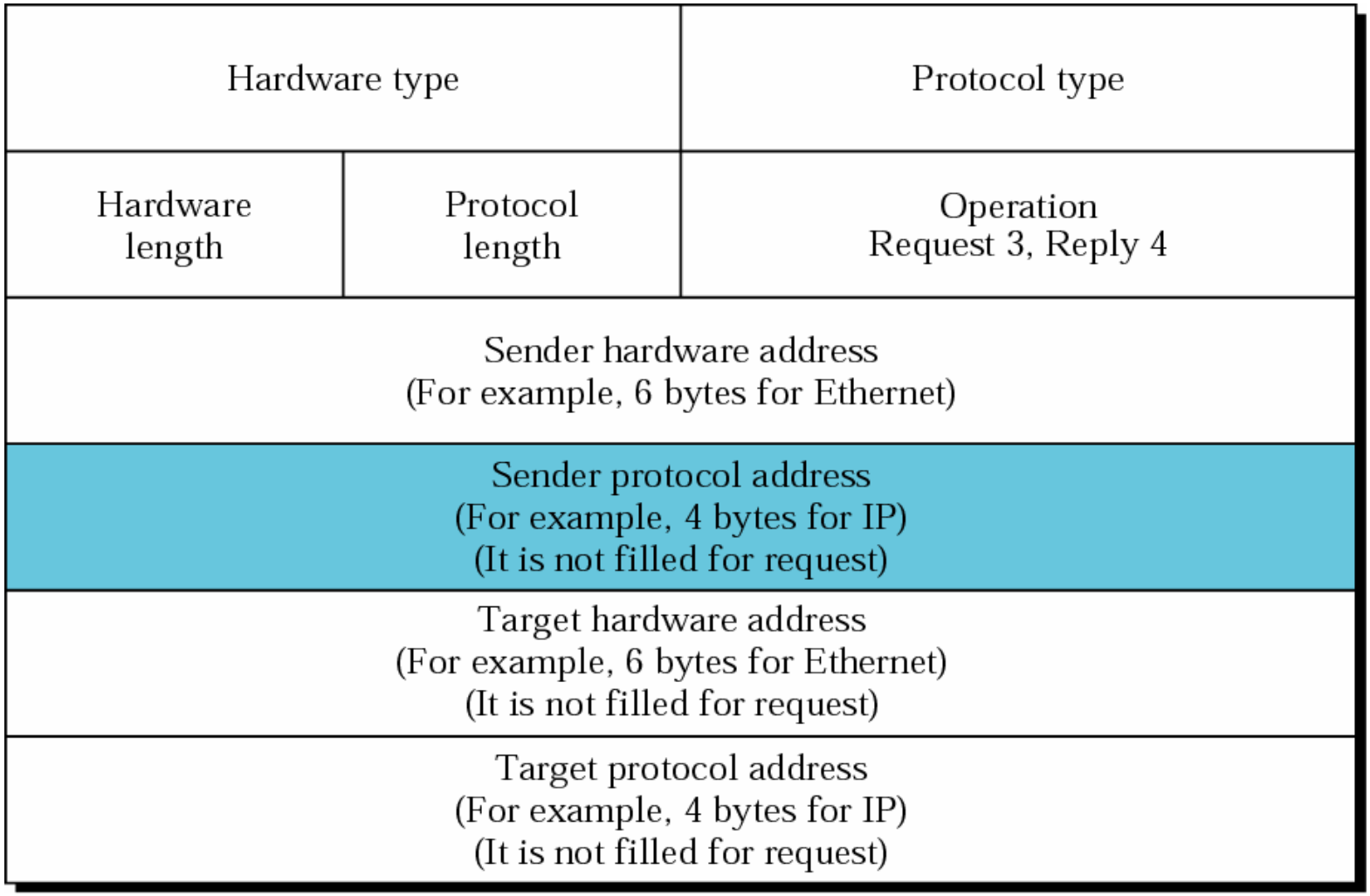


a. RARP request is broadcast

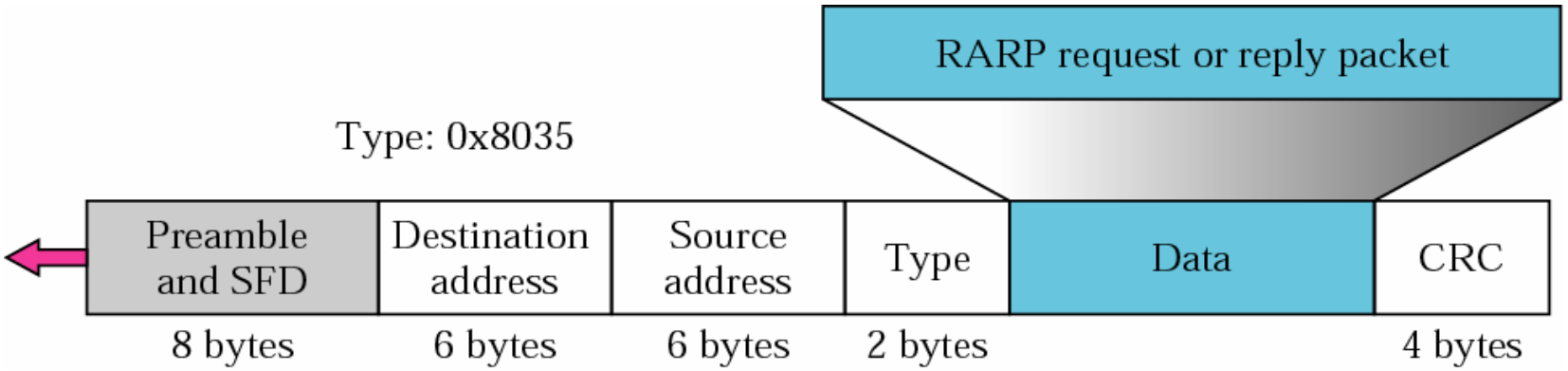


b. RARP reply is unicast

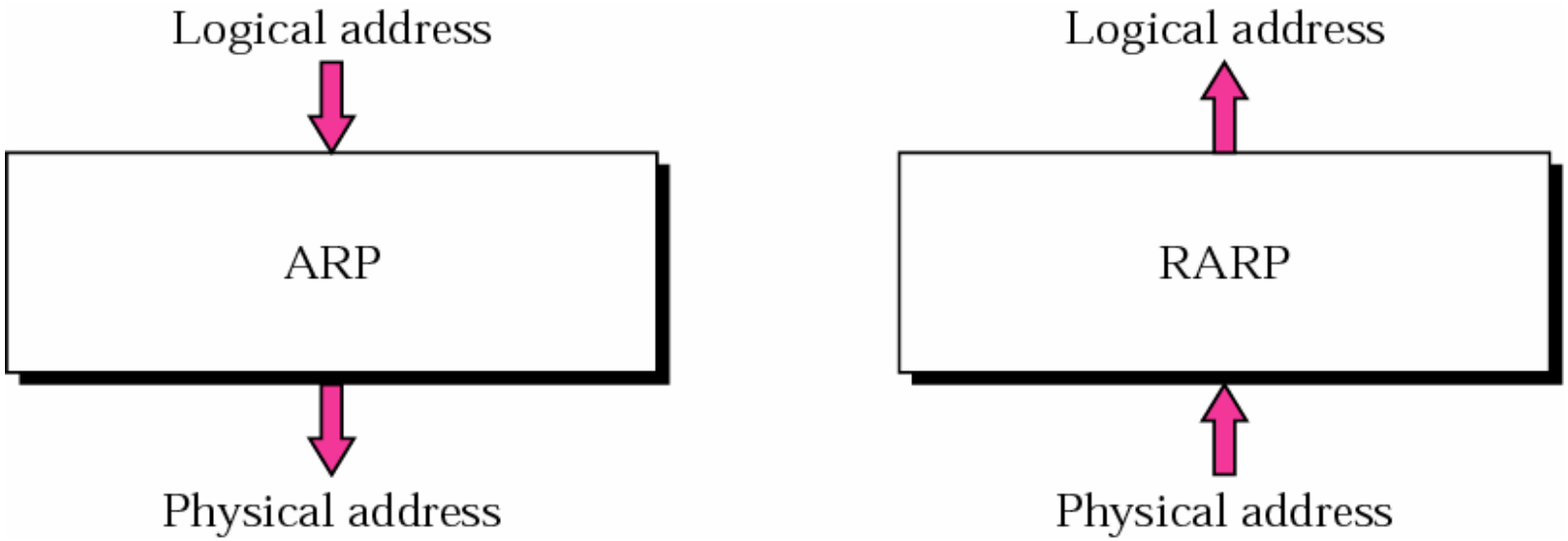
RARP – Khuôn dạng



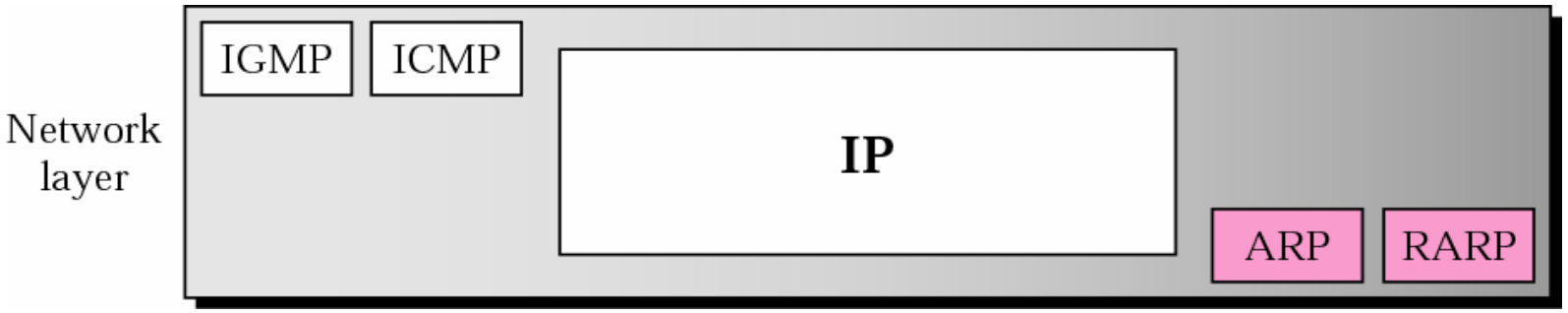
RARP – Bao Boc


RARP request or reply packet
Type: 0x8035
Preamble and SFD
Destination address
Source address
Type
Data
CRC
8 bytes
6 bytes
6 bytes
2 bytes
4 bytes

RARP và ARP


Logical address
ARP
Physical address
Logical address
RARP
Physical address

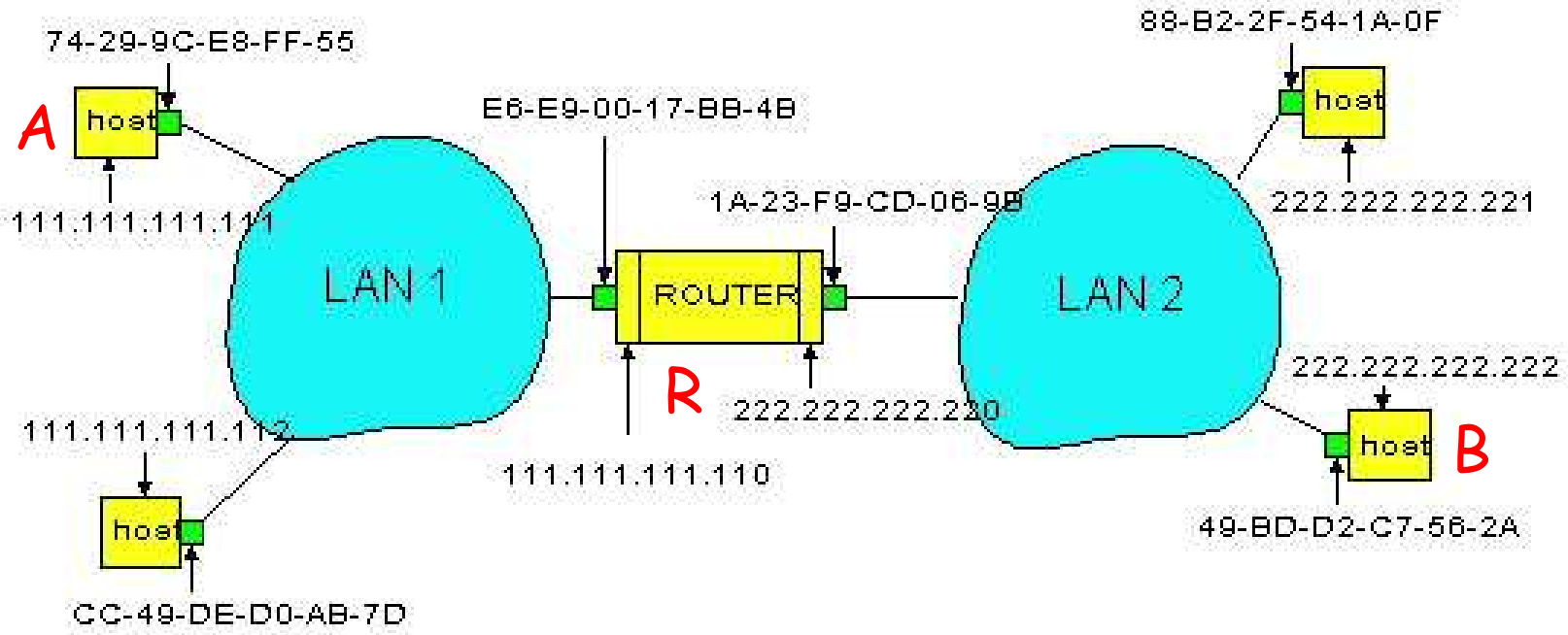
Vị trí RARP và ARP



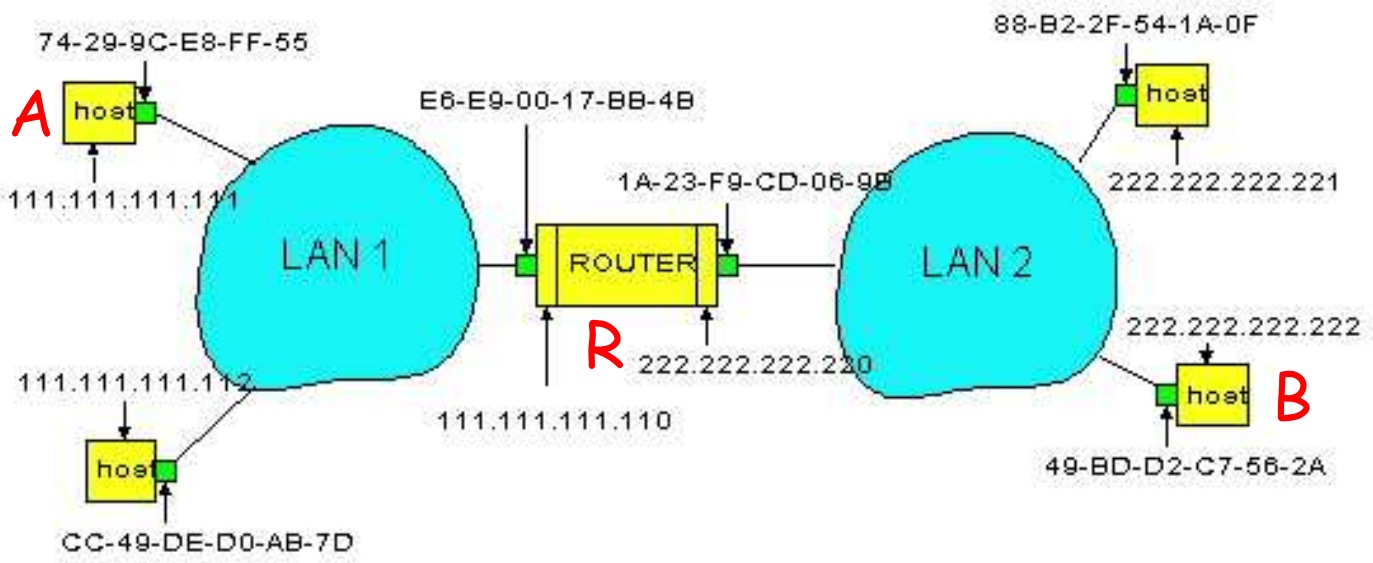
Nằm cùng với IP ở tầng Mạng. Bổ trợ cho IP

Định tuyến sang mạng LAN khác

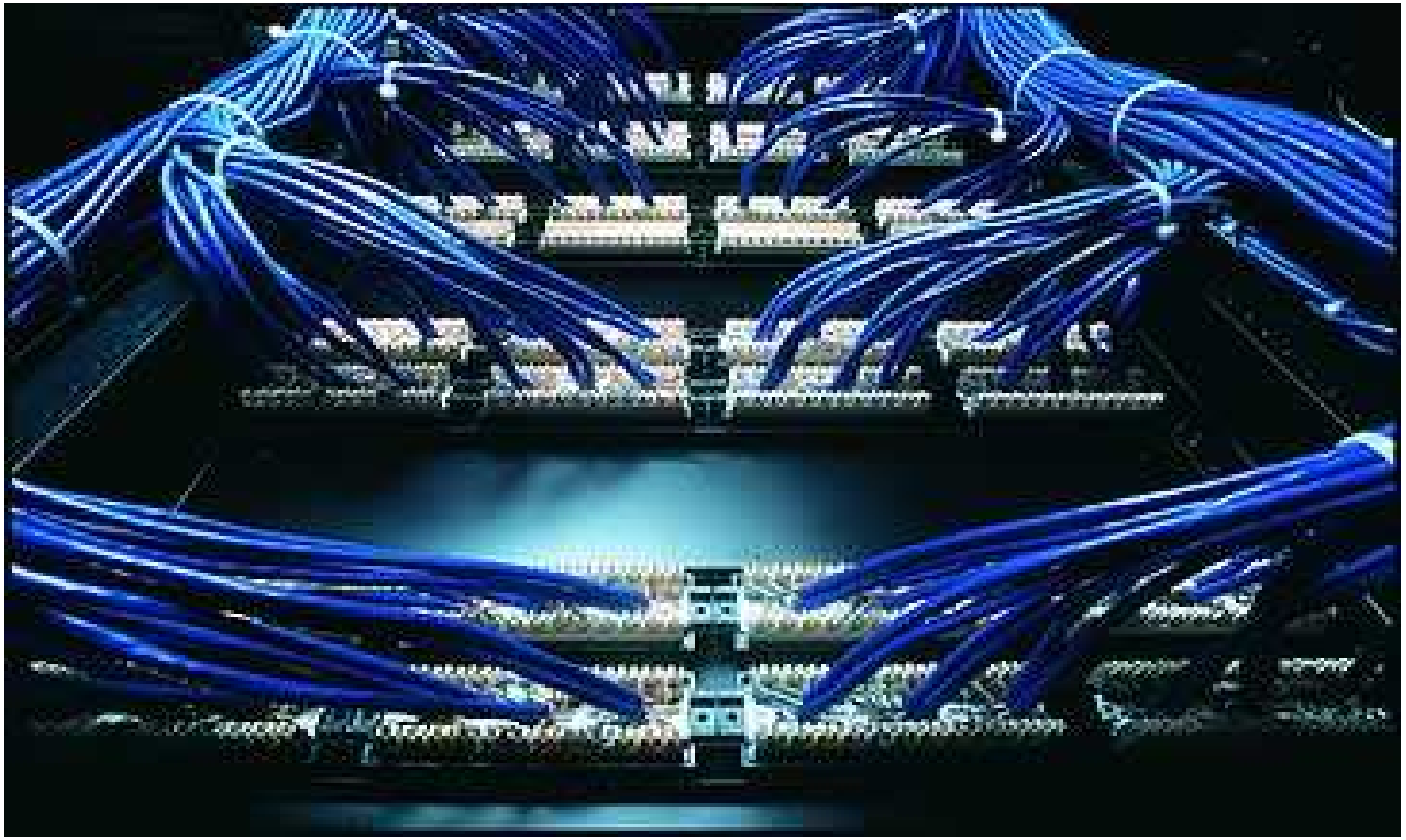
Gửi từ A tới B qua R



- A tạo ra một IP packet với địa chỉ Gửi là A, Nhận là B
- A sử dụng ARP để xác định Địa chỉ Vật lý ứng với IP 111.111.111.110
- A tạo ra Ethernet frame có đích là Địa chỉ Vật lý của R, frame này chứa IP datagram
- Tầng Liên kết dữ liệu của A gửi đi Ethernet frame
- Tầng Liên kết dữ liệu của R nhận được Ethernet frame
- R lấy ra IP datagram từ Ethernet frame, thấy Địa chỉ IP đích là B
- R sử dụng ARP để xác định Địa chỉ Vật lý của B
- R tạo ra frame chứa IP datagram Gửi-A-Nhận-B rồi gửi tới B



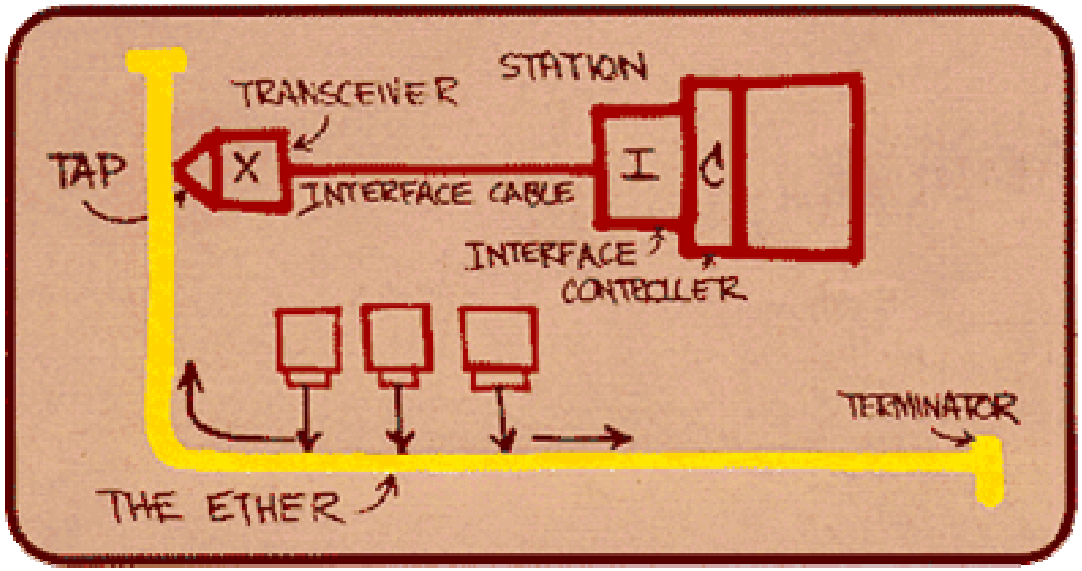
Công nghệ Kết nối Ethernet



Ethernet

Là Công nghệ hiện nay ***Thống trị*** thị trường LAN:

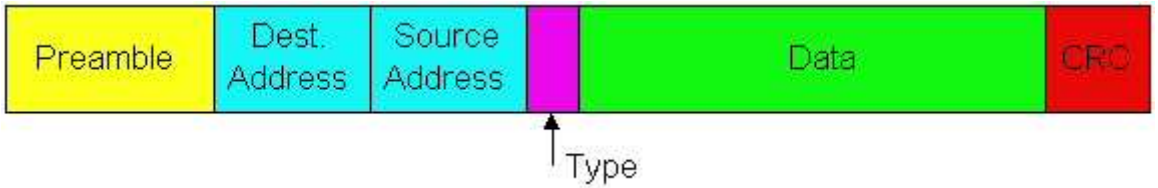
- Hiện tại 5USD (năm 2006)
- Là công nghệ LAN được sử dụng rộng rãi đầu tiên
- Đơn giản, Rẻ tiền hơn so với token LANs và ATM
- Liên tục nâng cao tốc độ : 10, 100, 1000 Mbps



Thủ bút của Metcalfe

Cấu trúc Frame Ethernet

Adapter phía Gửi đặt IP datagram (hay bất kỳ gói tin ở tầng Mạng nào) bên trong Ethernet frame

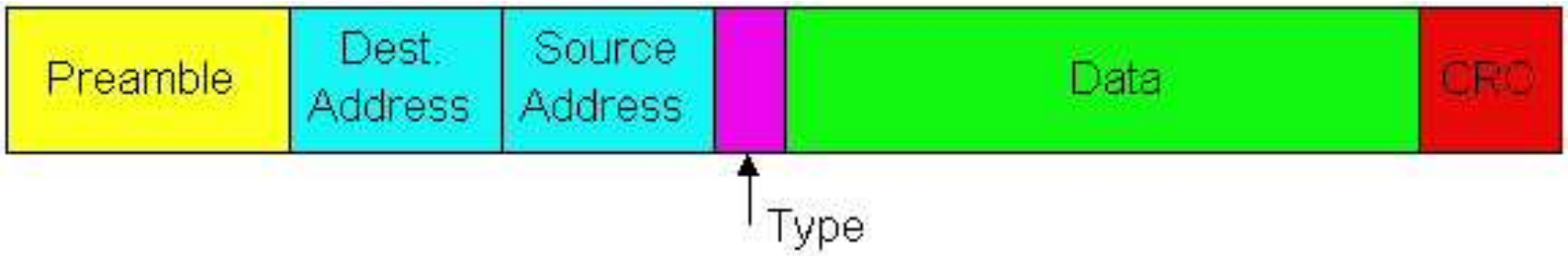


Preamble:

- 7 bytes có giá trị 10101010 sau đó là 1 byte có giá trị 10101011
- Để đồng bộ hóa tốc độ đồng hồ bên Gửi, bên Nhận

Ethernet Frame Structure (more)

- Addresses: 6 bytes, frame được tất cả các adapter trong LAN nhận, nhưng bỏ qua nếu địa chỉ không phù hợp
- Type: Chỉ ra giao thức nào ở tầng mạng sẽ nhận dữ liệu, chủ yếu là IP nhưng có thể có một vài giao thức khác (như Novell IPX hay AppleTalk)
- CRC: Được kiểm tra ở phía nhận, nếu phát hiện được lỗi, frame sẽ bị loại bỏ



Ethernet: Sử dụng CSMA/CD

A: Cảm nhận Kênh truyền, **if** Rỗi

then {

Truyền và Cảm nhận Kênh truyền;

If Phát hiện ra có một cuộc truyền khác

then {

Hủy bỏ và Gửi thông điệp báo tắc nghẽn;

Cập nhật # collisions;

Làm trễ theo thuật toán exponential backoff;

goto A

}

else {truyền xong frame; đặt biến collisions=0}

}

else {Đợi cuộc truyền kia truyền xong và goto A}

Ethernet: CSMA/CD (tiếp)

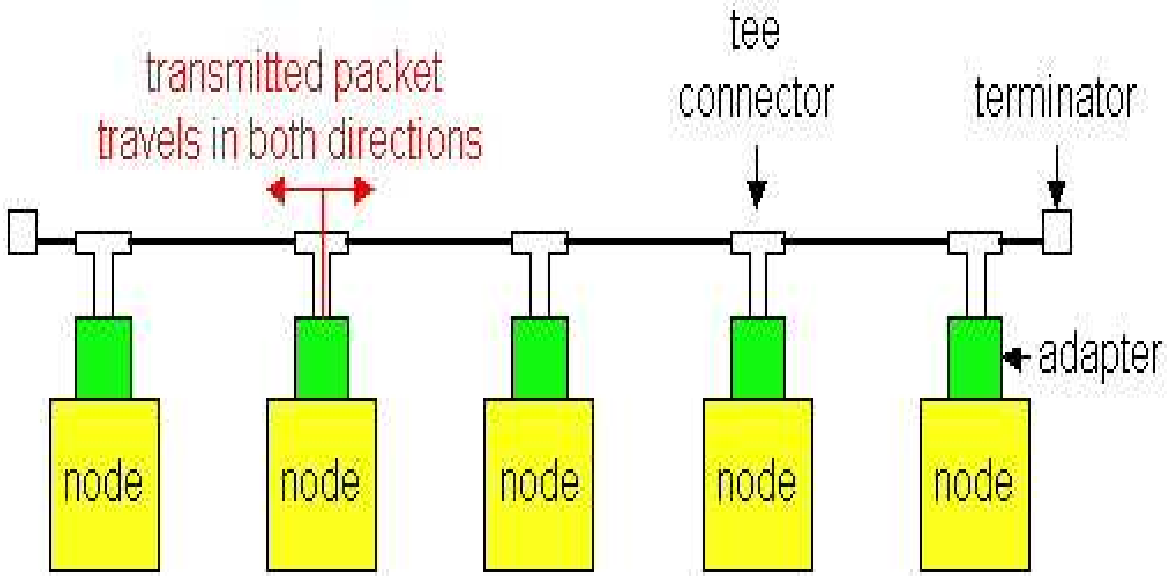
Tín hiệu báo tắc nghẽn (Jam Signal): đảm bảo tất cả các nút đang truyền nhận biết được Tắc nghẽn; 48 bits;

Exponential Backoff:

- *Mục tiêu*: Việc truyền lại thích nghi với tải hệ thống
  - Tải nhiều: Khoảng thời gian đợi ngẫu nhiên sẽ dài hơn
- Lần xung đột thứ nhất: chọn K trong khoảng {0,1}; độ trễ là K x 512 thời gian truyền đi 1 bit
- Sau lần xung đột thứ 2: chọn ngẫu nhiên K trong {0,1,2,3}…
- Sau lần xung đột thứ 10, chọn K trong {0,1,2,3,4,…,1023}

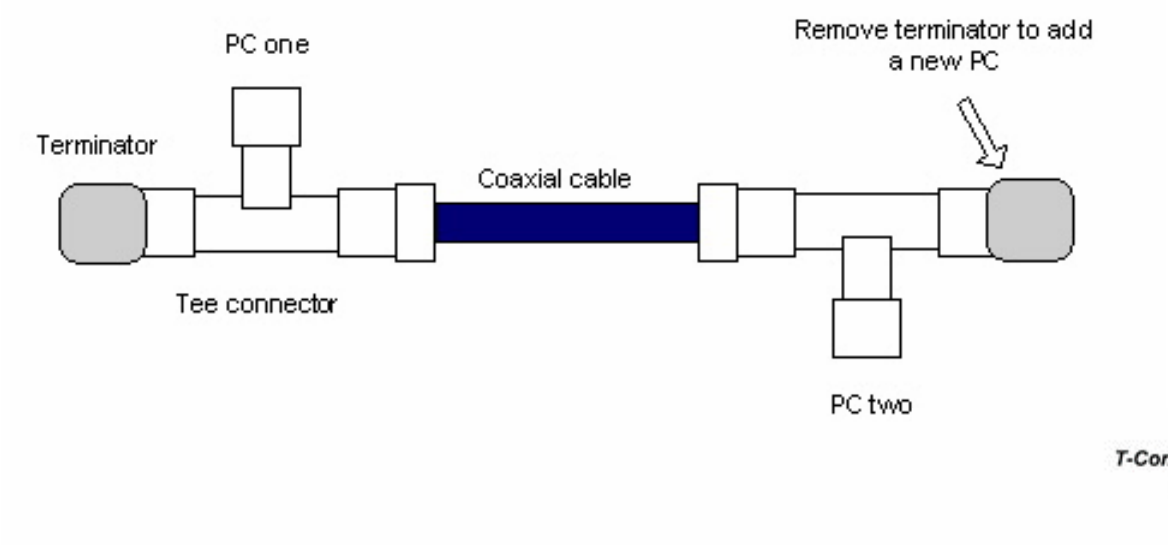
Công nghệ Ethernet : 10Base2

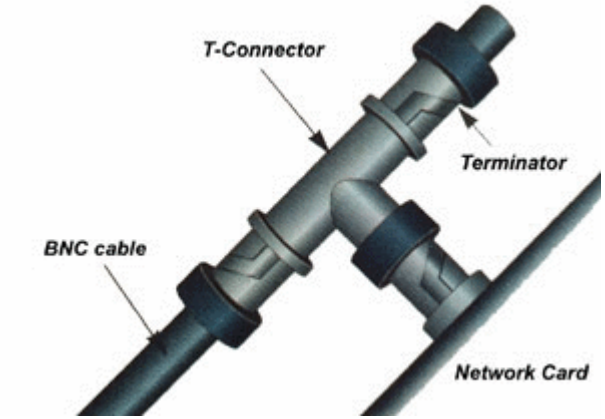
- 10: 10Mbps; 2: Khoảng cách tối đa của cáp là 200m
- Cáp gầy, Hình trạng (topo) BUS



- Có thể sử dụng Bộ tiếp sức (repeater) để mở rộng
- Repeater sao chép các tín hiệu từ một interface ra tất cả các interface khác (Chỉ là thiết bị ở tầng Vật lý)

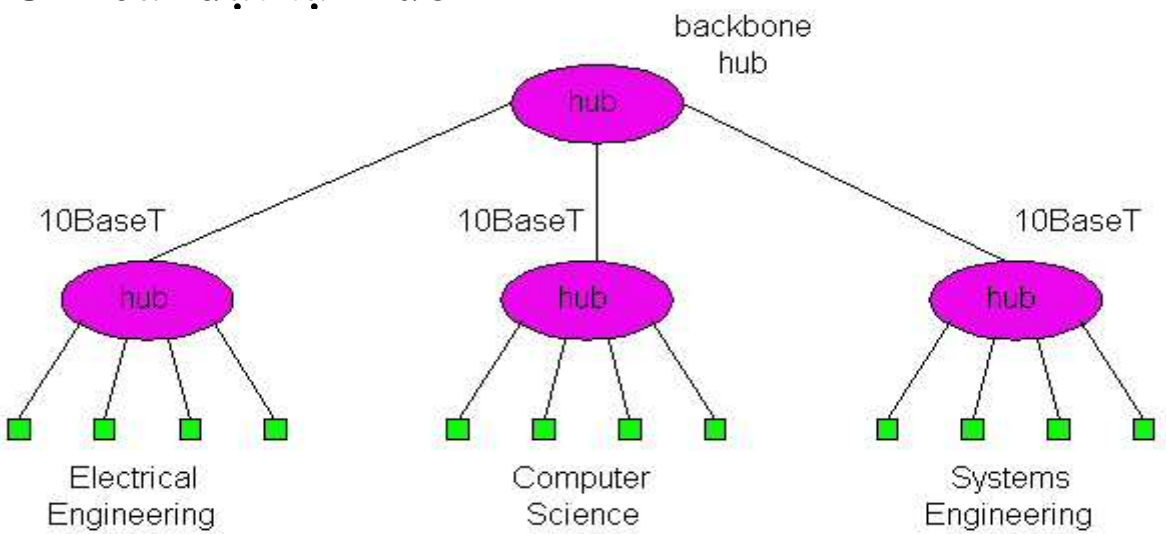
Lắp đặt


PC one
Remove terminator to add a new PC
Terminator
Coaxial cable
Tee connector
PC two


T-Connector
Terminator
BNC cable
Network Card

10BaseT và 100BaseT

- Tốc độ 10/100 Mbps rate; 100Mbps còn được gọi là “fast ethernet”
- T là viết tắt của Twisted Pair
- Các nút kết nối tới Hub tthông qua cáp đồng trục, Topo Hình sao
- CSMA/CD cài đặt tại hub



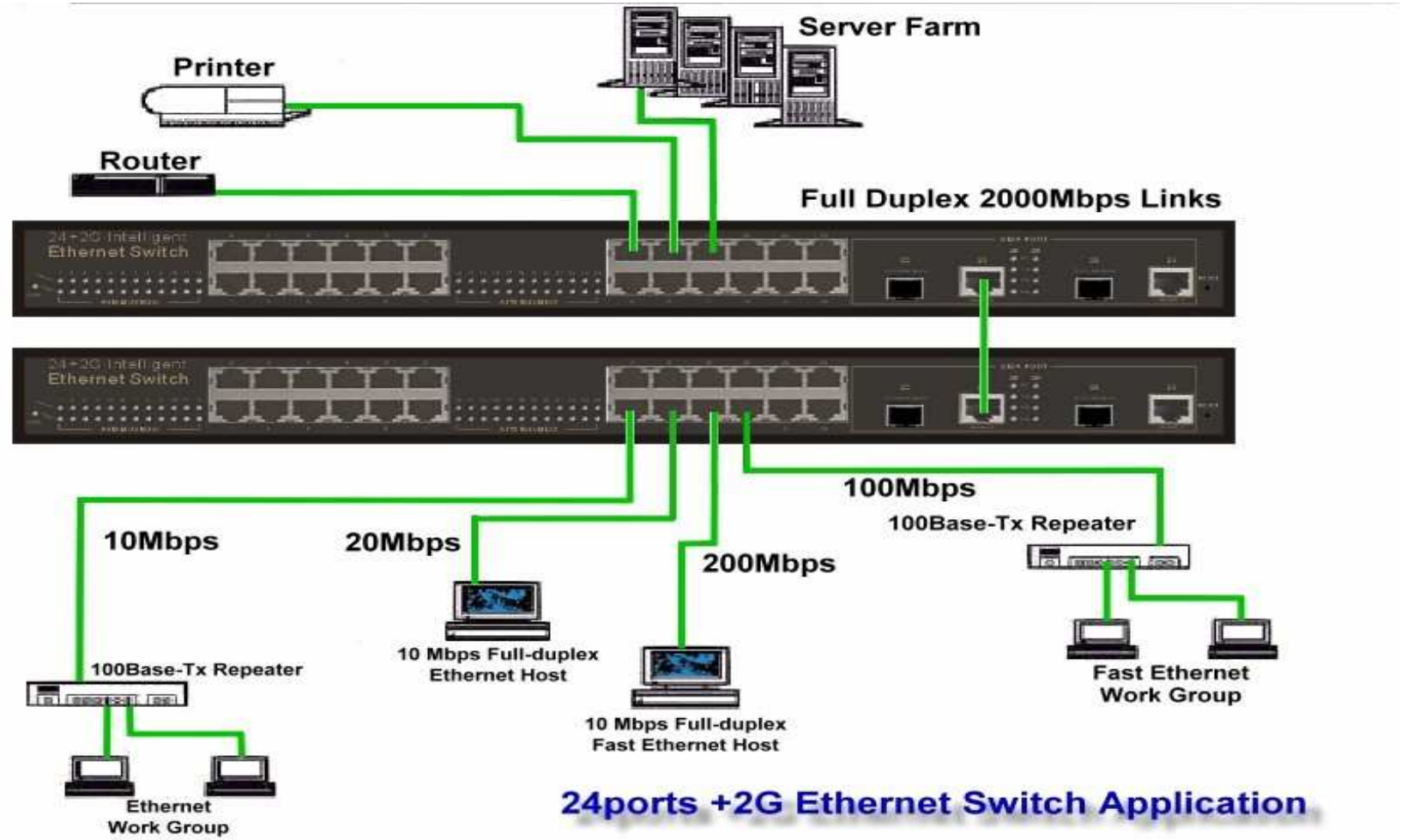
10BaseT và 100BaseT (tiếp)

- Khoảng cách cực đại từ nút tới Hub là 100m
- Hub có thể phong tỏa các adapter gây nhiễu (jabbering)
- Hub có thể giúp người quản trị mạng LAN thu thập các thông tin mang tính kiểm soát, các thông tin thống kê

Gbit Ethernet

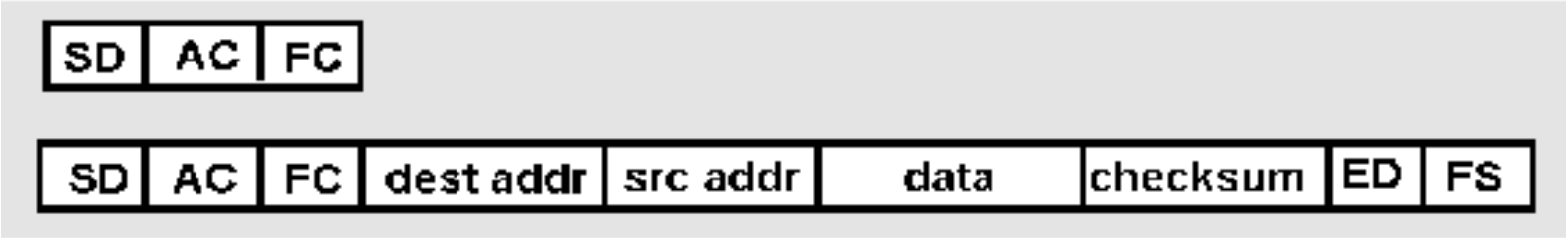
- Sử dụng khuôn dạng Ethernet frame chuẩn
- Cho phép kênh truyền Điểm-nối-Điểm và kênh truyền Dùng chung chia sẻ
- Trong chế độ dùng chung, sử dụng CSMA/CD; để có hiệu suất cao, khoảng cách giữa các nút phải nhỏ
- Cơ chế truyền song công ở tốc độ 1 Gbps cho kênh truyền Điểm nối Điểm

Ví dụ về Switch Ethernet 1G



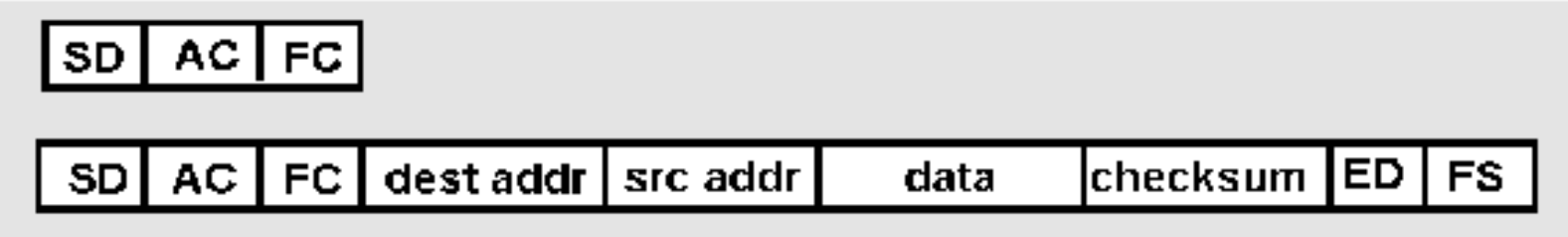
Chuyển Thẻ bài: Chuẩn IEEE802.5

- 4 Mbps
- Thời gian cực đại giữ token: 10 ms, kích thước frame bị hạn chế



- SD, ED : Đánh dấu Khởi đầu và Kết thúc của packet
- AC: Byte điều khiển (Access Control):
  - token bit: giá trị 0 có nghĩa là có thể lấy token, giá trị 1 nghĩa là dữ liệu đi sau FC
  - priority bits: Độ ưu tiên của packet
  - reservation bits: Trạm có thể thiết lập những bit này để ngăn ngừa những trạm có độ ưu tiên thấp hơn chiếm token khi token rỗi

Chuyển Thẻ bài: Chuẩn IEEE802.5



- FC: frame control được sử dụng để kiểm soát và bảo trì
- source, destination address: 48 bit địa chỉ Vật lý giống như trong Ethernet
- data: packet từ tầng Mạng
- checksum: CRC
- FS: frame status: Được phía Nhận đặt, phía Gửi sẽ đọc
  - Được thiết lập để xác nhận nút Đích đã nhận frame từ Vòng
  - Biên nhận ở mức Liên kết Dữ liệu

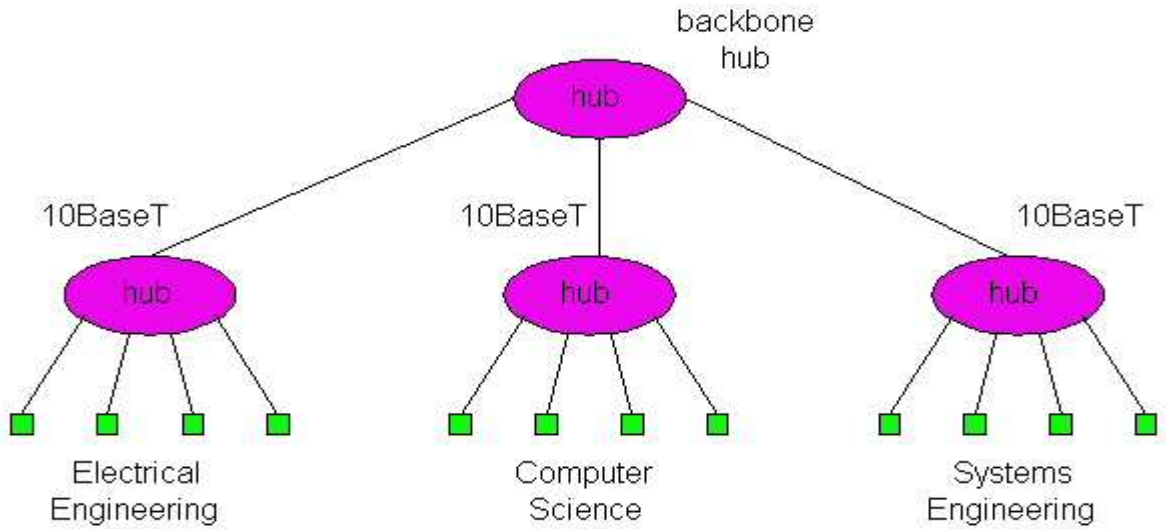
Kết nối nhiều mạng LAN

Q: Tại sao không có một mạng LAN lớn duy nhất?

- Hạn chế băng thông: trên mạng LAN duy nhất, tất cả các trạm chia sẻ môi trường truyền dẫn chung
- Độ dài hạn chế: 802.3 xác định độ dài cực đại của cáp
- "miền xung đột" lớn (Có thể xung đột với nhiều trạm khác)
- Hạn chế số lượng trạm: 802.5 chuyển thẻ bài có độ trễ ở mỗi trạm

Hubs

- Thiết bị ở Tầng Vật lý: Chủ yếu là repeater hoạt động ở mức bit: mỗi bit nhận được trên một interface sẽ được gửi ra trên tất cả các interface khác
- Hub có thể được sắp xếp có thứ bậc (Thiết kế nhiều tầng), với hub trục chính (backbone) ở vị trí cao nhất



Hubs (tiếp)

- LAN **segment :** phân đoạn mạng LAN. Là miền xung đột
- Hub Không cô lập các miền xung đột: 2 nút ở hai LAN segment khác nhau vẫn có thể bị xung đột

Ưu điểm của Hub :

- Đơn giản, Rẻ tiền
- Kết nối nhiều mức : Khả năng chống chọi lỗi
- Có thể mở rộng khoảng cách bằng cách lắp thêm Hub

Hub: Hạn chế

- Miền xung đột duy nhất không thể tăng thông lượng toàn bộ Hệ thống
  - Thông lượng trên cả hệ thống bằng thông lượng trên một phân đoạn
- Hạn chế về số lượng nút trong một phân mạng
- Không thể kết nối các kiểu công nghệ Ethernet khác nhau (ví dụ 10BaseT và 100baseT)

Bridges

- Thiết bị ở tầng Liên kết Dữ liệu: thao tác trên Ethernet frames, lấy ra tiêu đề gói tin và chuyển tiếp frame một cách có chọn lọc căn cứ theo địa chỉ đích
- Bridge cô lập các miền xung đột vì lưu tạm (buffer) các frame
- Khi cần chuyển frame vào segment nào, bridge sử dụng CSMA/CD để truy cập và truyền trên segment

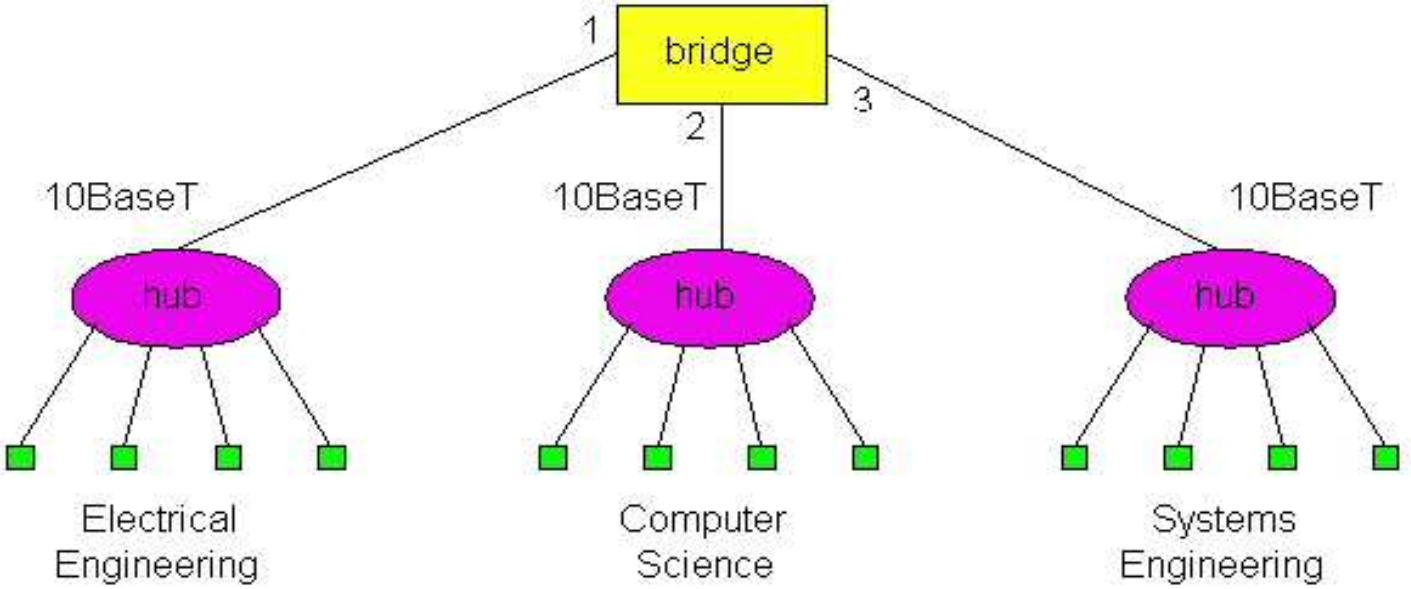
Bridges (tiếp)

- Ưu điểm của Bridge:
  - Cô lập các miền xung đột nên tổng dung lượng cả hệ thống được nâng cao và không hạn chế số lượng nút trong mạng
  - Có thể Kết nối nhiều kiểu công nghệ Ethernet khác nhau vì Hành vi Giữ và Chuyển
  - Trong suốt: Không cần thay đổi các LAN adapter

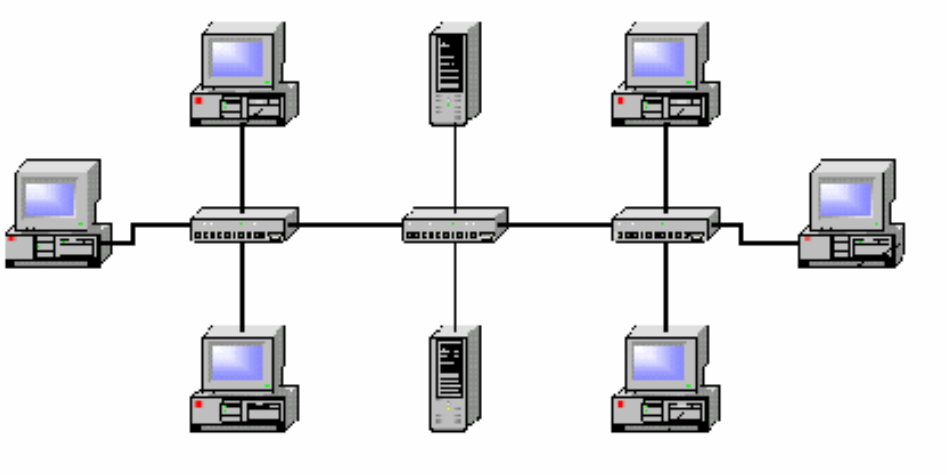
Bridges: Lọc và Chuyển các frame

- bridges LỌC các packet
  - Frame mà nút gửi và nhận trên cùng LAN segment không cần chuyển qua LAN segment khác
- CHUYỂN:
  - Làm thế nào để xác định interface có thể chuyển gói tin đến đúng đích?
  - Có vẻ giống Định tuyến ?

Backbone Bridge



Kết nối không cần Backbone



- Không được sử dụng do hai nguyên nhân sau:
  - Nếu hub ở Computer Science hỏng thì mạng sụp đổ
  - Tất cả truyền thông qua EE và SE phải chuyển qua CS segment

Bridge : Lọc (Filtering)

- bridge *Học* có thể gửi đến máy tính nào qua interfave nào: Xây dựng Bảng lọc
  - Khi nhận được frame, bridge "học" được vị trí của máy gửi: LAN segment đến
  - Ghi lại vị trí nút gửi trong bảng Lọc
- Các mục trong bảng Lọc:
  - (Địa chỉ logic của nút, Bridge Interface, Time Stamp)
  - TTL : 60 phút sẽ xóa đi mục tương ứng

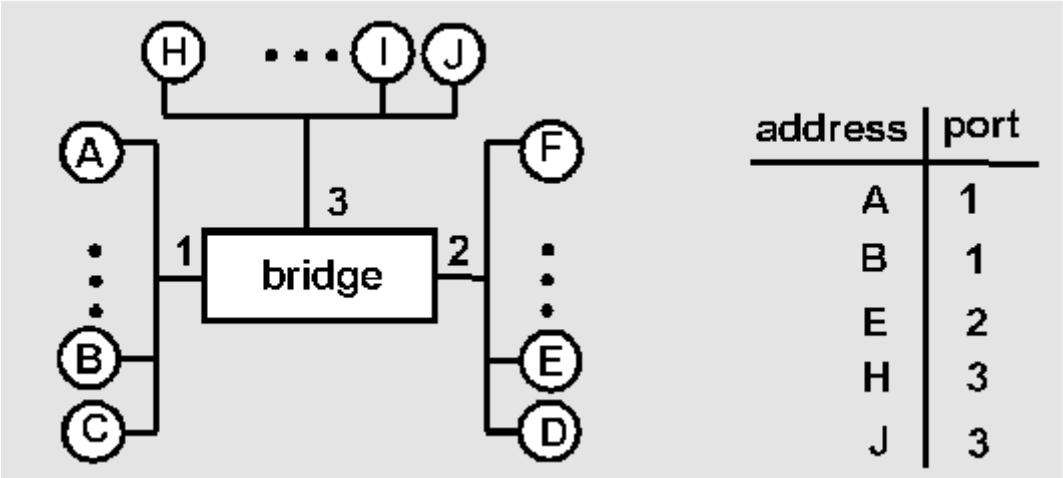
Bridge : Chức năng Lọc

- Thuật toán Lọc:

```
if Đích của frame nằm trong cùng một phân đoạn nơi gửi frame tới
    then Loại bỏ frame
    else { Tìm kiếm trên bảng Lọc
        if Tìm thấy mục ứng với đích trong bảng Lọc
            then Chuyển tiếp frame trên cổng ra tương ứng;
            else Gửi tràn ngập;    /* Gửi đi tất cả các
              interface ngoại trừ interface đến của frame*/
    }
```

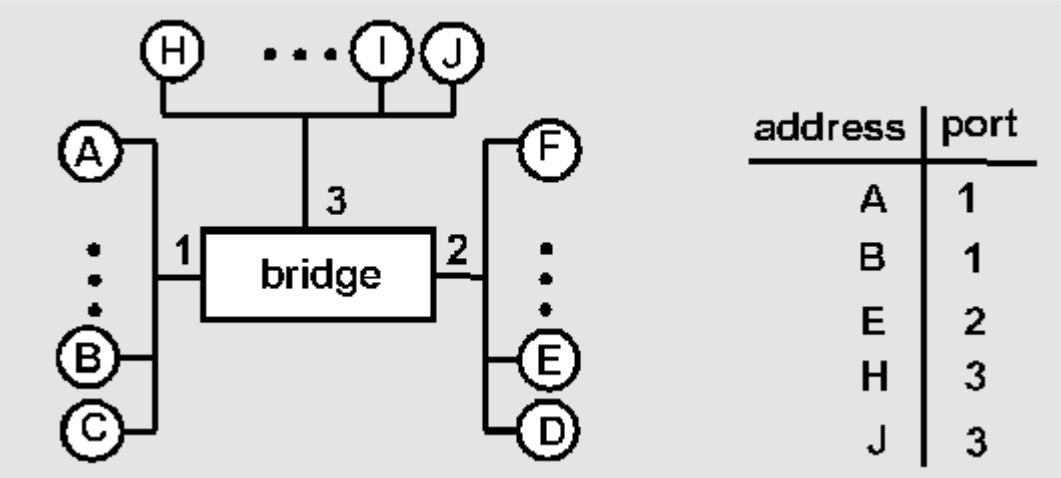
Bridge : Ví dụ Chức năng Học

Giả sử C gửi frame tới D và D gửi frame trả lời C



- C gửi frame, bridge chưa có thông tin gì về D, do vậy gửi tràn ngập trên toàn bộ mạng LAN
  - bridge nhận thấy C được gửi từ port 1
  - Mạng LAN ở phía trên bỏ qua frame này
  - Cuối cùng D nhận được frame

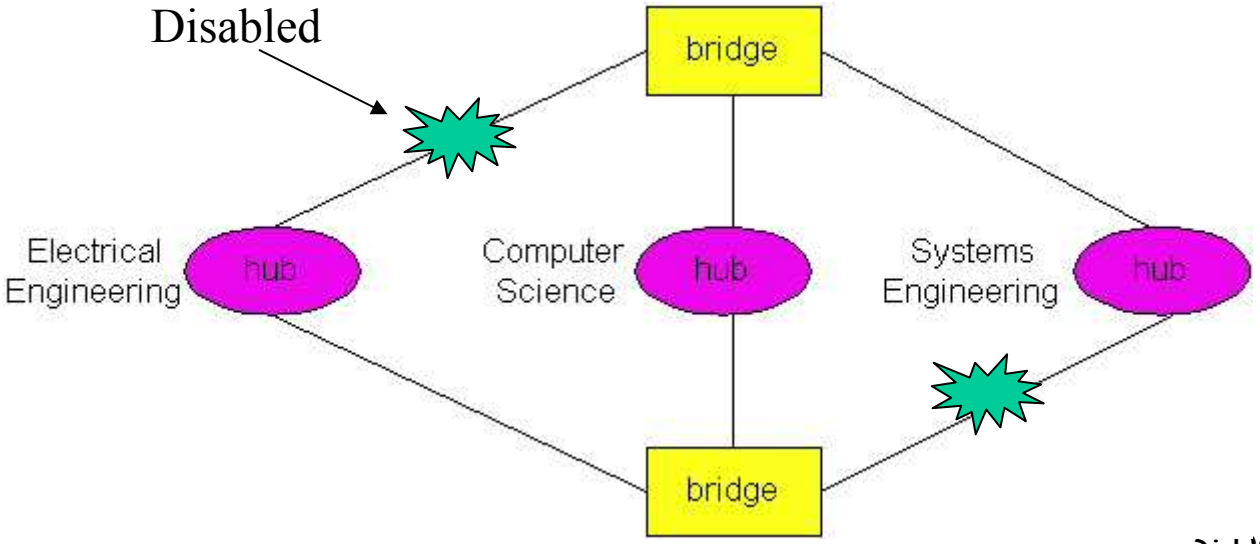
Bridge Learning: example



- D tạo ra câu trả lời gửi cho C, sau đó gửi
  - bridge nhìn thấy frame đến từ D
  - bridge ghi nhớ D nằm trên interface 2
  - Vì bridge biết C nằm trên interface 1, do vậy chuyển tiếp *có chọn lọc* frame qua interface 1

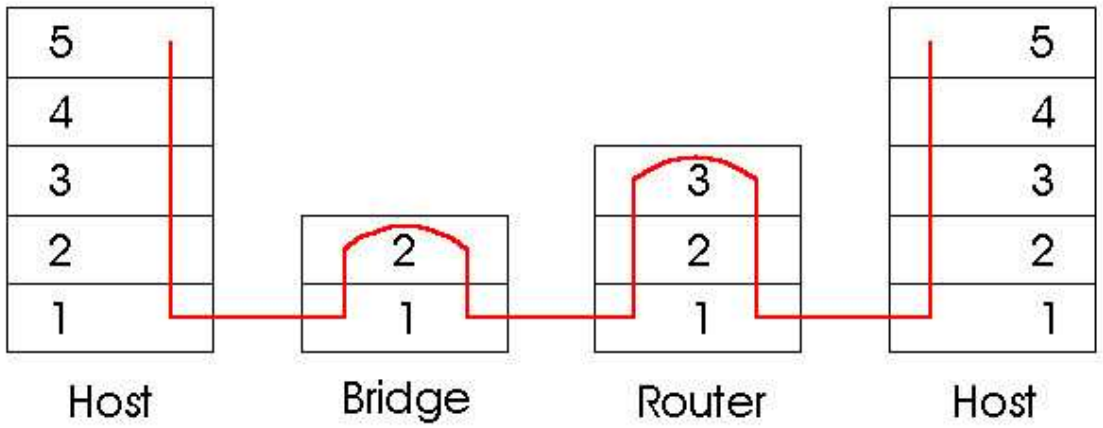
Bridges : Spanning Tree

- Để tăng cường tính tin cậy, người ta mong muốn giữa hai điểm có nhiều tuyến đường khác nhau
- Với nhiều tuyến đường, có thể nảy sinh chu trình – các frame luân lưu mãi mãi trong mạng
- **Giải pháp**: Sắp xếp các bridges thành cây spanning bằng cách cấm một số interface



So sánh Bridges và Routers

- Đều là Thiết bị ***Lưu giữ và Chuyển tiếp***
  - router: Thiết bị ở tầng Mạng (Kiểm tra tiêu đề ở tầng Mạng)
  - bridges : Thiết bị ở tầng liên kết dữ liệu
- Router sử dụng Thuật toán Định tuyến để xây dựng Bảng Định tuyến
- Bridge chạy các Thuật toán Học, Lọc, Chuyển tiếp để xây dựng Bảng Lọc



So sánh Bridges và Routers

Bridges + và -

+ Bridge : Thao tác Đơn giản hơn

- Topo Mạng bị hạn chế vì phải xây dựng cây spanning để tránh chu trình

- Bridges không có khả năng chống chọi với tấn công kiểu flood (tín hiệu đến từ 1 cổng sẽ được sao chép đến tất cả các cổng khác

So sánh Bridges và Routers

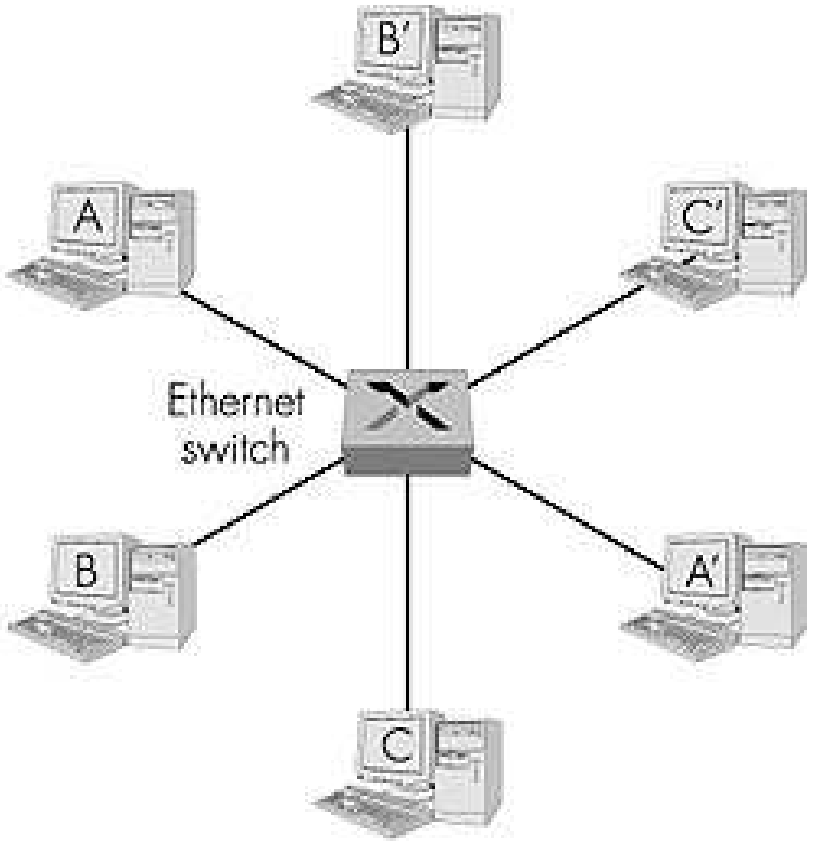
Routers + and -

+ Hình trạng tuỳ ý (Chu trình được khắc phục nhờ TTL và giao thức định tuyến tốt)
+ Có firewall để chống tấn công gửi tràn ngập
- Yêu cầu phải cấu hình địa chỉ IP (not plug and play)
- Tốn năng lực xử lý hơn

- Bridges : Mạng vài trăm máy
- Router : Mạng vài nghìn máy

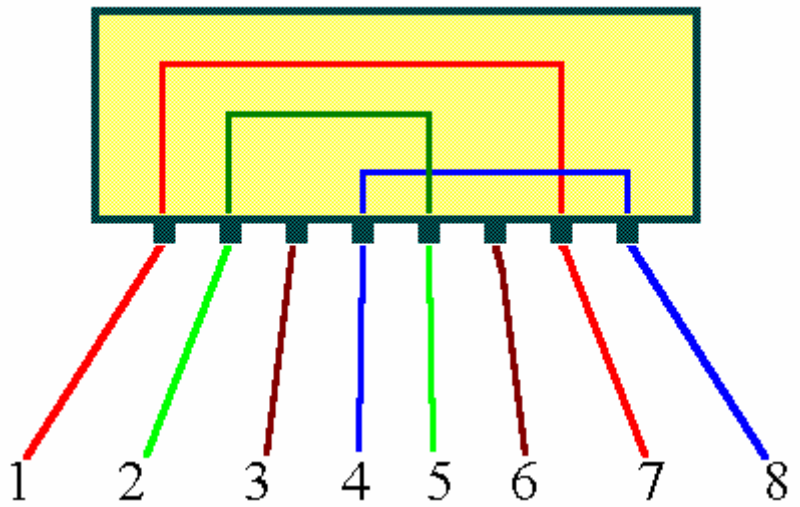
Ethernet : Switch

- Là Thiết bị ở tầng 2, Lọc và Chuyển tiếp frame trên cơ sở địa chỉ MAC
- Switching: A-tới-B và A'-tới-B' đồng thời, không bị xung đột
- Nhiều interface
- Thường các máy tính kết nối trực tiếp vào switch theo cấu hình sao
  - Ethernet, nhưng không có xung đột!

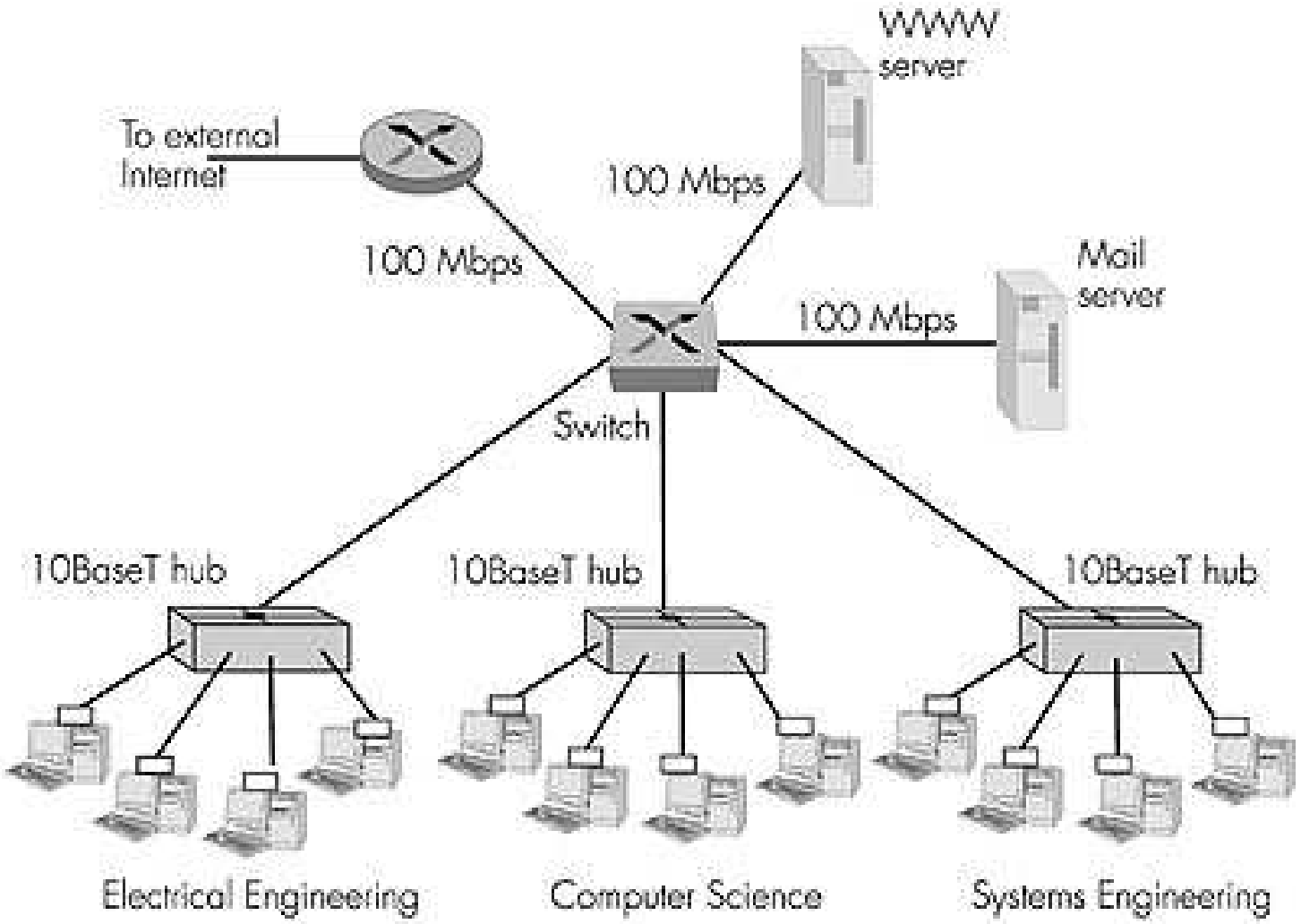


Ethernet Switches

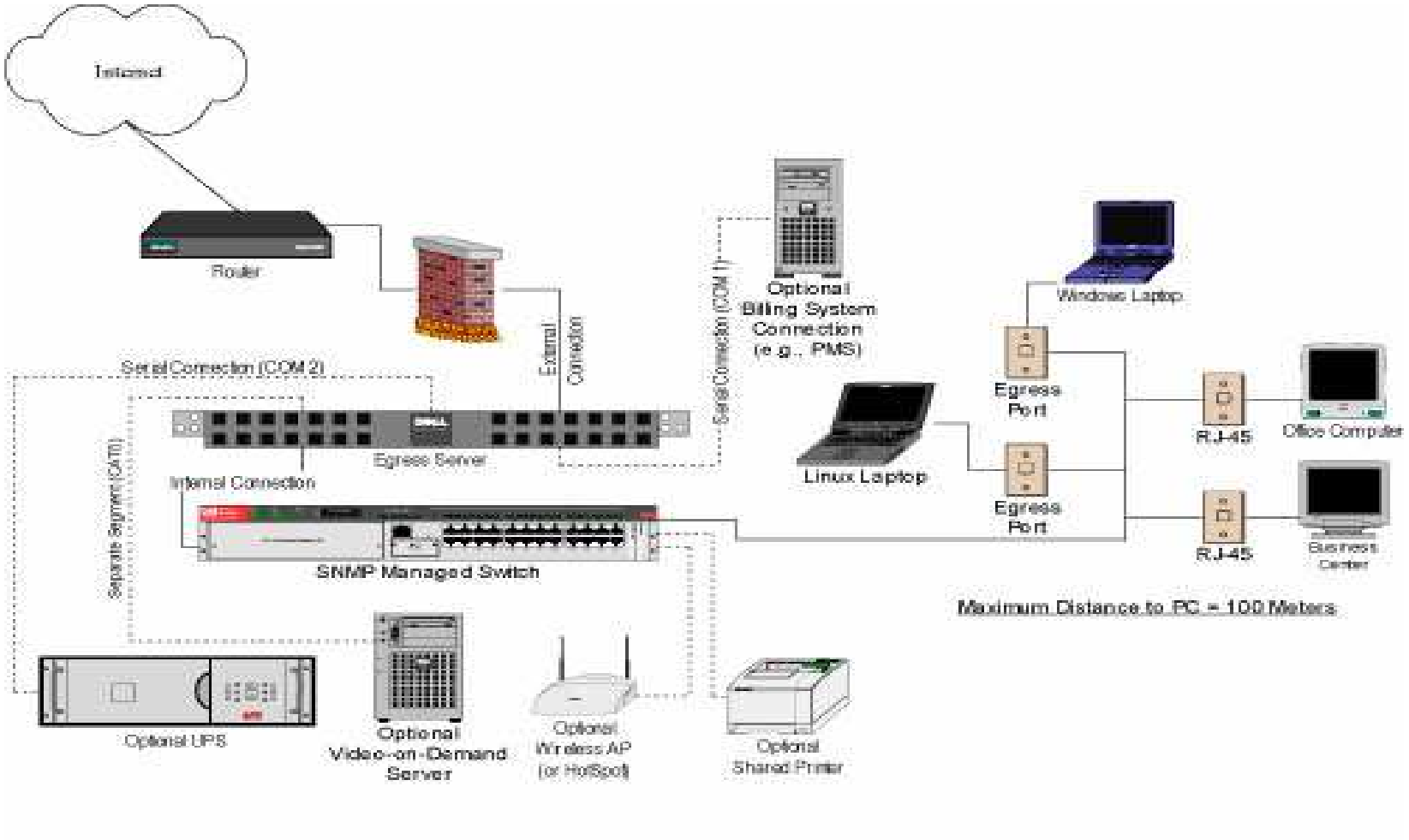
- Chuyển xuyên suốt: frame có thể chuyển ngay từ input port tới output port mà không cần nhận hết frame
  - Giảm đáng kể độ trễ
- Kết hợp Dùng chung\ Dùng riêng, Tốc độ 10/100/1000 Mbps



Ethernet Switch (tiếp)

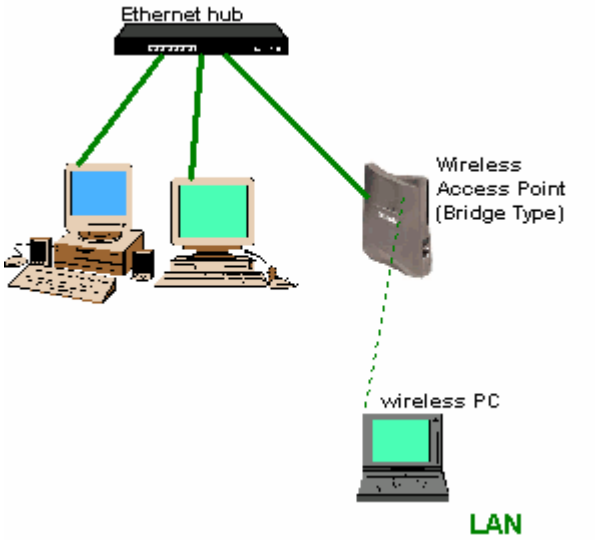


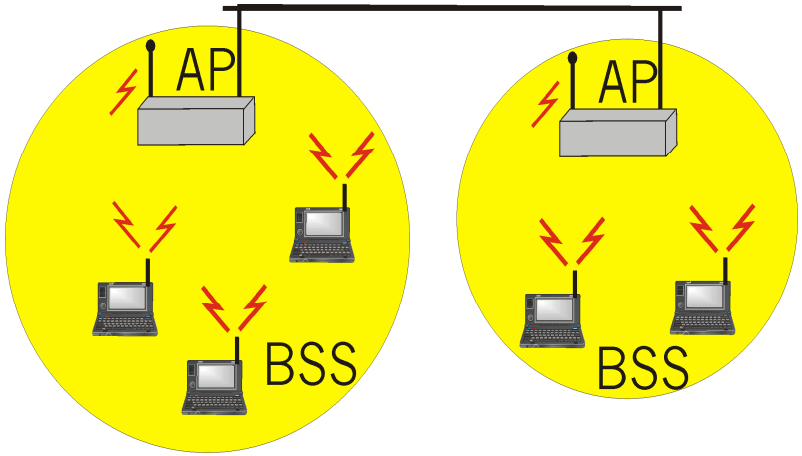
Ethernet Switch (tiếp)



IEEE 802.11 : LAN không dây

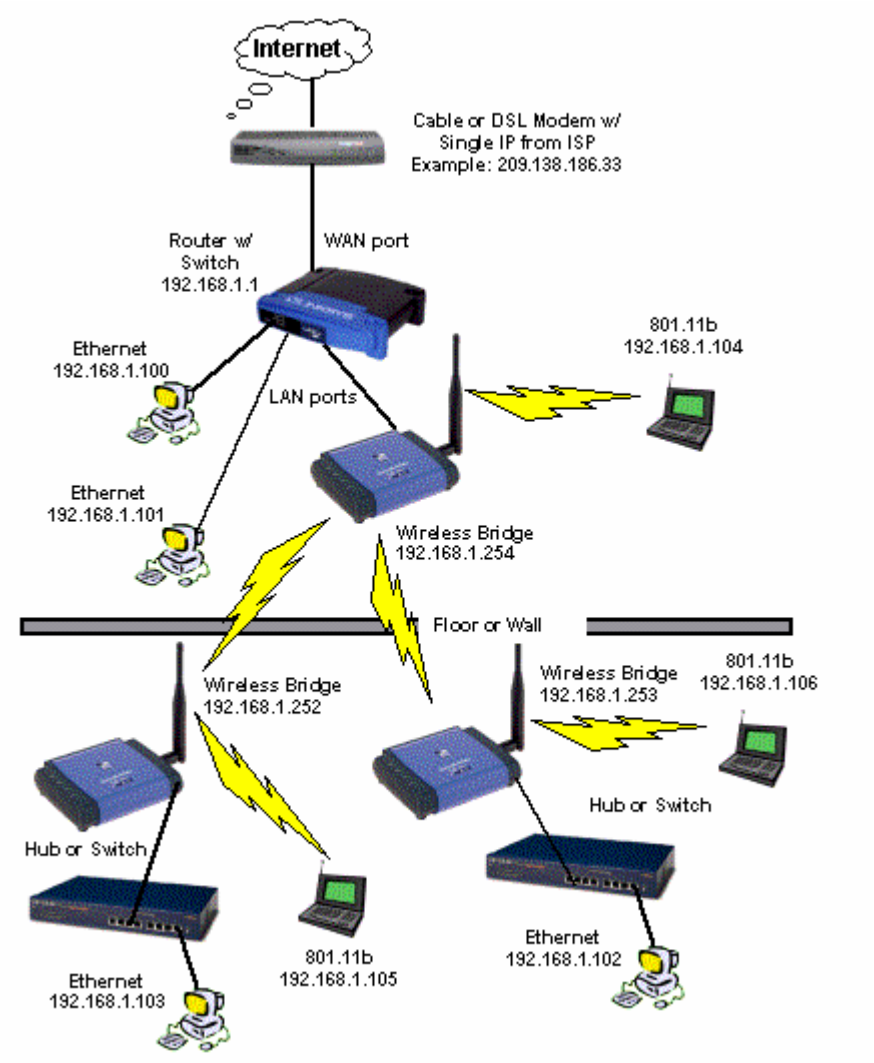
- LAN không dây: thường dành cho máy tính xách tay (có khả năng di động)
- Chuẩn IEEE 802.11 :
  - MAC protocol
  - Phổ tần số : 900Mhz, 2.4Ghz





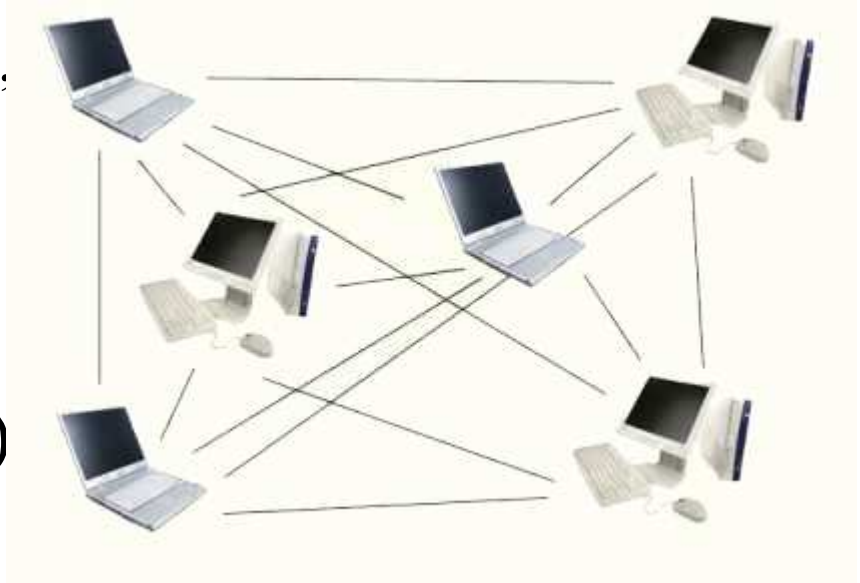
- Basic Service Set (BSS) bao gồm:
  - wireless hosts
  - access point (AP): base station
- Kết hợp các BSS thành hệ thống phân tán distribution system (DS)

Ví dụ

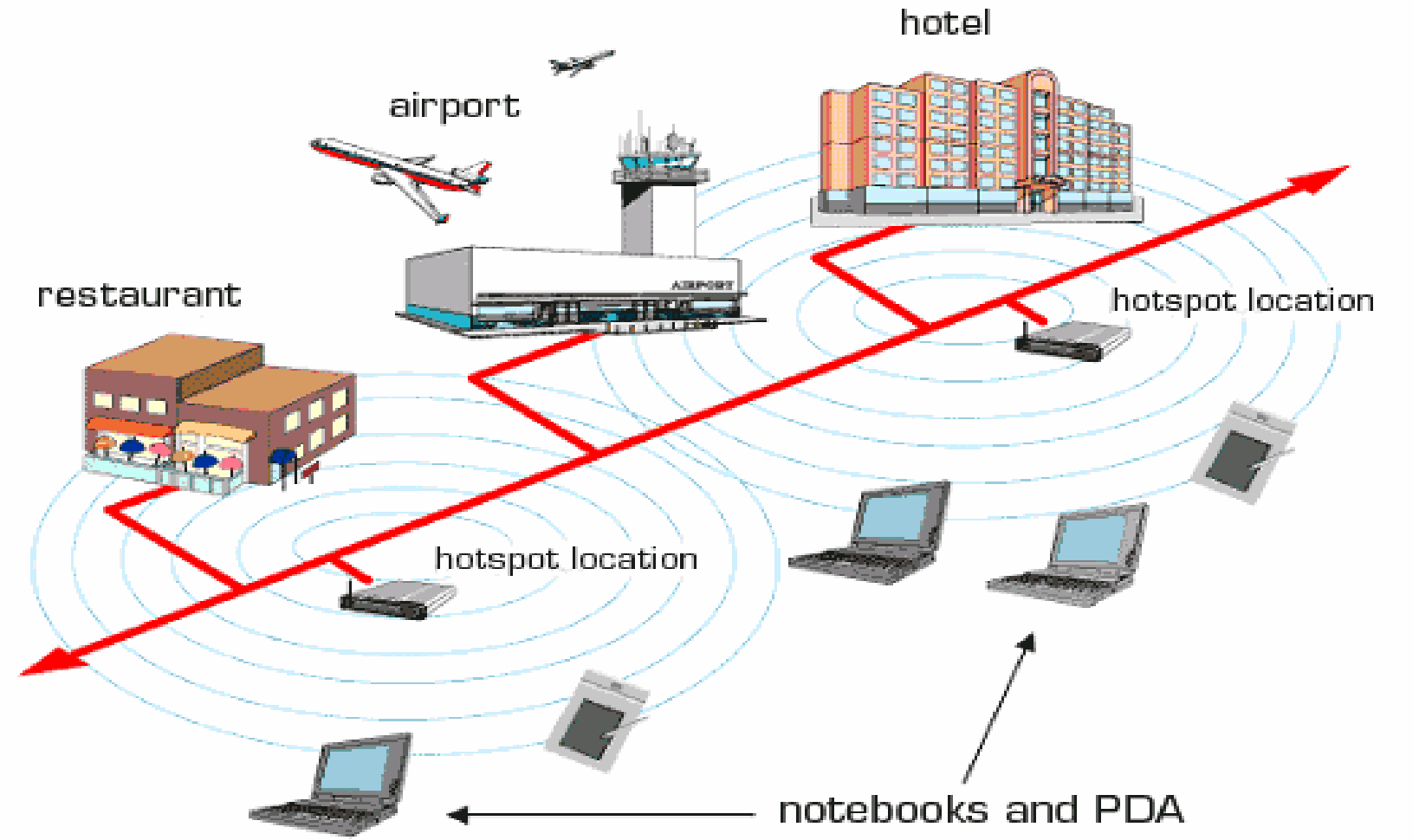

Internet
Cable or DSL Modem w/
Single IP from ISP
Example: 209.138.186.33
Router w/
Switch
192.168.1.1
WAN port
Ethernet
192.168.1.100
LAN ports
801.11b
192.168.1.104
Ethernet
192.168.1.101
Wireless Bridge
192.168.1.254
Floor or Wall
Wireless Bridge
192.168.1.252
Wireless Bridge
192.168.1.253
801.11b
192.168.1.106
Hub or Switch
Hub or Switch
801.11b
192.168.1.105
Ethernet
192.168.1.102
Ethernet
192.168.1.103

Mạng AdHoc

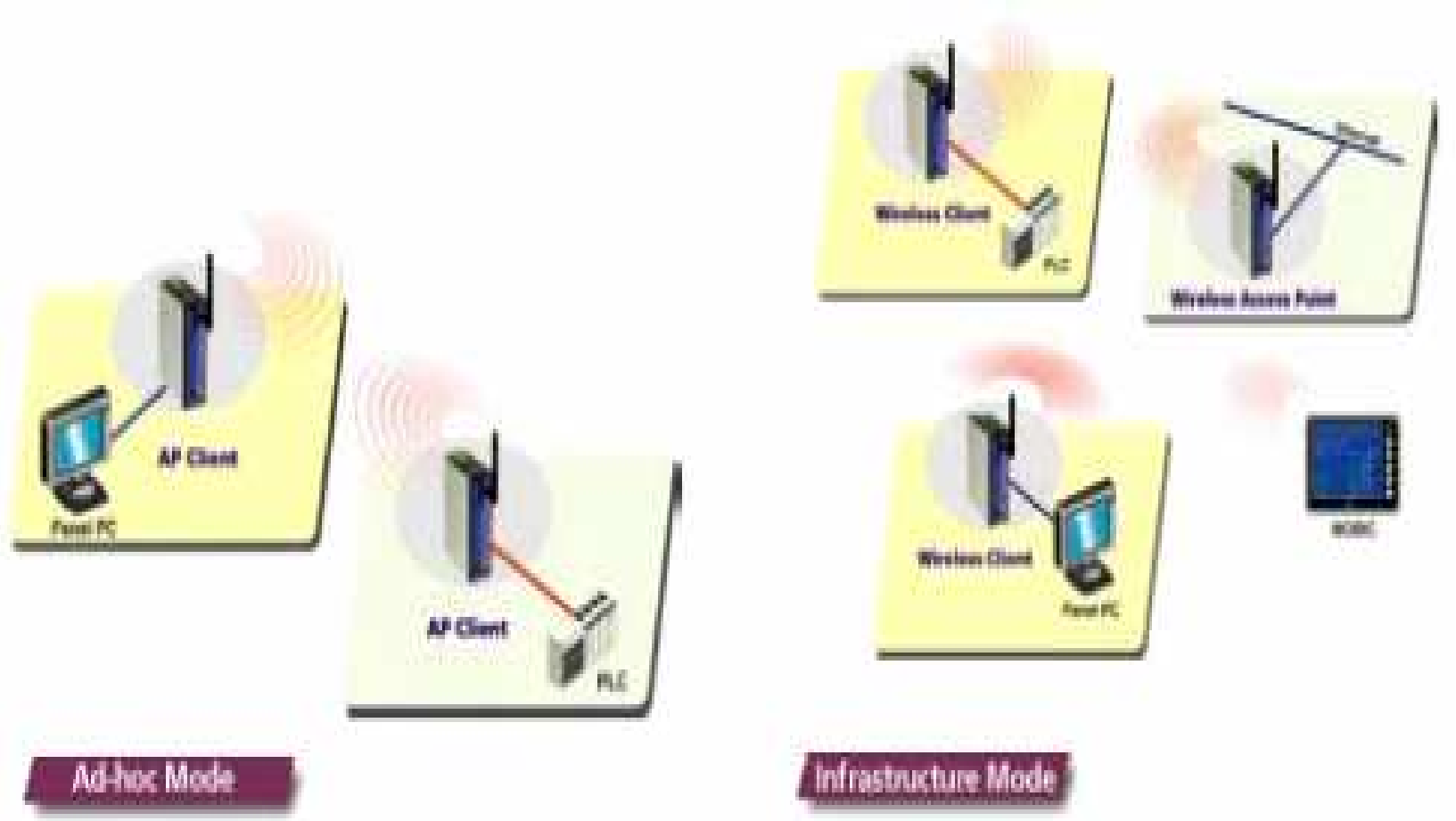
- Mạng Adhoc : IEEE 802.11. Các trạm có thể tự hình thành nên mạng *không cần có* AP
- Ứng dụng:
  - Các laptop trong phòng họp.
  - Kết nối các thiết bị cá nhân
  - Trên chiến trường
- IETF MANET (Mobile Ad hoc Networks) working group



Trong tương lai



So sánh : Có hạ tầng và Không có



IEEE 802.11 : CSMA/CA

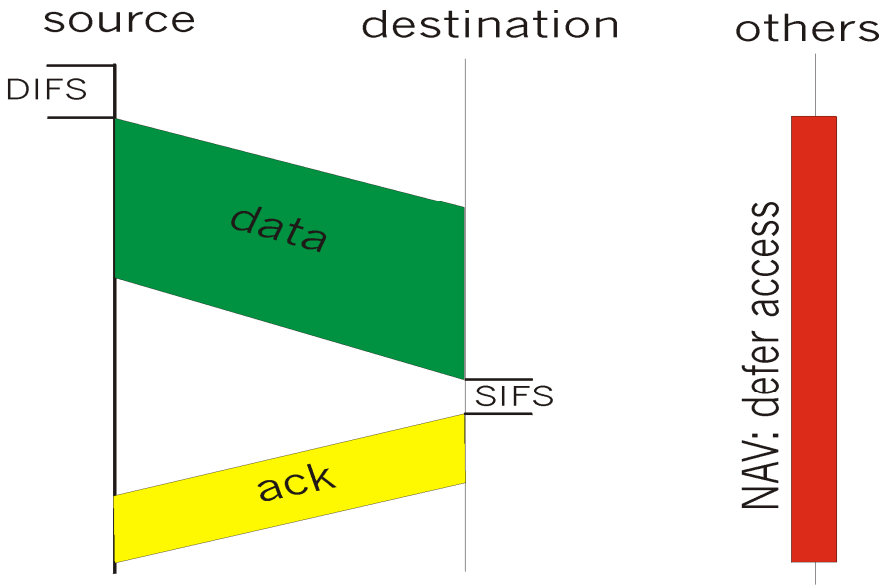
802.11 CSMA: Phía gửi

- **Nếu** kênh truyền rỗi trong **DISF** sec.

  thì truyền toàn bộ frame (không phát hiện xung đột)

- **Nếu** kênh truyền bận

  thì binary backoff



802.11 CSMA Phía Nhận:

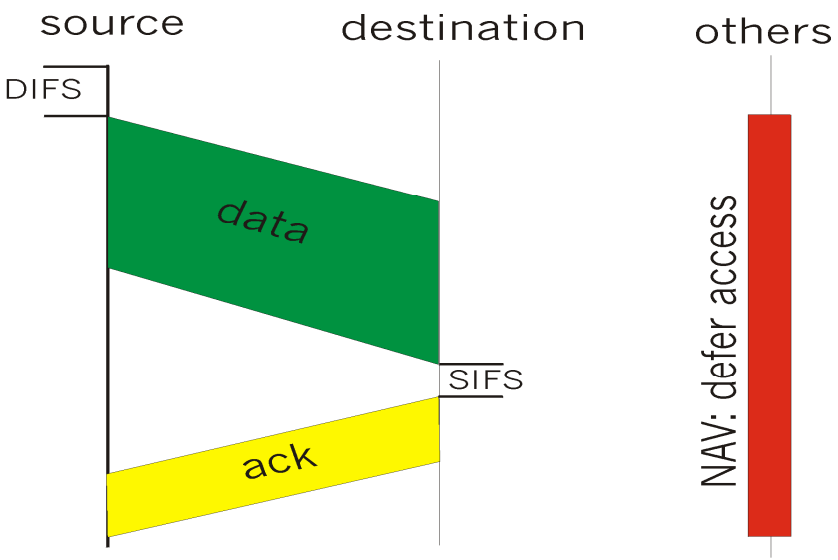
Nếu received OK

Biên nhận ACK sau SIFS

IEEE 802.11 MAC Protocol

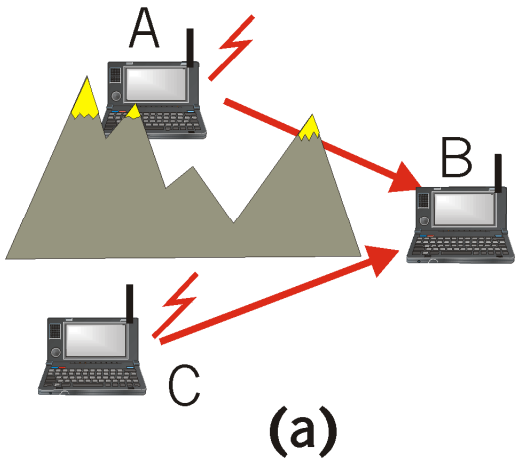
802.11 Một số giao thức CSMA khác

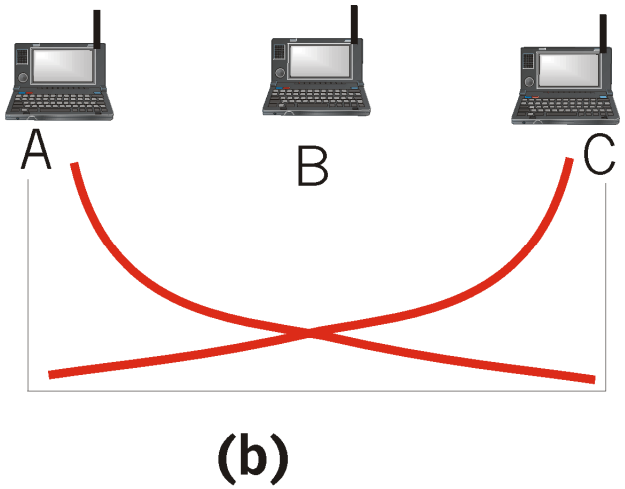
- **NAV**: Network Allocation Vector
- 802.11 frame có trường ***transmission time***
- Các trạm khác (khi nhận được frame) sẽ trì hoãn việc truyền trong NAV đơn vị thời gian

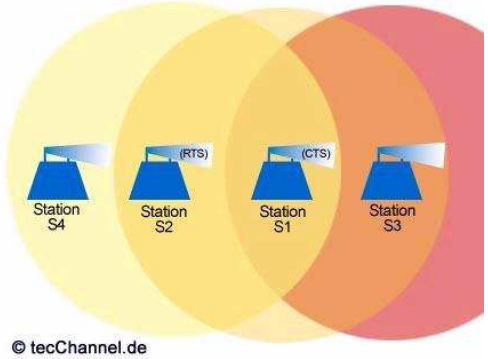


Vấn đề : Trạm ẩn

- **Trạm ẩn**: A và C không thể "nghe" từ nhau
  - Có vật cản, tín hiệu không lan toả qua được
  - Có thể xung đột ở B
- **Mục tiêu**: Tránh xung đột ở B
- **CSMA/CA: CSMA with Collision Avoidance**

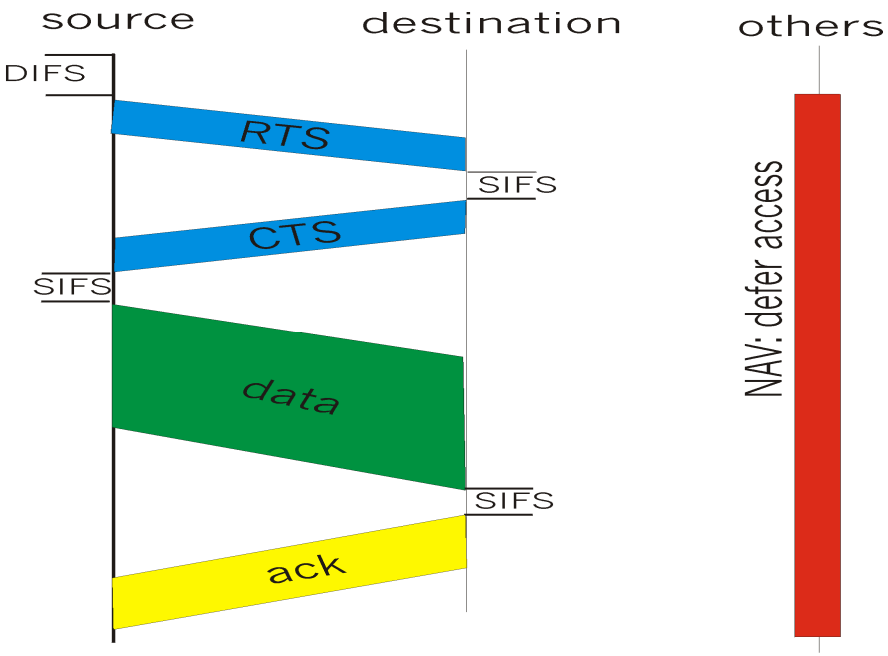






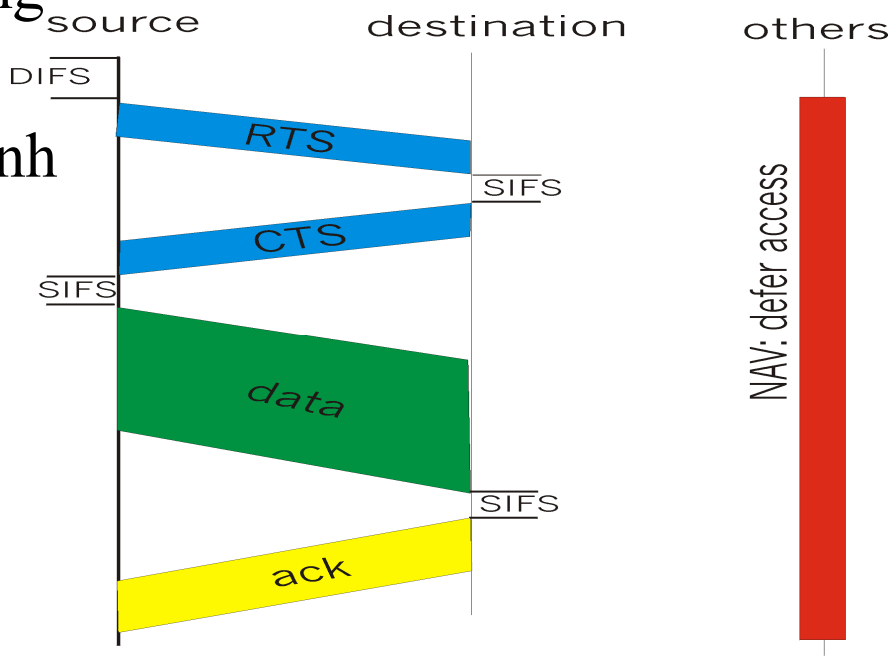
Collision Avoidance: Trao đổi RTS-CTS

- **CSMA/CA**: "đặt chỗ" sử dụng kênh truyền tường minh
  - **Gửi**: gửi yêu cầu RTS: request to send
  - **Nhận**: chấp nhận CTS: clear to send
- CTS đặt chỗ cho phía gửi, báo cho các trạm khác (có thể trạm ẩn)
- Tránh xung đột do trạm ẩn



Collision Avoidance: Trao đổi RTS-CTS

- RTS và CTS :
  - Giảm khả năng Xảy ra Xung đột
  - Kết quả tương ứng với Tránh xung đột



- IEEE 802.11 alows:
  - CSMA
  - CSMA/CA: “Đặt chỗ”
  - Thăm dò từ AP

Giao thức kiểu Điểm nối Điểm

- Một bên gửi, một bên nhận, một đường truyền:
  Đơn giản hơn Kênh truyền Quảng bá:
  - Không cần Media Access Control
  - Không cần địa chỉ MAC tường minh
  - Ví dụ : Đường dial up, đường ISDN

- Một vài giao thức Điểm-nối-Điểm tiêu biểu:
  - PPP (point-to-point protocol)
  - HDLC: High level data link control

PPP [RFC1557] : Yêu cầu Thiết kế

- Đặt packet vào frame: Gói gói tin của tầng Mạng vào frame của tầng Liên kết dữ liệu
  - Gửi *đồng thời* nhiều gói tin của các giao thức khác nhau ở tầng Mạng
- Trong suốt: Có thể chuyển bất kỳ khuôn dạng Dữ liệu nào
- Phát hiện Lỗi (Không phải Sửa)
- Kiểm soát kênh truyền: Phát hiện khi kênh truyền bị lỗi, báo lại cho tầng Mạng bên trên
- Thảo luận Địa chỉ tầng Mạng: Có thể xác định/ cấu hình địa chỉ IP cho các điểm đầu cuối

PPP : Không cần phải

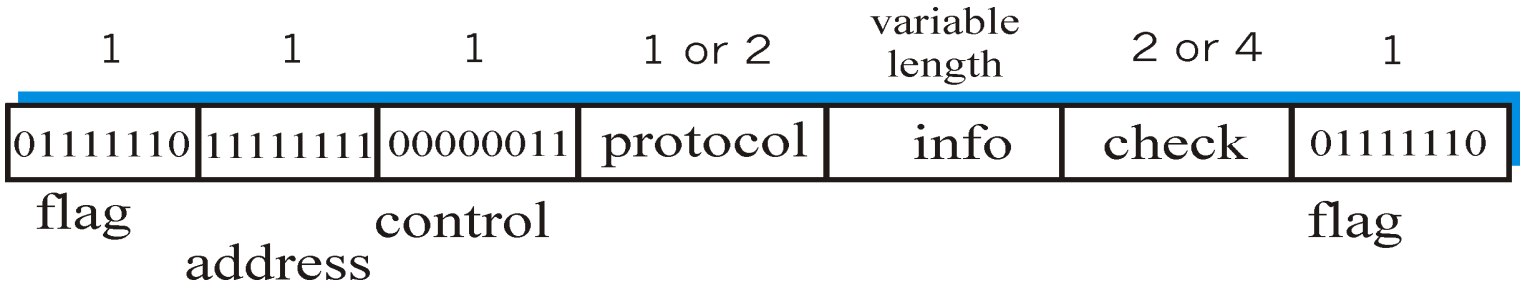
- Không có cơ chế Khắc phục / Sửa lỗi
- Không Điều khiển lưu lượng
- Không đảm bảo Gửi Nhận theo đúng thứ tự
- Không hỗ trợ đa điểm (Ví dụ thăm dò)

Khắc phục lỗi, Điều khiển lưu lượng, Thứ tự Gửi & Nhận Dữ liệu

Đã được xử lý ở các tầng trên !!

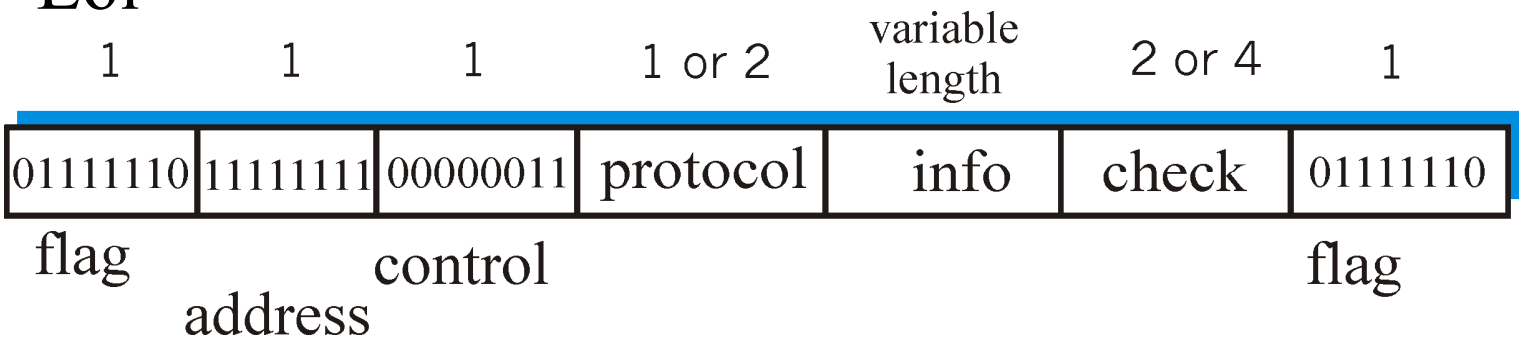
PPP Data Frame

- Flag: Đánh dấu (mở đầu khung)
- Address: Chưa sử dụng
- Control: Chưa sử dụng, có thể sử dụng trong tương lai
- Protocol: Giao thức ở tầng Mạng bên trên lấy Dữ liệu (Ví dụ PPP-LCP, IP, IPCP, …)



PPP Data Frame

- info: Chứa Dữ liệu của tầng trên
- check: Mã Kiểm tra Dư thừa vòng để Phát hiện Lỗi

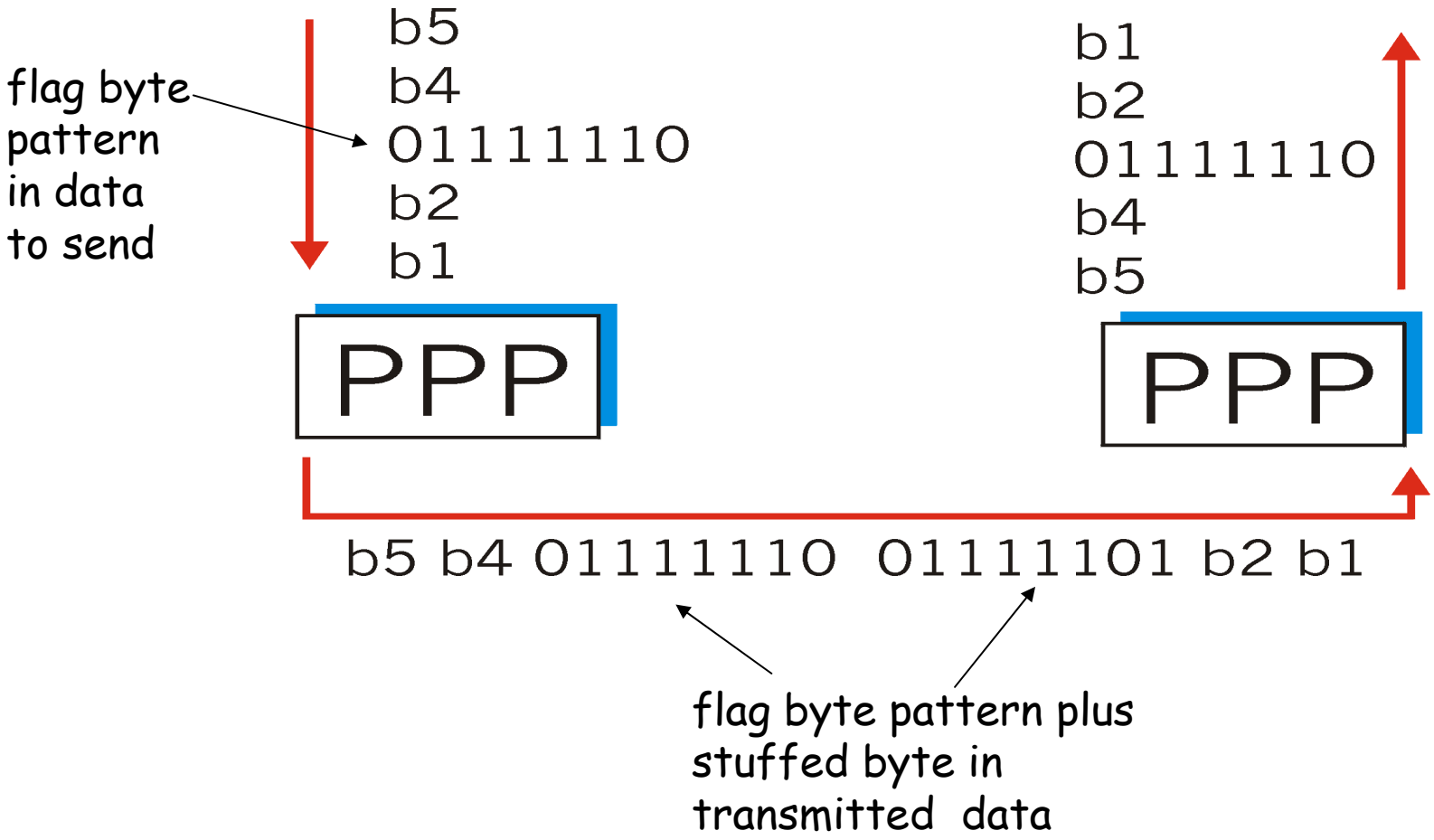


Chèn Byte

- Điều kiện "Trong suốt" : Mẫu cờ <01111110> phải có trong trường dữ liệu
  - Q: <01111110> nhận được là Dữ liệu hay Cờ?

- GỬI: Chèn thêm byte dư thừa < 01111110> sau mỗi byte < 01111110> *dữ liệu*
- NHẬN:
  - Hai byte 01111110 liên tiếp: Loại bỏ byte đầu tiên, tiếp tục nhận dữ liệu
  - Một byte 01111110: byte cờ

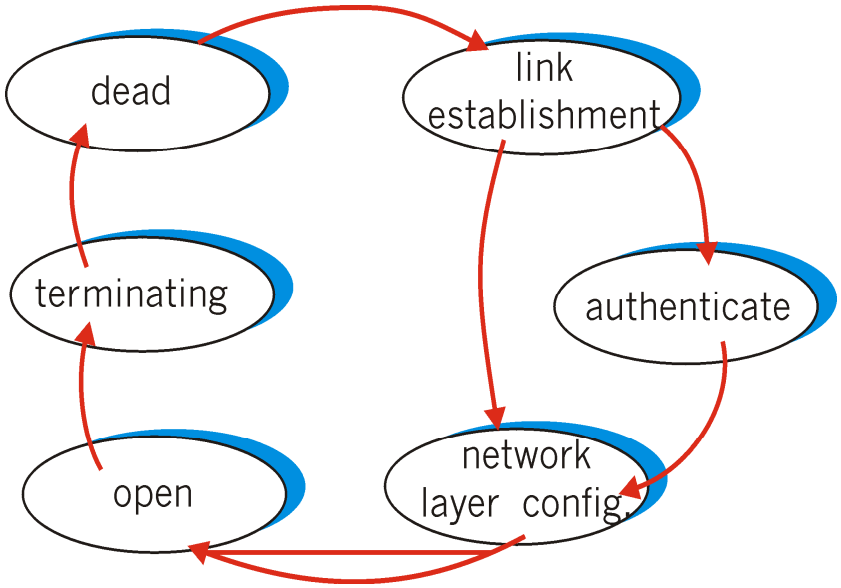
Chèn Byte : Ví dụ



PPP :Giao thức Kiểm soát Dữ liệu

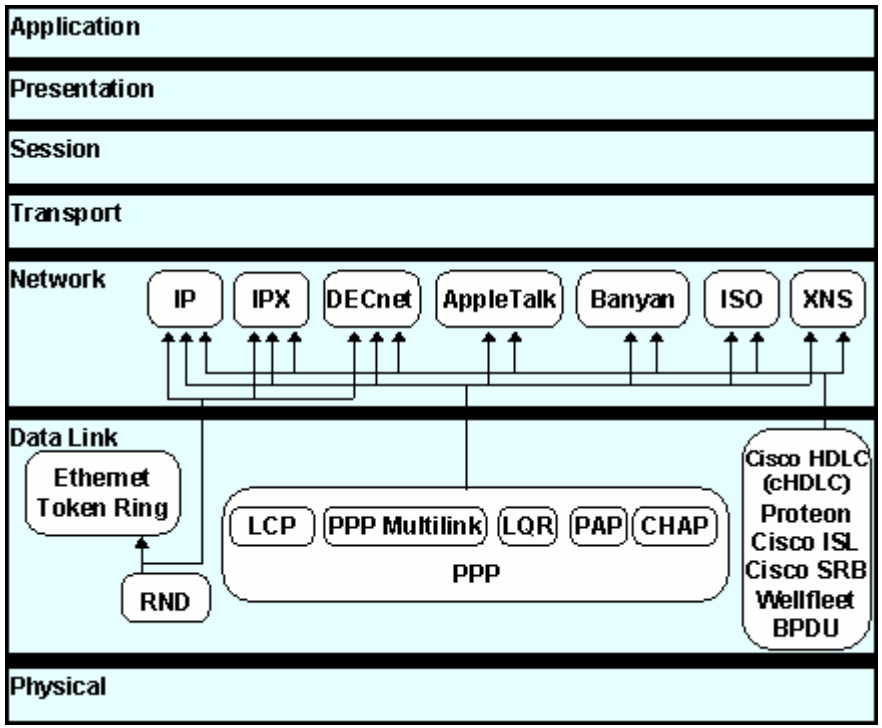
Trước khi trao đổi dữ liệu của tầng Mạng, các thực thể Liên kết dữ liệu phải :

- Cấu hình Đường truyền PPP (Kích thước frame cực đại, Kiểm chứng người dùng)
- Học/ Cấu hình các thông tin tầng Mạng
  - Với IP: mang thông điệp IP Control Protocol (IPCP) (mã giao thức: 8021) để cấu hình địa chỉ IP



Chương 5: Tổng Kết

- Các Nguyên lý Hoạt động của tầng Liên kết Dữ liệu:
  - Phát hiện và Sửa Lỗi
  - Đa truy cập : Sử dụng chung kênh truyền
  - Địa chỉ tầng link, ARP
- Các Công nghệ khác nhau:
  - Ethernet
  - hub, bridge, switche
  - IEEE 802.11 LAN
  - PPP



Hành trình qua các tầng đã *KẾT THÚC !*

Lời Cuối …