

HỒ ĐẶC PHƯƠNG

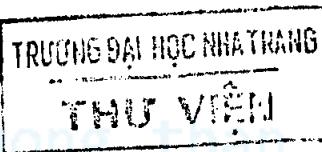
# GIÁO TRÌNH NHẬP MÔN MẠNG MÁY TÍNH



**HỒ ĐẮC PHƯƠNG**

**Giáo trình**

**NHẬP MÔN MẠNG MÁY TÍNH**



cuuduongthancong.com

**NHÀ XUẤT BẢN GIÁO DỤC VIỆT NAM**

# MỤC LỤC

<b>Chương 1. GIỚI THIỆU CHUNG .....</b>	<b>7</b>
1.1. MẠNG TRUYỀN THÔNG VÀ CÔNG NGHỆ MẠNG.....	7
1.1.1. Giới thiệu chung .....	7
1.1.2. Mạng máy tính .....	9
1.1.3. Phân loại mạng máy tính.....	12
1.1.4. Địa chỉ mạng, định tuyến, tính tin cậy, tính liên tác và an ninh mạng .....	15
1.1.5. Chuẩn mạng .....	17
1.2. MÔ HÌNH OSI.....	19
1.2.1. Mô hình.....	20
1.2.2. Chức năng các tầng .....	24
1.2.3. Bộ giao thức TCP/IP – Mô hình Internet .....	35
<b>Chương 2. TẦNG ỨNG DỤNG .....</b>	<b>37</b>
2.1. GIAO THỨC TẦNG ỨNG DỤNG.....	37
2.1.1. Giao thức tầng ứng dụng.....	38
2.1.2. Các yêu cầu của ứng dụng .....	41
2.1.3. Dịch vụ của các giao thức giao vận Internet .....	43
2.1.4. Một số ứng dụng phổ biến.....	46
2.2. WORLD WIDE WEB: HTTP .....	46
2.2.1. Tổng quan về HTTP .....	47
2.2.2. Kết nối liên tục và không liên tục .....	49
2.2.3. Khuôn dạng thông điệp HTTP .....	51
2.2.4. Tương tác giữa người dùng và HTTP server.....	55
2.2.5. GET có điều kiện .....	57
2.3. THƯ TÍN ĐIỆN TỬ (e-mail) TRÊN INTERNET.....	59
2.3.1. SMTP .....	60
2.3.2. So sánh SMTP với HTTP .....	63
2.3.3. Giao thức truy nhập mail .....	64

2.4. DỊCH VỤ TÊN MIỀN – DNS .....	69
2.4.1. Các dịch vụ của DNS.....	70
2.4.2. Cơ chế hoạt động của DNS.....	72
2.4.3. Bản ghi DNS .....	78
2.4.4. Thông điệp DNS .....	79
2.5. LẬP TRÌNH SOCKET .....	80
<b>Chương 3. TẦNG GIAO VẬN .....</b>	<b>86</b>
3.1. DỊCH VỤ VÀ NGUYỄN TẮC CỦA TẦNG GIAO VẬN .....	86
3.1.1. Quan hệ giữa tầng giao vận và tầng mạng .....	88
3.1.2. Tổng quan về tầng giao vận trong Internet.....	89
3.2. DỊCH VỤ DÒN KÊNH, PHÂN KÊNH.....	91
3.3. UDP – GIAO THỨC KHÔNG HƯỚNG NÓI .....	95
3.3.1. Cấu trúc UDP segment.....	99
3.3.2. UDP checksum .....	99
3.4. CÁC NGUYỄN TẮC TRUYỀN DỮ LIỆU TIN CÂY .....	100
3.4.1. Xây dựng giao thức truyền dữ liệu tin cậy.....	102
3.4.2. Giao thức truyền dữ liệu tin cậy liên tục .....	112
3.4.3. Go-back-N (GBN) .....	114
3.4.4. Giao thức lặp lại có lựa chọn.....	119
3.5. TCP – GIAO THỨC GIAO VẬN HƯỚNG NÓI.....	121
3.5.1. Kết nối TCP .....	121
3.5.2. Cấu trúc TCP Segment.....	123
3.5.3. Số thứ tự và số biên nhận .....	125
3.5.4. Truyền dữ liệu tin cậy .....	127
3.5.5. Kiểm soát lưu lượng .....	129
3.5.6. Quản lý kết nối TCP.....	131
3.6. KIỂM SOÁT TẮC NGHẼN TRONG TCP .....	135
<b>Chương 4. TẦNG MẠNG .....</b>	<b>139</b>
4.1. CÁC MÔ HÌNH DỊCH VỤ CỦA TẦNG MẠNG .....	139
4.1.1. Mô hình dịch vụ mạng .....	141
4.1.2. Nguồn gốc của dịch vụ chuyển mạch gói và chuyển mạch ảo.....	146
4.2. CÁC NGUYỄN LÝ ĐỊNH TUYẾN .....	147
4.2.1. Thuật toán định tuyến link state.....	150
4.2.2. Thuật toán Distance vector.....	154

4.3. ĐỊNH TUYẾN PHẦN CẤP .....	158
4.4. INTERNET PROTOCOL .....	161
4.4.1. Địa chỉ IPv4.....	163
4.4.2. Chuyển datagram từ nguồn tới đích: vấn đề địa chỉ và định tuyến .....	169
4.4.3. Khuôn dạng gói dữ liệu IP .....	172
4.4.4. Phân mảnh (Fragmentation) và hợp nhát (Reassembly) gói tin IP .....	175
4.4.5. Giao thức kiểm soát lỗi ICMP .....	178
4.5. ĐỊNH TUYẾN TRÊN INTERNET.....	180
4.5.1. Định tuyến trong một miền (Intra-AS routing) (Định tuyến nội miền).....	181
4.5.2. Định tuyến giữa các miền (Inter-AS routing) (Định tuyến liên miền).....	184
4.6. CẤU TẠO CỦA THIẾT BỊ ĐỊNH TUYẾN (ROUTER) .....	185
4.6.1. Cổng vào .....	186
4.6.2. Kết cấu chuyển .....	188
4.6.3. Cổng ra .....	189
4.6.4. Hàng đợi ở router .....	190
4.7. IPv6 .....	192
4.8. CƠ CHẾ DỊCH CHUYỂN ĐỊA CHỈ (NAT) .....	195
<b>Chương 5. TẦNG LIÊN KẾT DỮ LIỆU.....</b>	<b>201</b>
5.1. CÁC KHÁI NIỆM CHUNG, DỊCH VỤ CỦA TẦNG DATA LINK....	201
5.1.1. Những dịch vụ của tầng liên kết dữ liệu .....	202
5.1.2. Bộ điều hợp .....	205
5.2. KỸ THUẬT PHÁT HIỆN VÀ SỬA LỖI.....	207
5.2.1. Kiểm tra tính chẵn lẻ .....	208
5.2.2. Phương pháp tính tổng kiểm tra.....	210
5.2.3. Kiểm tra dư thừa vòng (CRC) .....	211
5.3. GIAO THỨC ĐA TRUY CẬP VÀ MẠNG CỤC BỘ .....	213
5.3.1. Giao thức phân chia kênh truyền .....	216
5.3.2. Giao thức truy cập ngẫu nhiên .....	218
5.3.3. Giao thức truy cập lần lượt .....	224
5.3.4. Mạng cục bộ LAN .....	225

5.4. ĐỊA CHỈ LAN VÀ ARP .....	227
5.4.1. Địa chỉ LAN .....	227
5.4.2. Giao thức giải mã địa chỉ (ARP) .....	229
5.5. ETHERNET .....	233
5.5.1. Những khái niệm cơ bản của Ethernet.....	234
5.5.2. CSMA/CD – Giao thức đa truy cập của Ethernet.....	238
5.5.3. Những công nghệ Ethernet .....	241
5.6. HUB, BRIDGE VÀ SWITCH .....	245
5.6.1. Hub .....	245
5.6.2. Bridge.....	247
5.6.3. Switch .....	256
5.7. MẠNG LAN KHÔNG DÂY .....	259
5.7.1. Giới thiệu chung .....	259
5.7.2. Lớp giao thức IEEE 802.11 .....	261
5.7.3. Một số vấn đề hay gặp đối với mạng không dây.....	264
5.8. PPP – GIAO THỨC ĐIỀM NỐI ĐIỀM .....	266
5.8.1. Khuôn dạng gói dữ liệu (Frame PPP) .....	268
5.8.2. Giao thức điều khiển đường truyền PPP (LCP) và kiểm soát mạng.....	271
TÀI LIỆU THAM KHẢO.....	274

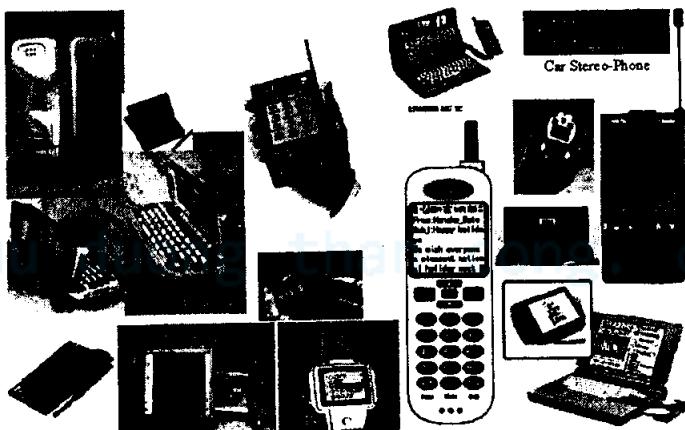
# **Chương 1**

## **GIỚI THIỆU CHUNG**

## **1.1. MẠNG TRUYỀN THÔNG VÀ CÔNG NGHỆ MẠNG**

### **1.1.1. Giới thiệu chung**

*Truyền thông máy tính* (computer communications) là quá trình truyền dữ liệu từ thiết bị này sang thiết bị khác. Trước đây, người ta thường hiểu thiết bị là máy tính, nhưng ngày nay thiết bị (end system, device) không chỉ đơn thuần là máy tính mà bao gồm nhiều chủng loại thiết bị khác, ví dụ: điện thoại di động, máy tính PAM,... (Hình 1.1). Số lượng các kiểu thiết bị có khả năng lấy thông tin từ Internet ngày càng tăng. Một từ phổ biến có nghĩa tương tự như vậy là *truyền dữ liệu*. Mặc dù hai cụm từ này có thể sử dụng thay thế cho nhau, một số người coi thuật ngữ *dữ liệu* (data) chỉ bao gồm những sự kiện đơn giản và thô (chưa được xử lý), sử dụng thuật ngữ *thông tin* (information) để chỉ việc tổ chức những sự kiện này thành dạng thông tin có nghĩa đối với con người.



Hình 1.1. Các thiết bị có khả năng kết nối Internet

Khái niệm *mạng* (networking) chỉ khái niệm kết nối các thiết bị lại với nhau, nhằm mục đích chia sẻ thông tin. Khái niệm mạng liên quan đến nhiều vấn đề, bao gồm:

- ☛ **Giao thức truyền thông** (protocol): Mô tả những nguyên tắc mà tất cả các *thành phần mạng* cần tuân thủ để có thể trao đổi được với nhau.
- ☛ **Topo (mô hình ghép nối mạng/hình trạng mạng)**: Mô tả cách thức nối các thiết bị với nhau.
- ☛ **Địa chỉ**: Mô tả cách thức định vị một đối tượng trên mạng.
- ☛ **Định tuyến** (routing): Mô tả cách thức dữ liệu truyền từ thiết bị này sang thiết bị khác trên mạng.
- ☛ **Tính tin cậy** (reliability): Giải quyết tính toàn vẹn của dữ liệu, đảm bảo dữ liệu nhận được chính xác như dữ liệu gửi đi.
- ☛ **Khả năng liên tác** (interoperability): Chỉ mức độ các sản phẩm phần mềm và phần cứng của các hãng sản xuất khác nhau có thể làm việc cùng nhau.
- ☛ **An ninh** (security): Đảm bảo an toàn, hoặc bảo vệ tất cả các thành phần của mạng.
- ☛ **Chuẩn** (standard): Thiết lập các quy tắc và luật lệ cụ thể cần phải tuân theo.

Trên thực tế, khái niệm mạng xuất hiện với nhiều kiểu ứng dụng khác nhau. Ví dụ, trong công nghiệp giải trí, các công ty truyền thanh, truyền hình,... đều có những mạng độc lập riêng của mình, với nhiều trạm phát. Thông qua những mạng này, các chương trình tin tức, thể thao, điện ảnh, phim truyện,... được dùng chung giữa các trạm phát. Mạng truyền thông ra đời sớm nhất và phổ biến nhất là mạng điện thoại. Khi nói đến mạng điện thoại, người ta muốn nhắc đến hệ thống điện thoại kiểu cũ (plain old telephone system – POTS), hoặc mạng điện thoại chuyển mạch công cộng (PSTN – public switched telephone network). Mạng PSTN mô tả hệ thống điện thoại truyền thông dựa trên tín hiệu tương tự, được sử dụng để truyền tiếng nói. Một mạng truyền thông khá quen thuộc ngày nay là mạng máy tính Internet – là một tập hợp các mạng, hay mạng mạng.

### 1.1.2. Mạng máy tính

Mạng bao gồm nhiều thành phần, các thành phần được nối với nhau theo một cách thức nào đó và cùng sử dụng chung một ngôn ngữ:

- ☞ **Các thiết bị đầu/cuối** (end system) kết nối với nhau tạo thành mạng có thể là các máy tính (computer), hoặc các thiết bị khác. Ngày càng có nhiều loại thiết bị có khả năng kết nối vào mạng máy tính như điện thoại di động, PDA, tivi,...
- ☞ **Môi trường truyền** (media) thực hiện việc truyền dẫn các tín hiệu vật lý. Môi trường truyền có thể là các loại dây dẫn (cáp), sóng (đối với các mạng không dây).
- ☞ **Giao thức** (protocol) là quy tắc quy định cách thức trao đổi dữ liệu giữa các thực thể.

Nói chung, ba khái niệm trên đưa đến một định nghĩa chuẩn về mạng máy tính như sau:

*Mạng máy tính là tập hợp các máy tính và các thiết bị phụ trợ khác sử dụng chung một nhóm giao thức để chia sẻ tài nguyên thông qua các phương tiện truyền thông mạng.*

#### a) Các thành phần mạng (thiết bị, nút, máy tính)

Theo nghĩa chung nhất, thuật ngữ *thiết bị* (device) chỉ bát cứ một thực thể phần cứng nào. Chẳng hạn, các thiết bị đầu/cuối, máy in, máy tính, hoặc một thiết bị phần cứng đặc biệt liên quan đến mạng, ví dụ: máy chủ (server), repeater (bộ lặp), bridge (cầu), switch, router (bộ định tuyến) và rất nhiều thiết bị đặc biệt khác. Tất cả các kiểu thiết bị này sẽ được trình bày ở các chương sau.

Nói chung, có nhiều phương pháp gán cho thiết bị mạng một định danh duy nhất, thường thì thiết bị được chính hãng sản xuất gắn một số nhận dạng duy nhất. Việc này tương tự như in số seri trên tivi, hoặc các đồ dùng điện tử khác. Ví dụ, mỗi card Ethernet được hãng sản xuất gán cho một địa chỉ duy nhất – địa chỉ này không trùng với bất kỳ card Ethernet nào khác.

Khi mô tả các thành phần mạng, cần phân biệt giữa khái niệm thiết bị (device) và máy tính (computer). Nếu xét ở khía cạnh thiết bị, máy tính thường được gọi là host (hoặc server), hay trạm làm việc (workstation)

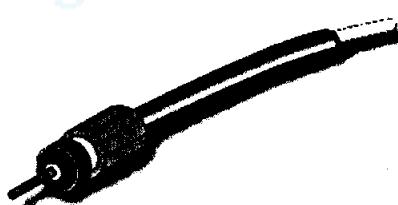
(cũng còn được gọi là desktop hay client). Thuật ngữ này thường dùng để chỉ những hệ thống máy tính có cài đặt hệ điều hành riêng của mình (ví dụ như Windows, UNIX). Vì vậy, workstation có thể là máy tính cá nhân như máy Apple Macintosh, hoặc bất cứ máy tính họ Intel nào (thường được gọi là IBM-PC); cũng có thể là một workstation đồ họa (ví dụ các workstation đồ họa được sản xuất bởi Sun Microsystems, Silicon Graphics, IBM, Hewlett-Packard, Compaq Computer Corporation), một superminicomputer như Compaq's VAX, hay một hệ thống IBM AS/400, một super-microcomputer như Compaq's Alpha; hoặc có thể là một máy tính lớn (mainframe) như IBM ES-9000.

### b) Phương tiện và các giao thức truyền thông mạng

Để chia sẻ thông tin và sử dụng dịch vụ trên mạng, các thành phần của mạng phải có khả năng truyền thông được với nhau. Để đáp ứng được yêu cầu này, người ta phải xét tới hai tiêu chí cụ thể của mạng như *khả năng liên kết* (connectivity) và *ngôn ngữ* (language). Khả năng liên kết chỉ đường truyền, hoặc kết nối vật lý giữa các thành phần; ngôn ngữ chỉ một bảng từ vựng cùng các quy tắc truyền thông mà các thành phần phải tuân thủ.

### c) Phương tiện truyền thông (media)

Môi trường vật lý sử dụng để kết nối các thành phần của mạng thường được gọi là *môi trường truyền thông* (medium, media). Môi trường truyền thông mạng được chia thành hai loại là *cáp* (cable) và *không dây* (wireless). Ví dụ, cáp truyền thông có thể là cáp xoắn đôi (twisted – pair), cáp đồng trục (coaxial) và cáp sợi quang (fiber – optic cable),... Truyền thông không dây có thể là sóng radio (gồm sóng cực ngắn, hay truyền thông qua vệ tinh), bức xạ hồng ngoại. Các môi trường truyền thông mạng được thảo luận chi tiết trong Chương 5.



Hình 1.2. Môi trường truyền, sợi cáp quang

#### d) Giao thức (Protocols)

Ngôn ngữ được sử dụng bởi các thực thể mạng, gọi là *giao thức truyền thông mạng*. Các bên truyền thông "hiểu nhau" do giao thức định nghĩa một ngôn ngữ chung giữa các thành phần mạng. Từ ý nghĩa khái quát này, có thể hiểu giao thức truyền thông mạng là các thủ tục, quy tắc, hoặc các đặc tả chính thức đã được chấp nhận, nhằm xác định hành vi và ngôn ngữ trao đổi giữa các bên. Nói chung, trong cuộc sống hàng ngày, người ta cũng áp dụng những quy tắc nào đó. Ví dụ, khi đi đến những nơi đòi hỏi tính trang trọng, mọi người phải tuân theo những nghi thức đặc biệt về ăn mặc. Nhưng khi đến các quán ăn bình dân, thì không cần ăn mặc trang trọng như vậy. Trong mạng và truyền thông máy tính, giao thức mạng là bản đặc tả chính thức quy định cách thức "ứng xử" của các thực thể tham gia truyền thông. Ở đây, khái niệm thực thể bao gồm cả thiết bị phần cứng cũng như tiến trình phần mềm. Giao thức mạng cũng định nghĩa khuôn dạng dữ liệu được trao đổi giữa các bên. Nói một cách ngắn gọn, giao thức mạng định nghĩa bằng từ vựng và các quy tắc áp dụng để truyền thông dữ liệu.

Không có môi trường truyền, không thể trao đổi thông tin giữa các thực thể mạng; không có ngôn ngữ chung, không thể hiểu được nhau. Vì vậy, đường truyền cung cấp môi trường để thực hiện truyền thông, trong khi đó, ngôn ngữ chung đảm bảo hai bên truyền thông hiểu được nhau. Điều này cũng giống như cuộc nói chuyện điện thoại giữa một người chỉ nói được tiếng Ý với một người chỉ nói được tiếng Nga. Nếu không có đường điện thoại (đường truyền mạng) cho cuộc nói chuyện, thì hai người không thể nói chuyện được với nhau (không thể trao đổi dữ liệu); khi đã có đường điện thoại rồi, hai người có thể nói và nghe thấy giọng nói của nhau (truyền dữ liệu được thực hiện), nhưng họ không giao tiếp được với nhau, vì không ai trong số họ hiểu được ngôn ngữ của người kia (họ nói chuyện bằng hai thứ tiếng khác nhau).

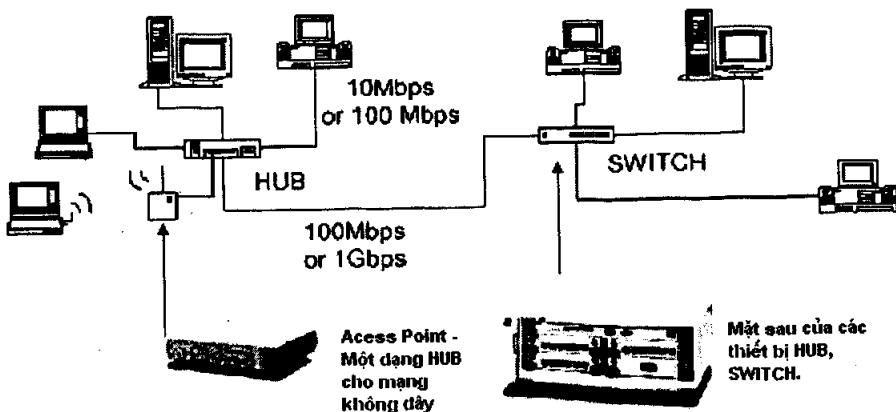
Ví dụ, TCP/IP là một giao thức mạng quen thuộc – một trong những giao thức của bộ giao thức TCP/IP (Transmission Control Protocol/Internet Protocol), TCP/IP được coi là xương sống của Internet. Tuy tên gọi TCP/IP chỉ gồm hai giao thức cụ thể là TCP và IP, nhưng thường được sử dụng để chỉ nhóm gồm nhiều giao thức khác ngoài TCP và IP. Tập hợp các giao thức này được gọi là *bộ giao thức TCP/IP*. Có thể kể đến một số giao thức trong

bộ giao thức TCP/IP như: FTP (Transfer Protocol) – định nghĩa cách chuyển file; HTTP (the Hypertext Transport Protocol) – dùng cho World Wide Web (WWW); định nghĩa cách các server cần phải truyền các tài liệu (trang Web) tới các client (Web browser) như thế nào? Ngoài ra, cũng phải kể đến ba giao thức được sử dụng cho thư điện tử (email) là Post Office Protocol (POP), Simple Mail Transfer Protocol (SMTP) và Internet Mail Access Protocol (IMAP). Ngày nay, các mạng sử dụng rất nhiều giao thức khác nhau từ đơn giản đến phức tạp. Có thể nói các giao thức là "keo dán" ràng buộc mạng máy tính lại với nhau, bởi vì chúng định nghĩa cách thực hiện các hoạt động cụ thể.

Một số bộ giao thức khác có thể kể đến là *AppleTalk* – bộ giao thức mạng của công ty Apple Computer, đầu tiên chỉ sử dụng cho các máy tính Macintosh, không thể phục vụ cho các hệ điều hành khác; bộ giao thức của hệ điều hành *Windows 2000* của Microsoft; *DECnet* của Digital Equipment Corporation (hiện giờ là Compad) – sử dụng giao thức mạng Digital Network Architecture (DNA). DECnet được thiết kế cho các máy VAX, hoặc Alpha chạy dưới hệ điều hành Open-VMS, hoặc cho các hệ thống DEC dùng hệ điều hành DEC trước đây, ví dụ như: RSX-11M, RT-11, RSTS/E cũng như một số hệ điều hành phổ biến khác như MS-DOS, Windows và một vài biến thể của Unix. Mạng máy tính đôi khi cũng được đặt tên theo giao thức chúng dùng. Ví dụ, một mạng gồm các thiết bị hỗ trợ Apple Talk thường được gọi là một mạng Apple Talk. Tương tự, mạng TCP/IP là mạng của các thiết bị được liên kết với nhau và sử dụng bộ giao thức TCP/IP để truyền thông.

### 1.1.3. Phân loại mạng máy tính

Có rất nhiều kiểu mạng máy tính khác nhau. Việc phân loại chúng thường dựa trên các đặc điểm chung. Ví dụ, mạng máy tính thường được phân loại theo vùng địa lý (diện hoạt động) (ví dụ: mạng cục bộ, mạng diện rộng,...); theo topo (mô hình ghép nối mạng) (ví dụ: điểm – điểm (point to point), hay broadcast); hoặc theo kiểu đường truyền thông mà mạng sử dụng.



Hình 1.3. Một mạng LAN đơn giản

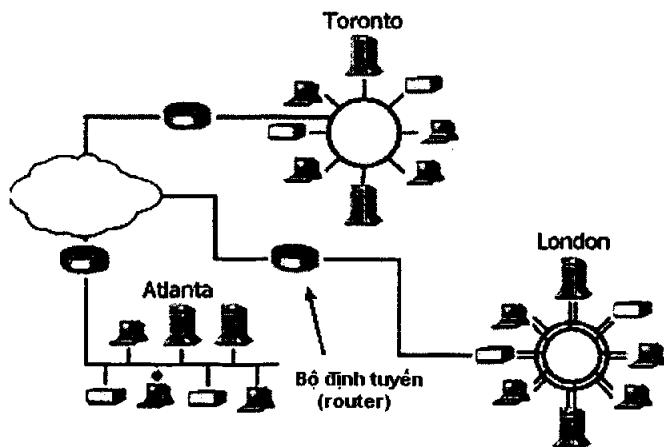
#### a) Phân loại mạng theo diện hoạt động

Nếu phân loại theo diện hoạt động, mạng máy tính có thể được phân chia thành:

- Mạng cục bộ (Local Area Network – LAN);
  - Mạng diện rộng (Wide Area network – WAN);
  - Mạng đô thị (Metropolitan Area network – MAN);
  - Mạng toàn cầu (Global Area network – GAN);
  - Mạng cá nhân (Personal Area network – PAN);
  - Mạng lưu trữ (Storage Area Network – SAN).
- ☞ **Mạng cục bộ (LAN)** liên kết các tài nguyên máy tính trong một vùng địa lý có kích thước hạn chế. Đó có thể là một phòng, vài phòng trong một tòa nhà, hoặc vài tòa nhà trong một khu nhà. Cụm từ "kích thước hạn chế" không được xác định cụ thể, nên người ta xác định phạm vi của mạng LAN bằng cách định bán kính nằm trong khoảng vài chục mét đến vài kilômét. IEEE (Institute of Electrical and Electronics Engineers) xác định bán kính của mạng LAN nhỏ hơn 10km. Ví dụ, một số công nghệ mạng LAN như Ethernet/802.3, Token Ring, mạng FDDI (Fiber Distributed Data Interface).

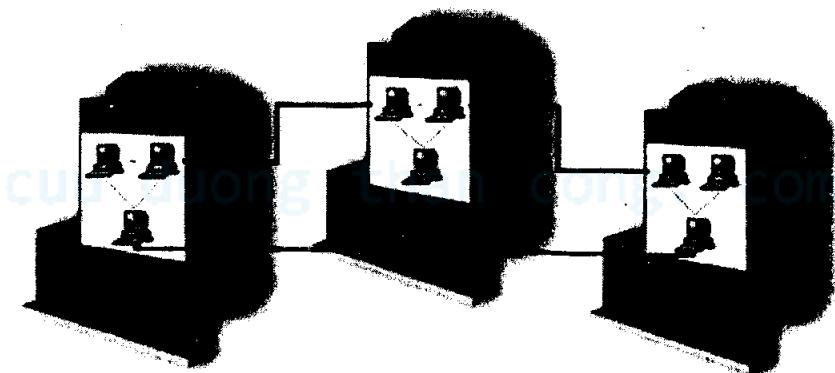
- ☞ **Mạng diện rộng (WAN)** liên kết các tài nguyên máy tính trong một vùng địa lý rộng (có bán kính trên 100km) như thị xã, thành phố, tỉnh/bang, quốc gia. Có thể coi mạng WAN gồm nhiều mạng

LAN khác nhau. Ví dụ một số công nghệ mạng WAN như ISDN (Integrated Services Data Network) frame relay, SMDS (Switched Multimegabit Data Service) và ATM (Asynchronous Transfer Mode).



Hình 1.4. Mạng WAN – Kết hợp của nhiều mạng LAN qua các router

Người ta còn phân biệt rõ hơn giữa mạng LAN và WAN, còn có **Mạng đô thị (MAN)**. MAN liên kết các tài nguyên máy tính trong vùng lãnh thổ (tỉnh, thành phố). Giả sử có một công ty kinh doanh có nhiều toà nhà trong tỉnh/thành phố. Mỗi toà nhà có một mạng LAN riêng, những mạng LAN này được kết nối với nhau, kết quả ta có một mạng MAN, (vì tất cả các toà nhà là ở trong cùng một tỉnh/thành phố). Nhìn chung, MAN được dùng để chỉ các mạng có diện hoạt động lớn hơn LAN, nhưng nhỏ hơn WAN.



Hình 1.5. Mạng MAN – Kết hợp nhiều mạng LAN trong một khu vực địa lý

Một loại mạng nữa là **Mạng cá nhân (PAN)**, chỉ mạng máy tính nhỏ sử dụng trong gia đình. Giá máy tính ngày càng rẻ, làm cho số gia đình có nhiều máy tính ngày càng tăng nhanh, dẫn đến nhu cầu xuất hiện mạng PAN (vì người ta nhận ra ưu điểm của việc kết nối các máy tính trong gia đình). Ví dụ, có thể nối các máy tính trong nhà đến cùng một máy in, không cần phải mua máy in cho mỗi máy tính. PAN cũng cho phép cả gia đình sử dụng một máy làm server, chứa tất cả phần mềm ứng dụng và dữ liệu người dùng; có thể truy cập đến server này từ bất cứ máy nào nối với mạng gia đình. PAN cũng giúp các thành viên trong gia đình truy cập đến bất cứ tài nguyên nào được dùng chung trong gia đình từ các vị trí khác nhau trong nhà.

- ☞ **Mạng toàn cầu (GAN)** là mạng của các mạng WAN, trải rộng trên phạm vi toàn cầu. Ví dụ, nhiều công ty xuyên quốc gia hoạt động trên nhiều nước trên thế giới. Việc kết nối mạng của các công ty con lại với nhau tạo thành mạng GAN. Mạng toàn cầu Internet cũng là một mạng GAN đặc biệt.

#### b) *Phân loại mạng theo mô hình ghép nối*

Một cách khác để phân loại mạng là theo topo – mô hình ghép nối mạng, hay còn gọi là *hình trạng mạng*. Topo mạng gần giống như bản đồ đường phố. Nó mô tả chi tiết cách thức kết nối các thành phần chính của mạng (các nút) và các đường truyền. Có thể so sánh topo mạng với bản thiết kế của một ngôi nhà, trong đó hệ thống điện, sưởi, điều hoà và nước được tích hợp với nhau trong một thiết kế chung nhất, hoàn chỉnh. Có 3 chiến lược kết nối tổng quát: điểm – điểm (point to point), broadcast (điểm – nhiều điểm) và multidrop (đa chặng).

### 1.1.4. Địa chỉ mạng, định tuyến, tính tin cậy, tính liên tác và an ninh mạng

Khái niệm mạng máy tính quan đến nhiều yếu tố, trong đó có địa chỉ, định tuyến, tính tin cậy, tính liên tác và an ninh mạng. Dưới đây trình bày sơ qua về những yếu tố này.

### **a) Địa chỉ (Address)**

Khái niệm *địa chỉ* liên quan đến việc gán cho mỗi nút mạng một địa chỉ duy nhất – cho phép các thiết bị khác định vị được nó. Điều này giống như địa chỉ của một ngôi nhà – tên phố sẽ chỉ cho biết khu vực cần đi đến, số nhà xác định chính xác nhà cần đến. Một ví dụ khác là hệ thống điện thoại, mỗi điện thoại có mã vùng và một số "địa chỉ". Mã vùng cung cấp thông tin về vị trí vùng của điện thoại, còn số điện thoại là số xác định duy nhất máy điện thoại trong vùng. Hệ thống các thiết bị chuyển mạch trong công ty điện thoại được lập trình để tạo nên một kênh truyền giữa hai thiết bị. Về thực chất, mã vùng lại được phân cấp thành mã quốc gia và mã khu vực.

### **b) Routing – định tuyến**

*Định tuyến* xác định tuyến đường mà dữ liệu sẽ đi qua trong quá trình chuyển từ nút nhận đến nút gửi. Chức năng định tuyến được thực hiện bởi một thiết bị phần cứng đặc biệt là router (thiết bị định tuyến). Việc lựa chọn tuyến đường tốt nhất phải dựa trên một tiêu chuẩn cụ thể – được gọi là *độ đo* (metric). Các độ đo định tuyến phổ biến là khoảng cách, số chặng (hop) và băng thông.

### **c) Tính tin cậy**

*Tính tin cậy* chỉ tính toàn vẹn dữ liệu – đảm bảo dữ liệu nhận được giống hệt dữ liệu gửi đi. Mạng máy tính không phải là hệ thống không có lỗi. Trong thực tế, lỗi có thể xuất hiện trên tất cả các môi trường truyền. Vì vậy, cần phải thiết kế sao cho hệ thống có khả năng xử lý lỗi. Một trong những phương pháp điển hình là thêm thông tin dư thừa vào dữ liệu chuyển đi, sao cho phía nhận phát hiện được lỗi (nếu có). Khi phát hiện ra lỗi, phía nhận có thể: (1) yêu cầu truyền lại dữ liệu bị lỗi, hoặc (2) kiểm tra xem dữ liệu đúng là gì và sửa lại dữ liệu bị lỗi. Cách thứ nhất là sửa lỗi bằng cách yêu cầu truyền lại, cách thứ hai là tự sửa lỗi. Để sửa được lỗi, phải dò tìm lỗi. Việc tự sửa lỗi nói chung khó thực hiện. Hầu hết các mạng ngày nay đều được thiết kế có khả năng phát hiện lỗi (error detection). Có hai cách để phát hiện lỗi thông dụng là kiểm tra bit chẵn/lẻ và mã dư thừa vòng (CRC – Cyclic Redundancy Check). Hai kỹ thuật này được trình bày trong Chương 5.

#### **d) Tính liên tác (interoperability)**

Tính liên tác (interoperability) chỉ khả năng các sản phẩm (phần cứng và phần mềm) của các hãng sản xuất khác nhau có thể giao tiếp được với nhau trong mạng. Trong thời kỳ hoàng kim của các mạng độc quyền (của tư nhân, hãng sản xuất, hoặc một tổ chức), không cần phải quan tâm đến tính liên tác, miễn là các thành phần cấu thành mạng đều là sản phẩm và giao thức của cùng một hãng sản xuất. Khi hãng sản xuất thứ ba phát triển ứng dụng có tính năng được cải tiến hơn ứng dụng của hãng sản xuất độc quyền, hãng sản xuất thứ ba phải được sự đồng ý của nhà sản xuất độc quyền – tức là, hãng sản xuất thứ ba phải trả phí bản quyền. Ngày nay, với bộ giao thức "mở" TCP/IP, các hãng sản xuất – những người viết và bán các ứng dụng dựa trên TCP/IP được tự do làm những điều họ muốn, không phải lo ngại về việc vi phạm bản quyền. Hầu hết các hãng sản xuất máy tính đều cố gắng để sản phẩm của mình tương thích với sản phẩm của các hãng sản xuất khác.

#### **e) An ninh**

An ninh mạng chỉ việc bảo vệ mọi thứ trong mạng, bao gồm dữ liệu, phương tiện truyền thông và các thiết bị. An ninh mạng còn bao gồm các chức năng quản trị, các công cụ kỹ thuật và thiết bị như các sản phẩm mã hoá, các sản phẩm kiểm soát truy cập mạng (ví dụ: tường lửa firewall – thiết bị phần cứng đặc biệt bảo vệ mạng với thế giới bên ngoài). An ninh mạng cũng bao gồm việc định ra những chính sách sử dụng tài nguyên mạng, kiểm tra xem tài nguyên mạng có được sử dụng phù hợp với chính sách đã định trước hay không, quy định và kiểm tra chỉ những người có đủ quyền mới được sử dụng các tài nguyên đó,...

### **1.1.5. Chuẩn mạng**

Chuẩn mạng định nghĩa các giao tiếp phần cứng, giao thức truyền thông, kiến trúc mạng,... Chuẩn mạng thiết lập những quy tắc hay các quy ước cụ thể mà các bên tham gia truyền thông cần tuân thủ. Chúng làm tăng khả năng giao tiếp giữa sản phẩm phần cứng và phần mềm của các hãng sản xuất khác nhau. Chuẩn được xây dựng thông qua các tổ chức chuẩn hoá.

Những tổ chức này được chia thành bốn loại chính: (a) quốc gia, (b) vùng, (c) quốc tế và (d) ngành/hiệp hội thương mại/hiệp hội nghề. Thành viên của tổ chức chuẩn thường là đại diện của chính phủ, viện nghiên cứu và hãng sản xuất. Quá trình xây dựng chuẩn phải đảm bảo được tính thống nhất, vì vậy thường kéo dài, đôi khi phải mất nhiều năm mới cho ra đời được một chuẩn chính thức. Quá trình này cũng bị ảnh hưởng bởi các yếu tố khác như kinh tế, chính trị.

#### **a) Chuẩn chính thức (*De jure standard*)**

*Chuẩn chính thức* được công nhận bởi những tổ chức chuẩn hóa chuyên nghiệp. Ví dụ, những giao thức về modem được xây dựng bởi Hiệp hội Truyền thông quốc tế (International Telecommunications Union – ITU), hay chuẩn EIA/TIA-568 dùng cho Commercial Building Telecommunications Wiring được xây dựng bởi Electronic Industries Association – EIA và Telecommunications Industries Association – TIA, hoặc các chuẩn cho mạng cục bộ được xây dựng bởi Institute for Electrical and Electronic Engineers – IEEE (Các chuẩn này sẽ được trình bày chi tiết trong các chương sau).

#### **b) Chuẩn thực tế (*De facto standard*)**

*Chuẩn thực tế* là chuẩn tồn tại trong thực tế, chứ không phải do các tổ chức chuẩn hóa xây dựng nên. Chúng được phát triển thông qua sự chấp nhận của toàn ngành đối với chuẩn nào đó của một hãng sản xuất cụ thể. Ví dụ, một chuẩn thực tế là Network File System (NFS) – giao thức chia sẻ file của hãng Sun Microsystems. Sun đã công khai đặc tả của giao thức này, do đó những nhà sản xuất khác có thể tự do triển khai. Kết quả NFS được sử dụng rộng rãi và được coi như một chuẩn thực tế. Hiện tại, NFS được cài đặt trên rất nhiều hệ thống UNIX khác nhau (Sun, IBM, Silicon Graphics, Compaq, và HP), cũng như các hệ thống dựa trên Macintosh và Intel. Một chuẩn thực tế khác là Java – ngôn ngữ lập trình Web được phát triển bởi hãng Sun Microsystems.

#### **c) Chuẩn của hãng**

*Chuẩn của hãng* quy định những yêu cầu cụ thể của một nhà sản xuất nào đó. Những đặc tả này không được công khai, chỉ được tuân theo và chấp

nhận bởi chính hãng sản xuất đề nghị ra nó. Trong thời kỳ đầu của mạng, các chuẩn của hãng thống trị. Mặc dù ngày nay những chuẩn như vậy không còn được tán thành nữa, song chúng vẫn tồn tại rất nhiều. Được biết đến nhiều nhất phải kể đến các chuẩn của IBM (ví dụ: SNA – kiến trúc hệ thống mạng của IBM, giao thức IPX của Novell – dựa trên giao thức XNS của Xerox). Chuẩn riêng của hãng trói buộc khách hàng vào giải pháp của một nhà sản xuất cụ thể, làm cho họ gặp khó khăn khi sử dụng sản phẩm (phần cứng hoặc phần mềm) của các hãng sản xuất khác.

#### *d) Chuẩn hiệp hội*

*Chuẩn hiệp hội* tương tự như chuẩn chính thức theo nghĩa chúng là sản phẩm của quá trình chuẩn hoá. Điểm khác nhau là quá trình lập kế hoạch và chuẩn hoá những chuẩn này không chịu sự quản lý của các tổ chức chuẩn hoá chuyên nghiệp. Thay vào đó, đặc tả cho các chuẩn được thiết kế và thỏa thuận bởi nhóm các nhà sản xuất thành lập nên hiệp hội, với một mục đích cụ thể để đạt được mục tiêu chung. Những nhà sản xuất này cam kết hỗ trợ cho các chuẩn được phát triển bởi hiệp hội và phát triển những sản phẩm tuân theo chuẩn này. Ví dụ, về các chuẩn hiệp hội như Fast Ethernet, Asynchronous Transfer Mode (ATM Forum) hay Gigabit Ethernet.

## **1.2. MÔ HÌNH OSI**

Tổ chức Tiêu chuẩn quốc tế ISO (International Standards Organization) được thành lập năm 1971, với mục đích xây dựng các tiêu chuẩn quốc tế. Một trong các chuẩn được ISO công nhận các mặt của truyền thông mạng là mô hình OSI – Open Systems Interconnection (Mô hình liên kết giữa các hệ thống mở). Đây là mô hình cho phép bất cứ hai hệ thống nào (cho dù khác nhau) cũng có thể truyền thông với nhau, mà không cần quan tâm đến kiến trúc bên dưới của chúng. Các giao thức của riêng một hãng sản xuất thường ngăn ngừa việc truyền thông giữa hai hệ thống không cùng một kiểu. Mô hình OSI ra đời với mục đích cho phép hai hệ thống bất kỳ truyền thông với nhau mà không cần thay đổi bất cứ phần cứng hoặc phần mềm nào bên dưới. Mô hình OSI không phải là một giao thức, nó là một mô hình để nhận biết và thiết kế một kiến trúc mạng linh động, vững chắc và có khả năng liên tác. (Chú ý: ISO là tên tổ chức; OSI là một mô hình).

### 1.2.1. Mô hình

Mô hình OSI được phân tầng với mục đích thiết kế các hệ thống mạng, cho phép tất cả các kiểu hệ thống máy tính khác nhau có thể truyền thông với nhau. Mô hình gồm 7 tầng riêng biệt, nhưng có liên quan đến nhau, mỗi tầng định nghĩa một phần của quá trình truyền thông tin trên mạng. Những quy tắc cơ bản của mô hình OSI là nền tảng cơ bản để nghiên cứu chi tiết truyền thông dữ liệu.

Thực ra trong cuộc sống, con người gặp khá nhiều ví dụ về việc phân tầng. Giả sử người A viết thư gửi cho người B; sau khi viết thư xong, A cho thư vào phong bì, dán kín, ghi địa chỉ của B, dán tem và cho bức thư vào hộp thư ở bưu điện. Giữa A và B, đơn vị dữ liệu trao đổi là các lá thư. Bức thư có thể xem là dữ liệu thực sự trong khi phong bì thư có thể xem là một loại tiêu đề chứa các thông tin điều khiển. Hệ thống bưu điện (bao gồm nhiều bưu cục – là các trạm trung gian mà bức thư sẽ đi qua) chịu trách nhiệm chuyển bức thư tới địa chỉ của B. Với ví dụ này, tầng dưới (hệ thống bưu điện) sẽ cung cấp dịch vụ chuyển thư cho tầng trên (A và B). A và B chỉ quan tâm đến nội dung bức thư, khuôn dạng thư, ngôn ngữ viết trong thư,... mà không cần quan tâm đến làm thế nào để thư có thể chuyển tới B. Đây chính là ưu điểm của việc phân tầng (tầng trên sử dụng dịch vụ của tầng dưới, nhưng không cần quan tâm đến cách thức thực hiện dịch vụ đó).

#### a) Kiến trúc phân tầng

Mô hình OSI gồm 7 tầng (Hình 1.6):

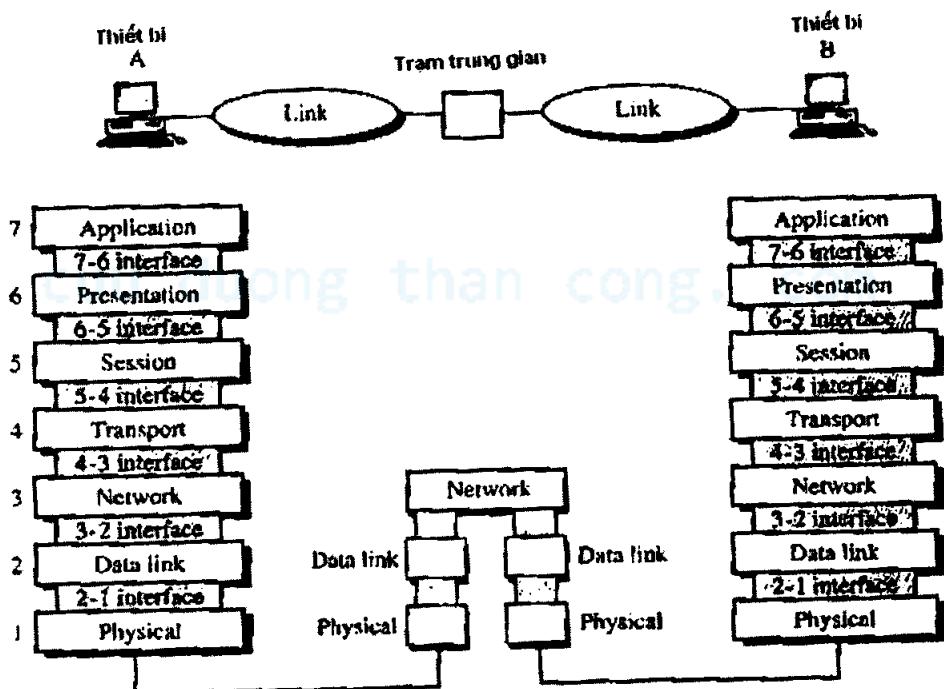
- Tầng vật lý (Physical layer);
- Tầng liên kết dữ liệu (Data Link layer);
- Tầng mạng (Network layer);
- Tầng giao vận (Transport layer);
- Tầng phiên (Session layer);
- Tầng trình diễn (Presentation layer);
- Tầng ứng dụng (Application layer).

Hình 1.7 minh họa mối quan hệ giữa các tầng khi thông điệp được gửi từ thiết bị A đến thiết bị



Hình 1.6. 7 tầng  
trong mô hình OSI

B. Khi đi từ A đến B, thông điệp có thể đi qua nhiều nút trung gian khác. Những nút trung gian này thường chỉ liên quan đến 3 tầng đầu của mô hình OSI. Khi phát triển mô hình, các nhà thiết kế đã phân tích quá trình truyền dữ liệu thành những chức năng cơ bản nhất, sau đó nhóm những chức năng có mục đích (sử dụng) liên quan đến nhau vào các nhóm riêng – còn gọi là *tầng* (layer). Mỗi tầng đều có chức năng, nhiệm vụ xác định. Bằng cách xác định và khoanh vùng các chức năng trong mô hình, nhà thiết kế đã đưa ra một kiến trúc đạt được cả tính toàn diện và linh hoạt. Quan trọng nhất, mô hình OSI tạo ra tính trong suốt hoàn toàn giữa hai hệ thống không tương thích với nhau.



Hình 1.7. Mô hình OSI

### b) Các tiến trình ngang hàng (*peer - to - peer*)

Tại thiết bị đầu/cuối, mỗi tầng sử dụng các dịch vụ do tầng bên dưới cung cấp. Ví dụ, tầng 3 sử dụng các dịch vụ do tầng 2 cung cấp và đến lượt mình lại cung cấp dịch vụ cho tầng 4. Giữa các máy tính, tầng N trên một thiết bị giao tiếp với tầng N trên thiết bị khác. Việc giao tiếp này được tiến hành theo các quy tắc và quy ước đã được thoả thuận trước – gọi là *giao thức*.

Tại tầng vật lý, truyền thông diễn ra trực tiếp (A gửi một luồng bit đến B dưới dạng tín hiệu). Tuy nhiên, tại các tầng cao hơn trên máy A, dữ liệu được chuyển dần xuống các tầng bên dưới, đến máy B và tiếp tục đi ngược lên các tầng cao hơn (của B). Mỗi tầng trong máy gửi (A) thêm các thông tin của mình vào thông điệp nhận được từ phía trên, rồi sau đó chuyển toàn bộ gói dữ liệu xuống tầng phía dưới. Các thông tin được thêm vào này – được gọi là *header* (tiêu đề chèn trước) và *trailer* (tiêu đề chèn sau) là các thông tin điều khiển được thêm vào đầu hay cuối gói dữ liệu. Header được thêm vào thông điệp tại mỗi tầng 6, 5, 4, 3 và 2; trailer được thêm vào tại tầng 2.

Tại tầng 1, gói dữ liệu được chuyển thành dạng tín hiệu có thể truyền đi tới máy nhận. Tại thiết bị nhận, các tiêu đề được lấy ra dần dần trong quá trình chuyển dữ liệu lên trên. Ví dụ, tầng 2 loại bỏ các tiêu đề của tầng 2 và chuyển phần còn lại (dữ liệu) cho tầng 3. Tầng 3 loại bỏ các tiêu đề tầng 3 và chuyển phần dữ liệu cho tầng 4,...

#### c) Giao diện giữa các tầng

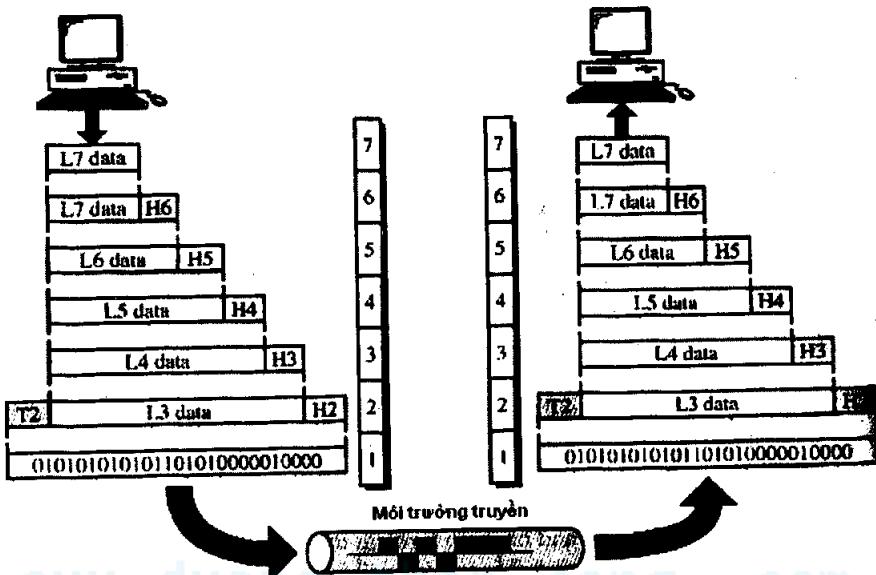
Trên cùng một máy tính, hai tầng kề nhau trao đổi dữ liệu với nhau qua các giao diện (interface). Giao diện định nghĩa cách thức và khuôn dạng dữ liệu trao đổi giữa hai tầng kề nhau trên cùng một thiết bị. Định nghĩa giao diện giữa các tầng một cách rõ ràng cho phép thay đổi cách thức triển khai tại một tầng, mà không ảnh hưởng đến các tầng khác.

Trong thuật ngữ mạng, người ta thường gọi giao diện giữa các tầng là *điểm truy cập dịch vụ* (Service Access Point – SAP), vì tầng trên yêu cầu dịch vụ của tầng dưới thông qua giao diện.

#### d) Tổ chức các tầng

Có thể chia bảy tầng vào ba nhóm. Nhóm *tầng hỗ trợ mạng* gồm ba tầng là vật lý, liên kết dữ liệu và mạng chịu trách nhiệm về các mặt liên quan đến khía cạnh vật lý khi truyền dữ liệu từ thiết bị này sang thiết bị khác (ví dụ: đặc tả điện, các kết nối vật lý, định địa chỉ vật lý, định thời gian truyền và tính tin cậy). Tầng phiên trình diễn và ứng dụng thuộc nhóm tầng hỗ trợ người dùng; chúng tạo ra khả năng liên tác giữa các hệ thống phần mềm khác nhau. Tầng 4 – tầng giao vận, đảm bảo việc chuyển dữ liệu đầu/cuối (end – to – end) tin cậy, trong khi tầng 2 đảm bảo việc truyền dữ

liệu tin cậy trên một đường truyền vật lý riêng lẻ. Nói chung, các tầng trên của mô hình OSI thường được triển khai qua phần mềm, trong khi nhóm các tầng dưới được triển khai bằng sự kết hợp của cả phần cứng lẫn phần mềm. Tầng vật lý hầu như được triển khai bởi phần cứng.



Hình 1.8. Dữ liệu được chuyển dọc theo các tầng đi xuống phía dưới

Hình 1.8 mô tả tổng quan các tầng trong mô hình OSI. Trong hình vẽ, (dữ liệu) L7 là đơn vị dữ liệu tại tầng 7, (dữ liệu) L6 là đơn vị dữ liệu tại tầng 6,... Trong thuật ngữ mạng, người ta gọi Li là đơn vị dữ liệu giao thức tầng i (iPDU – Protocol Data Unit). Đối với tầng thứ i + 1, dữ liệu tầng i truyền cho được gọi là đơn vị dữ liệu dịch vụ (service data unit – SAP). Nói chung, PDU chứa SAP và một số thông tin tiêu đề khác. Quá trình được bắt đầu tại tầng 7 (tầng ứng dụng), sau đó chuyển xuống các tầng dưới. Tại mỗi tầng, ngoại trừ tầng 7 và tầng 1, header được chèn thêm vào đơn vị dữ liệu. Tại tầng 2, ngoài header, trailer cũng được thêm vào đơn vị dữ liệu. Khi đi qua tầng vật lý (tầng 1), đơn vị dữ liệu (đã định khuôn dạng) được chuyển thành tín hiệu vật lý và truyền đi qua môi trường vật lý.

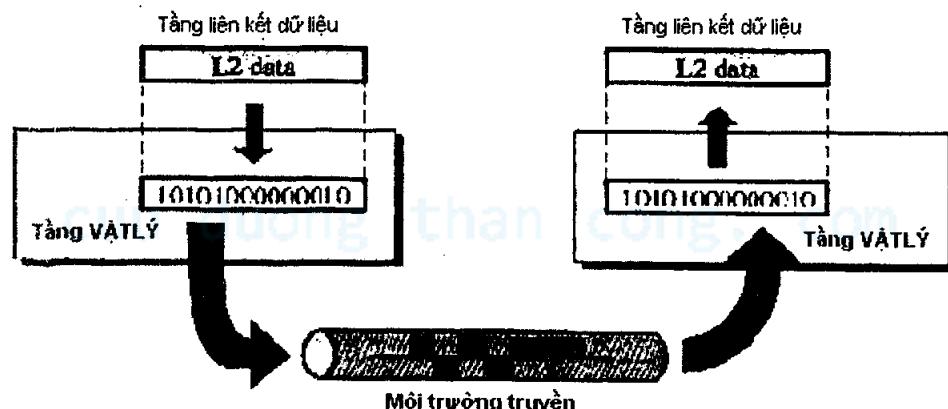
Tại nơi nhận, tín hiệu vật lý đến tầng 1 và được chuyển ngược lại thành chuỗi bit. Các đơn vị dữ liệu sau đó sẽ được chuyển dần từ tầng 1 lên các tầng trên trong mô hình OSI. Tại mỗi tầng, các header và trailer được thêm

vào khối dữ liệu ở tầng tương ứng bên máy gửi được lấy ra. Khi đến tầng 7, thông điệp đã ở dạng dữ liệu phù hợp và ứng dụng có thể sử dụng.

### 1.2.2. Chức năng các tầng

Trong phần này sẽ mô tả chi tiết chức năng của từng tầng trong mô hình OSI.

Tầng vật lý thực hiện các chức năng cần thiết để truyền luồng bit dữ liệu đi qua môi trường vật lý. Nó giải quyết những vấn đề liên quan đến đặc điểm kỹ thuật về cơ và điện giữa card ghép nối (interface) với môi trường truyền dẫn. Nó cũng xác định các thủ tục, chức năng mà thiết bị vật lý và thiết bị giao tiếp cần phải tuân thủ. Hình 1.9 minh họa mối quan hệ giữa tầng vật lý với môi trường truyền dẫn và tầng liên kết dữ liệu.



Hình 1.9. Vị trí, vai trò của tầng vật lý

#### a) Tầng vật lý

Trong ví dụ chuyển thư, tầng vật lý liên quan đến công nghệ chuyển thư, chẳng hạn như xe đạp, máy bay, tàu hỏa, tàu thuỷ,... dùng chuyên chở các túi thư. Tầng liên kết dữ liệu chuyển lá thư cho tầng vật lý và hy vọng tầng vật lý chuyển lá thư sang phía bên kia của kênh truyền.

Tầng vật lý liên quan đến:

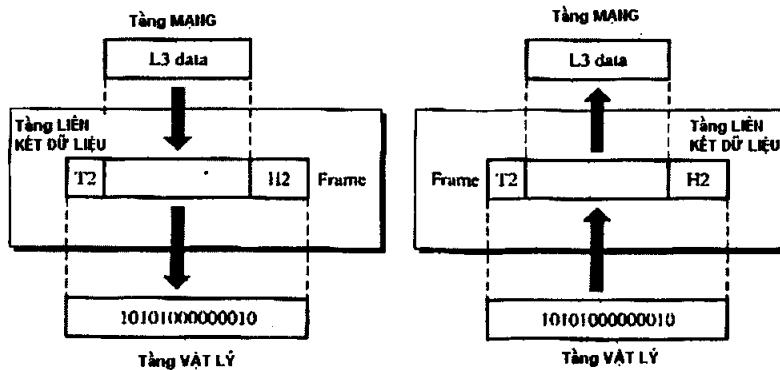
- ☛ **Đặc điểm vật lý của môi trường (thiết bị) giao tiếp và truyền thông:** Tầng vật lý xác định đặc điểm/đặc tính giao diện giữa thiết bị và môi trường truyền dẫn. Nó cũng xác định kiểu môi trường truyền dẫn thông tin (xem Chương 5).

- ☞ **Biểu diễn bit:** Dữ liệu tầng vật lý là luồng bit liên tục (các chuỗi 0 và 1). Để truyền đi, bit phải được mã hoá thành tín hiệu điện hoặc quang. Tầng vật lý xác định phương thức mã hoá (các bit 0 và 1 được chuyển thành tín hiệu như thế nào).
- ☞ **Tốc độ truyền dữ liệu:** Tốc độ truyền dẫn – số bit được gửi đi trong một đơn vị thời gian. Nói cách khác, tầng vật lý xác định khoảng thời gian để truyền đi một bit.
- ☞ **Đồng bộ hoá các bit:** Máy gửi và nhận phải được đồng bộ hoá.
- ☞ **Cấu hình đường truyền:** Tầng vật lý liên quan đến việc kết nối các thiết bị vào môi trường truyền thông. Trong cấu hình điểm – điểm (point – to – point), hai thiết bị được nối với nhau qua một đường truyền riêng. Trong cấu hình điểm – nhiều điểm (multipoint), một đường truyền được nhiều thiết bị dùng chung.
- ☞ **Topo (mô hình ghép nối) vật lý:** Topo vật lý xác định cách nối các thiết bị với nhau để tạo thành mạng. Có thể sử dụng topo dạng lưới (mesh topology) – mỗi thiết bị được nối với tất cả các thiết bị còn lại; topo dạng sao (star topology) – mỗi thiết bị được nối với một thiết bị trung tâm; topo dạng vòng (ring topology) – mỗi thiết bị được nối với một thiết bị bên cạnh (cứ như vậy tạo thành vòng); hay topo dạng bus – mỗi thiết bị được nối đến một đường truyền chung.
- ☞ **Chế độ truyền dẫn:** Tầng vật lý xác định hướng truyền dữ liệu giữa hai thiết bị gồm đơn công (simplex), bán song công (half – duplex), hay song công (full – duplex). Trong chế độ đơn công, một thiết bị chỉ có thể gửi hoặc nhận dữ liệu. Chế độ đơn công là truyền thông một chiều. Trong chế độ bán song công, thiết bị có thể gửi và nhận dữ liệu, nhưng không phải tại cùng một thời điểm. Trong chế độ song công, thiết bị có thể nhận và gửi dữ liệu tại cùng một thời điểm.

### b) Tầng liên kết dữ liệu

Nhiệm vụ của tầng liên kết dữ liệu là truyền thông giữa hai nút nối trực tiếp với nhau. Nó biến tầng vật lý không tin cậy thành đường truyền tin cậy

cho tầng mạng bên trên. Hình 1.10 minh họa mối quan hệ giữa tầng liên kết dữ liệu với tầng mạng và tầng vật lý.

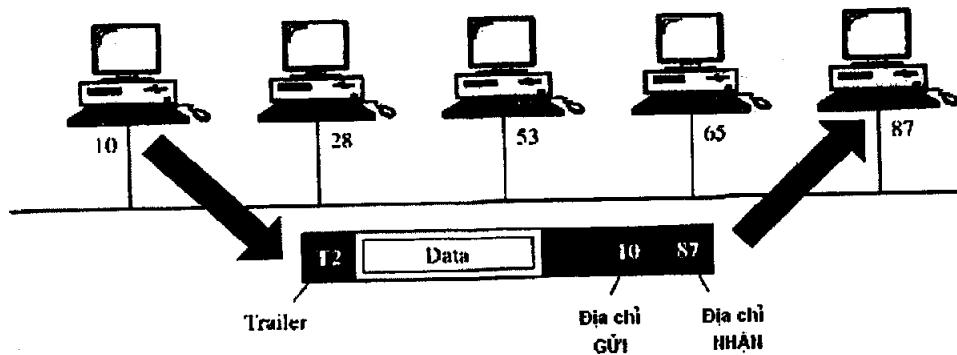


Hình 1.10. Vị trí, vai trò của tầng liên kết dữ liệu

Tầng liên kết dữ liệu chịu trách nhiệm:

- ☛ **Framing – đóng gói dữ liệu:** Tầng liên kết dữ liệu chia luồng bit nhận được từ tầng mạng thành các đơn vị dữ liệu gọi là frame.
- ☛ **Định địa chỉ vật lý:** Nếu gói dữ liệu được chuyển đến thiết bị khác trong mạng, tầng liên kết dữ liệu thêm vào tiêu đề của frame địa chỉ vật lý của nơi nhận (địa chỉ đích) và địa chỉ vật lý của nơi gửi (địa chỉ nguồn). Nếu gói dữ liệu được chuyển đến các thiết bị bên ngoài mạng, địa chỉ nhận sẽ là địa chỉ của thiết bị trung gian kết nối mạng ra bên ngoài.
- ☛ **Kiểm soát lưu lượng:** Nếu tốc độ nhận dữ liệu nhỏ hơn tốc độ gửi dữ liệu, tầng liên kết dữ liệu phải thực hiện kỹ thuật kiểm soát lưu lượng để ngăn ngừa tình trạng quá tải tại nơi nhận.
- ☛ **Kiểm soát lỗi:** Tầng liên kết dữ liệu làm tăng tính tin cậy của tầng vật lý, bằng cách sử dụng kỹ thuật phát hiện và truyền lại các frame bị lỗi hoặc bị mất. Nó cũng sử dụng kỹ thuật xử lý các frame trùng lặp. Kiểm soát lỗi thường được thực hiện bằng cách thêm một trailer vào phần cuối của frame.
- ☛ **Kiểm soát truy cập:** Khi nhiều thiết bị dùng chung đường truyền, các giao thức ở tầng liên kết dữ liệu sẽ quyết định thiết bị nào được quyền sử dụng đường truyền tại một thời điểm xác định.

Trong Hình 1.11, nút có địa chỉ vật lý 10 gửi frame đến nút có địa chỉ vật lý 87. Hai nút này được nối với nhau qua đường truyền dùng chung. Ở tầng liên kết dữ liệu, header của frame chứa các địa chỉ vật lý. Phần còn lại của header là Trailes (tiêu đề cuối) chứa các thông tin cần thiết cho tầng liên kết dữ liệu. Trailer thường chứa các bit dư thừa để kiểm soát lỗi.



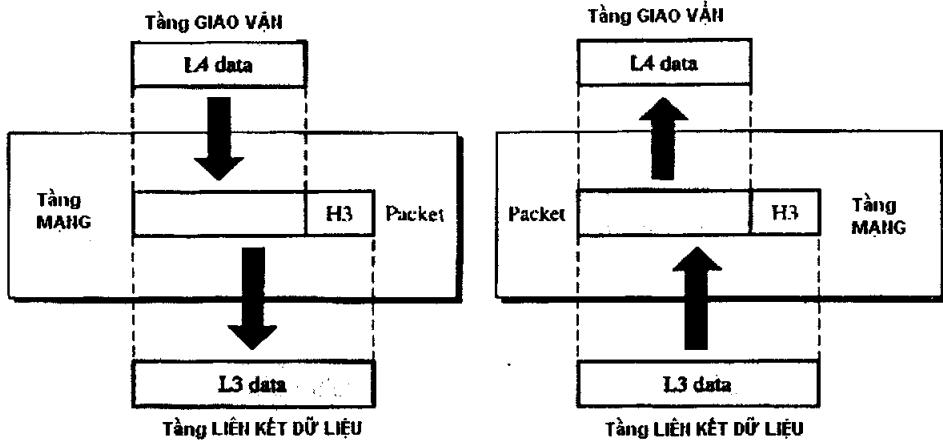
**Hình 1.11.** Ví dụ về địa chỉ của tầng liên kết dữ liệu

Hãy xét tiếp ví dụ chuyển thư: Sau khi thu thập thư tại hòm thư, nhân viên bưu chính thực hiện việc phân loại thư ra hai nhóm, nhóm thứ nhất gồm các thư gửi tới địa chỉ nằm trong vùng do bưu cục quản lý, nhóm thứ hai chuyển ra phía ngoài. Đối với nhóm thứ hai, nhân viên bưu cục đặt tất cả thư trong một túi thư lớn và chuyển cho bưu cục cấp cao hơn. Các túi thư có thể xem là các "frame". Hơn thế nữa, địa chỉ trên túi thư sẽ cho phép túi thư được chuyển đến bưu cục cấp cao thích hợp.

### c) Tầng mạng

Tầng mạng chịu trách nhiệm chuyển gói dữ liệu từ nơi gửi đến nơi nhận, gói dữ liệu có thể phải đi qua nhiều mạng (các chặng trung gian). Tầng liên kết dữ liệu thực hiện truyền gói dữ liệu giữa hai thiết bị trong cùng một mạng, còn tầng mạng đảm bảo gói dữ liệu sẽ được chuyển từ nơi gửi đến đúng nơi nhận.

Nếu hai thiết bị nằm trên cùng một môi trường truyền thì rõ ràng không cần tầng mạng. Tuy nhiên, nếu hai thiết bị ở trên hai mạng khác nhau và giữa chúng có nhiều thiết bị kết nối trung gian, thì cần phải có tầng mạng để thực hiện việc chuyển dữ liệu từ nguồn đến đích. Hình 1.12 minh họa mối quan hệ giữa tầng mạng với tầng giao vận và liên kết dữ liệu.

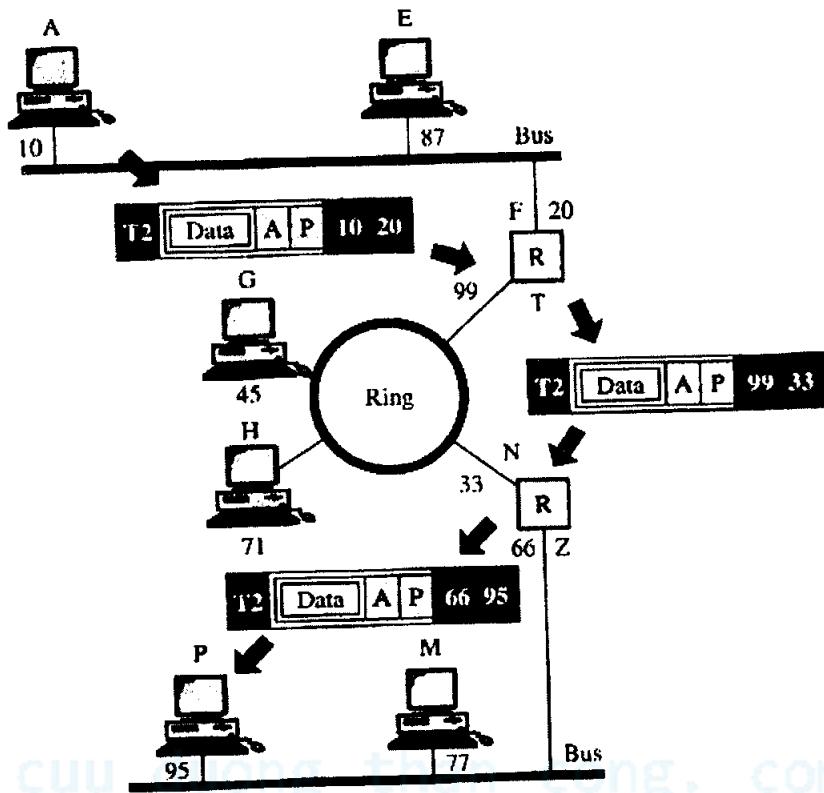


Hình 1.12. Vị trí tầng mạng

Tầng mạng có nhiệm vụ:

- ☞ **Định địa chỉ logic:** Địa chỉ vật lý của tầng liên kết dữ liệu chỉ giải quyết được vấn đề định địa chỉ cục bộ trên một mạng nhỏ. Nếu gói dữ liệu được chuyển sang mạng khác, cần có hệ thống địa chỉ khác để phân biệt giữa hệ thống gửi và hệ thống nhận. Tầng mạng bổ sung thêm tiêu đề – có chứa địa chỉ logic của thiết bị nhận và thiết bị gửi vào mỗi gói dữ liệu gửi đi.
- ☞ **Định tuyến:** Khi mạng hoặc các nút riêng rẽ nối với nhau tạo thành một liên mạng (mạng của các mạng), các thiết bị kết nối trung gian (router hoặc gateway) phải xác định tuyến đường (định tuyến) cho gói dữ liệu để chúng đến được đích.

Giả sử trong Hình 1.13, dữ liệu được gửi từ nút có địa chỉ mạng A với địa chỉ vật lý 10 trong một mạng cục bộ tới nút có địa chỉ mạng P với địa chỉ vật lý 95 trong mạng cục bộ khác. Do hai thiết bị thuộc hai mạng khác nhau, người ta không thể chỉ sử dụng địa chỉ vật lý, vì nó chỉ có tác dụng trong mạng cục bộ. Ở đây, cần một địa chỉ toàn thế để có thể chuyển packet giữa các mạng khác nhau. Địa chỉ logic có đặc điểm này. Gói dữ liệu tại tầng mạng chứa địa chỉ logic, nó không thay đổi khi packet đi từ nơi gửi đến nơi nhận (A và P). Địa chỉ logic không thay đổi khi gói dữ liệu đi từ mạng này sang mạng khác. Ngược lại, địa chỉ vật lý thay đổi khi packet đi từ mạng này sang mạng khác. Trong hình 1.13, R là router – kiểu thiết bị này sẽ được mô tả kỹ trong Chương 3.



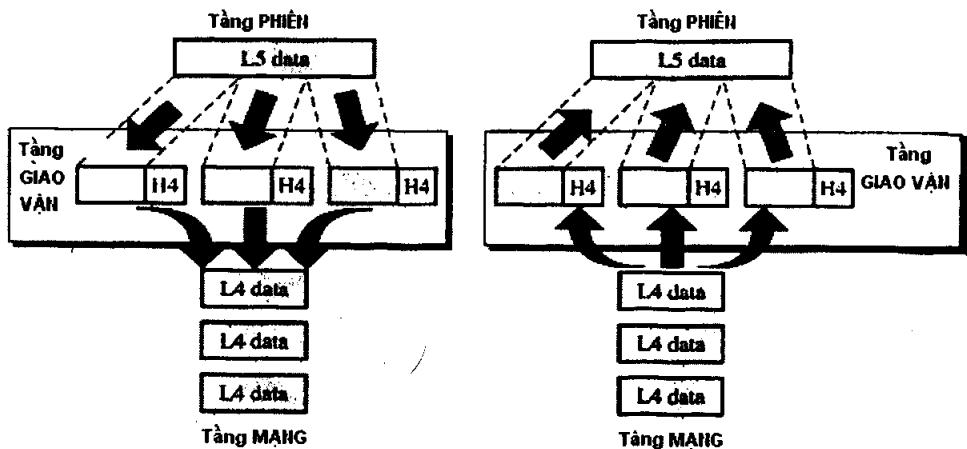
**Hình 1.1. Ví dụ về địa chỉ tầng mạng**

Các bưu cục sẽ cung cấp dịch vụ giống với tầng mạng. Trong mỗi bưu cục sẽ có một "bảng định tuyến" cho phép bưu tá xác định nơi cần chuyển tiếp bức thư đến. Bức thư sẽ được chuyển đến trạm kế tiếp (bưu cục hay địa chỉ nhận) nhờ vào dịch vụ của tầng liên kết dữ liệu. Rõ ràng rằng, thư được chuyển trên các chặng có thể bằng những phương thức hoàn toàn khác nhau (bằng ô tô, máy bay,...). Điểm khác biệt duy nhất giữa mạng bưu chính và mạng máy tính là topo của mạng bưu chính gần như không thay đổi theo thời gian, do vậy việc định tuyến gần như là "tĩnh".

#### *d) Tầng giao vận*

Tầng giao vận chịu trách nhiệm chuyển toàn bộ thông điệp từ nơi gửi đến nơi nhận. Tầng mạng chuyển từng gói dữ liệu riêng lẻ từ nơi gửi đến nơi nhận mà không quan tâm đến quan hệ giữa các gói dữ liệu. Tầng mạng xử lý mỗi gói dữ liệu một cách độc lập mà không quan tâm các gói có thuộc vào cùng một thông điệp hay không. Nói cách khác, tầng giao vận đảm bảo gửi

thông điệp đến nơi nhận một cách toàn vẹn. Hình 1.14 minh họa mối quan hệ của tầng giao vận với tầng mạng và tầng phiên.



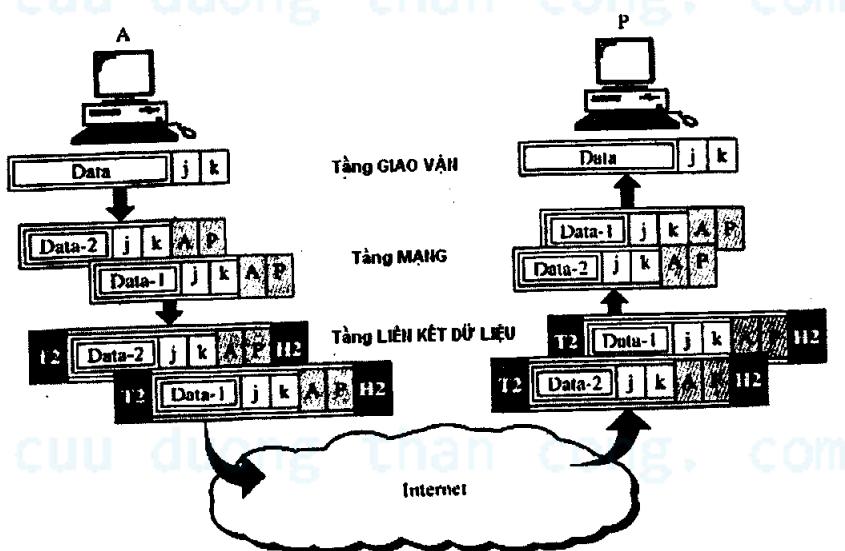
**Hình 1.2. Quan hệ giữa tầng giao vận, tầng phiên và tầng mạng**

Tầng giao vận tạo ra một kết nối logic giữa hai cổng đầu/cuối: tất cả các gói dữ liệu của cùng một thông điệp được truyền theo đường kết nối đó. Có ba giai đoạn của kết nối gồm thiết lập kết nối, truyền dữ liệu, giải phóng kết nối. Do phải truyền tất cả các gói dữ liệu trên một kết nối, tầng giao vận còn phải kiểm soát thứ tự truyền, lưu lượng, phát hiện và sửa lỗi.

Tầng giao vận chịu trách nhiệm:

- ☞ **Địa chỉ cổng (port number):** Các máy tính thường chạy nhiều chương trình tại cùng một thời điểm. Vì vậy, việc chuyển thông điệp không chỉ là truyền dữ liệu từ một máy tính này đến một máy tính khác, mà phải chuyển thông điệp từ tiến trình cụ thể trên máy tính này đến tiến trình cụ thể trên máy tính khác. Header được thêm vào tại tầng giao vận phải chứa thông tin về một kiểu địa chỉ – địa chỉ cổng hay địa chỉ tiến trình. Sau khi tầng mạng chuyển gói dữ liệu tới thiết bị nhận, tầng giao vận ở phía nhận phải chuyển toàn bộ thông điệp đến đúng tiến trình nhận.
- ☞ **Phân mảnh và tái hợp nhất:** Mỗi thông điệp được chia thành các đoạn (segment) nhỏ, được truyền độc lập với nhau. Mỗi segment có một số thứ tự cho phép tầng giao vận phía nhận ghép các segment lại thành thông điệp hoàn chỉnh, hay phát hiện segment nào bị mất trong khi truyền.

- ☞ **Kiểm soát kết nối:** Tầng giao vận có thể hướng nối hoặc không hướng nối. Thực thể giao vận không hướng nối xử lý segment như một gói dữ liệu độc lập và chuyển nó đến tầng giao vận của máy nhận. Một tầng giao vận hướng nối thực hiện kết nối với tầng giao vận của máy nhận trước, sau đó mới chuyển các gói dữ liệu đi. Sau khi tất cả dữ liệu được chuyển đi, kết nối được giải phóng.
- ☞ **Kiểm soát lưu lượng:** Giống như tầng liên kết dữ liệu, tầng giao vận chịu trách nhiệm kiểm soát lưu lượng. Tuy nhiên, việc kiểm soát lưu lượng được thực hiện ở thiết bị đầu/cuối chứ không phải trên một đường truyền vật lý.
- ☞ **Kiểm soát lỗi:** Giống tầng liên kết dữ liệu, tầng giao vận chịu trách nhiệm kiểm soát lỗi. Tuy nhiên, việc kiểm soát lỗi ở tầng này được thực hiện tại các thiết bị đầu/cuối chứ không phải trên đường truyền trung gian. Tầng giao vận ở phía gửi đảm bảo rằng, toàn bộ thông điệp đến tầng giao vận nhận là không bị lỗi (hỏng, mất, dư thừa). Việc khắc phục lỗi thường được thực hiện bằng cách yêu cầu truyền lại.



Hình 1.15. Ví dụ về tầng giao vận

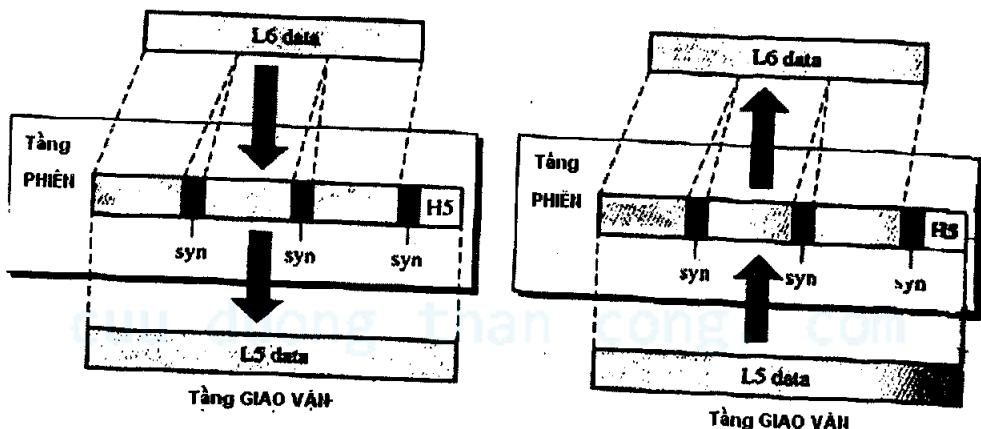
Hình 1.15 minh họa tầng giao vận. Dữ liệu đến từ các tầng trên có địa chỉ cổng là j (của ứng dụng gửi) và k (của ứng dụng nhận). Do kích thước dữ liệu lớn hơn khả năng xử lý của tầng mạng, dữ liệu được chia thành hai

gói nhỏ, mỗi gói dữ liệu vẫn chứa địa chỉ cổng j và k. Tiếp theo, tại tầng mạng, địa chỉ mạng (A và P) được thêm vào mỗi packet. Các gói dữ liệu có thể đi theo các tuyến đường khác nhau, đến nơi nhận có thể không theo đúng thứ tự. Hai gói dữ liệu được chuyển đến tầng mạng của nơi nhận, tại đây header của tầng mạng được lấy ra khỏi gói dữ liệu. Hai gói dữ liệu tiếp tục được chuyển lên tầng giao vận, tại đây chúng được ghép lại để chuyển lên tầng trên.

Hệ thống bưu cục không có tầng giao vận. Trong ví dụ chuyển thư, tầng giao vận sẽ được triển khai ở người gửi và người nhận thư. Giả sử A gửi thư cho B mỗi ngày một lá thư. Hệ thống bưu cục có thể làm mất, hay gửi trễ một lá thư nào đó. B có thể phát hiện ra điều đó nếu A ghi ngày tháng viết thư trong mỗi lá thư. Nếu B không nhận thư của một ngày nào đó trong một khoảng thời gian tương đối dài, B có thể cho rằng thư đó bị mất và yêu cầu A gửi lại. Nói chung đây sẽ là cơ chế hoạt động của tầng giao vận.

### e) Tầng phiên

Các dịch vụ của ba tầng đầu (vật lý, liên kết dữ liệu và mạng) chưa đủ để hai tiến trình trên hai thiết bị có thể truyền thông. Tầng phiên đóng vai trò "kiểm soát viên" hội thoại (dialog) của mạng với nhiệm vụ thiết lập, duy trì và đồng bộ hóa tính liên tác giữa hai bên.



Hình 1.16. Vai trò của tầng phiên

Tầng phiên chịu trách nhiệm về:

- ☞ **Kiểm soát hội thoại:** Tầng phiên cho phép hai thực thể (tiến trình) cùng tham gia vào một cuộc hội thoại. Nó cho phép truyền thông

giữa hai tiến trình được thực hiện, hoặc theo chế độ bán song công hoặc song công. Ví dụ, hội thoại giữa một thiết bị đầu/cuối với một mainframe có thể theo kiểu bán song công.

- ☞ **Đồng bộ hoá:** Tầng phiên cho phép một tiến trình thêm các mốc (trong thuật ngữ mạng gọi là *điểm đồng bộ* – *synchronization point*) vào luồng dữ liệu. Ví dụ, nếu hệ thống cần gửi đi một file có 2000 trang, cứ sau 100 trang nên chèn thêm các điểm đồng bộ để đảm bảo rằng việc nhận từng cụm 100 trang được thực hiện độc lập. Trong trường hợp này nếu như có lỗi khi đang truyền đi trang 523, việc truyền lại sẽ được bắt đầu từ trang 501, không cần phải truyền lại các trang từ 1 đến 500.

Trong một công ty nào đó có hai thư ký – một người chuyên nhận thư và một người chuyên gửi thư. Hai người thư ký này đóng vai trò tầng giao vận. Người thư ký trưởng phụ trách cả hai thư ký này đóng vai trò tầng phiên.

### f) Tầng trình diễn

Tầng trình diễn thực hiện biểu diễn cú pháp và ngữ nghĩa các thông tin được trao đổi giữa hai hệ thống.



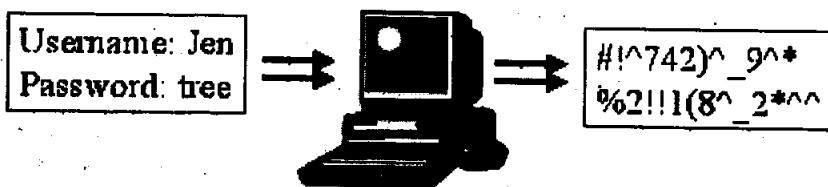
Hình 1.17. Nhiệm vụ của tầng trình diễn

### Tầng trình diễn có nhiệm vụ:

- ☞ **Phiên dịch (Translation):** Tiến trình trên hai thiết bị trao đổi các thông tin dưới dạng chuỗi ký tự, số,... Các thông tin này sau đó được chuyển thành chuỗi bit trước khi truyền. Do các hệ thống máy tính khác nhau sử dụng các hệ thống mã hoá khác nhau, tầng trình diễn chịu trách nhiệm chuyển đổi giữa các cách mã hoá khác nhau. Tầng trình diễn tại phía gửi chuyển thông tin theo khuôn dạng của

mình thành thông tin theo khuôn dạng chung. Tầng trình diễn tại máy nhận sẽ chuyển thông tin trong khuôn dạng chung thành thông tin theo khuôn dạng của máy nhận.

- **Mã hoá:** Hệ thống phải có khả năng đảm bảo tính bí mật khi chuyển những thông tin. Do vậy, phía gửi sẽ biến đổi thông tin ban đầu thành một dạng khác và gửi nó đến phía nhận – đây là công việc mã hoá. Phía nhận thực hiện quá trình ngược lại – biến thông điệp nhận được thành dạng ban đầu. Quá trình này được gọi là *giải mã*.

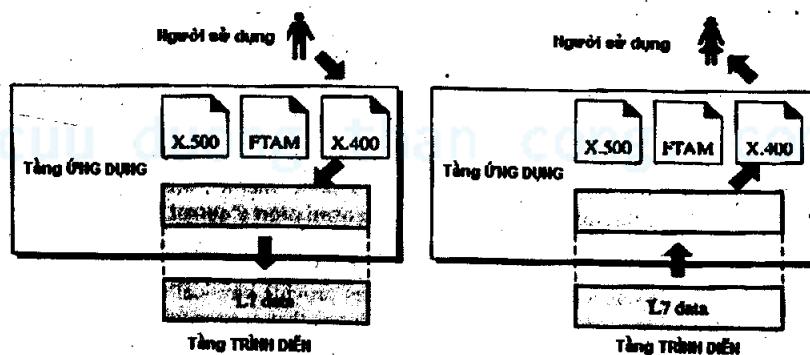


Hình 1.18. Mã hoá dữ liệu

- **Nén:** Nén dữ liệu làm giảm số lượng bit trên đường truyền. Nén dữ liệu ngày càng trở nên quan trọng, đặc biệt trong việc truyền các dữ liệu đa phương tiện như âm thanh, hình ảnh.

### g) Tầng ứng dụng

Tầng ứng dụng cho phép người dùng (con người hay phần mềm) truy cập vào mạng, bằng cách cung cấp giao diện người sử dụng, hỗ trợ các dịch vụ như gửi thư điện tử, truy cập và chuyển file từ xa, quản lý cơ sở dữ liệu dùng chung và một số dịch vụ khác về thông tin.



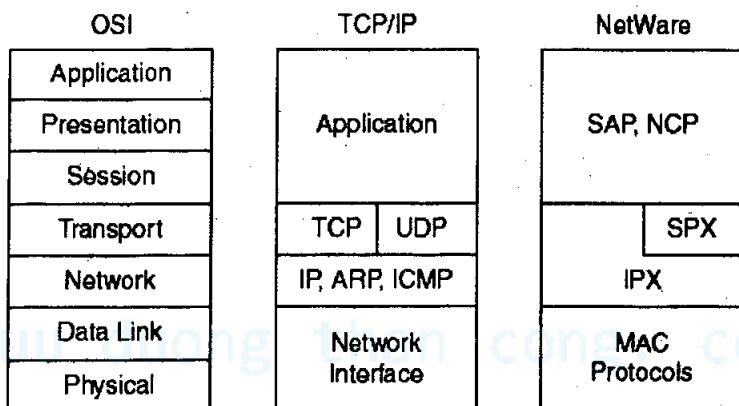
Hình 1.19. Tầng ứng dụng

Hình 1.19 minh họa mối quan hệ giữa tầng ứng dụng với người dùng và với tầng trình diễn. Có rất nhiều ứng dụng có sẵn, ở đây chỉ đề cập đến 3 ứng dụng: X.400 (dịch vụ xử lý thông điệp), X.500 (dịch vụ thư mục) và dịch vụ truy cập, chuyển và quản lý file (FTAM). Người dùng trong ví dụ dưới đây dùng X.400 để gửi đi một thông điệp điện tử. Chú ý rằng, tầng ứng dụng sẽ tạo ra dữ liệu thực sự chứ không có các thông tin tiêu đề.

Tầng ứng dụng cung cấp các dịch vụ:

- ☛ **Thiết bị đầu/cuối ảo của mạng:** Thiết bị đầu/cuối ảo là phiên bản phần mềm của thiết bị đầu/cuối vật lý, cho phép người dùng đăng nhập vào một máy từ xa.
- ☛ **Quản lý, truy cập và chuyển file:** Ứng dụng cho phép người dùng truy cập file (để viết hoặc đọc dữ liệu), lấy file, quản lý hoặc kiểm soát các file trên máy tính khác.
- ☛ **Các dịch vụ khác:** Sử dụng thư điện tử cho phép hai người trao đổi thông tin với nhau; ứng dụng Web cho phép người sử dụng xem thông tin được lưu trữ trên các server,... Số lượng các ứng dụng mạng tăng lên rất nhanh.

### 1.2.3. Bộ giao thức TCP/IP – Mô hình Internet



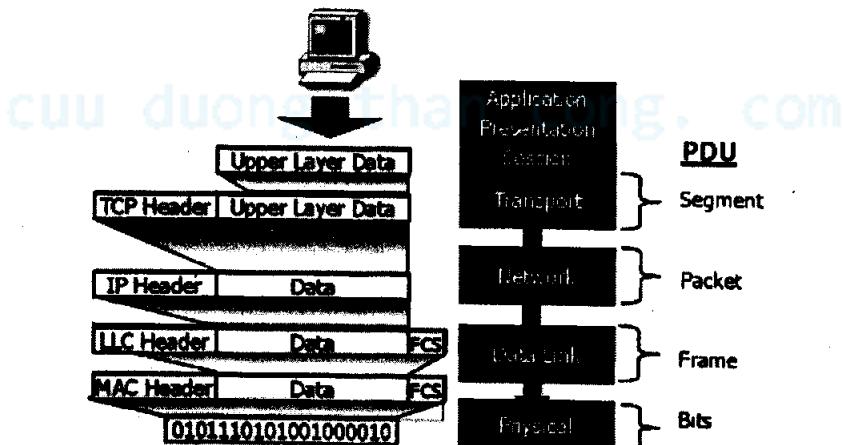
Hình 1.20. Đối chiếu mô hình OSI, mô hình Internet và NetWare

Bộ giao thức TCP/IP (được sử dụng trên Internet) ra đời trước khi có mô hình OSI. Các tầng trong bộ giao thức TCP/IP không giống hệt các tầng trong mô hình OSI. Bộ giao thức TCP/IP có 5 tầng là vật lý, liên kết dữ liệu, mạng, giao vận và ứng dụng. Bốn tầng đầu tiên cung cấp các chuẩn vật lý,

giao tiếp mạng, liên mạng và chức năng giao vận tương ứng với 4 tầng đầu tiên trong mô hình OSI. Tuy nhiên, 3 tầng trên cùng trong mô hình OSI được nhập thành tầng ứng dụng trong mô hình Internet (Hình 1.21).

TCP/IP là giao thức phân cấp, được tạo thành bởi các module độc lập, mỗi module cung cấp một chức năng nhất định. Tuy nhiên, các module này không nhất thiết phải độc lập với nhau. Mô hình OSI xác định rõ chức năng nào thuộc về tầng nào; trong khi đó, các tầng của bộ giao thức TCP/IP chưa có các giao thức tương đối độc lập với nhau, nhưng các giao thức này vẫn có thể kết hợp với nhau tùy thuộc vào nhu cầu hệ thống. Thuật ngữ "phân cấp" mang nghĩa mỗi giao thức ở tầng trên được hỗ trợ bởi một hoặc nhiều giao thức ở tầng dưới.

Tại tầng giao vận, mô hình Internet có hai giao thức: Transmission Control Protocol (TCP) và User Datagram Protocol (UDP). Tại tầng mạng là giao thức Internetworking Protocol, thường được gọi là IP.



Hình 1.21. Dữ liệu đi từ trên xuống trong mô hình INTERNET

# Chương 2

## TẦNG ỨNG DỤNG

---

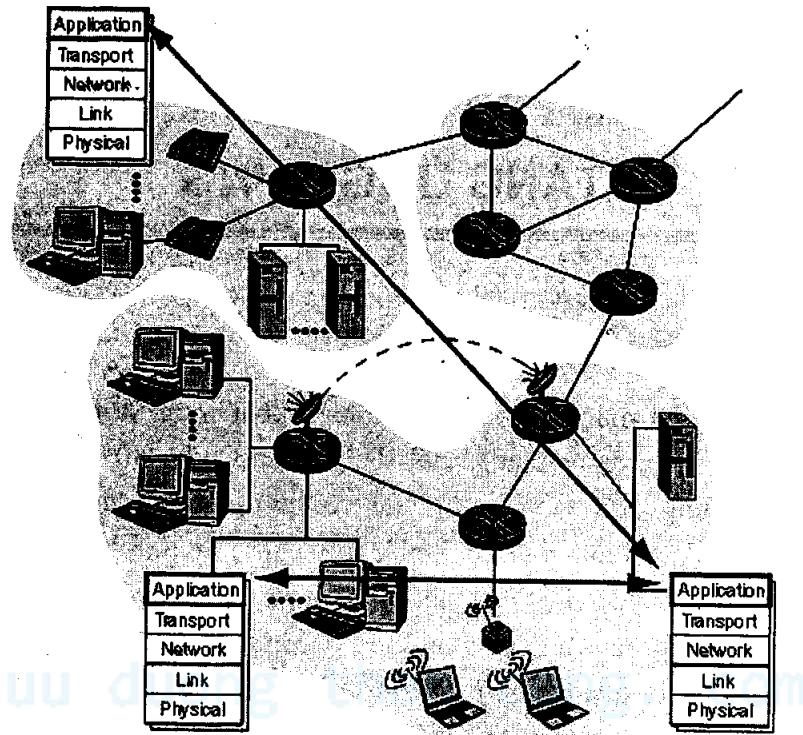
### 2.1. GIAO THỨC TẦNG ỨNG DỤNG

Trong hơn ba mươi năm qua, có nhiều phát minh đột phá trong việc phát triển các ứng dụng mạng. Bắt đầu từ thập niên 80 của thế kỷ XX, những ứng dụng đơn giản tương tác với người dùng qua chế độ lệnh (text – based) đã trở nên phổ biến, như truy cập máy tính từ xa (telnet), thư điện tử (email), truyền file (ftp), nhóm thông tin (newsgroup) và trò chuyện từ xa (chat). Hiện nay, những ứng dụng đa phương tiện phức tạp hơn như World Wide Web, điện thoại trực tuyến, hội thảo từ xa, chia sẻ file,... đã ngày càng trở nên quen thuộc.

Phần mềm ứng dụng mạng được cài đặt trên các *thiết bị đầu/cuối* (end – system) như máy tính, điện thoại di động,... Ví dụ, với Web, có hai phần mềm tương tác với nhau là phần mềm trình duyệt trong máy tính của người dùng (PC, hay trạm làm việc) và phần mềm Web server.

Trong thuật ngữ hệ điều hành, việc kết nối được thực hiện giữa các *tiến trình* (process) chứ không phải giữa các chương trình phần mềm. Tiến trình là một chương trình chạy trên thiết bị đầu/cuối. Khi các tiến trình chạy trên cùng một thiết bị, chúng sẽ kết nối, trao đổi dữ liệu với nhau thông qua cơ chế truyền thông liên tiến trình (interprocess communication) của hệ điều hành. Trong cuốn sách này quan tâm đến việc kết nối giữa các tiến trình trên những thiết bị khác nhau (và có thể trên những hệ điều hành khác nhau). Việc kết nối như vậy sẽ được thực hiện bằng cách trao đổi thông điệp qua mạng máy tính. Tiến trình gửi sẽ tạo và gửi thông điệp, tiến trình nhận sẽ nhận thông điệp (message) và có thể phản hồi lại bằng cách gửi một thông điệp trả lời (Hình 2.1). Ứng dụng mạng có các giao thức định nghĩa khuôn

dạng, thứ tự trao đổi các thông điệp cũng như hành vi của mỗi bên khi nhận được thông điệp.



Hình 2.1. Các ứng dụng trên mạng

### 2.1.1. Giao thức tầng ứng dụng

Giao thức tầng ứng dụng chỉ là một phần (cho dù là phần quan trọng) của ứng dụng mạng. Ví dụ, Web – ứng dụng mạng cho phép người dùng lấy các đối tượng từ Web server, bao gồm nhiều thành phần, như tiêu chuẩn định dạng văn bản (HTML), trình duyệt Web (Netscape Navigator hay Microsoft Internet Explorer), Web server (Apache, Microsoft và Netscape server) và giao thức tầng ứng dụng. Giao thức tầng ứng dụng của Web – HTTP (HyperText Transfer Protocol [RFC 2616]), định nghĩa cách thức truyền thông điệp giữa Web client (trình duyệt) và Web server.

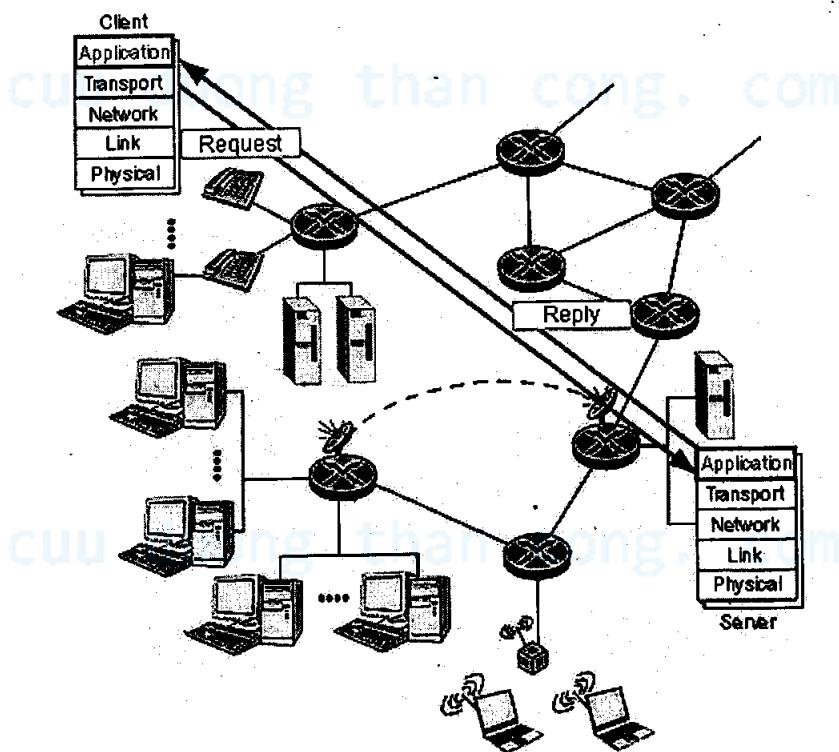
Như đã nói ở trên, giao thức tầng ứng dụng định nghĩa cách thức truyền thông điệp giữa các tiến trình ứng dụng chạy trên các thiết bị khác nhau. Nó xác định:

- ☞ Kiểu thông điệp trao đổi, ví dụ như thông điệp yêu cầu hay thông điệp trả lời.
- ☞ Cú pháp của thông điệp, ví dụ các trường trong thông điệp cũng như cách xác định chúng.
- ☞ Ý nghĩa của các trường.
- ☞ Quy tắc xác định khi nào và như thế nào tiên trình gửi và trả lời thông điệp?

Nhiều giao thức tầng ứng dụng được đặc tả trong các RFC. Ví dụ, đặc tả của HTTP là HTTP RFC.

#### a) *Mô hình khách hàng/người phục vụ (Client/Server)*

Giao thức ứng dụng mạng thường chia ra hai phía, phía client và phía server (Hình 2.2). Phía client trong thiết bị này liên lạc với phía server trong thiết bị khác. Ví dụ, trình duyệt Web là phía client, Web server là phía server của HTTP.

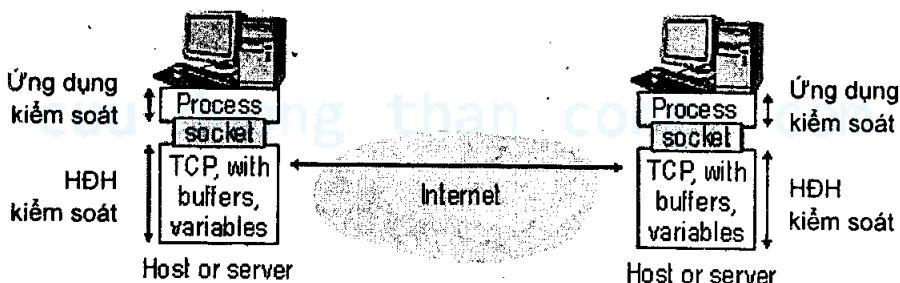


**Hình 2.2. Tương tác client/server**

Trong nhiều ứng dụng, máy tính sẽ thực hiện cả phần client và phần server của ứng dụng. Ví dụ, xét phiên làm việc Telnet giữa máy A và máy B (Telnet là ứng dụng đăng nhập từ xa). Nếu máy A bắt đầu trước (có nghĩa là người dùng ở máy A đăng nhập vào máy B), khi đó máy A chạy phía client và máy B chạy phía server của ứng dụng. Mặt khác, nếu máy B bắt đầu trước thì máy B chạy phía client của ứng dụng.

### b) Truyền thông giữa các tiến trình

Ứng dụng bao gồm hai tiến trình trên hai thiết bị khác nhau và liên lạc với nhau qua mạng. Hai tiến trình liên lạc với nhau bằng cách gửi và nhận thông điệp qua socket của chúng. Socket có thể xem như "cửa" của tiến trình, vì tiến trình nhận và gửi thông điệp thông qua "cửa". Khi muốn gửi thông điệp tới tiến trình khác, tiến trình đầy thông điệp cần gửi qua "cửa" với giả định rằng, thực thể giao vận nằm bên kia "cửa" sẽ chuyển thông điệp đến "cửa" của tiến trình nhận.



**Hình 2.3. Tiến trình ứng dụng, socket và giao thức giao vận**

Hình 2.3 minh họa truyền thông qua socket giữa hai tiến trình trên Internet (Tầng giao vận trên hình 2.3 là TCP, mặc dù ở đây có thể sử dụng giao thức khác như UDP). Qua hình 2.3 ta thấy, socket là giao diện giữa tầng ứng dụng và tầng giao vận trong máy tính. Nó được xem là API (giao diện lập trình ứng dụng) giữa ứng dụng và mạng. Người thiết kế ứng dụng kiểm soát mọi khía cạnh phía bên trên socket – là tầng ứng dụng, nhưng chỉ có khả năng kiểm soát rất ít tầng giao vận phía dưới socket. Với tầng giao vận, người lập trình ứng dụng chỉ có thể kiểm soát được: (1) chọn giao thức giao vận nào; (2) xác định một vài tham số ở tầng giao vận như độ lớn bộ đệm và kích thước tối đa của gói tin. Khi người lập trình lựa chọn giao thức giao vận nào, ứng dụng được tạo ra sử dụng tầng giao vận ứng với giao thức đó.

### c) Địa chỉ tiến trình

Để gửi thông điệp cho tiến trình trên máy tính khác, tiến trình gửi phải xác định được tiến trình nhận. Tiến trình được xác định qua hai phần: (1) tên hay địa chỉ của máy tính; (2) định danh xác định tiến trình trên máy tính nhận.

Trong ứng dụng Internet, máy tính được xác định qua địa chỉ IP (Địa chỉ IP sẽ được trình bày trong Chương 4). Nay giờ chúng ta chỉ cần biết địa chỉ IP là một số 32 bit, dùng để xác định duy nhất một thiết bị (chính xác hơn, nó xác định duy nhất giao diện (interface) của thiết bị kết nối vào Internet). Vì địa chỉ IP của thiết bị mang tính xác định duy nhất, nên việc phân phối địa chỉ IP được quản lý chặt chẽ. Mạng ATM có một chuẩn địa chỉ khác. ITU-T lấy số điện thoại làm địa chỉ, gọi là địa chỉ E.164 sử dụng trên mạng ATM.

Ngoài địa chỉ thiết bị nhận, phía gửi phải thêm vào thông tin giúp phía nhận chuyển tiếp thông điệp cho tiến trình phù hợp (vì trong máy tính nhận có thể có nhiều tiến trình đồng thời hoạt động). Thông tin này là cổng phía nhận (destination port). Các giao thức tầng ứng dụng phổ biến đều được gán số hiệu cổng (port number) là một số cụ thể. Ví dụ, tiến trình Web server (giao thức HTTP) sử dụng cổng 80. Tiến trình mail server (giao thức SMTP) sử dụng cổng 25. Danh sách các cổng cho tất cả giao thức thường gặp trên Internet được liệt kê trong RFC 1700. Khi xây dựng ứng dụng mạng mới thì ứng dụng đó phải được đăng ký một số hiệu cổng mới.

#### 2.1.2. Các yêu cầu của ứng dụng

Socket là giao diện giữa tiến trình ứng dụng và thực thể giao vận. Ứng dụng gửi thông điệp qua "cửa". Ở sau cánh cửa thực thể giao vận có trách nhiệm chuyển thông điệp qua mạng máy tính tới "cửa" tiến trình nhận. Nhiều kiểu mạng, kể cả Internet, có nhiều kiểu giao thức giao vận khác nhau. Khi thiết kế ứng dụng, phải lựa chọn một giao thức giao vận có sẵn nào đó, bằng cách nghiên cứu các dịch vụ được các giao thức giao vận cung cấp và sau đó sẽ chọn giao thức nào đáp ứng đầy đủ nhất các yêu cầu của mình. Nhìn chung, chúng ta có thể phân loại theo ba nhóm là mát mát dữ liệu, băng thông và thời gian.

### **a) Mất mát dữ liệu (Data loss)**

Một số ứng dụng như thư điện tử, truyền file, truy cập từ xa, truyền các đối tượng Web và ứng dụng tài chính đòi hỏi dữ liệu phải được truyền chính xác và đầy đủ, tức là không được mất mát dữ liệu. Đặc biệt, mất mát file dữ liệu, hoặc dữ liệu trong các giao dịch tài chính có thể gây nên hậu quả nghiêm trọng. Tuy nhiên, một số ứng dụng khác như ứng dụng đa phương tiện (real – time audio/video hay stored audio/video) chấp nhận mất mát dữ liệu. Trong các ứng dụng kiểu này, mất dữ liệu có thể gây nên một số nhiễu trong dữ liệu đa phương tiện, nhưng điều này có thể chấp nhận được trong giới hạn nào đó. Ảnh hưởng do mất mát dữ liệu tới chất lượng ứng dụng cũng như số lượng cho phép các gói dữ liệu bị mất phụ thuộc vào chính ứng dụng và phương pháp mã hoá.

### **b) Băng thông (bandwidth)**

Để hoạt động hiệu quả, một số ứng dụng phải truyền dữ liệu với một tốc độ nhất định. Ví dụ, ứng dụng gọi điện thoại qua Internet (Internet telephony) mã hoá âm thanh với tốc độ 32Kbs, thì sau đó dữ liệu tạo ra phải được chuyển tới ứng dụng nhận với tốc độ trên. Nếu không có đủ băng thông cần thiết, ứng dụng cần phải mã hoá âm thanh với tốc độ khác hay phải kết thúc – bởi vì nếu không đủ băng thông thì ứng dụng không thể đáp ứng yêu cầu người sử dụng. Những ứng dụng đa phương tiện hiện nay là ứng dụng phụ thuộc vào băng thông (bandwidth sensitive), nhưng trong tương lai ứng dụng đa phương tiện sẽ sử dụng các kỹ thuật mã hoá thích nghi để mã hoá tốc độ cho phù hợp với dài tần hiện có.

### **c) Thời gian (timing)**

Những ứng dụng thời gian thực (real – time) mang tính chất tương tác, như Internet telephone, hội thảo qua điện thoại, hay các trò chơi nhiều người tham gia cùng một lúc (multiplayer game) yêu cầu những ràng buộc chặt chẽ về thời gian trong việc trao đổi dữ liệu. Ví dụ, những ứng dụng này đòi hỏi độ trễ (delay) từ tiến trình gửi đến tiến trình nhận không vượt quá vài trăm phần nghìn giây. Độ trễ lớn trong ứng dụng Internet telephone khiến cuộc đàm thoại bị đứt đoạn giữa chúng. Trong trò chơi nhiều người cùng tham gia, hay trong môi trường tương tác ảo, độ trễ từ lúc đưa ra yêu cầu cho đến khi nhận được kết quả phản ứng từ môi trường (ví dụ, từ một người chơi khác) lớn sẽ làm giảm tính chân thực của trò chơi. Đối với ứng dụng

không tính đến yếu tố thời gian thực, người ta vẫn mong muốn có một độ trễ thấp, song không có ràng buộc chặt chẽ đối với độ trễ.

Hình 2.4 tóm tắt các yếu tố như độ tin cậy, băng thông và các đòi hỏi về thời gian của một số ứng dụng Internet phổ biến. Hình 2.4 chỉ phác họa một vài đòi hỏi quan trọng của những ứng dụng Internet này. Ở đây, chúng ta không có đầy đủ các phân loại hoàn chỉnh, nhưng cũng đủ để nhận biết một vài đặc trưng quan trọng nhất để phân loại ứng dụng.

Ứng dụng	Mất mát dữ liệu	Băng tần	Thời gian
File transfer	Không	Elastic	Không
E-mail	Không	Elastic	Không
Web Documents	Không	Elastic (few Kbps)	Không
Real-time Audio/Video	Chấp nhận mất mát	Audio: Few Kbps – 1Mb Video: 10Kb–5 Mb	Có: 100s of msec
Stored Audio/Video	Chấp nhận mất mát	Same as Above	Có: Few Seconds
Interactive games	Chấp nhận mất mát	Few Kbps – 10Kb	Có: 100s of msec
Financial Applications	Không	Elastic	Có hoặc không

Hình 2.4. Các yêu cầu cho một số ứng dụng

### 2.1.3. Dịch vụ của các giao thức giao vận Internet

Internet (và nói chung TCP/IP) cung cấp hai giao thức giao vận cho tầng ứng dụng là UDP và TCP. Khi xây dựng ứng dụng cho Internet,

một trong những quyết định đầu tiên mà nhà thiết kế phải đưa ra là sử dụng UDP hay TCP. Mỗi giao thức cung cấp một kiểu phục vụ khác nhau cho ứng dụng.

### a) Dịch vụ TCP

Đặc trưng của giao thức TCP là hướng kết nối và cung cấp dịch vụ truyền dữ liệu tin cậy. Khi sử dụng giao thức giao vận TCP, ứng dụng sẽ nhận được cả hai loại dịch vụ này.

- ☛ **Dịch vụ hướng kết nối (connection oriented):** TCP client và TCP server trao đổi các thông tin điều khiển với nhau trước khi truyền dữ liệu ứng dụng. Quá trình "bắt tay" giữa client và server như vậy cho phép cả hai bên sẵn sàng xử lý các gói dữ liệu. Sau quá trình này, xuất hiện một đường kết nối TCP (TCP connection) giữa socket của hai tiến trình. Đây là *kết nối hai chiều (song công – full duplex)* vì cho phép hai tiến trình có thể đồng thời gửi và nhận thông điệp. Khi ứng dụng kết thúc việc gửi thông điệp, nó đóng kết nối lại. Dịch vụ này chỉ là hướng kết nối chứ không phải mạch ảo (virtual circuit), bởi vì hai tiến trình được kết nối một cách lỏng lẻo. Trong Chương 3, sẽ trình bày chi tiết về dịch vụ hướng kết nối và cách thực hiện chúng.
- ☛ **Dịch vụ giao vận tin cậy:** Tiến trình gửi có thể sử dụng TCP để truyền dữ liệu chính xác và đúng thứ tự. Gửi đi một luồng byte qua socket, tiến trình ứng dụng có thể tin tưởng TCP sẽ chuyển luồng byte này đến socket nhận, không bị lỗi hay trùng lắp byte.

TCP cũng có cơ chế kiểm soát tắc nghẽn, cơ chế này đáp ứng cho cả Internet chứ không phải cho hai tiến trình truyền thông với nhau. Kỹ thuật kiểm soát tắc nghẽn của TCP là giảm tốc độ gửi dữ liệu của mỗi tiến trình (client hay server) khi mạng bị tắc nghẽn. Đặc biệt, như chúng ta sẽ thấy trong Chương 3, cơ chế kiểm soát tắc nghẽn của TCP có găng giới hạn mỗi kết nối TCP để chia sẻ công bằng thông tin giữa các tiến trình.

Giới hạn tốc độ truyền có thể không thỏa mãn với các ứng dụng audio và video theo thời gian thực, những ứng dụng đòi hỏi phải có một băng thông tối thiểu. Hơn nữa, ứng dụng thời gian thực chấp nhận mất mát dữ liệu và không thực sự cần đến một dịch vụ giao vận tin cậy hoàn toàn. Vì các lý do đó, các ứng dụng thời gian thực thường chạy trên nền UDP.

Bây giờ trình bày một số dịch vụ mà TCP không cung cấp. Thứ nhất, TCP không bảo đảm một tốc độ truyền tối thiểu. Tiền trình gửi không được phép truyền với bất kỳ tốc độ nào nó đề nghị, tốc độ này được kiểm soát bởi cơ chế kiểm soát tắc nghẽn của TCP. Đôi khi cơ chế này khiến tiền trình gửi phải gửi với tốc độ trung bình tương đối thấp. Thứ hai, TCP không đưa ra bất kỳ sự bảo đảm nào về độ trễ. Khi tiền trình gửi chuyển dữ liệu cho socket TCP, dữ liệu cuối cùng sẽ đến được socket nhận, nhưng TCP không bảo đảm dữ liệu sau bao lâu mới tới được đích. Với những quan sát trên môi trường Internet thực, có thể phải chờ vài giây, thậm chí đến vài phút để TCP gửi thành công một thông điệp (ví dụ, một trang Web HTML từ Web server đến Web client). Nói tóm lại, TCP bảo đảm việc truyền tất cả dữ liệu một cách chính xác, nhưng không bảo đảm về tốc độ truyền và độ trễ.

### b) Dịch vụ UDP

UDP là giao thức giao vận khá đơn giản, với mô hình phục vụ tối thiểu. UDP không hướng nối, nghĩa là không có giai đoạn "bắt tay" trước khi hai tiền trình bắt đầu trao đổi dữ liệu. UDP không cung cấp dịch vụ truyền tin cậy. Khi tiền trình gửi chuyển thông điệp qua cổng UDP, UDP không đảm bảo thông điệp sẽ đến được cổng tiền trình nhận. Hơn nữa, các thông điệp đến đích có thể không đúng thứ tự.

Mặt khác, UDP không có cơ chế kiểm soát tắc nghẽn, vì vậy tiền trình gửi có thể đẩy dữ liệu ra cổng UDP với tốc độ bất kỳ. Mặc dù, không phải tất cả dữ liệu đều tới được đích, nhưng phần lớn dữ liệu có thể tới được. Ứng dụng thời gian thực thường lựa chọn UDP ở tầng giao vận. Giống TCP, UDP không bảo đảm về độ trễ.

Hình 2.5 trình bày các giao thức giao vận của các ứng dụng mạng phổ biến. Thư điện tử, truy cập từ xa, Web và truyền file sử dụng TCP do TCP cung cấp dịch vụ truyền dữ liệu tin cậy, bảo đảm rằng mọi dữ liệu sẽ tới được đích.

Người ta thấy rằng, điện thoại qua Internet chạy trên nền UDP. Mỗi phía của ứng dụng này cần gửi dữ liệu qua mạng với tốc độ tối thiểu nào đó (Hình 2.5). Hơn nữa, ứng dụng điện thoại qua Internet chấp nhận mất mát dữ liệu, vì thế chúng không cần dịch vụ truyền tin cậy của TCP.

<b>Ứng dụng</b>	<b>Giao thức ứng dụng</b>	<b>Giao thức giao vận</b>
Thư 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
Remote File Server	NFS	UDP hoặc TCP
Streaming Multimedia	Giao thức riêng, không công bố (ví dụ Real Networks)	UDP hoặc TCP
Điện thoại Internet	Giao thức riêng, không công bố (ví dụ Vocaltec)	Thường là UDP

**Hình 2.5. Các ứng dụng phổ biến và giao thức giao vận tương ứng**

Nhu đã biết, TCP và UDP đều không bảo đảm về thời gian. Điều này có nghĩa là ứng dụng có ràng buộc về thời gian không thể chạy trên mạng ngày nay? Câu trả lời chắc chắn là không – Internet đã có một kế hoạch cho ứng dụng kiểu này trong nhiều năm tới.

#### **2.1.4. Một số ứng dụng phổ biến**

Các kiểu ứng dụng mạng ngày càng đa dạng và phong phú. Chúng ta sẽ tập trung nghiên cứu một số ứng dụng quan trọng thường gặp. Trong chương này trình bày khá chi tiết ba ứng dụng phổ biến là Web, thư điện tử và dịch vụ tên miền (DNS).

### **2.2. WORLD WIDE WEB: HTTP**

Vào đầu thập niên 90 của thế kỷ XX, ứng dụng quan trọng nhất của Internet – World Wide Web xuất hiện và nhanh chóng được mọi người chấp

nhận. Nó thay đổi cách thức tương tác giữa con người và môi trường làm việc.

## 2.2.1. Tổng quan về HTTP

Hyper Text Transfer Protocol (HTTP) là giao thức tầng ứng dụng của Web. HTTP được triển khai trên cả hai phía client và server. Các tiến trình client và server trên các hệ thống đầu/cuối khác nhau giao tiếp với nhau thông qua việc trao đổi các thông điệp HTTP. HTTP quy định cấu trúc thông điệp cũng như cách thức trao đổi thông điệp giữa client và server. Trước khi nói về HTTP, cần nói lại các thuật ngữ về Web.

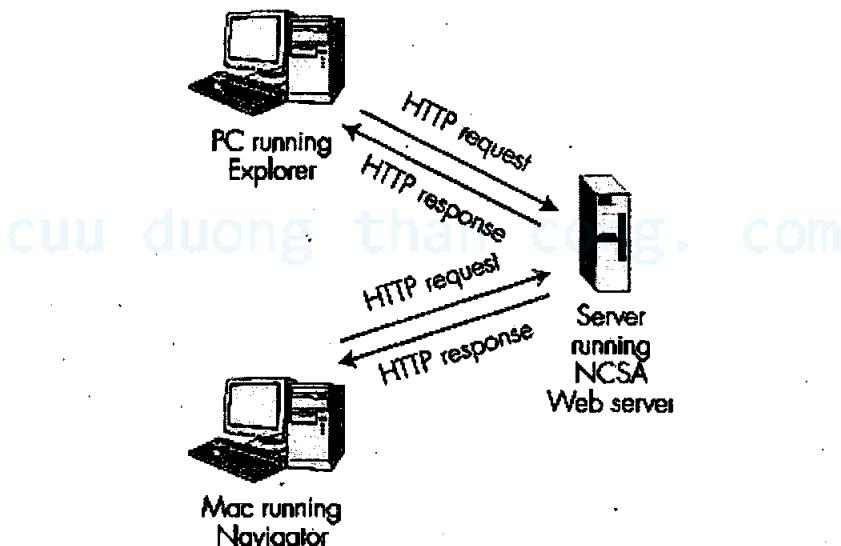
- ☞ **Trang Web** (Webpage – hay còn gọi là một tập tin) chứa các đối tượng (Object). Đơn giản đối tượng chỉ là một file như file HTML, file ảnh JPEG, file ảnh GIF, file java applet, một đoạn âm thanh,... Đối tượng được xác định qua địa chỉ URL. Trang Web chứa một file HTML cơ sở và tham chiếu đến các đối tượng khác. Ví dụ, một trang Web chứa một file HTML văn bản và 5 đối tượng ảnh JPEG, khi đó trang Web có 6 đối tượng gồm 1 file văn bản HTML và 5 file ảnh. File HTML cơ sở này tham chiếu đến các đối tượng khác thông qua địa chỉ URL. Mỗi địa chỉ URL có hai thành phần là tên của máy chủ và vị trí của đối tượng trên máy chủ. Ví dụ một địa chỉ URL là:

**www. someschool.edu/somedepartment/picture.gif**

**www.someschool.edu** là tên máy chủ; **somedepartment/picture.gif** là đường dẫn đối tượng.

- ☞ **Trình duyệt** (Browser): Chương trình giao tiếp người dùng của ứng dụng Web cho phép hiển thị trang Web. Browser là phía client của giao thức HTTP. Hiện nay có rất nhiều phần mềm trình duyệt nhưng phổ biến nhất là Nestcape Communication và Microsoft Internet Explorer. Web server lưu giữ các đối tượng Web và được xác định qua địa chỉ URL. Phần mềm Web server là phía server của giao thức HTTP. Một số phần mềm Web server phổ biến là Apache, Microsoft Internet Information Server và Nestcape Enterprise Server.

HTTP xác định cách thức trình duyệt yêu cầu trang Web từ Web server cũng như cách thức server gửi trang Web được yêu cầu tới trình duyệt. Dưới đây, sẽ nói rõ hơn về quá trình trao đổi giữa client và server. Hình 2.6 minh họa quá trình này. Khi người dùng yêu cầu một đối tượng (ví dụ kích chuột vào một siêu liên kết), browser sẽ gửi thông điệp HTTP tới server yêu cầu đối tượng đó. Server nhận được yêu cầu và trả lời bằng cách gửi lại một thông điệp trả lời chứa đối tượng được yêu cầu. Cho tới những năm 1997, phần lớn các trình duyệt Web và Web server tuân thủ phiên bản HTTP 1.0 (đặc tả trong RFC 1945). Từ năm 1998, một số browser và Web server sử dụng phiên bản 1.1 theo khuyến nghị RFC 2616. Phiên bản mới này tương thích với phiên bản 1.0, nghĩa là Web server dùng phiên bản 1.1 có thể "nói chuyện" được với trình duyệt sử dụng phiên bản 1.0 và ngược lại.



Hình 2.6. Tương tác client/server

Cả phiên bản 1.0 và 1.1 đều sử dụng TCP làm giao thức ở tầng giao vận phía dưới. HTTP client khởi tạo một kết nối TCP tới HTTP server. Sau khi thiết lập được kết nối, cả hai trình browser và Web server đều truy cập tới TCP thông qua socket. Như đã trình bày ở mục 2.1, socket là "cửa" giữa tiến trình ứng dụng và thực thể TCP. Client gửi thông điệp yêu cầu qua socket. Server nhận thông điệp yêu cầu này và gửi thông điệp trả lời qua socket. Sau khi gửi thông điệp qua socket, thông điệp nằm ngoài tầm "kiểm soát" của client và chính thực thể TCP chịu trách nhiệm chuyển nó sang phía bên kia. Trong mục 2.1 ta thấy rằng, TCP cung cấp dịch vụ truyền tin cậy cho

HTTP, như vậy thông điệp của tiền trình client sẽ được chuyển tải nguyên vẹn đến server và ngược lại. Đến đây ta đã thấy được ưu điểm của kiến trúc phân tầng. HTTP không giải quyết việc mất mát dữ liệu mà việc này là trách nhiệm của TCP và các tầng bên dưới.

TCP sử dụng cơ chế tránh tắc nghẽn, cơ chế này sẽ được nghiên cứu chi tiết ở Chương 3. Ở đây chỉ cần biết rằng, khi kết nối TCP mới khởi tạo cơ chế này đòi hỏi tốc độ truyền dữ liệu tương đối thấp, nhưng sẽ tăng nhanh khi trên mạng không có tắc nghẽn. Giai đoạn bắt đầu với tốc độ thấp gọi là giai đoạn bắt đầu chậm (slow start).

Một chú ý quan trọng là, server gửi các đối tượng được yêu cầu cho client mà không ghi lại bất kỳ một thông tin nào về trạng thái của client. Nếu client yêu cầu lại cùng một đối tượng, thì server sẽ không thể trả lời cho client rằng, đối tượng đó vừa được gửi cho client, server sẽ gửi lại cho client đối tượng đó như nó không biết việc gửi lần trước. HTTP server không nhớ các thông tin về client, vì thế HTTP được gọi là *giao thức không trạng thái*.

## 2.2.2. Kết nối liên tục và không liên tục (persistent/nonpersistent)

HTTP hỗ trợ cả hai cách kết nối liên tục và không liên tục. HTTP 1.0 sử dụng kết nối không liên tục. Chế độ mặc định của HTTP 1.1 là kết nối liên tục.

### a) Kết nối không liên tục (nonpersistent)

Xét các bước client thực hiện để yêu cầu trang Web từ server trong trường hợp sử dụng kết nối không liên tục. Giả sử trang Web có chứa một file HTML cơ sở và 10 file ảnh JPEG, đồng thời cả 11 đối tượng này cùng ở trên một server, địa chỉ của file HTML này là:

[www.someschool.edu/somedepartment/home.index](http://www.someschool.edu/somedepartment/home.index)

Các bước thực hiện như sau:

- ☞ HTTP client khởi tạo một kết nối TCP tới server có địa chỉ là [www.someschool.edu](http://www.someschool.edu). Cổng 80 là cổng được HTTP server sử dụng để "lắng nghe" các yêu cầu lấy trang Web từ client thông qua giao thức HTTP.

- ☞ HTTP client gửi thông điệp yêu cầu qua socket tới thực thể TCP đã được kết nối ở bước trước. Thông điệp bao gồm đường dẫn **somedepartment/home.index** (ý nghĩa thông điệp sẽ được giải thích ở phần dưới).
- ☞ HTTP server nhận được thông điệp yêu cầu từ socket, lấy đối tượng **somedepartment/home.index** trong bộ nhớ của mình (ở cùng hoặc RAM), đặt đối tượng này vào trong một thông điệp trả lời và gửi đi qua socket.
- ☞ HTTP server yêu cầu thực thể TCP kết thúc kết nối (nhưng nó không đóng lại thực sự cho đến khi client nhận được thông điệp).
- ☞ HTTP client nhận được thông điệp trả lời, kết nối được đóng lại. Thông điệp chỉ ra rằng, nó chứa một đối tượng là file HTML. Client sẽ lấy file đó ra từ thông điệp trả lời. File HTML tham chiếu đến 10 đối tượng ảnh JPEG.

Người ta thấy rằng, 4 bước đầu được lặp lại cho mỗi đối tượng ảnh được tham chiếu trong file HTML.

Khi nhận được thông điệp trả lời có chứa trang Web, browser sẽ hiển thị trang Web. Các browser khác nhau thì có thể có các cách hiển thị khác nhau đối với cùng một trang Web. HTTP không ảnh hưởng gì đối với cách hiển thị trang Web của client. Các đặc tả trong HTTP chỉ định nghĩa giao thức truyền thông giữa tiến trình client và server mà thôi.

Các bước ở trên sử dụng cách kết nối không liên tục, vì sau khi gửi đi một đối tượng, server sẽ đóng kết nối TCP lại, kết nối không được sử dụng để lấy các đối tượng khác. Lưu ý rằng, mỗi kết nối TCP chuyển duy nhất một thông điệp yêu cầu và một thông điệp trả lời. Như vậy, trong ví dụ trên, client yêu cầu toàn bộ đối tượng trên trang Web thì sẽ có thể có tới 11 kết nối TCP được thiết lập.

Trong ví dụ trên, người ta không hề nói đến việc client nhận được 10 file ảnh JPEG qua 10 liên kết TCP riêng rẽ hay một số file được nhận qua cùng một kết nối. Trên thực tế, người dùng có thể cấu hình cho trình duyệt điều khiển mức độ song song của các kết nối. Chế độ mặc định của trình duyệt thường là từ 5 đến 10 kết nối TCP song song, mỗi kết nối kiểm soát một cặp thông điệp yêu cầu/trả lời. Nhưng nếu người dùng không thích thì

có thể đặt số kết nối song song tối đa là 1, trong trường hợp này, 10 kết nối được thiết lập riêng lẻ. Trong chương sau chúng ta sẽ thấy rằng, cách kết nối song song làm giảm thời gian nhận được kết quả từ client.

### b) Kết nối liên tục

Có một vài nhược điểm trong kết nối không liên tục. Thứ nhất, khi kết nối mới được tạo ra, phía client và server phải tạo ra vùng đệm TCP (buffer) cũng như lưu giữ các biến TCP. Điều này chính là gánh nặng cho server khi có nhiều client cùng yêu cầu một lúc.

Với cách kết nối liên tục, server không đóng liên kết TCP sau khi gửi thông điệp trả lời. Các thông điệp yêu cầu và trả lời sau đó (giữa cùng một client và server) được gửi qua cùng một kết nối. Trong ví dụ trên, toàn bộ đối tượng trong trang Web (một file HTML và 10 file ảnh JPEG) được truyền nối tiếp nhau trên cùng một kết nối TCP. Ngoài ra, có thể các trang Web khác trên cùng server có thể được truyền qua một kết nối TCP. Thông thường thì HTTP server đóng liên kết khi liên kết không được sử dụng trong một khoảng thời gian nào đó.

Chế độ làm việc mặc định của phiên bản HTTP 1.1 là gửi liên tục. Trong trường hợp này, HTTP client gửi yêu cầu khi nhận được một tham chiếu (ví dụ, một siêu liên kết hay tham chiếu đến file ảnh), vì vậy client có thể gửi các yêu cầu liên tiếp. Khi nhận được yêu cầu thì server sẽ gửi các đối tượng nối tiếp nhau.

## 2.2.3. Khuôn dạng thông điệp HTTP

Các đặc tả HTTP 1.0 (RFC 1945) và HTTP 1.1 (RFC 2016) đặc tả khuôn dạng thông điệp HTTP. Có hai kiểu khuôn dạng HTTP là thông điệp yêu cầu và thông điệp trả lời.

### a) Thông điệp yêu cầu HTTP (HTTP request message)

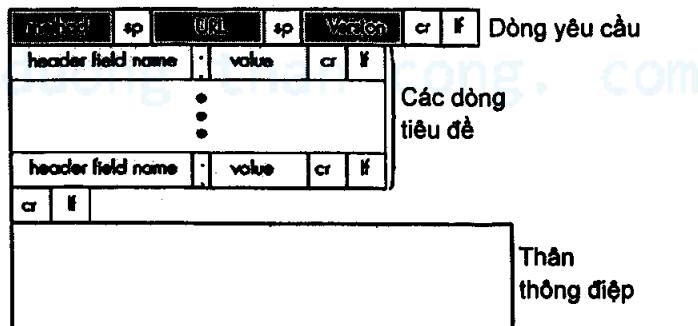
Một thông điệp yêu cầu thường có dạng sau:

**GET/somedir.page.html HTTP/1.1**  
**Host: www.someschool.edu**  
**Connection: close**  
**User-agent: Mozilla /4.0**  
**Accept-language: Fr**  
**(extra carry return line feed)**

Trước hết thấy rằng, thông điệp được viết bằng mã ASCII, vì thế bất kỳ máy tính thông thường nào cũng có thể đọc được. Thứ hai, thông điệp gồm 5 dòng và mỗi dòng đều kết thúc bởi cặp ký tự đặc biệt Carriage Return (CR = 13h) và Line Feed (LF = 10h). Trên thực tế, một thông điệp có thể có nhiều dòng hơn. Dòng đầu tiên của thông điệp được gọi là *dòng yêu cầu* (request line), các dòng sau gọi là *tiêu đề* (header). Dòng yêu cầu có 3 trường là trường method, trường địa chỉ URL và trường phiên bản HTTP. Trường method nhận một trong ba giá trị là GET, POST và HEAD. Phần lớn các yêu cầu sử dụng phương thức GET. Phương thức này được trình duyệt sử dụng để yêu cầu đối tượng có địa chỉ URL. Trong ví dụ trên, trình duyệt yêu cầu đối tượng **somedir/page.html**. Trường phiên bản xác định phiên bản giao thức HTTP (trong ví dụ là 1.1).

Bây giờ xét các trường trong tiêu đề. Host: **www.someschool.edu** là địa chỉ của máy tính có chứa đối tượng được yêu cầu. Ý nghĩa của trường **Connection: close** là trình duyệt yêu cầu server không sử dụng cách kết nối liên tục và yêu cầu server đóng kết nối lại sau khi đã gửi đi đối tượng được yêu cầu. Mặc dù client sử dụng phiên bản HTTP 1.1, nhưng lại không sử dụng kết nối liên tục. Trường **User-agent** xác định phần mềm trình duyệt của người sử dụng. Phần mềm trình duyệt ở đây là **Mozilla**, một sản phẩm của hãng Nestcape. Trường này rất quan trọng, vì server có thể gửi các bản khác nhau của cùng một đối tượng đến các trình duyệt khác nhau (các bản đối tượng này đều được xác định qua cùng một địa chỉ URL duy nhất). Cuối cùng là trường **Accept language**, trong ví dụ này người sử dụng yêu cầu bản tiếng Pháp của đối tượng – nếu server có bản này. Trong trường hợp không có, server gửi đi bản mặc định.

Hình 2.7 minh họa khuôn dạng chung của thông điệp yêu cầu.



Hình 2.7. Khuôn dạng thông điệp yêu cầu

Khuôn dạng tổng quát của thông điệp có thêm trường **Entity Body** sau các dòng tiêu đề. Trường này không được sử dụng trong phương thức GET, nhưng được sử dụng trong phương thức POST. HTTP client sử dụng phương thức POST khi người dùng điền vào một form – ví dụ khi muốn tìm kiếm qua máy tìm kiếm Google. Với phương thức POST, người dùng vẫn yêu cầu trang Web, nhưng nội dung cụ thể phụ thuộc vào nội dung điền trong form. Nếu giá trị của trường **method** là POST, thì phần **entity body** sẽ chứa nội dung mà người dùng điền vào form. Phương thức HEAD cũng tương tự như phương thức POST. Khi nhận được yêu cầu với phương thức POST, server sẽ gửi lại thông điệp HTTP trả lời, nhưng không gửi đối tượng được yêu cầu. Thường người ta sử dụng phương thức HEAD để gõ lỗi.

### b) Thông điệp trả lời (HTTP response message)

Sau đây là một ví dụ về thông điệp trả lời, thông điệp này có thể là trả lời cho thông điệp yêu cầu trên.

**HTTP /1.1 200 OK**

**Connection:close**

**Date: Thu, 06 Aug 1998 12:00:15 GMT**

**Server Apache/1. 3. 0 (unix)**

**Last modified:Mon, 22 Jun 1998 09:23:24 GMT**

**Connect lenght:6821**

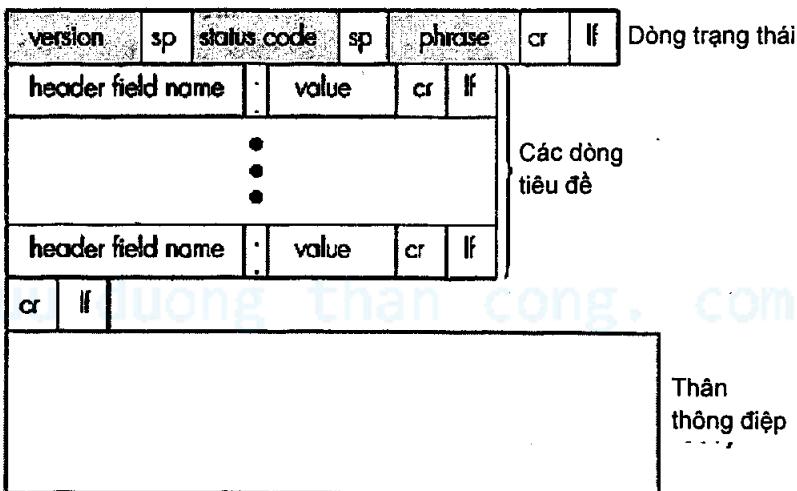
**Connect type:text/html**

**(data data,...,...)**

Thông điệp trên gồm có 3 phần: dòng đầu tiên là *dòng trạng thái* (status line), 6 dòng tiêu đề và cuối cùng là *phần thân* (Entity body) chứa đối tượng được yêu cầu (là phần data data,...). Dòng trạng thái có 3 trường là trường phiên bản của giao thức, mã trạng thái và trường trạng thái thông điệp trả lời. Trong ví dụ này, dòng trạng thái cho biết server sử dụng phiên bản HTTP 1.1 và trạng thái là sẵn sàng (server đã nhận được yêu cầu và gửi đối tượng được yêu cầu).

Trường **Connection: close** báo cho client biết server sẽ đóng kết nối sau khi gửi đi thông điệp. Trường **Date** cho biết thời gian khi server tạo ra thông điệp và gửi đi. Chú ý rằng, đây không phải là thời gian khi đối tượng được tạo ra hay lần cuối cùng đối tượng được cập nhật, mà đó là thời điểm mà

server tìm thấy đối tượng trong hệ thống file của mình, chèn đối tượng vào thông điệp trả lời và gửi đi. Trường **Server** cho biết thông điệp trả lời này được tạo ra từ phần mềm Web server Apache, ý nghĩa của nó giống với trường **User agent** trong thông điệp yêu cầu. Trường **Last modified** là thời gian cuối cùng đối tượng được cập nhật. Ta sẽ nghiên cứu kỹ hơn về trường này, nhưng chú ý rằng, nó có vai trò quan trọng đối với cả client và Web cache (proxy server). Trường **Content length** cho biết độ dài của đối tượng được gửi. Trường **Content type** xác định kiểu của đối tượng là file văn bản HTML (kiểu của đối tượng được đặt ở đây, chứ không phải trong phần mở rộng của tên file).



**Hình 2.8.** Khuôn dạng thông điệp trả lời

Chú ý, khi nhận được một thông điệp yêu cầu HTTP 1.0, server cũng sẽ không sử dụng kết nối liên tục ngay cả khi server dùng phiên bản 1.1. Server sẽ đóng kết nối ngay sau khi gửi đối tượng. Điều này là cần thiết, vì client sử dụng phiên bản HTTP 1.0 sẽ chờ server đóng kết nối lại.

Khuôn dạng chung của một thông điệp trả lời được minh họa trên Hình 2.8. Khuôn dạng này tương thích với ví dụ trên. Tuy nhiên, cần phải nói thêm về mã trạng thái (status code) và ý nghĩa của chúng. Mã trạng thái cùng với cụm từ đi sau cho biết kết quả đáp ứng yêu cầu. Sau đây là một vài giá trị thông dụng và ý nghĩa của chúng:

- ☞ **200 OK:** Yêu cầu được đáp ứng và dữ liệu được yêu cầu nằm trong thông điệp.

- ☞ **301 Moved permanently:** Cho biết đối tượng đã được chuyển và địa chỉ URL mới của đối tượng được đặt trong trường **Location** của thông điệp trả lời, phần mềm tại client sẽ tự động lấy đối tượng tại địa chỉ URL mới (đây là hiện tượng redirection thường gặp khi duyệt Web).
- ☞ **400 Bad Request:** Server không hiểu được yêu cầu từ client.
- ☞ **404 Not found:** Đối tượng không còn được lưu trên server.
- ☞ **505 HTTP version not support:** Server không hỗ trợ giao thức của client.

Trong phần này đã trình bày một số trường trong tiêu đề của thông điệp HTTP. HTTP (đặc biệt là bản 1.1) định nghĩa rất nhiều trường có thể được browser, Web server và Web cache chèn vào trong thông điệp. Ở trên mới đề cập đến một phần nhỏ, chi tiết có thể xem trong các đặc tả của HTTP.

Làm thế nào để trình duyệt cũng như server biết được phải chèn trường nào vào tiêu đề thông điệp? Thông điệp yêu cầu phụ thuộc vào chức năng trình duyệt cũng như phiên bản HTTP (HTTP 1.0 không thể tạo ra thông điệp kiểu HTTP 1.1). Người sử dụng có thể định cấu hình cho trình duyệt.

## 2.2.4. Tương tác giữa người dùng và HTTP server

Nhu đã nói trên, HTTP server không lưu giữ trạng thái. Điều này đơn giản hóa kiến trúc và làm tăng hiệu suất hoạt động của server. Tuy nhiên, server muốn phân biệt người dùng không chỉ vì muốn hạn chế sự truy cập, mà còn muốn phục vụ theo định danh người dùng. HTTP có 2 cơ chế để server phân biệt người dùng là Authentication và cookies.

### a) Authentication (Kiểm chứng)

Nhiều server yêu cầu người dùng phải cung cấp tên (username) và mật khẩu (password) để có thể truy cập được vào tài nguyên trên máy chủ. Yêu cầu này được gọi là kiểm chứng. HTTP có các mã trạng thái và trường để thực hiện quá trình kiểm chứng. Giả sử client yêu cầu một đối tượng từ server và server yêu cầu client cung cấp tên và mật khẩu. Đầu tiên client vẫn gửi một thông điệp yêu cầu thông thường. Server sẽ trả lời với thông điệp có phần thân rỗng và trường mã trạng thái là **401 Authentication required**. Trong thông điệp trả lời này, trường **www-Authenticate** xác định phương

thức kiểm chứng mà người dùng phải thực hiện, thông thường là đưa tên và mật khẩu. Nhận được thông điệp này, client yêu cầu người dùng cung cấp tên và mật khẩu. Sau đó, client sẽ gửi lại thông điệp yêu cầu có trường **Authoziration** trong tiêu đề, trường này chứa tên và mật khẩu của người dùng.

Sau khi nhận được đối tượng đầu tiên, client tiếp tục gửi tên và mật khẩu trong các thông điệp kế tiếp (Thường thì cho đến khi người dùng đóng trình duyệt lại. Khi trình duyệt còn mở, tên và mật khẩu được lưu lại trong cache để người dùng không phải nhập lại nữa). Theo cách này, server có thể phân biệt các người dùng khác nhau. HTTP phân biệt người dùng khá lỏng lẻo và không khó để vượt qua. Chúng ta sẽ nghiên cứu thêm về vấn đề bảo mật và sơ đồ xác nhận người dùng trong chương sau.

### b) *Cookie*

Cookie là kỹ thuật khác, được sử dụng để ghi lại thông tin của người truy cập. Nó được đặc tả trong RFC 2109. Ví dụ, lần đầu tiên người dùng truy cập vào một server nào đó có sử dụng cookie. Thông điệp trả lời của server có trường **Set-cookies** trong tiêu đề, cùng với một chuỗi ký tự do Web server tạo ra.

Ví dụ, **Set-cookies:1678453**. Khi nhận được thông điệp trả lời, client xác định được trường **Set-cookies** và chuỗi ký tự đi kèm, trình duyệt sẽ thêm một dòng vào cuối file cookie (là một file đặc biệt nằm trên máy client). Dòng này thường là dòng chứa tên máy chủ và chuỗi ký tự cookie. Giả sử một tuần sau, client gửi thông điệp yêu cầu đến server, client sẽ tự động chèn trường **Cookies** trong tiêu đề của thông điệp yêu cầu với giá trị là chuỗi giá trị cookie lưu trong file cookie. Trong ví dụ trên, tiêu đề chứa trường **Cookies** là **Cookie:1678453**. Theo cách này, server không xác định được tên của người dùng (user name), nhưng xác định được user này chính là người đã truy cập một tuần trước đó.

Web server sử dụng cookie cho nhiều mục đích:

- ☛ Nếu server yêu cầu kiểm chứng, nhưng không muốn đòi hỏi người dùng đăng nhập qua tên và mật khẩu, thì có thể sử dụng cookie cho mỗi lần người dùng truy cập vào server.
- ☛ Server sử dụng cookie nếu muốn ghi nhớ các hoạt động của người dùng, phục vụ mục đích quảng cáo.

- ☞ Nếu user mua hàng trên mạng (mua một đĩa CD chẳng hạn), thì server sử dụng cookie để ghi lại những gì mà user đã mua. Đó chính là các cửa hàng ảo.

Sử dụng cookie gây khó khăn cho người dùng không có máy cố định mà truy cập vào server từ nhiều máy khác nhau. Server sẽ coi đó là những người dùng phân biệt.

### 2.2.5. GET có điều kiện (Conditional GET)

Lưu giữ lại các đối tượng đã từng được lấy, Web cache có thể làm giảm thời gian chờ từ khi gửi yêu cầu đến khi nhận đối tượng và làm giảm lưu lượng thông tin truyền trên Internet. Web cache được triển khai trên trình duyệt hay các cache server. Chúng ta sẽ nghiên cứu network cache ở phần sau. Trong phần này chỉ quan tâm đến cache tại trình duyệt.

Mặc dù Web cache làm giảm thời gian chờ nhận đối tượng, nhưng vấn đề này sinh là bản sao của đối tượng được lưu giữ trên client có thể đã "cũ". Nói cách khác, đối tượng trên server có thể đã thay đổi từ khi client lấy đối tượng đó về. Tuy nhiên, HTTP có cơ chế cho phép sử dụng cache trong khi vẫn đảm bảo đối tượng trong cache chưa bị "cũ". Cơ chế này chính là *GET có điều kiện* (conditional GET). Một thông điệp HTTP được gọi là có điều kiện nếu: (1) thông điệp sử dụng phương thức GET; (2) thông điệp có trường **If-modified-since** trong tiêu đề. Ví dụ, trình duyệt yêu cầu một đối tượng từ server mà trong cache của nó chưa có:

**GET /fruit/banana.gif HTTP/1.0**

**User-agent:Mozilla/4.0**

Sau đó, server gửi thông điệp trả lời kèm với đối tượng

**HTTP /1.0 200 OK**

**Date: wed 12 aug 1998 15:38:29**

**Server: Apache/1.3.0 (Unix)**

**Last-modified: mon, 22 jun 1998 09:23:24**

**Content-type:image/gif**

**(data data,...,...)**

Trình duyệt hiển thị đối tượng đồng thời lưu lại đối tượng trong cache cục bộ cùng với thời gian trong trường **Last-modified** kèm theo đối tượng.

Một tuần sau, người sử dụng lại yêu cầu đối tượng này và đối tượng vẫn còn được lưu trên cache. Nhưng trên server đối tượng có thể đã bị thay đổi trong thời gian một tuần, nên trình duyệt phải thực hiện kiểm tra bằng cách gửi một thông điệp GET có điều kiện, cụ thể browser gửi đi:

**GET /fruit/kiwi.gif HTTP/1.0**

**User-agent: Mozilla /4.0**

**If-modified-since: Mon, 22 Jun 1998 09:23:24**

Chú ý, giá trị trường **If-modified-since** là giá trị của trường **Last-modified** trong tiêu đề mà server đã gửi cho client tuần trước. Thông điệp **GET** có điều kiện yêu cầu server chỉ gửi đối tượng cho client nếu như đối tượng đó bị thay đổi sau thời gian được chỉ ra ở trên. Giá sử đối tượng đó không thay đổi gì từ 9 giờ 23 phút 24 giây ngày 22 tháng 6 năm 1998, thì server sẽ gửi cho client thông điệp:

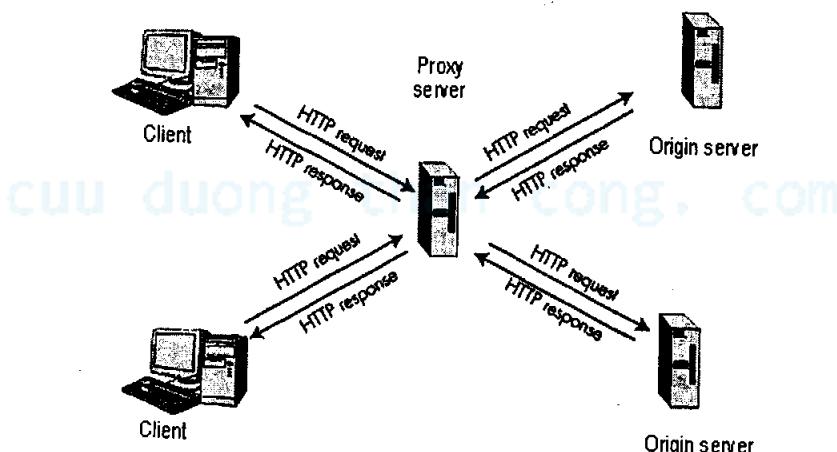
**HTTP /1.0 304 Not modified**

**Date: wed, 19 Aug 1998 15:39:29**

**Server: Apache /1.3.0 (Unix)**

**(empty entity body)**

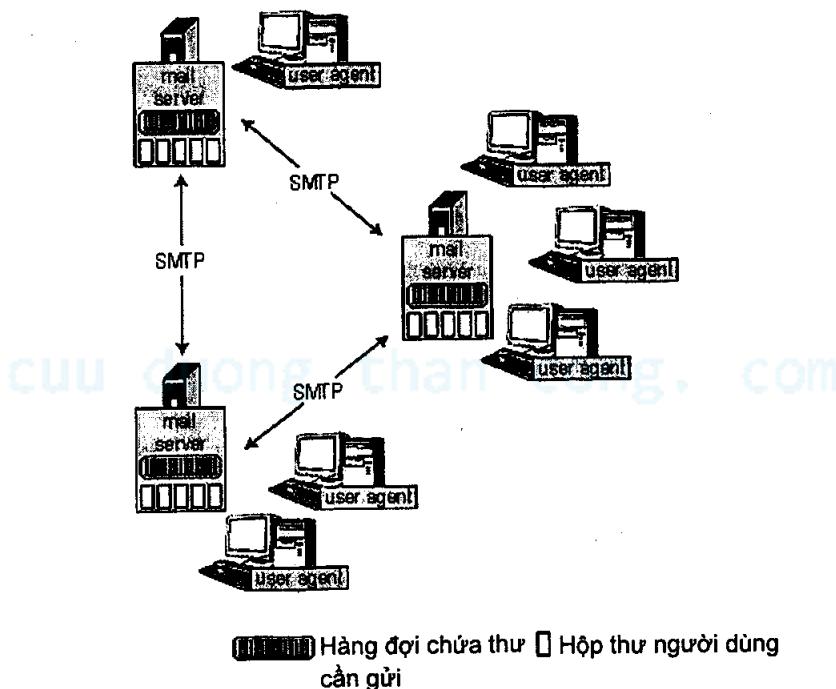
Thông điệp trả lời này không kèm theo đối tượng. Việc gửi kèm đối tượng chỉ làm lãng phí đường truyền và làm tăng thời gian client phải chờ để nhận được đối tượng, đặc biệt khi đối tượng có kích thước lớn. Giá trị trường trạng thái là **304 Not modified** báo cho client biết đối tượng mà client lưu trong cache giống đối tượng gốc tại server, do đó client có thể sử dụng lại đối tượng này.



**Hình 2.9. Client yêu cầu đối tượng thông qua cache**

## 2.3. THƯ TÍN ĐIỆN TỬ (e-mail) TRÊN INTERNET

Cùng với Web, thư điện tử là một trong những ứng dụng Internet thông dụng nhất. Gần giống thư tín thông thường, e-mail là dịch vụ không đòi hỏi đồng bộ, nghĩa là mọi người gửi và đọc thư khi thấy thuận tiện, không cần theo kế hoạch trước. Nhưng khác với thư tín thường, e-mail nhanh, dễ gửi và chi phí thấp. Hơn nữa, những thông điệp e-mail ngày nay có thể chứa đựng các hyperlink, văn bản định dạng HTML, hình ảnh, âm thanh và cả video. Trong phần này sẽ khảo sát các giao thức trao đổi thư thuộc tầng ứng dụng trên Internet.



Hình 2.10. Mô hình hệ thống email đơn giản

Hình 2.10 minh họa hệ thống mail trên Internet, gồm có 3 thành phần chính là user agent, mail server và SMTP (Simple Mail Transfer Protocol). Để tiện theo dõi, xét ví dụ Alice gửi e-mail cho Bob để mô tả 3 thành phần trên. Chương trình giao tiếp người dùng cho phép đọc, hồi âm, gửi, lưu giữ và soạn thảo các thư (user agent dành cho e-mail, còn được gọi là mail reader – trình đọc thư. Mặc dù vậy, trong cuốn sách này sẽ tránh sử dụng thuật ngữ đó). Khi Alice soạn thảo xong thư, user agent của Alice sẽ gửi thư tới mail server của Alice, tại đây thư được đặt vào hàng đợi để gửi ra ngoài. Khi Bob muốn đọc thư, user agent của Bob sẽ lấy thư trên hộp thư

(mail box) của Bob tại mail server. Trong những năm 90 của thế kỷ XX, các user agent có giao diện đồ họa GUI (Graphic User Interface) khá thông dụng, chúng cho phép người dùng có thể xem và soạn thảo các thư có gắn tài liệu đa phương tiện. Hiện nay, những phần mềm soạn e-mail thông dụng là Eudora, Microsoft Outlook và Nescape Messenger. Có nhiều chương trình user agent có giao diện dựa trên nền văn bản gõ lệnh như là mail, pine và elm.

Máy chủ phục vụ thư (Mail server) là thành phần cốt lõi trong hệ thống e-mail. Mỗi người có một hộp thư đặt trên mail server. Hộp thư của Bob quản lý, lưu giữ các thư gửi tới Bob. Thư được tạo ra tại user agent của người gửi, được gửi tới mail server của người gửi, rồi tới mail server của người nhận, và cuối cùng được chuyển vào hộp thư của người nhận. Khi Bob muốn truy cập vào hộp thư của mình, mail server chứa hộp thư của Bob sẽ kiểm chứng Bob ( thông qua username và password). Mail server của Alice cần phải xử lý khi mail server của Bob gặp sự cố. Nếu không thể gửi thư cho mail server của Bob, mail server của Alice sẽ giữ những thư đó trong hàng đợi gửi thông điệp và sẽ cố gắng gửi lại thông điệp. Quá trình gửi lại được tiến hành thường xuyên, 30 phút một lần trong năm ngày. Và sau đó, nếu vẫn không thành công thì server sẽ huỷ bỏ thư và gửi thư báo cho người gửi (Alice).

SMTP (Simple Mail Transfer Protocol) là giao thức gửi thư điện tử của tầng ứng dụng. SMTP sử dụng dịch vụ truyền dữ liệu tin cậy của TCP để truyền thư từ mail server của người gửi đến mail server của người nhận. Giống các giao thức khác ở tầng ứng dụng, SMTP có 2 phía là phía client trên mail server của người gửi và phía server trên mail server của người nhận. Tất cả các mail server đều chạy cả hai phía client và server của SMTP. Mail server đóng vai trò client khi gửi thư, đóng vai trò server khi nhận thư.

### 2.3.1. SMTP

SMTP là trái tim của dịch vụ gửi thư trên Internet và được đặc tả trong RFC 821. SMTP truyền các thông điệp (thư) từ mail server của người gửi đến mail server của người nhận. SMTP ra đời trước HTTP khá lâu (RFC đặc tả SMTP có từ năm 1982 và SMTP đã xuất hiện trước đó một thời gian dài).

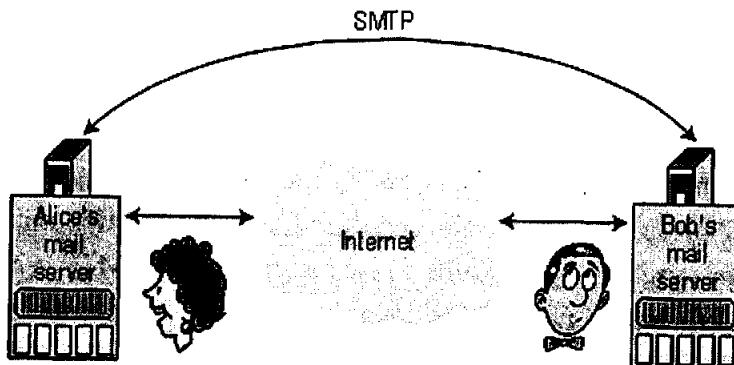
Mặc dù có nhiều ưu điểm, được tất cả mail server trên Internet sử dụng, SMTP vẫn là một công nghệ cũ, nên chắc chắn có những đặc tính "lạc hậu". Ví dụ, SMTP đòi hỏi phần thân của tất cả các thông điệp e-mail phải mã hoá theo bảng mã ASCII 7 bit. Sự hạn chế này là do trong những năm đầu thập kỷ 80 của thế kỷ XX, với số đường truyền ít ỏi, không ai gửi thư cùng với những phần đính kèm lớn, hay gửi kèm các file hình ảnh, âm thanh có kích thước lớn. Nhưng trong kỷ nguyên đa phương tiện ngày nay, việc giới hạn mã ASCII 7 bit là một hạn chế lớn, vì dữ liệu đa phương tiện nhị phân phải được chuyển sang mã ASCII trước khi được gửi đi qua SMTP, và sau đó lại phải giải mã thành mã nhị phân sau khi thư đến đích. Trong mục 2.2 đã biết rằng, HTTP không yêu cầu dữ liệu đa phương tiện phải mã hoá sang mã ASCII trước khi truyền.

Để minh họa hoạt động cơ bản của SMTP, xét ví dụ sau: Giả sử Alice muốn gửi cho Bob một thông điệp ASCII đơn giản:

- Đầu tiên, Alice sử dụng user agent của mình, đánh địa chỉ e-mail của Bob (bob@somechool.edu), soạn e-mail và yêu cầu user agent gửi thư đi.
- User agent của Alice gửi thư tới mail server của Alice. Tại đây thư được đặt vào hàng thư đợi gửi.
- SMTP client chạy trên mail server của Alice thấy thư trong hàng đợi. Nó tạo kết nối TCP tới SMTP server trên mail server của Bob.
- Sau giai đoạn khởi tạo 3 bước, SMTP client gửi thư của Alice qua kết nối TCP.
- Tại mail server của Bob, SMTP server nhận thư và đặt thư vào mail box của Bob.
- Cuối cùng, khi thuận tiện Bob sẽ sử dụng user agent của mình để đọc thư.

Kịch bản này được minh họa trên Hình 2.11. Một điểm quan trọng cần chú ý là SMTP không sử dụng mail server trung gian để gửi thư – ngay cả khi mail server gửi và nhận ở xa nhau. Ví dụ, nếu mail server của Alice đặt ở Hồng Kông và mail server của Bob ở Mobile tiểu bang Alabama, thì giữa hai mail server ở Hồng Kông và Mobile vẫn có đường kết nối TCP trực tiếp. Đặc biệt, nếu mail server của Bob bị hỏng, thì thư vẫn còn trong mail server

của Alice và đợi cho lần gửi sau. Thông điệp không được gửi qua mail server trung gian.



**Hình 2.11. Thư được gửi từ mail server của Alice đến mail server của Bob**

Bây giờ có thể xem chi tiết cách thức các mail server gửi thư bằng SMTP. Người ta thấy rằng, SMTP có nhiều đặc điểm tương tự như những quy tắc trong giao tiếp trực diện của con người. Đầu tiên, SMTP client (chạy trên mail server gửi) thiết lập kết nối TCP với cổng 25 tại SMTP server (chạy trên mail server nhận). Trong trường hợp server không làm việc, client sẽ cố gắng thử lại lần sau. Ngay khi kết nối được thiết lập, server và client thực hiện một vài thủ tục "bắt tay". Quá trình này tương tự như hai người tự giới thiệu về bản thân trước khi tiến hành nói chuyện. Trong thủ tục trao đổi, SMTP client thông báo với SMTP server địa chỉ e-mail người gửi và địa chỉ e-mail người nhận. Ngay sau quá trình giới thiệu, client sẽ gửi thư bằng dịch vụ truyền dữ liệu tin cậy của TCP. Sau đó, client sẽ lặp lại các bước này khi vẫn còn thông điệp khác để gửi tới server, còn nếu không, client yêu cầu TCP đóng kết nối lại.

Ví dụ sau là đoạn "hội thoại" giữa client (C) và server (S). Tên máy tính client là crepes.fr và server là hamburger.edu. Dòng hội thoại mở đầu bằng chữ C là đoạn hội thoại client gửi qua socket TCP và dòng hội thoại bắt đầu với chữ S là đoạn hội thoại server gửi đi thông qua socket TCP. Đoạn hội thoại bắt đầu ngay sau khi thiết lập được kết nối TCP:

**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.
```

Trong ví dụ trên, client gửi một thông điệp ("Do you like ketchup? How about pickles?") từ mail server crepes.fr tới mail server hamburger.edu. Client sử dụng 5 câu lệnh: HELO (viết tắt của HELLO), MAIL FROM, RCPT TO, DATA và QUIT. Ý nghĩa của những câu lệnh này có thể đoán được qua tên gọi của nó. Server gửi trả kết quả thực hiện mỗi lệnh, kết quả này chứa một mã trạng thái và một lời giải thích tiếng Anh. Ở đây SMTP sử dụng kết nối liên tục. Nếu có nhiều thư để gửi tới cùng một mail server, thì mail server gửi sẽ gửi tất cả các thư trên cùng một kết nối TCP. Với mỗi thông điệp, client bắt đầu tiến trình gửi bằng lệnh HELO crepes.fr và chỉ gửi lệnh QUIT sau khi gửi tất cả thư.

Thường thường, người ta sử dụng Telnet để xem một đoạn hội thoại trực tiếp với SMTP server. Sử dụng Telnet serverName 25, trong đó serverName là tên mail server. Khi đó bạn đã thiết lập kết nối TCP giữa máy tính của bạn và mail server. Sau khi đánh lệnh này, bạn sẽ nhận được ngay lập tức mã trả lời 220 từ server. Sau đó hãy sử dụng các lệnh SMTP như HELO, MAIL FROM, RCPT TO, DATA, QUIT ở những thời điểm tương ứng. Nếu bạn Telnet qua SMTP server của ai đó, bạn có thể gửi tới họ theo cách này (không phải dùng user agent).

### 2.3.2. So sánh SMTP với HTTP

Cả hai giao thức đều được sử dụng để gửi file giữa các máy tính. HTTP chuyển file hoặc đối tượng từ Web server tới Web client (trình duyệt Web), SMTP chuyển file (là thông điệp thư điện tử) giữa các mail server. Khi

truyền file, cả 2 giao thức HTTP và SMTP cùng sử dụng kết nối liên tục. Điểm khác biệt cơ bản giữa hai giao thức là, HTTP là giao thức kiểu kéo (Pull protocol) – client "kéo" thông tin từ server về. Phía nhận (client) là phía thiết lập kết nối TCP. SMTP lại là giao thức theo kiểu đẩy (Push protocol) – client "đẩy" thông tin lên server. Phía gửi (client) là phía thiết lập kết nối TCP trước.

Ngoài dữ liệu văn bản, thông điệp còn có thể chứa các kiểu dữ liệu khác như âm thanh, hình ảnh. HTTP đặt các đối tượng này trong các thông điệp riêng rẽ để gửi. Với SMTP tất cả các đối tượng này được đặt trong cùng một thư điện tử.

### 2.3.3. Giao thức truy nhập mail

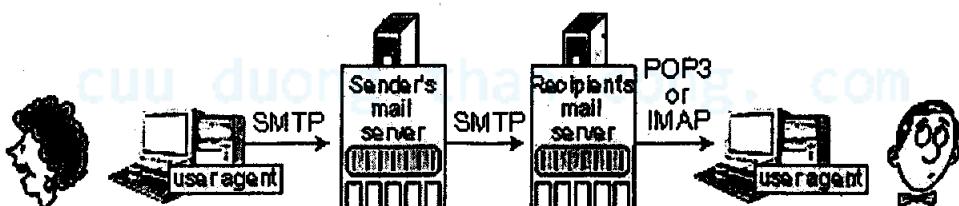
Mỗi khi SMTP gửi thư từ mail server của Alice tới mail server của Bob, thư được đặt trong mail box của Bob. Từ trước tới giờ, ta chấp nhận giả thiết để đọc thư, Bob phải đăng nhập vào mail server và sử dụng một chương trình đọc thư (mail reader) nào đó cài ngay trên mail server. Tới tận đầu những năm 90 của thế kỷ XX, mọi người vẫn thực hiện như vậy. Nhưng ngày nay, mọi người thường đọc thư qua một user agent chạy trên máy tính cá nhân của mình. Chạy user agent trên máy tính cá nhân, người sử dụng có được nhiều tính năng cao cấp, kể cả việc gửi và nhận những thông điệp đa phương tiện.

Giả sử Bob (người nhận) chạy user agent của mình trên máy tính cá nhân. Có thể cài đặt mail server ngay trên máy tính cá nhân của Bob. Tuy nhiên, cách này có nhiều nhược điểm. Mail server quản lý nhiều mail box, thực hiện cả chức năng client và server của SMTP. Nếu cài mail server trên máy tính cá nhân của Bob, thì máy tính đó lúc nào cũng phải bật và kết nối vào Internet để có thể nhận thư mới (mà thư thì có thể đến bất cứ lúc nào). Điều này không thực tế với đa số người sử dụng Internet. Thông thường, người sử dụng chạy chương trình user agent trên máy tính cá nhân, truy cập vào hộp thư trên một mail server dùng chung (mail server này luôn luôn kết nối tới Internet và được chia sẻ giữa nhiều người dùng khác). Mail server thường được ISP của người dùng (là trường đại học hoặc công ty) quản lý.

Do user agent chạy trên máy tính cá nhân và mail server được quản lý bởi các ISP, nên cần có một giao thức cho phép user agent và mail server

trao đổi với nhau. Đầu tiên, xét trường hợp thư được tạo ra tại PC của Alice được chuyển tới mail server của Bob như thế nào. Công việc này có thể được thực hiện một cách đơn giản bằng việc user agent của Alice trao đổi trực tiếp với mail server của Bob bằng giao thức SMTP. User agent của Alice sẽ khởi tạo một kết nối TCP tới mail server của Bob, gửi những lệnh khởi tạo SMTP, tải thư lên bằng lệnh DATA và sau đó đóng kết nối lại. Cách tiếp cận này hoàn toàn có thể thực hiện được, nhưng ít khi được dùng, vì nó không hỗ trợ trường hợp mail server phía nhận bị trục trặc. Trên thực tế, user agent gửi khởi tạo SMTP để tải thư của Alice tới chính mail server của Alice (chứ không phải là mail server của người nhận thư). Mail server của Alice sau đó sẽ thiết lập một phiên làm việc SMTP tới mail server của Bob để gửi tiếp thư tới mail server của Bob. Nếu mail server của Bob ngừng làm việc, thì mail server của Alice sẽ giữ thư lại, sau đó cố gắng gửi lại. RFC SMTP có những lệnh để gửi tiếp thư qua nhiều SMTP server.

Vậy user agent chạy trên máy tính cá nhân của Bob lấy thông điệp trong hộp thư trên mail server của Bob như thế nào? Giải pháp là phải có một giao thức lấy thư cho phép chuyển thư từ mail server của Bob tới máy tính cục bộ. Hiện nay có 2 giao thức lấy thư thông dụng là POP3 (Post Office Protocol – Version 3) và IMAP (Internet Mail Access Protocol). Dưới đây, sẽ trình bày cả hai giao thức này. Lưu ý rằng, user agent của Bob không thể sử dụng SMTP để lấy thư, bởi vì lấy thư giống như việc "kéo", trong khi SMTP là một giao thức "đẩy". Hình 2.12 minh họa về việc gửi và nhận thư. SMTP được dùng để chuyển thư giữa các mail server, hay giữa user agent của người gửi và mail server của người gửi. POP3 hay IMAP được dùng để chuyển thư từ mail server tới user agent của người nhận.



Hình 2.12. Giao thức email và các thành phần của truyền thông

#### a) Giao thức POP3

POP3 được đặc tả trong RFC 1939, là giao thức lấy thư cực kỳ đơn giản và có rất ít chức năng. POP3 được khởi tạo khi user agent (client) tạo kết nối

TCP tới mail server (server) qua cổng 110. Sau khi thiết lập được kết nối TCP, POP3 gồm 3 giai đoạn là kiểm chứng, tiến hành xử lý và cập nhật. Trong giai đoạn kiểm chứng đầu tiên, user agent sử dụng tên và mật khẩu để xác nhận người sử dụng. Trong giai đoạn tiến hành xử lý thứ hai, user agent tiến hành lấy thư. Nó có thể đánh dấu các thư để xoá hay huỷ bỏ việc đánh dấu xoá. Giai đoạn ba là cập nhật, xảy ra sau khi client ra lệnh QUIT để kết thúc phiên làm việc POP3. Tại thời điểm đó, mail server xoá tất cả thư được đánh dấu.

Trong giai đoạn xử lý, user agent gửi lệnh và server trả lời kết quả thực hiện của mỗi lệnh đó. Mỗi lệnh có hai trạng thái kết quả là +OK thông báo lệnh vừa gửi được thực hiện đúng và -ERR thông báo lệnh vừa gửi không thực hiện được.

Giai đoạn kiểm chứng có 2 lệnh là **user <username>** và **pass <password>**. Để minh họa hai lệnh này, nên Telnet trực tiếp qua một server POP3 sử dụng cổng 110 để thực hành. Giả sử mail server là tên mail server, có thể làm như sau:

```
telnet mail server 110
+OK POP3 server ready
user alice
+OK
pass hungry
+OK user successfully logged on
```

Nếu đánh sai một lệnh, thì server POP3 sẽ đáp lại bằng một thông điệp -ERR.

Người sử dụng có thể cấu hình user agent ở một trong hai chế độ "tải và xoá" ("download and delete") hay "tải và giữ" ("download and keep"). Chuỗi lệnh được user agent gửi phụ thuộc vào cấu hình này. Trong chế độ đầu, user agent sẽ phát ra chuỗi lệnh **list**, **retr** và **dele**. Giả sử người dùng có 2 thông điệp trong hộp thư của mình. Trong đoạn hội thoại dưới đây C (client) là user agent và S (server) là mail server. Khi đó giai đoạn xử lý công việc sẽ như sau:

```
C: list
S: 1 498
S: 2 912
```

```
S:.  
C: retr 1  
S: (blah blah,...  
S:.....,.....,....  
S:..... blah)  
S:.  
C: dele 1  
C: retr 2  
S: (blah blah,...  
S:.....,.....,....  
S:.....,.....blah)  
S:.  
C: dele 2  
C: quit  
S: +OK POP3 server signing off
```

Đầu tiên, user agent yêu cầu mail server liệt kê kích thước của tất cả thư lưu trữ trong hộp thư. Sau đó, user agent lấy và xoá từng thư trong hộp thư. Lưu ý rằng, sau giai đoạn kiểm chứng người dùng chỉ còn 4 câu lệnh là **list**, **retr**, **dele** và **quit**. Cú pháp của các lệnh này được đặc tả trong RFC 1939. Sau khi xử lý lệnh **quit**, server POP3 vào giai đoạn cập nhật và xoá thư 1, 2 trong mailbox.

Trong chế độ này, khi Bob lấy thư từ những địa điểm khác nhau, thư của Bob sẽ nằm rải rác trên nhiều máy. Đặc biệt, nếu Bob đã lấy thư từ máy tính ở nhà thì sau đó sẽ không thể đọc lại thư đó trên máy tính ở cơ quan. Trong chế độ thứ hai "download and keep", user agent vẫn để lại thư trên mail server sau khi đã tải về. Khi đó Bob vẫn có thể đọc thư từ nhiều máy khác nhau.

Trong phiên làm việc POP3 giữa user agent và mail server, server POP3 sẽ ghi nhớ một vài thông tin trạng thái, ví dụ các thư đã bị đánh dấu xoá. Tuy nhiên, server POP3 không chuyển thông tin trạng thái giữa các phiên làm việc khác nhau, ví dụ không có thư nào được đánh dấu xoá ở đầu mỗi phiên làm việc. Điều này làm đơn giản công việc xây dựng một server POP3.

### b) Giao thức IMAP

Sau khi tải thư về từ máy tính cá nhân, Bob có thể tạo những thư mục chứa thư và chuyển thư vào trong các thư mục đó. Sau đó Bob có thể xoá,

chuyển thư giữa các thư mục, hay tìm kiếm thư theo tên người gửi và chủ đề thư. Phương thức như vậy bất tiện với người sử dụng muốn đọc thư từ nhiều nơi, vì họ thích duy trì phân cấp thư mục trên mail server để có thể truy cập được từ bất kỳ máy tính nào. POP3 không đáp ứng được yêu cầu này.

IMAP (đặc tả trong RFC 2060) có thể giải quyết vấn đề này. Giống POP3, IMAP cũng là giao thức lấy thư. Nó có nhiều đặc tính phức tạp hơn POP3. IMAP được thiết kế cho phép người dùng thao tác trên những hộp thư ở xa một cách dễ dàng. IMAP cho phép Bob tạo những thư mục thư khác nhau trong mailbox. Bob có thể đặt thư vào trong thư mục, hay dịch chuyển thư từ thư mục này đến những thư mục khác. IMAP cũng có lệnh cho phép tìm kiếm trên thư mục theo tiêu chí xác định. IMAP phức tạp hơn POP3 nhiều, vì server IMAP phải duy trì hệ thống thư mục cho mọi người dùng. Những thông tin trạng thái như thế phải được mail server lưu giữ cho tất cả các phiên làm việc. Nên nhớ rằng, POP3 server không lưu giữ trạng thái của mỗi người dùng sau khi phiên làm việc kết thúc.

IMAP có một đặc tính quan trọng là có những lệnh cho phép user agent chỉ lấy một số thành phần trong thư. Ví dụ, user agent có thể lấy phần tiêu đề hoặc một phần trong thư có nhiều phần. Điều này rất có ích khi kết nối giữa useragent và mail server chậm, người dùng có thể không cần tải tất cả thư trong hộp thư của mình. Đặc biệt có thể tránh tải những thư chứa nội dung âm thanh hay hình ảnh có kích thước lớn.

Phiên làm việc IMAP gồm 3 giai đoạn là giai đoạn thiết lập kết nối giữa client (user agent) và IMAP server; giai đoạn server chấp nhận kết nối và giai đoạn tương tác client/server. Tương tác client/server của IMAP tương tự, nhưng phong phú hơn nhiều tương tác trong POP3. Server luôn ở một trong bốn trạng thái. Trong trạng thái *chưa kiểm chứng* (nonauthenticated) là trạng thái khởi đầu, khi đó người dùng phải đăng nhập hệ thống trước khi thực hiện các lệnh. Trạng thái *đã kiểm chứng* (authenticated), người dùng phải chọn một thư mục trước khi gửi lệnh. Trong trạng thái *lựa chọn* (selected), người dùng có thể sử dụng những lệnh có thể tác động tới thông điệp như lấy, xoá, chuyển thư. Cuối cùng là trạng thái *thoát* (logout) khi kết thúc phiên làm việc.

### c) Giao thức HTTP

Ngày nay, nhiều người sử dụng Webmail – dịch vụ có thể truy cập email qua trình duyệt (ví dụ, Hotmail hay Yahoomail). Khi đó, user agent là trình duyệt Web thông thường và mọi người kết nối tới hộp thư của mình trên mail server qua HTTP. Khi Bob muốn đọc thư, thư được gửi từ mail server của Bob tới trình duyệt nhờ giao thức HTTP chứ không phải là giao thức POP3 hay IMAP. Khi người gửi (Alice) có một tài khoản (account) trên mail server muốn gửi thư, thư sẽ được gửi từ trình duyệt của Alice tới mail server nhờ giao thức HTTP chứ không sử dụng SMTP. Tuy nhiên, mail server đó vẫn gửi và nhận thư với những mail server khác bằng giao thức SMTP. Giải pháp truy cập mail kiểu như vậy vô cùng thuận tiện cho người sử dụng hay phải di chuyển. Họ chỉ cần sử dụng một trình duyệt để gửi và nhận thư. Trình duyệt đó có thể ở một quán cafe Internet, ở nhà của bạn bè, ở khách sạn với Web TV,... Tương tự như IMAP, người dùng có thể tổ chức thư của mình trong hệ thống thư mục trên mail server. Sự thật là, e-mail dựa trên Web ngày càng thuận tiện và dần dần thay thế POP3 và IMAP trong những năm tới. Nhược điểm chính yếu của nó là chậm, bởi vì server và client ở xa, giao tiếp giữa chúng phải thông qua CGI.

## 2.4. DỊCH VỤ TÊN MIỀN – DNS

Đặc điểm cá nhân mỗi con người có thể được xác định theo nhiều cách. Ví dụ, chúng ta có thể được nhận biết qua tên trong giấy khai sinh, bằng số chứng minh thư nhân dân. Dù có nhiều cách nhận biết để phân biệt mọi người, nhưng lựa chọn phương thức nhận biết phụ thuộc vào hoàn cảnh. Ví dụ, công an sử dụng số chứng minh thư nhân dân chứ không sử dụng tên; bình thường mọi người thích nhớ tên nhau hơn là số chứng minh thư.

Cũng tương tự như việc xác định đặc điểm cá nhân của con người, máy tính trên Internet cũng có thể được xác định bằng nhiều cách. Tên máy tính (host name) là một cách, ví dụ cnn.com, www.yahoo.com, gais.umass.edu hay surf.eurecom.fr. Những tên đó tương đối dễ nhớ đối với con người. Tuy nhiên, tên máy tính cung cấp ít thông tin về vị trí trên Internet của máy tính (Tên máy tính surf.eurecom.fr chỉ cho người sử dụng biết máy tính đó ở nước Pháp, vì kết thúc bằng đuôi .fr, ngoài ra không còn thông tin nào khác). Hơn nữa, tên máy tính bao gồm nhiều ký tự, cả chữ cái và chữ số, có

độ dài thay đổi nên router khó có thể xử lý được. Vì lý do đó, máy tính được xác định thông qua địa chỉ IP. Địa chỉ IP được trình bày chi tiết trong Chương 4. Địa chỉ IP gồm có 4 byte và có cấu trúc phân cấp, giá trị mỗi byte lại được đổi ra số thập phân (từ 0 đến 255) và cách nhau bằng dấu chấm (ví dụ: 121.23.45.5). Địa chỉ IP phân cấp, khi duyệt địa chỉ từ trái qua phải, ta nhận được thêm nhiều thông tin xác định về vị trí của máy tính trên Internet (Vị trí ở trong mạng của các mạng, trong một mạng,...). Điều này tương tự khi xét địa chỉ bưu điện từ dưới lên, ta nhận được nhiều thông tin về địa chỉ đó.

#### 2.4.1. Các dịch vụ của DNS

Có hai cách để xác định một máy tính là dựa vào tên máy tính hoặc địa chỉ IP. Con người thích sử dụng tên máy vì dễ nhớ, trong khi router lại sử dụng địa chỉ IP có cấu trúc phân cấp và độ dài cố định (dễ xử lý). Để dung hoà giữa hai cách, cần một dịch vụ chỉ dẫn để chuyển đổi tên máy tính sang địa chỉ IP, đây chính là nhiệm vụ của hệ thống tên miền trên Internet (DNS). DNS là (1) Cơ sở dữ liệu phân tán được đặt trên một hệ thống phân cấp các máy phục vụ tên (name server) và (2) Giao thức thuộc tầng ứng dụng cho phép máy tính và máy chủ tên trao đổi thông tin phục vụ mục đích xác định địa chỉ IP. Máy chủ tên miền thường là các máy UNIX, có cài đặt phần mềm Berkely Internet Name Domain (BIND). Giao thức DNS chạy trên nền UDP với số hiệu cổng là 53.

Thông thường, DNS được các giao thức tầng ứng dụng khác như HTTP, SMTP và FTP sử dụng để xác định địa chỉ IP từ tên máy tính do người dùng đưa vào. Chuyện gì xảy ra khi trình duyệt (HTTP client) trên máy tính của người sử dụng yêu cầu đối tượng có địa chỉ URL là [www.someschool.edu/index.html](http://www.someschool.edu/index.html). Để gửi được thông điệp HTTP yêu cầu tới Web server, máy tính của người sử dụng phải xác định được địa chỉ IP của [www.someschool.edu](http://www.someschool.edu). Điều này được thực hiện như sau: Máy tính của người sử dụng chạy phía client của ứng dụng DNS. Trình duyệt sẽ lấy ra tên máy tính ([www.someschool.net](http://www.someschool.net)) từ địa chỉ URL và chuyển nó cho phần mềm client của DNS. DNS client gửi một truy vấn (query) chứa tên máy tính tới DNS server. DNS client sẽ nhận được một thông điệp trả lời từ DNS server chứa địa chỉ IP cần xác định. Sau đó trình duyệt sẽ mở một kết nối TCP tới tiến trình HTTP server trên máy tính có địa chỉ IP vừa được xác định.

Rõ ràng các ứng dụng Internet sử dụng DNS hoạt động chậm đi. Tuy nhiên, địa chỉ IP đã được xác định thường được ghi tạm (cache) trong một name server DNS ở gần và như vậy làm giảm tải cho hệ thống DNS cũng như độ trễ của ứng dụng.

Bên cạnh dịch vụ xác định địa chỉ IP từ tên máy, DNS cung cấp một số dịch vụ quan trọng sau:

**a) Dịch vụ đặt bí danh cho máy tính (host aliasing)**

Máy tính có tên phức tạp có thể có một hoặc nhiều bí danh (alias). Ví dụ, tên máy tính relay1.west-coast.enterprise.com có thể có hai bí danh là www.enterprise.com và enterprise.com. Trong trường hợp này, relay1.west-coast.enterprise.com là tên đầy đủ (canonical name). Tên bí danh thường dễ nhớ hơn tên đầy đủ. Một ứng dụng có thể yêu cầu DNS xác định tên đầy đủ cũng như địa chỉ IP của một tên bí danh.

**b) Dịch vụ đặt bí danh cho mail server (mail server aliasing)**

Hiển nhiên địa chỉ email cần dễ nhớ. Ví dụ, nếu Bob có tài khoản trên Hotmail, địa chỉ của Bob có thể chỉ đơn giản là bob@hotmail.com. Tuy nhiên, tên máy tính của máy phục vụ thư tại Hotmail phức tạp và vì thế khó nhớ hơn so với hotmail.com (Ví dụ, tên đầy đủ có thể là relay1.west-coast.hotmail.com). Ứng dụng có thể sử dụng DNS để xác định tên đầy đủ của một bí danh cũng như địa chỉ IP của máy tính đó. Trên thực tế, DNS cho phép mail server và Web server của các công ty có tên (bí danh) giống nhau, ví dụ: Web server và mail server của một công ty có thể cùng là enterprise.com.

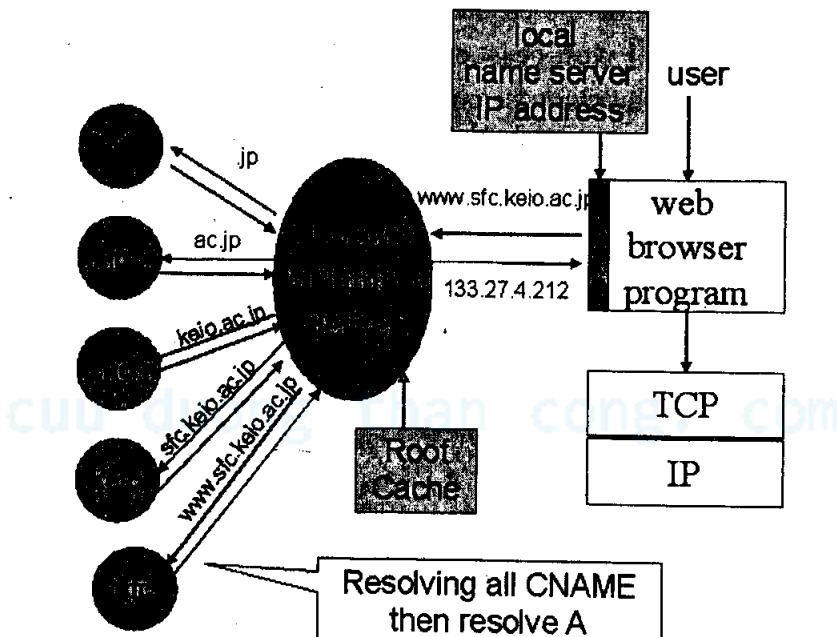
**c) Phân tán tải (load distribution)**

DNS thực hiện việc phân tán tải cho các server, đặc biệt là các Web server "nhân bản" (replicated) (là các server có nội dung giống hệt nhau). Những site có nhiều người truy cập như cnn.com được đặt trên nhiều server giống hệt nhau. Mỗi server là một hệ thống đầu/cuối (end system) khác nhau, có địa chỉ IP khác nhau. Đối với các server giống hệt nhau như vậy, một nhóm địa chỉ IP sẽ gắn với tên đầy đủ của một máy nào đó. Cơ sở dữ liệu DNS chứa toàn bộ nhóm địa chỉ IP đó. Khi client gửi truy vấn DNS để xác định địa chỉ IP, thì server sẽ gửi toàn bộ nhóm địa chỉ IP đó, nhưng server thay đổi thứ tự các địa chỉ IP trong nhóm. Thông thường, client gửi thông điệp HTTP tới máy tính có địa chỉ IP được liệt kê đầu tiên trong

nhóm. Sự hoán chuyển vị trí các địa chỉ IP mà DNS thực hiện đã phân tài cho các server. Việc hoán vị của DNS cũng được áp dụng cho e-mail khi nhiều mail server có chung bí danh.

DNS được đặc tả trong RFC 1034, RFC 1035 và cập nhật trong một số RFC khác. DNS là hệ thống phức tạp và chúng ta chỉ nghiên cứu một vài khía cạnh của nó.

#### 2.4.2. Cơ chế hoạt động của DNS



Hình 2.13. Ứng dụng sử dụng dịch vụ của DNS

Ở đây trình bày tổng quan cách thức hoạt động của DNS, tập trung vào dịch vụ xác định địa chỉ IP từ tên máy tính. Với client, DNS là một "hộp đen". Client gửi thông điệp truy vấn DNS vào hộp đen đó, trong thông điệp chứa tên máy cần xác định địa chỉ IP. Với hệ điều hành UNIX, `gethostname()` là một hàm mà ứng dụng có thể gọi để gửi thông điệp truy vấn. Sau một khoảng thời gian nào đó (từ vài phần nghìn giây đến vài chục giây), client nhận được thông điệp trả lời của DNS chứa địa chỉ IP cần xác định. Vì vậy, với client thì DNS là một dịch vụ xác định IP đơn giản và dễ hiểu. Nhưng "hộp đen" triển khai dịch vụ đó thực sự phức tạp, bao gồm nhiều máy chủ tên (name server) đặt khắp nơi trên thế giới và một giao thức

ở tầng ứng dụng xác định cách thức trao đổi thông tin giữa các name server và giữa name server với máy tính.

Để triển khai DNS, người ta có thể đưa ra một kiến trúc đơn giản sau: có một name server chứa tất cả các ánh xạ tên và địa chỉ IP. Theo thiết kế tập trung này, client chỉ cần gửi tất cả các truy vấn tới name server duy nhất và name server này sẽ trực tiếp trả lời mọi truy vấn. Mặc dù tính đơn giản của thiết kế này rất hấp dẫn, nhưng nó hoàn toàn không thích hợp cho Internet với số lượng lớn và ngày càng nhiều máy tính. Thiết kế tập trung như vậy này sinh một số vấn đề sau:

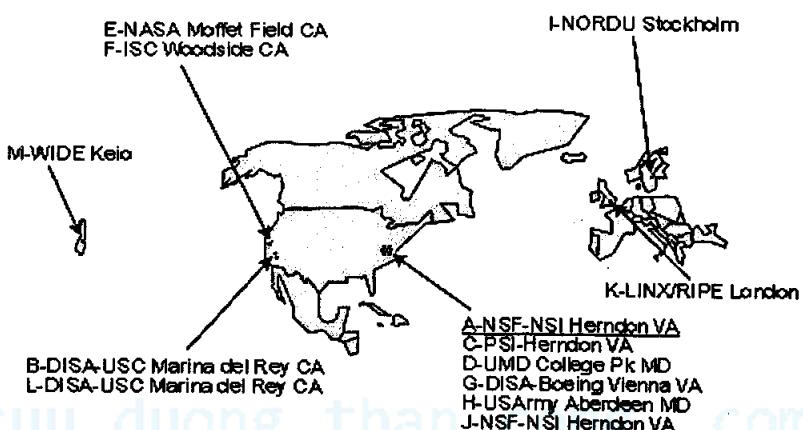
- ☞ **Điểm hỏng duy nhất** (A single point of failure): Nếu name server duy nhất ngừng làm việc cũng có nghĩa là toàn bộ Internet ngừng hoạt động.
- ☞ **Khối lượng công việc** (Traffic volume): Một name server duy nhất phải xử lý tất cả các truy vấn DNS (cho tất cả các thông điệp yêu cầu từ hàng triệu máy tính trên toàn cầu).
- ☞ **CSDL tập trung ở xa** (Distant centralized database): Name server duy nhất không thể gần tất cả các client. Nếu name server đặt ở New York, thì tất cả truy vấn từ Australia phải chuyển tới phía bên kia Trái Đất, có thể qua một đường kết nối chậm và tắc nghẽn. Hậu quả là các ứng dụng phải chịu độ trễ lớn.
- ☞ **Bảo trì** (Maintenance): Name server phải ghi nhớ thông tin về tất cả các máy tính trên Internet. Khi đó, cơ sở dữ liệu sẽ cực kỳ lớn, name server phải cập nhật thường xuyên thông tin cho mọi máy tính mới, cũng như phải giải quyết các vấn đề kiểm chứng và xác nhận khi người dùng sử dụng cơ sở dữ liệu tập trung.

Tóm lại, cơ sở dữ liệu tập trung trên một name server duy nhất không phù hợp khi quy mô hệ thống lớn. Do đó, DNS được thiết kế phân tán. Trên thực tế, DNS là một ví dụ tuyệt vời về triển khai cơ sở dữ liệu phân tán trên Internet. Để giải quyết vấn đề quy mô, DNS sử dụng nhiều name server tổ chức phân cấp và phân tán trên toàn cầu. Không có name server nào chứa tất cả tên và địa chỉ IP các máy tính trên Internet, những thông tin này được phân tán trên nhiều name server. Có ba loại name server là Local name server, Root name server và Authoritative name server. Các name server đó trao đổi thông tin với nhau và với các máy tính khác.

### a) Local name server

Mỗi ISP như trường đại học, công ty đều có local name server (còn được gọi là default name server). Khi máy tính trong cơ quan tạo ra một thông điệp truy vấn DNS, thì đầu tiên thông điệp đó được gửi tới local name server của tổ chức. Địa chỉ IP của local name server phải được cấu hình trong máy tính (Trong Win 95/98, có thể tìm thấy địa chỉ IP của local name server bằng cách mở Control Panel, sau đó chọn Network, chọn TCP/IP, rồi chọn cấu hình DNS). Local name server thường "gắn" với client, trong trường hợp tại cơ quan của một tổ chức, nó có thể ở trên cùng mạng LAN với máy tính client. Với ISP phục vụ kết nối từ nhà, thì khoảng cách giữa name server và các máy tính client chỉ là vài router. Nếu máy tính yêu cầu xác định địa chỉ IP của một máy tính khác trong cùng một ISP, thì local name server có thể ngay lập tức xác định được địa chỉ IP cần thiết. Ví dụ, nếu máy tính surf.eurecom.fr yêu cầu địa chỉ IP của baie.eurecom.fr, thì local name server ở eurecom ngay lập tức có thể đưa ra địa chỉ IP mà không phải liên hệ với bất kỳ name server nào khác.

### b) Root name server



Hình 2.14. Các root name server trên thế giới

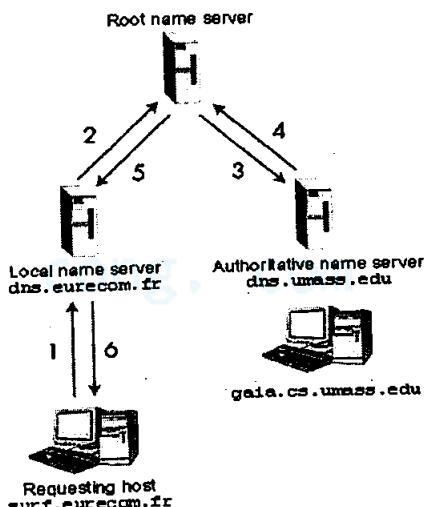
Trên Internet có 13 root name server, hầu hết đặt tại Bắc Mỹ. Vị trí các root name server vào thời điểm tháng 02/1998 được minh họa trên Hình 2.14. Khi local name server không thể trả lời truy vấn của một máy tính (bởi vì nó không có thông tin của máy tính được yêu cầu), thì local name server sẽ đóng vai trò client DNS và gửi câu hỏi truy vấn tới một trong số các root name server. Nếu root name server có thông tin của máy tính được hỏi, nó

sẽ gửi một thông điệp DNS hồi âm tới local name server và sau đó thông tin này được local name server gửi trả lời cho máy tính yêu cầu. Nhưng root name server có thể không có thông tin của máy tính đó, trong trường hợp này, root name server biết được địa chỉ IP của name server quản lý máy tính đó.

### c) Authoritative name server

Mỗi máy tính phải đăng ký tới một authoritative name server. Thông thường, authoritative name server của một máy tính là name server trong miền ISP của máy tính đó (thực tế mỗi máy tính phải có ít nhất hai authoritative name server để đề phòng trường hợp một name server bị hỏng). Có thể định nghĩa authoritative name server của một máy tính là name server luôn lưu trữ bản ghi DNS cho phép xác định địa chỉ IP của máy tính đó từ tên. Khi authoritative name server nhận được truy vấn từ root name server, nó sẽ gửi một thông điệp DNS trả lời chứa ánh xạ được yêu cầu. Sau đó, root name server gửi ánh xạ đó tới local name server và local name server lại tiếp tục gửi ánh xạ đó tới máy tính yêu cầu. Nhiều name server vừa là local name server, vừa là authoritative name server.

Xét ví dụ sau: Giả sử máy tính surf.eurecom.fr muốn có địa chỉ IP của máy tính gaia.cs.umass.edu, giả sử name server của miền Eurecom là dns.eurecom.com.fr và authoritative name server của gaia.cs.umass.edu là dns.umass.edu. Như đã trình bày trong Hình 2.15, đầu tiên máy tính surf.eurecom.fr gửi một thông điệp truy vấn tới local name server của nó là dns.eurecom.fr. Thông điệp đó chứa tên máy tính cần xác định địa chỉ IP là gaia.cs.umass.edu. Local name server gửi thông điệp tới root name server. Root name server gửi tiếp thông điệp tới name server có thể xác định tất cả các máy tính trong miền umass.edu, ví dụ là dns.umass.edu. Sau đó, authoritative name server này gửi kết quả cho surf.eurecom.fr thông qua root name server và local name server. Trong ví

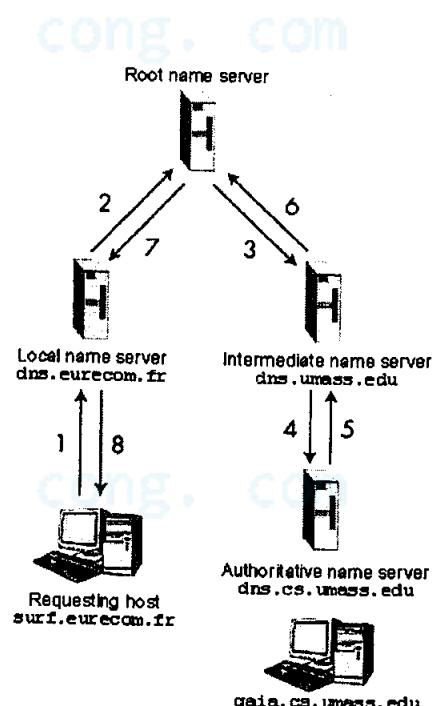


Hình 2.15. Các truy vấn DNS

dụ này, để xác định được địa chỉ IP, có 6 thông điệp DNS được trao đổi (3 thông điệp yêu cầu và 3 thông điệp trả lời).

Giả thiết root name server biết địa chỉ IP của authoritative name server của mọi máy tính như trên có thể không đúng. Với tên một máy tính, root name server có thể chỉ biết được địa chỉ IP của một name server trung gian mà chính name server trung gian này mới biết được địa chỉ IP của authoritative name server của máy tính đó. Để minh họa điều này, vẫn xét ví dụ máy tính surf.eurecom.fr cần xác định địa chỉ IP của gaia.cs.umass.edu. Giả sử trường Đại học Massachusetts (Univ of Massachusetts) có name server cho toàn bộ trường đại học, đó là dns.umass.edu. Ta cũng giả sử tiếp rằng, mỗi khoa trong trường đại học này có name server riêng, quản lý tên cho tất cả các máy tính trong khoa đó. Khi root name server nhận được yêu cầu xác định địa chỉ IP cho một tên máy tính có tận cùng là umass.edu, nó sẽ gửi yêu cầu tới name server dns.umass.edu. Name server này gửi tất cả các yêu cầu có tên máy tính tận cùng là cs.umass.edu cho authoritative name server quản lý tất cả máy tính có tên tận cùng là cs.umass.edu. Authoritative name server này (dns.cs.umass.edu) gửi kết quả tới name server trung gian (dns.umass.edu) và name server này sẽ gửi tiếp kết quả tới root name server. Root name server sẽ gửi tiếp kết quả tới local name server của máy tính yêu cầu. Trong ví dụ này, 8 thông điệp DNS được gửi (Hình 2.16). Thực ra có thể có nhiều hơn 8 thông điệp DNS được trao đổi, vì có thể có nhiều name server trung gian ở giữa root name server và authoritative name server.

Trong ví dụ trên, tất cả các truy vấn được gọi là đệ quy (recursive query). Khi máy tính hay name server A gửi thông điệp yêu cầu tới name server B, name server B sẽ thay mặt A nhận thông điệp chứa kết quả và sau

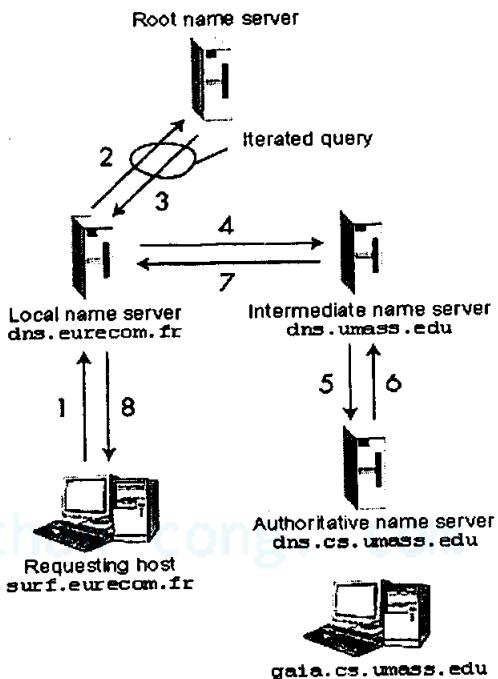


Hình 2.16. Các truy vấn DNS

đó gửi kết quả tới A. Tuy nhiên DNS cho phép các truy vấn tương tác (iterative query) ở bất kỳ giai đoạn nào trong quá trình từ máy tính yêu cầu đến authoritative name server. Khi name sever A gửi một truy vấn tương tác tới name server B, nếu name server B không có ánh xạ được yêu cầu, nó sẽ gửi cho A thông điệp trả lời chứa địa chỉ IP của name server kế tiếp trên chuỗi, giả sử là name sever C.

Sau đó name server A trực tiếp gửi thông điệp yêu cầu tới name server C.

Các truy vấn trong dãy truy vấn liên tiếp có thể là tương tác hoặc đệ quy như minh họa trên Hình 2.17. Thông thường, tất cả các truy vấn (ngoại trừ truy vấn từ local name server tới root name server) là đệ quy, truy vấn tới root name server thường là tương tác (bởi vì root name server phải xử lý một lượng lớn các yêu cầu, nên cần làm giảm số lượng truy vấn tới root name server).



Hình 2.17. Các truy vấn đệ quy và tương tác

Một đặc tính quan trọng của DNS là lưu trữ tạm thời các bản ghi DNS (DNS caching). Trên thực tế, DNS lưu trữ tạm thời (cache) để làm giảm độ trễ cũng như làm giảm số thông điệp DNS trao đổi trên mạng. Ý tưởng này rất đơn giản: Khi nhận được ánh xạ DNS của máy tính nào đó, bên cạnh việc gửi tiếp thông điệp, name server sẽ lưu ánh xạ này vào bộ nhớ cục bộ (ổ đĩa cứng hay RAM). Với ánh xạ tên máy – địa chỉ IP được lưu trữ sẵn, nếu có một truy vấn khác yêu cầu địa chỉ IP của cùng tên máy mà name server vừa lưu trữ, name server sẽ xác định ngay được địa chỉ IP mong muốn, kể cả khi nó không là authoritative name server cho máy tính đó. Để khắc phục tình trạng lưu trữ thông tin cũ, thông tin được lưu trữ tạm thời sẽ bị xoá bỏ sau một khoảng thời gian (thường là hai ngày).

### 2.4.3. Bản ghi DNS

Name server lưu giữ các bản ghi tài nguyên (resource record) cho các ánh xạ Tên máy / Địa chỉ IP. Mỗi thông điệp trả lời DNS chứa một hay nhiều bản ghi tài nguyên. Trong phần này sẽ trình bày qua về bản ghi tài nguyên và thông điệp DNS. Về chi tiết, bạn có thể xem trong DNS RFC [RFC 1034, RFC 1035].

Bản ghi tài nguyên gồm 4 trường là **Name, Value, Type, TTL**.

TTL là thời gian tồn tại của bản ghi tài nguyên, dùng để xác định thời điểm có thể xoá bản ghi tài nguyên khỏi bộ nhớ lưu trữ. Trong các bản ghi ví dụ dưới đây, ta bỏ qua trường TTL. Ý nghĩa của trường Name và Value phụ thuộc vào trường Type:

- ☞ Nếu **Type = A** thì Name là tên máy và Value là địa chỉ IP của máy đó. Bản ghi kiểu A là ánh xạ Tên máy → Địa chỉ IP chuẩn. Ví dụ, (relay1.bar.foo.com, 145.37.93.126, A) là một bản ghi Type A.
- ☞ Nếu **Type = NS** thì Name là một miền (như là foo.com) và Value là tên máy của authoritative name server của các máy tính trong miền đó. Bản ghi này thường được sử dụng để gửi tiếp các truy vấn DNS. Ví dụ, 1 bản ghi Type NS: (foo.com, dns.foo.com, NS).
- ☞ Nếu **Type = CNAME** thì Value là tên đầy đủ của máy có tên bí danh đặt trong Name. Bản ghi kiểu này cho phép xác định tên đầy đủ của một máy tính từ tên bí danh. Ví dụ, 1 bản ghi CNAME: (foo.com, relay1.bar.foo.com, CNAME).
- ☞ Nếu **Type = MX** thì Value là tên máy của mail server có tên bí danh đặt trong Name. Ví dụ, bản ghi kiểu MX (foo.com, mail.bar.foo.com, MX). Bản ghi MX cho phép mail server có tên bí danh đơn giản.
- ☞ Nếu một name server là authoritative name server cho một máy tính nào đó, thì name server sẽ chứa bản ghi kiểu A của máy tính đó (ngay cả nếu name server đó không là authoritative name server thì có thể nó chưa bản ghi kiểu A trong bộ nhớ cache của nó). Nếu name server không là authoritative name server của máy tính được hỏi, thì nó sẽ chứa một bản ghi kiểu NS cho miền của máy tính này, và nó cũng có một bản ghi kiểu A xác định địa chỉ IP của

name server của miền này đặt trong trường Value của bản ghi NS. Ví dụ, root name server không là authoritative name server cho máy tính gaia.cs.umass.edu, root server sẽ có một bản ghi cho miền chứa cs.umass.edu, ví dụ như umass.edu, dns.umass.edu, NS. Root server đó đồng thời cũng có một bản ghi kiểu A cho phép xác định địa chỉ IP của name server dns.umass.edu, chẳng hạn (dns.umass.edu, 128.119.40.111, A)

#### 2.4.4. Thông điệp DNS

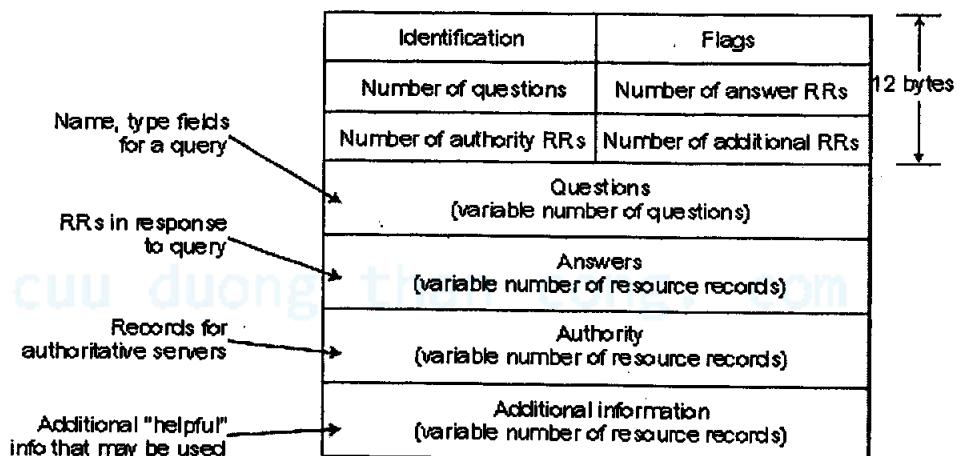
Có hai loại thông điệp DNS là thông điệp yêu cầu và thông điệp trả lời. Cả hai kiểu thông điệp này có chung khuôn dạng minh họa trên Hình 2.18.

Ý nghĩa các trường trong thông điệp như sau:

- ☞ 12 byte đầu tiên là phần tiêu đề. Phần tiêu đề có một số trường. Trường đầu tiên là một định danh 16 bit cho mỗi thông điệp yêu cầu. 16 bit định danh này được ghi lại vào thông điệp trả lời, cho phép client xác định được đây là câu trả lời cho thông điệp yêu cầu nào. Có nhiều cờ trong trường cờ (mỗi cờ ứng với một bit). *Cờ truy vấn* (query/reply flag) xác định thông điệp là yêu cầu (0) hay là trả lời (1). *Cờ authoritative* được đặt trong thông điệp trả lời khi name server là authoritative name server của tên máy tính cần xác định địa chỉ IP. *Cờ mong muốn để quy* (recursive – desired query) được đặt khi client (máy tính hay name server) mong muốn name server thực hiện truy vấn để quy khi nó không có bản ghi đó. *Cờ chấp nhận để quy* (recursion – available flag) được đặt trong thông điệp trả lời nếu name server đó hỗ trợ để quy. Trong phần tiêu đề cũng có 4 trường số lượng, các trường này xác định số lượng các bản ghi trong 4 phần dữ liệu sau phần tiêu đề.
- ☞ Phần câu hỏi (Question section) chứa thông tin về các câu hỏi được tạo ra. Nó bao gồm: (1) trường tên chứa tên đang được hỏi; (2) trường kiểu xác định kiểu câu hỏi cho tên máy tính đó (Kiểu A cho tên máy tính, kiểu MX cho mail server).
- ☞ Trong thông điệp trả lời từ server name, phần trả lời (answer section) chứa các bản ghi tài nguyên cho tên được yêu cầu trước đó. Chú ý rằng, mỗi bản ghi tài nguyên có 4 trường là Type (A, NS,

CNAME, MX), Name, Value, TTL. Thông điệp trả lời có thể có nhiều bản ghi tài nguyên, vì tên máy tính có thể ứng với nhiều địa chỉ IP.

- ☞ Mục thẩm quyền (authority section) chứa các bản ghi của các authoritative server.
- ☞ Mục phụ trợ (additional section) chứa các bản ghi "hữu ích" khác. Ví dụ, trường trả lời trong thông điệp trả lời một truy vấn MX sẽ chứa tên đầy đủ của mail server có tên bí danh đặt ở trong Name. Phần phụ trợ có thể chứa một bản ghi kiểu A cung cấp địa chỉ IP cho chính mail server đó.



Hình 2.18. Khuôn dạng thông điệp DNS

Các phần trên trình bày cách thức lấy dữ liệu trong cơ sở dữ liệu DNS. Vậy, làm thế nào để đưa được dữ liệu vào cơ sở dữ liệu? Cho tới gần đây, nội dung của server DNS được cấu hình tĩnh, ví dụ, thông qua file cấu hình được người quản trị hệ thống tạo ra. Gần đây, lựa chọn UPDATE được đưa vào giao thức DNS, cho phép dữ liệu được tự động thêm vào hay xoá bỏ khỏi cơ sở dữ liệu thông qua thông điệp DNS. RFC 2136 đặc tả quá trình cập nhật động của DNS.

## 2.5. LẬP TRÌNH SOCKET

Phần này giới thiệu quá trình phát triển ứng dụng mạng. Trong các phần trước, ta đã biết phần lớn các ứng dụng mạng gồm hai phía là Client và

Server. Hai phía trao đổi với nhau bằng cách gửi và nhận các thông điệp qua Socket. Phần này sẽ giới thiệu các bước chính trong quá trình phát triển các ứng dụng mạng. Sau đó sẽ giới thiệu một chương trình Web server.

Đầu tiên, chương trình phía server chạy và "lắng nghe" trên một cổng nào đó để đợi phía client kết nối tới. Nếu mọi việc diễn ra bình thường (kết nối thành công), cả hai phía đều có hai thể hiện của lớp Socket và dữ liệu sẽ được truyền qua hai socket này.

Để mở socket MyClient, phía client sẽ thực hiện khai báo và sau đó sử dụng phương thức tạo mới Socket như sau:

```
Socket MyClient;  
try {  
    MyClient = new Socket("Machine name", PortNumber);  
}  
catch (IOException e)  
{  
    System.out.println(e);  
}
```

trong đó **Machine name** là tên (hoặc địa chỉ IP) của máy tính server; **PortNumber** là số hiệu cổng mà tiến trình server đang chạy "lắng nghe" (đợi kết nối từ client). Chú ý, các port từ 0 đến 1023 được sử dụng cho các ứng dụng đặc biệt (HTTP, FTP, SMTP), nên hãy lựa chọn cổng có số hiệu lớn hơn 1023.

Phía server sẽ mở socket bằng cách sau:

```
ServerSocket MyService;  
try {  
    MyService = new ServerSocket(PortNumber);  
}  
catch (IOException e)  
{  
    System.out.println(e);  
}
```

Phía server cần phải khởi tạo đối tượng của lớp **ServerSocket** với mục đích lắng nghe và chấp nhận kết nối đến từ client.

```
Socket clientSocket = null;
try {
    serviceSocket = MyService.accept();
}
catch (IOException e)
{
    System.out.println(e);
}
```

### a) Tạo đối tượng Input Stream ở phía client

Phía client phải sử dụng lớp **DataInputStream** tạo input với mục đích nhận thông điệp trả lời từ server:

```
DataInputStream input;
try {
    input = new DataInputStream(MyClient.getInputStream());
}
catch (IOException e)
{
    System.out.println(e);
}
```

Lớp **DataInputStream** cho phép người sử dụng đọc từng dòng văn bản, hoặc các đối tượng thuộc kiểu cơ bản trong ngôn ngữ lập trình Java (chẳng hạn read, readChar, readInt, readDouble, và readLine).

Ở phía server, cũng sử dụng lớp **DataInputStream** để nhận thông điệp từ client:

```
DataInputStream input;
try {
    input = new DataInputStream(serviceSocket.getInputStream());
}
catch (IOException e)
{
    System.out.println(e);
}
```

### b) Tạo output stream

Phía client có thể sử dụng lớp **PrintStream** hoặc **DataOutputStream** của **java.io** để gửi thông điệp cho server.

```
PrintStream output;
try {
    output = new PrintStream(MyClient.getOutputStream());
}
catch (IOException e)
{
    System.out.println(e);
}
```

Lớp **PrintStream** có phương thức để hiển thị các kiểu dữ liệu cơ bản của Java dưới dạng văn bản (Chẳng hạn, các phương thức Write hoặc Println), còn lớp **DataOutputStream** có thể được sử dụng như sau:

```
DataOutputStream output;
try {
    output = new DataOutputStream(MyClient.getOutputStream());
}
catch (IOException e)
{
    System.out.println(e);
}
```

Lớp **DataOutputStream** cũng cho phép viết các kiểu dữ liệu cơ bản trong Java (ví dụ phương thức writeBytes).

Lớp **PrintStream** được phía server sử dụng để gửi thông điệp cho client.

```
PrintStream output;
try {
    output = new PrintStream(serviceSocket.getOutputStream());
}
catch (IOException e)
{
    System.out.println(e);
}
```

### c) Đóng kết nối

Phải đóng output và input stream trước khi đóng socket.

Ở phía client:

```

try {
    output.close();
    input.close();
    MyClient.close();
}
catch (IOException e)
{
    System.out.println(e);
}

```

Ở phía server:

```

try {
    output.close();
    input.close();
    serviceSocket.close();
    MyService.close();
}
catch (IOException e)
{
    System.out.println(e);
}

```

Phương thức **sendStaticResource** rất đơn giản, sử dụng tham số là đường dẫn đầy đủ của thư mục gốc (WEB\_ROOT) và tên file yêu cầu khởi tạo đối tượng file thuộc lớp **java.io.File**.

```
File file = new File(HttpServer.WEB_ROOT, request.getUri());
```

Đầu tiên sẽ kiểm tra xem file như vậy có tồn tại không. Nếu tồn tại, phương thức **sendStaticResource** sẽ tạo ra một đối tượng thuộc lớp **java.io.FileInputStream** từ đối tượng File. Kế tiếp sẽ sử dụng phương thức **read** của lớp **FileInputStream** để ghi luồng byte lên **OutputStream**. Nội dung của đối tượng tĩnh được gửi cho trình duyệt dưới dạng dữ liệu thô.

```

if (file.exists())
{
    fis = new FileInputStream(file);
    int ch = fis.read(bytes, 0, BUFFER_SIZE);
}

```

```
        while (ch != -1)
        {
            output.write(bytes, 0, ch);
            ch = fis.read(bytes, 0, BUFFER_SIZE);
        }
    }
```

Nếu file không tồn tại, phương thức **sendStaticResource** gửi thông báo lỗi cho trình duyệt

```
String errorMessage = "HTTP/1.1 404 File Not Found\r\n" +
"Content-Type: text/html\r\n" +
"Content-Length: 23\r\n" +
"\r\n" +
"<h1>File Not Found</h1>";
output.write(errorMessage.getBytes());
```

cuu duong than cong. com

cuu duong than cong. com

# Chương 3

## TẦNG GIAO VẬN

### 3.1. DỊCH VỤ VÀ NGUYÊN TẮC CỦA TẦNG GIAO VẬN

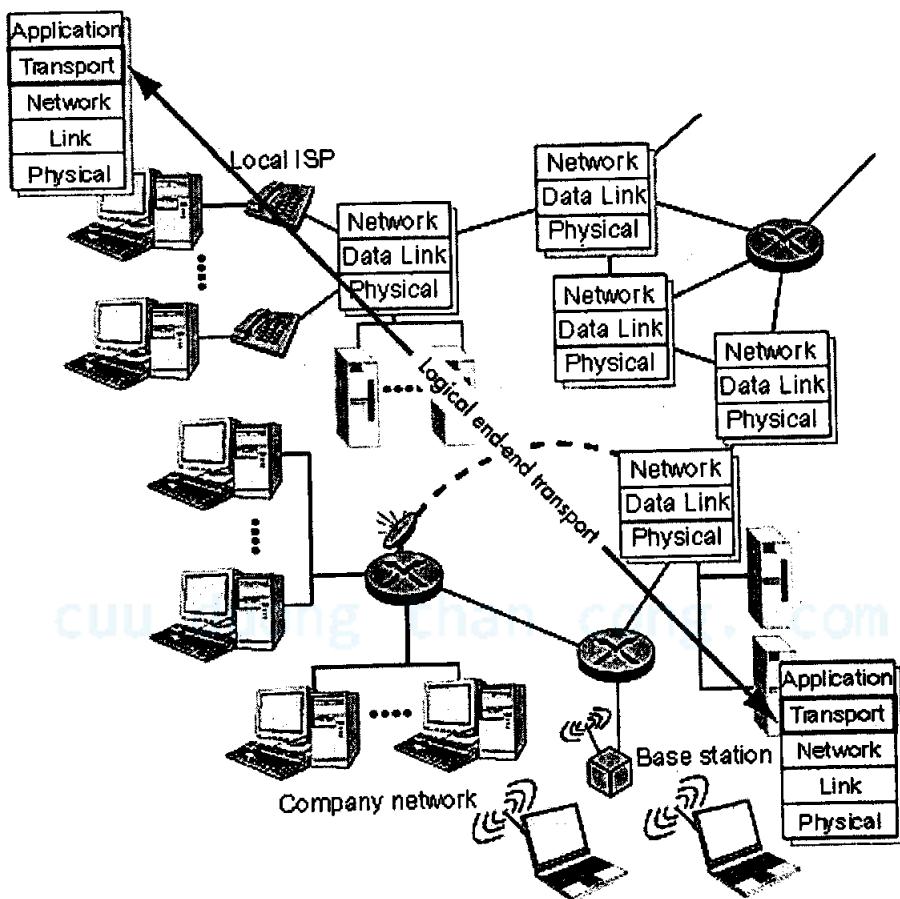
Tầng giao vận nằm giữa tầng ứng dụng và tầng mạng, nó là tầng trung tâm trong kiến trúc phân tầng, với nhiệm vụ cung cấp dịch vụ truyền thông giữa các tiến trình ứng dụng chạy trên các máy tính khác nhau. Chương này nghiên cứu tất cả dịch vụ của tầng giao vận cũng như các nguyên tắc cơ bản thực hiện điều này theo nhiều cách tiếp cận khác nhau; sẽ xem cách thức dịch vụ này được cài đặt trong các giao thức. Tầng giao vận của Internet có hai giao thức quan trọng là TCP và UDP.

Hai chương trước đã nói về vai trò và những dịch vụ mà tầng giao vận cung cấp. Giao thức tầng giao vận cung cấp một kênh truyền logic (ảo) giữa các tiến trình ứng dụng chạy trên máy tính khác nhau. Gọi là logic vì không tồn tại một đường truyền vật lý thực sự giữa hai tiến trình. Các tiến trình ứng dụng sẽ sử dụng đường truyền ảo này để trao đổi thông điệp mà không phải bận tâm về cơ sở hạ tầng của môi trường vật lý thực sự. Hình 3.1 minh họa điều này.

Trên Hình 3.1, tầng giao vận nằm trên các thiết bị đầu/cuối, chứ không phải ở các router. Router hoạt động ở tầng mạng.

Ở phía gửi, thực thể giao vận chèn thông điệp mà nó nhận được từ tiến trình ứng dụng vào các 4-PDU (là đơn vị dữ liệu của giao thức tầng giao vận – Protocol Data Unit). Công việc được thực hiện bằng cách chia thông điệp thành nhiều đoạn nhỏ, bổ sung vào đầu mỗi đoạn tiêu đề của tầng giao vận để tạo ra gói dữ liệu của tầng giao vận 4-PDU. Sau đó tầng giao vận truyền gói dữ liệu 4-PDU xuống tầng mạng, tại đây mỗi gói này được đặt trong gói dữ liệu của tầng mạng 3-PDU. Ở phía nhận, tầng giao vận nhận

gói dữ liệu từ tầng mạng, loại bỏ phần tiêu đề của gói dữ liệu 4-PDU, ghép chúng lại thành một thông điệp hoàn chỉnh và chuyển cho tiến trình ứng dụng nhận.



Hình 3.1. Tầng giao vận cung cấp dịch vụ truyền thông logic  
cho các tiến trình ứng dụng

Trên mạng máy tính có thể có nhiều giao thức giao vận khác nhau, cung cấp cho ứng dụng các dịch vụ với chất lượng khác nhau.

Tất cả giao thức tầng giao vận đều cung cấp dịch vụ *dồn kênh* (multiplex) và *phân kênh* (demultiplex), điều này sẽ nói cụ thể trong các phần sau. Như đã nói trong mục 2.1, ngoài dịch vụ dồn kênh/phân kênh, tầng giao vận còn có thể cung cấp các dịch vụ khác cho tiến trình ứng dụng như truyền dữ liệu tin cậy, đảm bảo băng thông hay giới hạn độ trễ.

### 3.1.1. Quan hệ giữa tầng giao vận và tầng mạng

Tầng giao vận nằm ở trên tầng mạng. Nếu giao thức tầng giao vận cung cấp đường truyền logic giữa tiến trình chạy trên các máy tính khác nhau, thì giao thức tầng mạng cung cấp đường truyền giữa các máy tính. Điểm khác biệt nhỏ này tuy khó nhận biết, nhưng rất quan trọng, xét ví dụ dưới đây.

Giả sử có hai nhà, một ở Hà Nội, một ở Huế, trong mỗi nhà có 12 đứa trẻ là anh em họ với nhau. Hàng tuần chúng trao đổi thư cho nhau, mỗi thư được đặt trong một phong bì riêng và được dịch vụ bưu chính gửi đi theo địa chỉ ghi trên phong bì. Hàng tuần mỗi nhà sẽ nhận 144 lá thư từ nhà bên kia (bọn trẻ có thể tiết kiệm được tiền nếu chúng sử dụng email). Ở mỗi nhà có một đứa trẻ chịu trách nhiệm thu thập và phân phát thư – An trong nhà phía tây, Bình trong nhà phía đông. Mỗi tuần, An lấy thư từ bọn trẻ trong nhà mình và chuyển cho nhân viên bưu cục – người thường xuyên ghé qua nhà để lấy và chuyển thư. Khi nhận thư từ nhân viên bưu tá, An chuyển tiếp thư cho người nhận. Bình cũng sẽ thực hiện công việc tương tự.

Trong ví dụ trên, dịch vụ bưu chính cung cấp đường truyền logic giữa hai nhà – chuyển thư từ nhà này đến nhà kia, chứ không phải từ người này đến người kia. Còn An và Bình cung cấp đường truyền logic giữa từng người trong hai nhà. Đối với lũ trẻ, An và Bình là dịch vụ chuyển thư, mặc dù An và Bình chỉ là một phần (phần đầu mút) của cả hệ thống chuyển thư. Qua ví dụ này ta hiểu được quan hệ giữa tầng giao vận và tầng mạng:

- ☞ Máy tính (hay thiết bị đầu/cuối) = Ngôi nhà.
- ☞ Tiến trình = Từng người trong ngôi nhà.
- ☞ Thông điệp ứng dụng = Thư trong phong bì.
- ☞ Giao thức tầng mạng = Dịch vụ bưu chính (gồm nhân viên bưu chính).
- ☞ Giao thức tầng giao vận = An và Bình.

Trong ví dụ trên, An và Bình thực hiện công việc phân phát thư tại chính ngôi nhà của mình, nhưng không thực hiện những việc như sắp xếp thư tại các bưu cục (là các trạm trung chuyển trên đường đi), hay gửi thư từ bưu cục này tới bưu cục khác. Tương tự, giao thức tầng giao vận chỉ hoạt động ở các thiết bị đầu/cuối. Tại thiết bị đầu/cuối, giao thức tầng giao vận

chuyển dữ liệu từ tiến trình ứng dụng xuống tầng mạng và ngược lại, nhưng không biết thông điệp được truyền đi như thế nào trong tầng mạng. Trên Hình 3.1, các router không xử lý bất kỳ thông tin tiêu đề nào mà tầng giao vận chèn vào bên cạnh thông điệp ứng dụng.

Giả sử An và Bình đi vắng, Hạnh và Phúc làm thay. Nhưng thật đáng tiếc hai đứa trẻ này còn quá nhỏ, không làm việc được cẩn thận như An và Bình, chúng làm mất thư. Tương tự như vậy, mạng máy tính có thể có nhiều giao thức giao vận, mỗi giao thức cung cấp các dịch vụ với chất lượng khác nhau cho chương trình ứng dụng.

Dịch vụ mà An và Bình cung cấp phụ thuộc vào dịch vụ của bưu điện. Ví dụ, nếu bưu điện không đảm bảo thời gian chuyển thư giữa hai nhà thì An và Bình cũng sẽ không đảm bảo được thời gian chuyển thư giữa những người trong hai nhà. Tương tự, dịch vụ của giao thức tầng giao vận cung cấp sẽ phụ thuộc vào dịch vụ của tầng mạng bên dưới. Nếu giao thức tầng mạng không đảm bảo thời gian trễ, hay đảm bảo về băng thông cho gói dữ liệu 4-PDU trong quá trình gửi giữa các máy tính, thì giao thức tầng giao vận cũng không thể cung cấp những dịch vụ này khi gửi thông điệp giữa các tiến trình ứng dụng.

Tuy nhiên, tầng giao vận vẫn có thể cung cấp những dịch vụ mà tầng mạng không cung cấp. Những dịch vụ như thế được nghiên cứu ngay trong chương này. Ví dụ, giao thức tầng giao vận cung cấp dịch vụ truyền dữ liệu tin cậy cho tầng ứng dụng ngay cả khi tầng mạng không đáng tin cậy – làm mất, gửi lỗi, hay gửi trùng lặp dữ liệu.

### 3.1.2. Tổng quan về tầng giao vận trong Internet

Trong mạng Internet hay mạng TCP/IP có hai giao thức ở tầng giao vận là UDP (User Datagram Protocol) và TCP (Transmission Control Protocol). **UDP cung cấp dịch vụ truyền không tin cậy, không hướng nội, còn TCP cung cấp dịch vụ tin cậy, hướng nội cho ứng dụng. Người viết ứng dụng phải lựa chọn một trong hai giao thức này cho ứng dụng của mình.**

Để đơn giản, trong mô hình Internet người ta coi 4-PDU là một segment. Trong các khuyến nghị RFC thì **4-PDU được coi là segment đối**

với TCP và *datagram* đối với UDP. Nói chung thuật ngữ *datagram* thường sử dụng cho PDU ở tầng mạng, nhưng trong cuốn sách nhập môn này, nói chung ít xảy ra nhầm lẫn khi sử dụng thuật ngữ *segment* cho cả TCP PDU và UDP PDU.

Để tiếp tục, trước hết trình bày qua về tầng mạng của Internet (Tầng mạng sẽ được nghiên cứu chi tiết trong Chương 4). Giao thức của tầng mạng là IP (Internet Protocol). IP cung cấp đường truyền logic giữa các máy tính và mô hình dịch vụ của nó theo kiểu *cố gắng tối đa* (best effort delivery service). Nghĩa là, IP cố gắng gửi các segment giữa các máy tính hay thiết bị đầu/cuối khác nhau với hết khả năng của mình, nhưng không đảm bảo điều này. Nói cụ thể hơn, IP không đảm bảo về thứ tự truyền, về tính toàn vẹn của dữ liệu trong segment. Chính vì thế, người ta xem IP là dịch vụ không tin cậy. Mỗi máy tính có một địa chỉ IP xác định. Trong chương này ta chỉ cần biết mỗi máy tính cần có một địa chỉ IP xác định duy nhất.

Nhiệm vụ chính của UDP và TCP là mở rộng dịch vụ IP (truyền dữ liệu giữa hai thiết bị đầu/cuối thành dịch vụ truyền dữ liệu giữa hai tiến trình chạy trên thiết bị đầu/cuối). Việc mở rộng từ truyền dữ liệu giữa các máy tính (host – to – host) đến truyền dữ liệu giữa các tiến trình (process – to – process) được gọi là *quá trình dồn kênh* (multiplex) và *phân kênh* (demultiplex). Vấn đề này sẽ nghiên cứu ở phần sau. UDP và TCP kiểm soát tính toàn vẹn (hay tính đúng đắn) của dữ liệu nhờ trường phát hiện lỗi đặt trong tiêu đề gói dữ liệu. UDP chỉ cung cấp dịch vụ phân phối dữ liệu giữa hai tiến trình và kiểm tra lỗi. Tương tự IP, UDP là dịch vụ không tin cậy, không đảm bảo dữ liệu được truyền đi một cách đúng đắn giữa các tiến trình. UDP được trình bày kỹ trong mục 3.1.

Ngoài phân kênh, dồn kênh, TCP còn cung cấp một số dịch vụ khác cho ứng dụng. Dịch vụ đầu tiên và quan trọng nhất là *truyền dữ liệu tin cậy* (reliable data transfer). Các cơ chế điều khiển lưu lượng, đánh số thứ tự, số thứ tự nhận, bộ định thời sẽ giúp TCP đảm bảo dữ liệu được truyền từ tiến trình gửi đến tiến trình nhận chính xác và đúng thứ tự. Như vậy, giao thức TCP đã biến dịch vụ truyền không tin cậy giữa các thiết bị đầu/cuối (IP) thành dịch vụ truyền dữ liệu tin cậy giữa các tiến trình.

Giao thức cung cấp dịch vụ truyền dữ liệu tin cậy và kiểm soát tắc nghẽn rất phức tạp.

### **3.2. DỊCH VỤ DÒN KÊNH, PHÂN KÊNH**

Trong phần này trình bày về công việc dồn kênh và phân kênh trong mạng. Để đơn giản, chỉ trình bày các dịch vụ của tầng giao vận trong mô hình Internet. Tuy nhiên cần nhấn mạnh rằng, đây là dịch vụ cần thiết đối với tất cả các mô hình kết nối mạng.

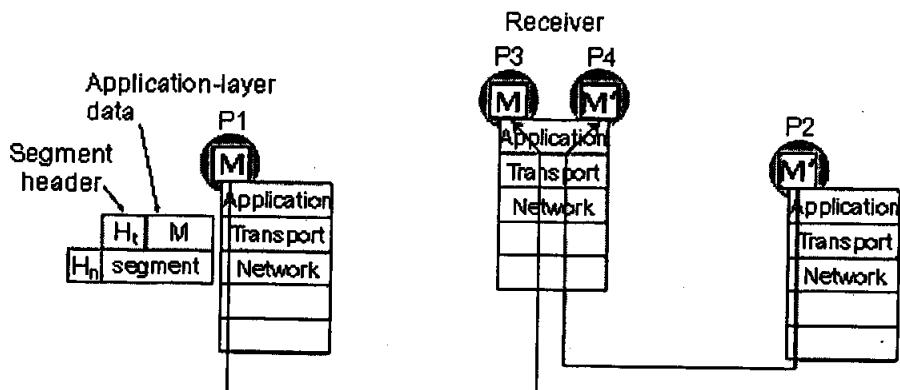
Mặc dù dồn kênh/phân kênh không phải là một trong những dịch vụ quan trọng nhất của tầng giao vận, nhưng nó cực kỳ cần thiết. Để hiểu tại sao như vậy, thấy rằng, tầng mạng truyền dữ liệu giữa hai thiết bị đầu/cuối, mỗi thiết bị có một địa chỉ tầng mạng nhất định. Tầng mạng không truyền dữ liệu giữa các tiến trình ứng dụng chạy trên các máy tính. Mở rộng việc gửi từ máy tính đến máy tính tới việc giữ tiến trình đến tiến trình là công việc dồn kênh và phân kênh.

Tại máy tính nhận, tầng giao vận nhận gói dữ liệu (hay còn gọi là segment) từ tầng mạng ngay phía dưới, đồng thời có trách nhiệm gửi dữ liệu bên trong segment này tới tiến trình ứng dụng thích hợp trên máy tính. Giả sử lúc nào đó máy tính đang tải trang Web xuống, chạy một phiên FTP và hai phiên Telnet cùng một lúc. Như vậy, đang chạy 4 tiến trình ứng dụng gồm 2 tiến trình Telnet, 1 tiến trình FTP và 1 tiến trình HTTP. Khi tầng giao vận trong máy tính nhận được dữ liệu từ tầng mạng chuyển lên, nó phải gửi dữ liệu tới 1 trong 4 tiến trình trên. Việc đó diễn ra như thế nào?

Mỗi segment của tầng giao vận có trường xác định tiến trình nhận dữ liệu. Tầng giao vận bên nhận sẽ sử dụng trường này để xác định rõ tiến trình nhận và gửi dữ liệu trong segment tới tiến trình đó. Công việc chuyển dữ liệu trong segment tới đúng tiến trình ứng dụng được gọi là *phân kênh*. Tại thiết bị gửi, tầng giao vận nhận dữ liệu từ nhiều tiến trình ứng dụng khác nhau, tạo segment chứa dữ liệu cùng với một số thông tin tiêu đề và cuối cùng chuyển segment xuống tầng mạng. Quá trình trên được gọi là *dồn kênh*. Hình 3.2 minh họa cả hai quá trình này.

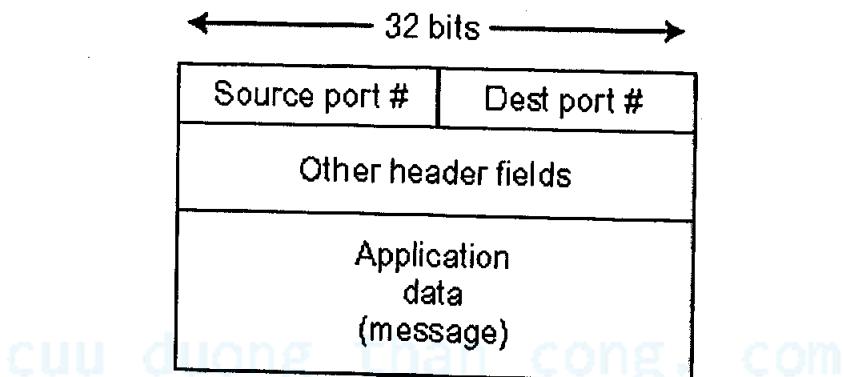
Để hiểu rõ hơn về dịch vụ dồn kênh, quay lại ví dụ trước. Mỗi đứa trẻ được xác định qua tên. Khi Bình nhận được thư từ người đưa thư, cậu bé sẽ thực hiện quá trình phân kênh bằng cách đọc tên trên phong bì thư để

chuyển cho đúng người nhận. Còn An thực hiện quá trình dồn kênh khi thu thập thư từ mọi người và chuyển cho người đưa thư.



Hình 3.2. Dịch vụ dồn kênh, phân kênh

UDP và TCP thực hiện việc dồn kênh và phân kênh nhờ hai trường đặc biệt ở đầu segment là trường định danh cổng tiến trình gửi (nguồn – source port number) và trường định danh cổng tiến trình nhận (đích – destination port number). Hai trường này được minh họa trên Hình 3.3. Chúng xác định một tiến trình ứng dụng duy nhất chạy trên máy tính. Tất nhiên, UDP và TCP còn có nhiều trường khác, sẽ được trình bày ở phần sau.



Hình 3.3. Trường địa chỉ tiến trình gửi, tiến trình nhận trong gói dữ liệu segment

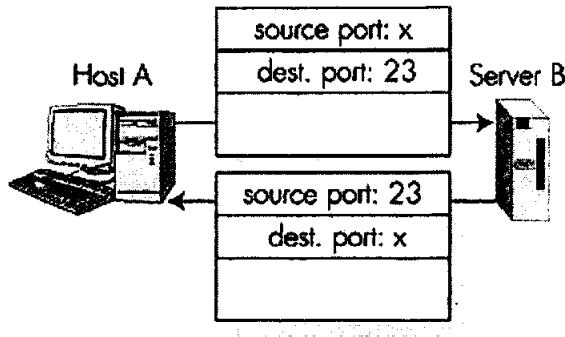
Khái niệm số hiệu cổng đã được giới thiệu qua trong Chương 2. Nó là một con số 16 bit, nhận giá trị từ 0 tới 65535. Giá trị từ 0 đến 1023 là các giá trị đặc biệt, được sử dụng rất hạn chế, chỉ dành cho các ứng dụng thông dụng như HTTP, FTP. HTTP sử dụng cổng 80, FTP sử dụng cổng 21. Danh

sách các cổng thông dụng có thể tham khảo trong RFC 1700. Khi xây dựng một ứng dụng mới, phải xác định số hiệu cổng cho ứng dụng này.

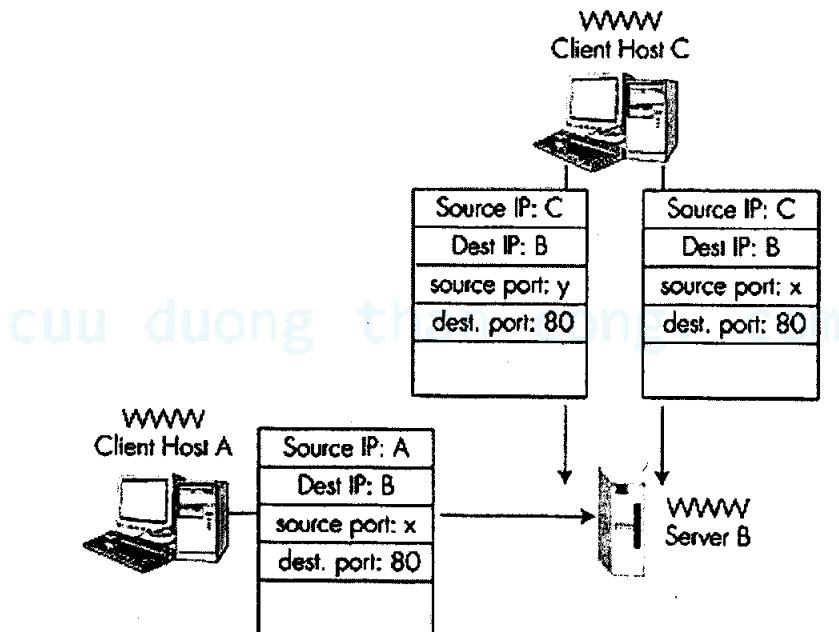
Mỗi ứng dụng chạy trên thiết bị đầu/cuối có số hiệu cổng nhất định. Bởi vậy, vấn đề đặt ra là: Tại sao mỗi segment ở tầng giao vận đều có trường số hiệu cổng nguồn và đích? Một thiết bị đầu/cuối có thể chạy đồng thời hai tiến trình cùng kiểu, như vậy, số hiệu cổng đích chưa đủ để phân biệt các tiến trình. Giả sử Web server chạy tiến trình HTTP, xử lý các thông điệp yêu cầu; khi Web server phục vụ nhiều yêu cầu cùng một lúc (điều này hết sức thông thường), thì server sẽ chạy nhiều tiến trình trên cổng 80. Để gửi dữ liệu đến tiến trình nhận, phải xác định số hiệu cổng của phía gửi (cổng nguồn).

Cổng nguồn được tạo ra như thế nào? Nhận giá trị bao nhiêu? Để trả lời câu hỏi này, nhớ lại rằng, ứng dụng mạng sử dụng kiến trúc khách hàng/người phục vụ. Thông thường, máy tính nào khởi đầu trước đóng vai trò client, máy tính kia đóng vai trò server. Xét ví dụ một tiến trình ứng dụng có số hiệu cổng là 23 (số hiệu cổng của ứng dụng Telnet server). Hãy quan sát segment ở tầng giao vận khi rời client (là máy tính chạy chương trình Telnet client) chuyển tới server. Số hiệu cổng nguồn và đích của segment này là bao nhiêu? Số hiệu cổng đích chính là số hiệu cổng tiến trình nhận (23). Còn số hiệu cổng nguồn ở phía client là một giá trị chưa được sử dụng bởi tiến trình nào, được phần mềm giao vận chạy trên máy tính client xác định tự động. Giả sử phía client chọn số hiệu cổng là x, thì mỗi segment được gửi tới ứng dụng Telnet có cổng nguồn là x, cổng đích là 23. Khi segment tới, server căn cứ vào số hiệu cổng để chuyển dữ liệu trong segment tới đúng tiến trình ứng dụng nhận. Cổng đích 23 xác định tiến trình Telnet, cổng nguồn x để xác định một tiến trình gửi cụ thể.

Segment truyền từ server tới client sẽ ngược lại. Cổng nguồn bây giờ sẽ là cổng của ứng dụng có giá trị 23, còn cổng đích sẽ là x (là số hiệu cổng nguồn trong segment gửi từ client tới server). Khi segment tới, client sẽ căn cứ vào số hiệu cổng nguồn và đích để gửi dữ liệu trong segment tới đúng tiến trình ứng dụng. Hình 3.4 minh họa quá trình trên.



**Hình 3.4.** Sử dụng số hiệu cổng nguồn và cổng đích trong trình ứng dụng khách/chủ



**Hình 3.5.** Hai client cùng số hiệu cổng đích truyền thông với cùng một server

Điều gì sẽ xảy ra nếu có hai client khác nhau cùng thiết lập phiên làm việc tới một server và mỗi client đều chọn cổng nguồn là x? Điều này rất dễ xảy ra với những Web server có nhiều người truy cập, phải phục vụ nhiều yêu cầu. Bên server phải phân kênh segment như thế nào khi hai phiên làm việc có cùng cặp số hiệu cổng? Khi đó server phải sử dụng địa chỉ IP trong gói dữ liệu IP (datagram) chứa segment. Trên Hình 3.5, máy C có hai phiên làm việc và máy A có một phiên làm việc HTTP tới cùng server B. Cả ba máy A, B, C đều có địa chỉ IP phân biệt lần lượt là A, B, C. Máy C sử dụng

hai cổng nguồn là (x, y) khác nhau cho hai kết nối HTTP tới B. A chọn số hiệu cổng nguồn độc lập với C, nên nó có thể gán cổng nguồn x cho kết nối HTTP của mình. Tuy nhiên, máy chủ B vẫn có thể thực hiện phân kênh hai gói segment có cặp cổng giống nhau do địa chỉ IP nguồn khác nhau. Tóm lại, bên nhận sử dụng cả ba giá trị (địa chỉ IP nguồn, số hiệu cổng nguồn, số hiệu cổng đích) để xác định tiến trình ứng dụng nhận.

Tiếp theo, trình bày một trong các giao thức giao vận của Internet – UDP. Trong phần này, sẽ thấy ngoài hai dịch vụ dồn kinh và phân kinh, UDP gần như không cung cấp dịch vụ nào khác.

### 3.3. UDP – GIAO THỨC KHÔNG HƯỚNG NÓI

Trong phần này ta sẽ nghiên cứu cơ chế hoạt động của UDP. Bạn đọc cần nhớ lại khái quát về dịch vụ UDP trình bày trong mục 2.1.

Bạn sẽ làm gì nếu muốn xây dựng một giao thức giao vận cực kỳ đơn giản – một giao thức giao vận "rỗng"? Khi đó, thực thể giao vận phía gửi nhận thông điệp từ tiến trình ứng dụng và chuyển xuống tầng mạng; thực thể giao vận phía nhận chuyển thông điệp tầng mạng đưa lên tới chương trình ứng dụng tương ứng. Tầng giao vận chỉ cung cấp dịch vụ dồn kinh/phân kinh bằng cách chuyển dữ liệu đến từ tầng mạng tới đúng tiến trình ứng dụng nhận.

UDP đặc tả trong RFC 768 là giao thức giao vận cực kỳ đơn giản. Như vậy, bên cạnh chức năng dồn kinh/phân kinh, UDP có thêm cơ chế phát hiện lỗi đơn giản. Có thể nói, nếu sử dụng UDP thì gần như ứng dụng làm việc trực tiếp với tầng mạng IP. UDP lấy thông điệp từ tiến trình ứng dụng, chèn thêm một số trường tiêu đề, trong đó có hai trường địa chỉ cổng nguồn và đích cho dịch vụ dồn kinh/phân kinh để tạo nên gói dữ liệu segment. Gói segment sau khi tạo ra được chuyển xuống tầng mạng. Tầng mạng đặt segment này trong gói dữ liệu IP datagram và "cố gắng" gửi gói IP datagram tới máy tính nhận. Nếu segment tới đích, UDP sử dụng số hiệu cổng và địa chỉ IP của tiến trình nhận để truyền dữ liệu trong segment tới đúng tiến trình ứng dụng nhận. Chú ý, UDP không đòi hỏi thực thể bên gửi và bên nhận

phải liên kết trước khi trao đổi dữ liệu. Vì thế, UDP được xem là dịch vụ *không hướng nối* hay *không liên kết trước* (connectionless).

DNS là một giao thức tầng ứng dụng chạy trên nền UDP. Khi muốn truy vấn, DNS tạo thông điệp truy vấn DNS, chuyển thông điệp tới socket. UDP bổ sung một số trường vào đầu mỗi thông điệp để tạo ra UDP segment, rồi gửi segment này xuống tầng mạng. Tầng mạng sẽ đóng gói UDP segment này trong IP datagram và gửi datagram tới đích (name server). Sau đó, DNS bên gửi đợi trả lời. Nếu không nhận được câu trả lời (điều này có thể xảy ra khi các tầng dưới làm mất thông điệp yêu cầu hay thông điệp trả lời), thì DNS gửi lại yêu cầu hoặc báo cho ứng dụng biết là không nhận được câu trả lời. Các đặc tả DNS cho phép DNS chạy trên nền TCP, nhưng trong thực tế DNS thường chạy trên UDP.

So với UDP, TCP có vẻ có nhiều ưu điểm hơn (TCP cung cấp dịch vụ truyền dữ liệu tin cậy, trong khi UDP không làm được). Tuy nhiên, trên thực tế nhiều ứng dụng lại sử dụng UDP với các lý do sau đây:

- ☛ **Không có giai đoạn thiết lập kết nối:** TCP sử dụng cơ chế "bắt tay" ba bước trước khi bắt đầu truyền dữ liệu thực sự. UDP không cần cơ chế này trước khi truyền dữ liệu. Vì thế, UDP sẽ không phải chịu thời gian trễ để thiết lập đường truyền. Đây chính là nguyên nhân DNS chạy trên UDP chứ không phải là TCP. DNS sẽ chạy chậm nếu sử dụng TCP. HTTP sử dụng TCP vì các đối tượng Web cần được tải về chính xác, do đó yêu cầu một đường truyền tin cậy. Nhưng như đã trình bày trong mục 2.2, giai đoạn thiết lập đường truyền trong TCP gây nên một thời gian trễ cho ứng dụng HTTP (tình trạng "world wide wait").
- ☛ **Không duy trì trạng thái kết nối:** TCP ghi nhớ trạng thái kết nối trên hệ thống đầu/cuối. Trạng thái kết nối bao gồm vùng đệm (buffer) của bên nhận và bên gửi, các tham số kiểm soát tắc nghẽn, số tuần tự phát và số biên nhận. Nó giúp TCP triển khai dịch vụ truyền tin tin cậy và cơ chế kiểm soát tắc nghẽn. Trong mục 3.5 sẽ giúp chúng ta hiểu ý nghĩa các trạng thái này. UDP không phải lưu giữ những thông tin như vậy. Do đó, nếu phía server sử dụng UDP thì có khả năng phục vụ đồng thời nhiều client hơn.

- Tiêu đề gói dữ liệu nhỏ:** Tiêu đề của TCP segment là 20 bytes trong khi UDP chỉ có 8 bytes.
- Không kiểm soát tốc độ gửi:** TCP có cơ chế kiểm soát tắc nghẽn, điều chỉnh tốc độ gửi khi xảy ra tắc nghẽn. Cơ chế điều chỉnh này có thể ảnh hưởng tới những ứng dụng thời gian thực – là những ứng dụng chấp nhận mất mát dữ liệu (trong phạm vi nào đó), nhưng lại đòi hỏi phải có một tốc độ truyền tối thiểu. Tốc độ truyền dữ liệu của UDP chỉ bị giới hạn bởi tốc độ sinh dữ liệu của ứng dụng, khả năng máy tính nguồn (CPU, tốc độ đồng hồ) và tốc độ truy cập mạng. Chú ý rằng, bên nhận không nhất thiết phải nhận toàn bộ dữ liệu. Khi mạng bị tắc nghẽn, một phần dữ liệu có thể bị mất do tràn vùng đệm ở router. Tốc độ nhận có thể bị giới hạn do tắc nghẽn ngay cả khi tốc độ gửi không bị giới hạn.

Ứng dụng	Giao thức tầng ứng dụng	Tầng giao vận tương ứng
Thư điện tử	SMTP	TCP
Truy cập từ xa	Telnet	TCP
Web	HTTP	TCP
Truyền file	FTP	TCP
File server	NFS	thường là UDP
Đa phương tiện	Phụ thuộc vào hãng sản xuất	thường là UDP
Điện thoại qua Internet	Phụ thuộc vào hãng sản xuất	thường là UDP
Quản lý mạng	SNMP	thường là UDP
Định tuyến	RIP	thường là UDP
Tên miền	DNS	thường là UDP

Hình 3.6. Các ứng dụng thông dụng và giao thức giao vận tương ứng

Hình 3.6 liệt kê một số ứng dụng phổ biến và giao thức giao vận của chúng. Email, truy cập từ xa, Web và truyền file chạy trên nền TCP vì chúng cần đến dịch vụ truyền dữ liệu tin cậy. Tuy nhiên, có một số ứng dụng khác thích hợp với UDP hơn TCP. Giao thức cập nhật bảng định tuyến RIP (xem Chương 4) sử dụng UDP, bởi vì việc cập nhật được thực hiện định kỳ (thường khoảng 5 phút một lần), cho nên dù cập nhật bị mất, nhưng sẽ có cập nhật mới sau một khoảng thời gian ngắn. UDP được sử dụng để gửi dữ liệu quản trị mạng (SNMP). Trong trường hợp này, UDP thích hợp hơn TCP, vì các tiến trình quản trị mạng thường hoạt động khi mạng có sự cố không thể truyền dữ liệu chính xác, hay cơ chế kiểm soát tắc nghẽn không làm việc. DNS sử dụng UDP, do đó có thể tránh được thời gian trễ trong giai đoạn thiết lập kết nối.

Ngày nay, UDP thường được các ứng dụng đa phương tiện như điện thoại Internet, hội thảo từ xa, các ứng dụng thời gian thực sử dụng. Các ứng dụng như thế có thể chấp nhận sai lệch, lỗi trên một phần dữ liệu. Vì thế, truyền dữ liệu tin cậy không phải là tiêu chí quan trọng nhất đánh giá kết quả của ứng dụng. Hơn nữa, các ứng dụng thời gian thực không thích ứng được với cơ chế kiểm soát tắc nghẽn của TCP. Do đó, chúng thường lựa chọn UDP ở tầng giao vận.

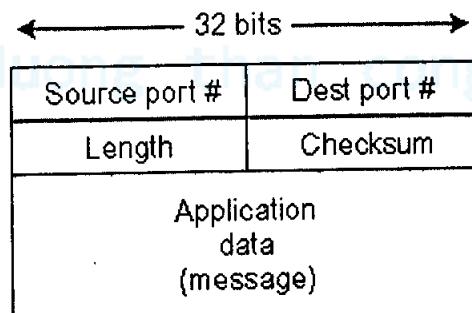
Mặc dù đã triển khai trong thực tế, song việc các ứng dụng đa phương tiện sử dụng UDP gây ra nhiều tranh cãi. Như đã nói ở trên, UDP không kiểm soát được tắc nghẽn, nên mạng rất dễ bị tắc nghẽn – khi đó chỉ rất ít thông tin được chuyển. Nếu tất cả mọi người đều xem phim trực tuyến, thì các gói tin sẽ bị tràn bộ đệm ở các router, và khi đó thì chẳng ai xem được gì cả. Thiếu cơ chế kiểm soát tắc nghẽn có thể sẽ là một vấn đề nghiêm trọng đối với UDP. Người ta đã đưa ra nhiều cơ chế đòi hỏi tất cả các thực thể, kể cả UDP thực hiện một cơ chế kiểm soát lưu lượng thích nghi.

Trước khi trình bày về cấu trúc UDP, cần chú ý rằng, tuy sử dụng UDP, nhưng ứng dụng vẫn có thể có một đường truyền tin cậy. Điều này được thực hiện bằng cách đảm bảo tính tin cậy ngay trong bản thân ứng dụng (bằng các cơ chế đánh số thứ tự, truyền lại). Công việc này sẽ làm ứng dụng cồng kềnh và phức tạp. Tuy nhiên, ưu điểm là ứng dụng có thể truyền thông tin cậy với tốc độ không bị cơ chế kiểm soát tắc nghẽn của TCP không chế. Ngày nay, một số phần mềm chuyên dụng đa phương tiện sử dụng cơ chế

danh số thứ tự và truyền lại ngay trong chương trình ứng dụng để giảm bớt việc mất dữ liệu.

### 3.3.1. Cấu trúc UDP segment

Cấu trúc UDP segment đặc tả trong RFC 768 được minh họa trên Hình 3.7. Dữ liệu của ứng dụng nằm trong trường dữ liệu của UDP datagram. Ví dụ đối với DNS, trường dữ liệu chứa thông điệp yêu cầu hay thông điệp trả lời. Tiêu đề UDP có bốn trường, độ lớn mỗi trường là 2 bytes. Như đã nói ở phần trước, số hiệu cổng cho phép thiết bị gửi chuyển dữ liệu tới đúng tiến trình chạy trên thiết bị nhận (chức năng phân kênh). Trường Checksum được bên nhận sử dụng để kiểm tra trong segment có lỗi hay không. Trên thực tế, kể cả tiêu đề của gói dữ liệu IP cũng được tính checksum. Nguyên tắc cơ bản của cơ chế phát hiện và sửa lỗi được trình bày trong mục 5.1. Trường độ dài (Length) cho biết độ dài (tính theo byte) của toàn bộ gói dữ liệu UDP segment – kể cả phần tiêu đề.



Hình 3.7. Cấu trúc gói UDP datagram

### 3.3.2. UDP checksum

UDP checksum được sử dụng để phát hiện lỗi. Checksum được tính như sau: tính giá trị *bù một* của tổng các từ 16 bit trong segment, giá trị nhận được được đặt vào trường checksum trong gói dữ liệu UDP segment. Có thể tìm hiểu phương thức triển khai trong RFC 1071 và hiệu quả trên dữ liệu thực trong [Stone 1998 và Stone 2000]. Giả sử có ba từ 16 bit sau đây:

0110011001100110

0101010101010101

0000111100001111

Tổng hai từ đầu là:

$$\begin{array}{r} 0110011001100110 \\ 0101010101010101 \\ \hline 1011101110111011 \end{array}$$

Cộng từ thứ ba vào, ta có:

$$\begin{array}{r} 1011101110111011 \\ 0000111100001111 \\ \hline 1100101011001010 \end{array}$$

Cách lấy *bù một* là đảo 0 thành 1 và 1 thành 0. Vì vậy, kết quả phép lấy bù một của 1100101011001010 là 0011010100110101, đó chính là giá trị checksum. Tại phía nhận, tất cả bốn từ (kể cả checksum) được cộng lại. Nếu dữ liệu không có lỗi thì tổng nhận được là 1111111111111111. Nếu có một bit nào đó bằng 0, ta biết dữ liệu nhận được có lỗi.

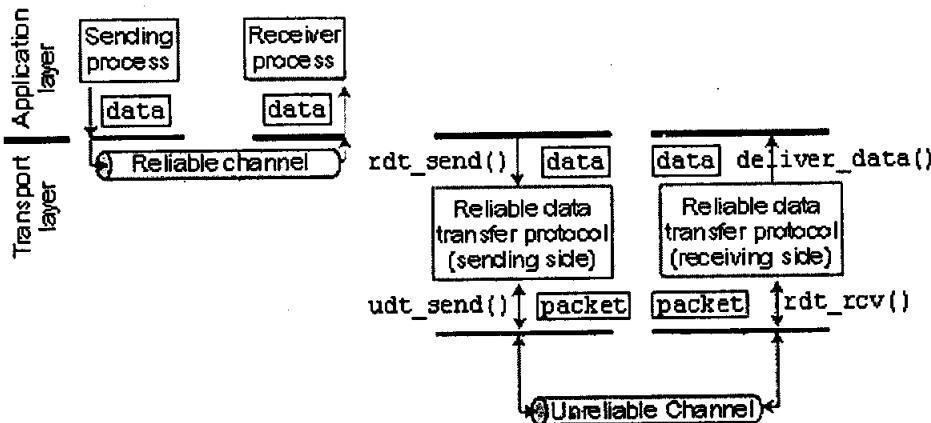
Vậy, tại sao UDP tính checksum, trong khi một vài giao thức tầng liên kết dữ liệu (kể cả giao thức Ethernet thông dụng) cũng có cơ chế kiểm tra lỗi? Lý do là chưa chắc tất cả các kết nối (link – đường truyền vật lý thực sự) giữa thiết bị gửi và thiết bị nhận đều có cơ chế kiểm tra lỗi, có thể một trong các kết nối đó sử dụng giao thức không cung cấp việc kiểm tra lỗi. Mặc dù UDP có thể phát hiện được lỗi, nhưng nó không làm gì khi phát hiện ra lỗi. Có thể nó sẽ loại bỏ segment bị lỗi, có thể nó sẽ chuyển segment bị lỗi cho ứng dụng nhận cùng với một thông điệp cảnh báo.

TCP cung cấp đường truyền tin cậy, do đó hiển nhiên triển khai nó phức tạp hơn UDP rất nhiều. Trước khi tìm hiểu về TCP, trong phần sau trình bày nguyên tắc chung để xây dựng một đường truyền tin cậy. TCP sẽ áp dụng đúng những nguyên tắc này khi triển khai.

### 3.4. CÁC NGUYÊN TẮC TRUYỀN DỮ LIỆU TIN CẬY

Phần này trình bày tổng quan dịch vụ truyền dữ liệu tin cậy. Dịch vụ này không chỉ nằm ở tầng giao vận mà còn có thể nằm ở tầng liên kết dữ liệu hay tầng ứng dụng. Có thể nói, truyền dữ liệu tin cậy là một trong những vấn đề quan trọng nhất của mạng. Trong phần kế tiếp về TCP sẽ nghiên cứu cách thức TCP áp dụng các nguyên tắc chung như thế nào?

Hình 3.8 là sơ đồ cấu trúc của quá trình truyền dữ liệu tin cậy. Tầng dưới cung cấp dịch vụ truyền tin cậy cho các thực thể ở tầng trên. Trên đường truyền tin cậy này, dữ liệu không bị lỗi (bit 0 biến thành bit 1 hoặc ngược lại), không bị mất và được nhận theo đúng thứ tự gửi. Đây chính là dịch vụ mà TCP cung cấp cho các ứng dụng của Internet.



a) Dịch vụ cung cấp

b) Cài đặt dịch vụ

Hình 3.8. Dịch vụ truyền dữ liệu tin cậy (mô hình và triển khai)

Để thực hiện công việc này, người ta cần đến những giao thức truyền dữ liệu tin cậy. Nguyên nhân là tầng phía dưới của giao thức tin cậy là không tin cậy. Ví dụ, TCP là giao thức truyền dữ liệu tin cậy nằm ở phía trên giao thức truyền không tin cậy (IP) giữa hai thiết bị đầu/cuối trên mạng. Để đơn giản trong trường hợp này, coi tầng phía dưới là một đường truyền điểm nối điểm (point – to – point) không tin cậy.

Trong phần này sẽ xây dựng dàn giao thức truyền dữ liệu tin cậy giữa phía gửi và phía nhận theo độ phức tạp tăng dần của kênh truyền bên dưới. Hình 3.8b minh họa điều này. Thực thể gửi sẽ nhận dữ liệu từ phía trên chuyển xuống qua hàm `rdt_send()` (Ở đây rdt là viết tắt của "reliable data transfer" và \_send chỉ rõ đây là phía gửi của giao thức rdt. Bước đầu tiên khi xây dựng một giao thức nào đó là chọn cho nó một cái tên dễ nhớ!). Phía nhận sử dụng hàm `rdt_rcv()` để lấy gói dữ liệu từ đường truyền. Để chuyển dữ liệu lên tầng trên, phía nhận sử dụng hàm `deliver_data()`. Trong phần này, sử dụng thuật ngữ "packet" thay thế "segment" với ý nghĩa là *đơn vị dữ liệu giao thức – PDU*. Ý tưởng trình bày trong phần này không chỉ áp dụng cho

tầng giao vận mà còn áp dụng chung cho toàn mạng máy tính, vì thế sử dụng thuật ngữ "packet" thích hợp hơn.

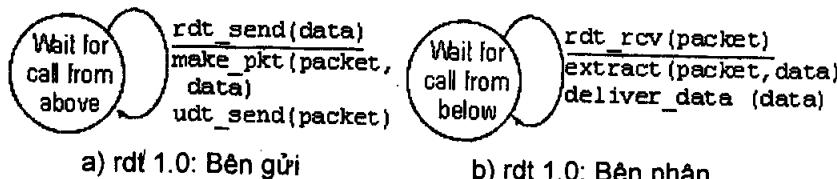
Trong phần này chỉ nghiên cứu trường hợp dữ liệu truyền theo một hướng từ nơi gửi đến nơi nhận. Trường hợp dữ liệu truyền theo hai hướng là một vấn đề không khó về mặt lý thuyết, nhưng triển khai cụ thể tương đối phức tạp. Mặc dù dữ liệu chỉ được truyền theo một hướng, nhưng các bên truyền thông trong giao thức rdt cần truyền dữ liệu theo cả hai hướng (Hình 3.8), bởi vì ngoài các gói dữ liệu thực sự, chúng còn phải trao đổi các gói dữ liệu chứa thông tin điều khiển. Cả bên gửi và bên nhận đều sử dụng hàm `udt_send()` để gửi dữ liệu đến phía bên kia (udt là viết tắt của "unreliable data transfer").

### 3.4.1. Xây dựng giao thức truyền dữ liệu tin cậy

Trong phần này sẽ từng bước trình bày các giao thức với độ phức tạp tăng dần, để cuối cùng đi đến giao thức truyền dữ liệu không lỗi; sẽ mô tả trạng thái của phía nhận và phía gửi bằng kỹ thuật "*máy hữu hạn trạng thái*" (Finite State Machine – FSM).

#### a) Truyền dữ liệu tin cậy trên kênh truyền tin cậy hoàn toàn (giao thức rdt 1.0)

Giao thức đầu tiên, đơn giản nhất được đưa ra là rdt 1.0, sử dụng kênh truyền tin cậy ở phía dưới. Giao thức rdt 1.0 cực kỳ đơn giản, FSM của bên gửi và bên nhận đều chỉ có một trạng thái (Hình 3.9). Mùi tên trong sơ đồ chỉ sự chuyển trạng thái của giao thức (mặc dù mỗi FSM trong Hình 3.9 chỉ có một trạng thái, vẫn cần đến sự chuyển trạng thái để quay về chính trạng thái cũ). Sự kiện kích hoạt việc chuyển trạng thái được đặt phía trên đường kẻ nằm ngang, đó là nhãn sự kiện. Phía bên dưới đường kẻ nằm ngang là những hành động mà thực thể phải thực hiện ngay khi sự kiện đó xảy ra (thực hiện trước khi thực thể chuyển sang trạng thái mới).



Hình 3.9. Giao thức cho kênh truyền tin cậy hoàn toàn

Với rdt 1.0, việc gửi đơn giản chỉ là nhận dữ liệu từ tầng trên thông qua sự kiện **rdt\_send(data)**, tạo ra gói dữ liệu (bằng hành động **make\_data(packet,data)**) và gửi gói dữ liệu (packet) lên kênh truyền. Trên thực tế, sự kiện **rdt\_send(data)** là kết quả của một thủ tục (ví dụ, khi ứng dụng phía trên sử dụng hàm **rdt\_send()**).

Ở phía nhận, rdt nhận gói dữ liệu (packet) từ kênh truyền bằng sự kiện **rdt\_rcv(packet)**, lấy dữ liệu ra khỏi gói dữ liệu (bằng hành động **extract(packet,data)**) và đưa dữ liệu lên tầng trên. Trên thực tế, sự kiện **rdt\_rcv(packet)** là kết quả của một thủ tục (ví dụ, khi ứng dụng phía trên sử dụng hàm **rdt\_rcv()**).

Trong giao thức đơn giản này, không có sự khác biệt giữa dữ liệu (data) với gói dữ liệu (packet). Như vậy, tất cả packet đều được truyền từ phía gửi cho phía nhận. Với kênh truyền tin cậy, phía nhận không cần thiết phải phản hồi cho phía gửi, vì nó chắc chắn rằng không có chuyện gì xảy ra. Chú ý rằng, ở đây đã giả thiết phía nhận có thể nhận dữ liệu với tốc độ phía gửi gửi. Vì vậy, phía nhận không cần yêu cầu phía gửi chậm lại.

### b) Truyền dữ liệu tin cậy trên kênh truyền có lỗi bit (giao thức rdt 2.0)

Một dạng kênh truyền thực tế hơn là gói dữ liệu trên kênh truyền có thể bị lỗi. Thường bit bị lỗi trên đường truyền vật lý của mạng. Tuy nhiên, giả thiết rằng, tất cả các gói dữ liệu truyền đi đều đến được đích và theo đúng thứ tự gửi, mặc dù các bit trong gói dữ liệu có thể bị lỗi.

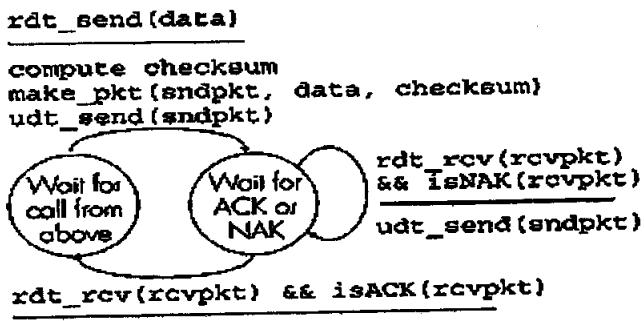
Trước khi tiếp tục, xét ví dụ sau: Giả sử bạn đọc một bài chính tả cho ai đó qua điện thoại. Thông thường, người chép sẽ nói "Xong rồi" sau khi đã nghe, hiểu và ghi lại một câu chính tả. Nếu câu nói của bạn bị nhiều, người kia nghe không rõ thì họ sẽ yêu cầu bạn nhắc lại. Giao thức truyền tin này sử dụng *phản hồi tích cực* (positive acknowledgement) ("Xong rồi") hay *phản hồi tiêu cực* (negative acknowledgement) ("Giờ cơ?"). Những thông điệp điều khiển này cho phép bên nhận báo cho bên gửi biết dữ liệu nào nhận đúng, dữ liệu nào bị lỗi và yêu cầu truyền lại dữ liệu bị lỗi. Trong mạng máy tính, giao thức truyền tin cậy dựa trên cơ chế truyền lại như vậy được gọi là *giao thức ARQ* (Automatic Repeat reQuest).

Các giao thức ARQ cần phải có ba khả năng sau để xử lý trong trường hợp dữ liệu có lỗi:

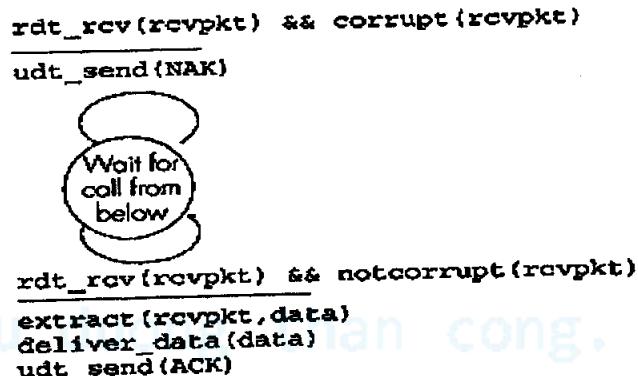
- ☞ **Phát hiện lỗi** (error detection): Là cơ chế cho phép bên nhận phát hiện được khi nào trong gói dữ liệu có bit bị lỗi. Trong phần trước, ta thấy UDP sử dụng trường Internet checksum cho mục đích này. Trong Chương 5 sẽ trình bày chi tiết một số kỹ thuật phát hiện và thậm chí có thể sửa được lỗi. Còn bây giờ, chỉ cần biết rằng, những kỹ thuật như vậy yêu cầu ngoài việc gửi dữ liệu gốc, bên gửi còn phải tạo ra và gửi kèm một lượng dữ liệu dư thừa (nhưng phụ thuộc vào dữ liệu gốc). Các bit dư thừa này được đặt trong trường checksum của gói dữ liệu rdt 2.0.
- ☞ **Phản hồi từ phía nhận** (receiver feedback): Khi phía gửi và phía nhận nằm trên các thiết bị đầu/cuối khác nhau – có thể cách nhau hàng nghìn km, cách duy nhất để phía gửi biết được kết quả gửi là phía nhận gửi thông tin phản hồi thông báo tình trạng nhận cho phía gửi. Báo nhận đúng (đôi khi gọi là báo nhận tích cực) ACK và báo nhận sai NAK, trong ví dụ chính là các thông tin phản hồi. Giao thức rdt 2.0 yêu cầu phía nhận gửi phản hồi các thông điệp ACK hay NAK cho phía gửi. Gói dữ liệu phản hồi chỉ cần sử dụng một bit, ví dụ giá trị 0 ứng với NAK và giá trị 1 ứng với ACK.
- ☞ **Truyền lại** (retransmission): Gói dữ liệu bị lỗi sẽ được bên gửi truyền lại.

Hình 3.10 là FSM của phía gửi và nhận trong giao thức rdt 2.0, với cơ chế phát hiện lỗi, phản hồi (ACK, NAK) và truyền lại.

Trong giao thức rdt 2.0, phía gửi có hai trạng thái. Ở trạng thái thứ nhất, phía gửi đợi dữ liệu từ tầng trên. Trong trạng thái thứ hai, phía gửi đợi phản hồi ACK hoặc NAK từ phía nhận. Nếu nhận được ACK (`rdt_rcv(rcvpkt) & isACK(rcvpkt)` trong Hình 3.10 tương ứng với sự kiện này), phía gửi biết được gói dữ liệu chuyển đến đích an toàn, vì vậy nó trở về trạng thái đợi dữ liệu từ tầng trên để chuyển tiếp. Nếu nhận được NAK, phía gửi gửi lại gói dữ liệu rồi quay lại trạng thái đợi phản hồi ACK hoặc NAK cho gói dữ liệu vừa gửi lại. Chú ý rằng, khi phía gửi ở trong trạng thái chờ phản hồi (ACK hoặc NAK), nó không thể nhận thêm dữ liệu từ tầng trên đưa xuống. Nó chỉ chấp nhận dữ liệu mới khi nhận được ACK và chuyển trạng thái. Phía gửi sẽ không gửi dữ liệu mới cho đến khi nó chắc chắn rằng, phía nhận đã nhận đúng gói dữ liệu đã gửi. Giao thức rdt 2.0 với hành vi như vậy thuộc kiểu *dừng và chờ* (stop and wait).



a) rdt 2.0: Bên gửi



b) rdt 2.0: Bên nhận

Hình 3.10. Giao thức cho kênh truyền có lỗi bit

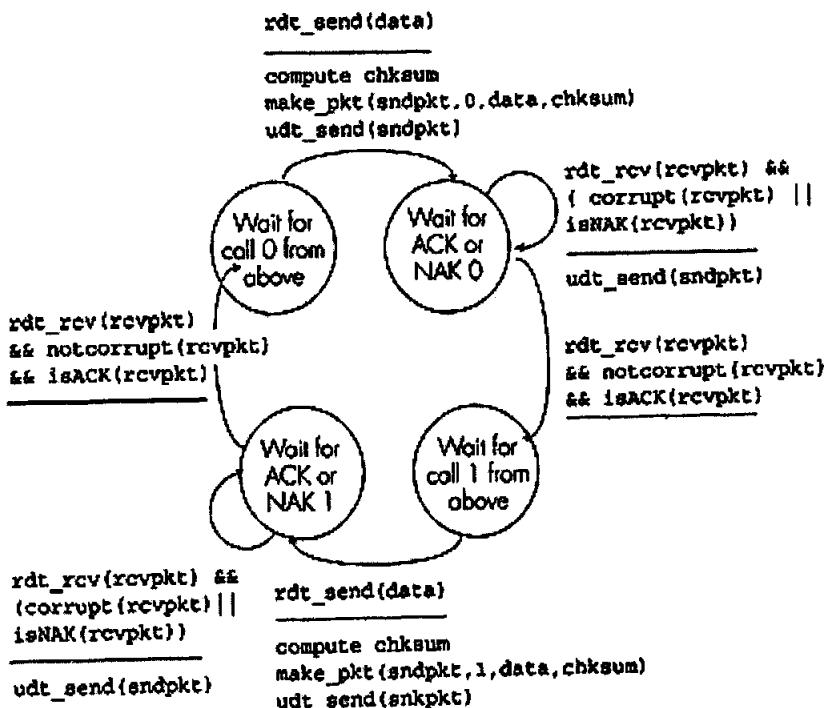
FSM bên nhận trong giao thức rdt 2.0 chỉ có một trạng thái duy nhất. Khi nhận được gói dữ liệu (packet), phía nhận gửi thông điệp phản hồi ACK hoặc NAK, phụ thuộc vào gói dữ liệu đã nhận có lỗi hay không. Trong Hình 3.10, rdt\_recv(rcvpkt) && corrupts(rcvpkt) tương ứng với sự kiện gói dữ liệu nhận được bị lỗi.

Giao thức rdt 2.0 vẫn còn nhược điểm, đó là chưa tính đến khả năng chính gói ACK hoặc NAK có lỗi. (Trước khi tiếp tục, cần nghĩ cách cải tiến giao thức này). Cần tạo checksum cho chính gói phản hồi (ACK hoặc NAK) để bên gửi (lúc này đóng vai trò bên nhận) có khả năng phát hiện lỗi trong chính gói phản hồi. Vấn đề ở đây là khi nhận được một gói phản hồi bị lỗi – phía gửi không thể xác định nó là ACK hay NAK, do đó không xác định được gói dữ liệu nó gửi tới đích có bị lỗi hay không. Trong trường hợp này, bên gửi sẽ phải làm gì?

Có ba giải pháp xử lý ACK hoặc NAK bị lỗi:

- ☞ Giải pháp thứ nhất, người đọc trong ví dụ đọc chính tả ở trên sẽ làm gì trong trường hợp này? Nếu không hiểu câu phản hồi "Xong rồi" hay "Gì cơ" thì họ có thể hỏi "Bạn nói gì?" (một dạng thông điệp điều khiển khác). Nếu nghe được, người bên kia sẽ lặp lại câu phản hồi. Nhưng chuyện gì xảy ra nếu chính câu "Bạn nói gì?" có lỗi? Khi đó phía nhận – do không xác định được câu có lỗi đó là một phần trong bài chính tả hay là yêu cầu nhắc lại câu phản hồi, nên có thể phản hồi lại bằng câu "Bạn nói gì?". Dĩ nhiên, câu trả lời này cũng có thể bị lỗi. Rõ ràng giải pháp này đã đi vào ngõ cụt.
- ☞ Giải pháp thứ hai là thêm vào trường checksum một số bit để không những cho phép phía nhận phát hiện mà còn sửa được các bit lỗi. Đây hoàn toàn có thể là giải pháp trung gian cho những kênh truyền có lỗi, nhưng không xử lý được trường hợp toàn bộ gói dữ liệu (packet) bị mất.
- ☞ Giải pháp thứ ba, phía gửi truyền lại gói dữ liệu nếu phát hiện lỗi trong gói phản hồi (ACK hoặc NAK). Tuy nhiên, phương pháp này có thể dẫn đến *sự trùng lặp dữ liệu* (duplicate packet). Phía nhận không biết được ACK/NAK mà nó gửi phản hồi có bị lỗi trên đường truyền không. Vì thế nó không xác định được gói dữ liệu vừa nhận được là gói dữ liệu mới hay gói cũ (sẽ bị trùng lặp).

Giải pháp đơn giản nhất cho vấn đề này (sẽ được áp dụng cho nhiều giao thức, kể cả TCP) là thêm trường số thứ tự cho gói dữ liệu (packet), phía gửi đánh số thứ tự các gói dữ liệu và đặt giá trị này vào trường số thứ tự (sequence number). Phía nhận chỉ cần kiểm tra số thứ tự để xác định gói dữ liệu nhận được là gói mới hay gói truyền lại. Với giao thức dừng và chờ đơn giản, chỉ cần một bit số thứ tự. Bên nhận có thể xác định bên gửi truyền lại gói dữ liệu đã gửi lần trước (số thứ tự của gói dữ liệu nhận được trùng với số thứ tự gói dữ liệu nhận được lần trước) hay gói dữ liệu mới (có số thứ tự khác nhau, tăng lên theo module 2). Vì vẫn giả định toàn bộ gói dữ liệu (packet) không bị mất trên kênh truyền, nên trong gói phản hồi (ACK/NAK) không cần chỉ ra số thứ tự của gói dữ liệu mà chúng nhận. Phía gửi biết rằng, gói ACK/NAK (có thể bị lỗi hoặc không) là biên nhận cho gói dữ liệu gần nhất nó gửi.



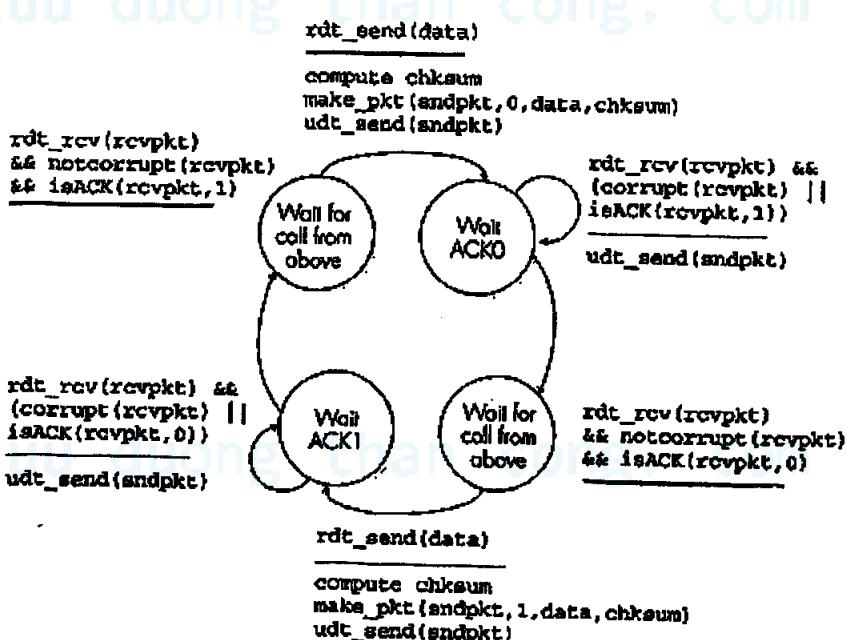
Hình 3.11. FSM của phía gửi trong rdt 2.1



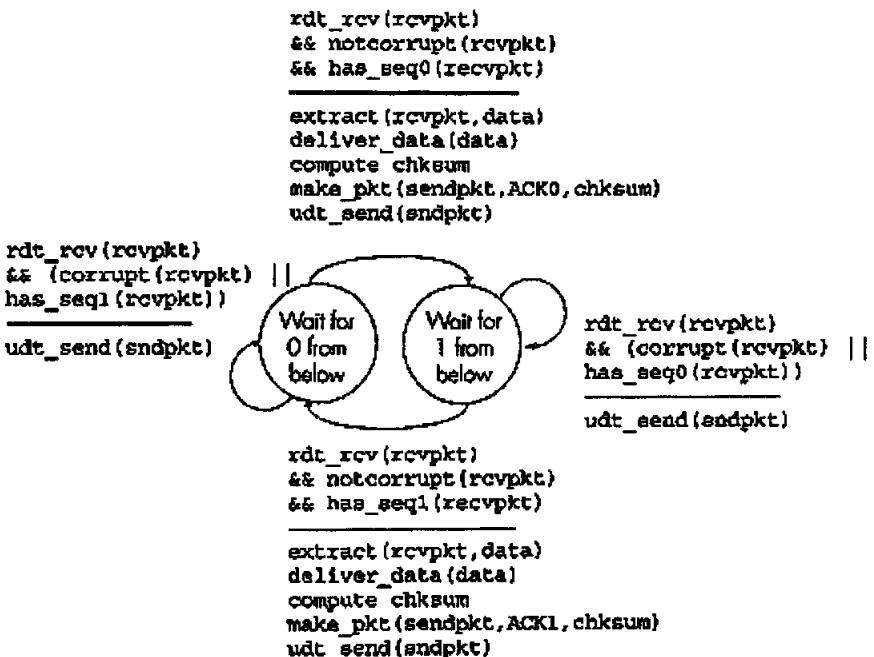
Hình 3.12. FSM của phía nhận trong rdt 2.1

Hình 3.11 và Hình 3.12 là FSM của bên gửi và nhận trong giao thức rdt 2.1 – phiên bản mới của rdt 2.0. Trong rdt 2.1, FSM của bên gửi và nhận đều có số trạng thái tăng gấp đôi. Đó là vì trạng thái giao thức phải biểu diễn gói dữ liệu được gửi (bởi bên gửi) và gói dữ liệu được đợi (tại bên nhận) có số thứ tự là 0 hay 1. Chú ý rằng, các hành động trong trạng thái gói dữ liệu có số thứ tự 0 được gửi (phía gửi) hoặc được mong đợi (phía nhận) ngược với trạng thái gói dữ liệu có số thứ tự 1 được gửi hay được đợi.

Giao thức rdt 2.1 sử dụng cả biên nhận đúng (ACK) và biên nhận sai (NAK). NAK được gửi khi nhận được gói dữ liệu bị lỗi hay không đúng số thứ tự. Có thể không cần sử dụng NAK, thay vì việc gửi NAK, gửi ACK cho gói dữ liệu cuối cùng đã được nhận đúng. Nếu nhận hai ACK cho cùng một gói dữ liệu (hiện tượng *trùng ACK* – duplicate ACK) bên gửi xác định được bên nhận không nhận đúng gói dữ liệu sau gói dữ liệu đã biên nhận ACK hai lần. TCP sử dụng sự kiện "3 lần nhận được ACK trùng nhau" ("triple duplicate ACKs") để kích hoạt việc gửi lại. rdt 2.2 là giao thức truyền dữ liệu tin cậy trên kênh truyền có bit lỗi không sử dụng NAK được minh họa trên Hình 3.13 và Hình 3.14.



Hình 3.13. FSM của phía gửi trong rdt 2.2



Hình 3.14. FSM của phía nhận trong rdt 2.1

c) *Truyền dữ liệu tin cậy trên kênh truyền mà dữ liệu bị mất, lỗi (giao thức rdt 3.0)*

Dữ liệu trên kênh truyền không những bị lỗi mà còn có thể bị mất, đây là tình huống khá phổ biến trong mạng máy tính ngày nay, kể cả Internet. Lúc này giao thức cần phải giải quyết hai vấn đề là: Làm thế nào để phát hiện gói dữ liệu bị mất và làm gì khi mất gói dữ liệu? Sử dụng cơ chế phát hiện lỗi nhờ checksum, số thứ tự, biên nhận ACK và truyền lại gói dữ liệu – đã được phát triển trong giao thức rdt 2.2 cho phép người ta giải quyết được vấn đề thứ hai. Để giải quyết vấn đề thứ nhất, cần phải có một cơ chế mới.

Có nhiều giải pháp xử lý việc mất mát dữ liệu. Ở đây trình bày giải pháp lựa chọn bên gửi là nơi phát hiện và xử lý việc mất dữ liệu. Giả sử phía gửi gửi đi gói dữ liệu, nhưng chính gói dữ liệu đó hoặc biên nhận ACK cho nó bị mất trên đường truyền. Trong cả hai trường hợp, bên gửi đều không nhận được biên nhận cho gói dữ liệu đã gửi. Giải pháp được đưa ra là, sau khi gửi một khoảng thời gian nào đó mà không nhận được biên nhận ACK (có thể gói dữ liệu bị mất) thì bên gửi sẽ truyền lại.

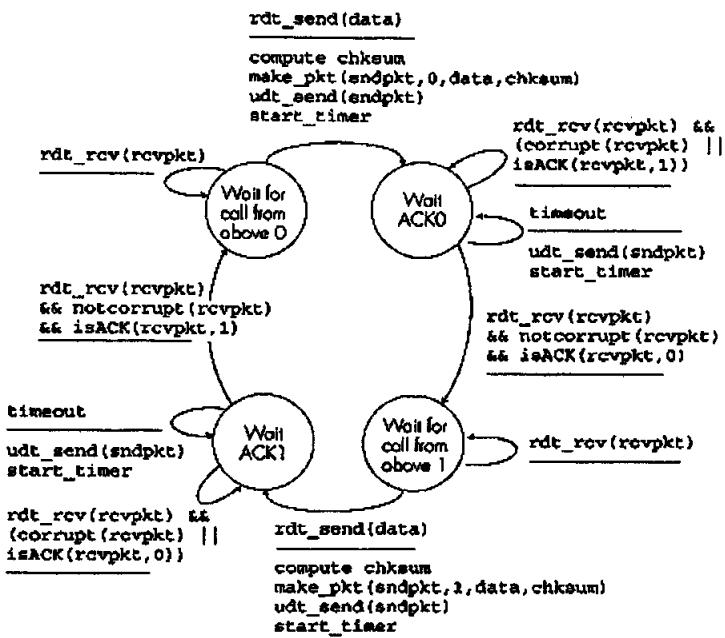
Nhưng phía gửi phải đợi trong bao lâu để chắc chắn rằng gói dữ liệu đã bị mất? Ít nhất phía gửi phải đợi trong khoảng thời gian để gói tin đi đến

được phía nhận, phía nhận xử lý gói tin và thông tin biên nhận quay lại. Trong nhiều mạng, rất khó dự đoán và ước lượng được thời gian này. Lý tưởng là phải xử lý việc mất gói tin ngay khi có thể, đợi một khoảng thời gian dài, đồng nghĩa với việc chậm trễ khi xử lý gói tin bị mất. Trên thực tế, phía gửi sẽ chọn một khoảng thời gian đợi nào đó, mặc dù không đảm bảo chắc chắn là gói tin có bị mất hay không. Nếu không nhận được ACK trong khoảng thời gian này, bên gửi sẽ gửi lại gói dữ liệu. Chú ý rằng, nếu gói dữ liệu đến trễ, phía gửi sẽ gửi lại gói dữ liệu (ngay cả khi gói dữ liệu đó và cả ACK đều không bị mất). Điều này gây ra trùng lặp dữ liệu tại phía nhận. Tuy nhiên, giao thức rdt 2.2 đã có đủ khả năng (nhờ số thứ tự) để ngăn chặn sự trùng lặp dữ liệu.

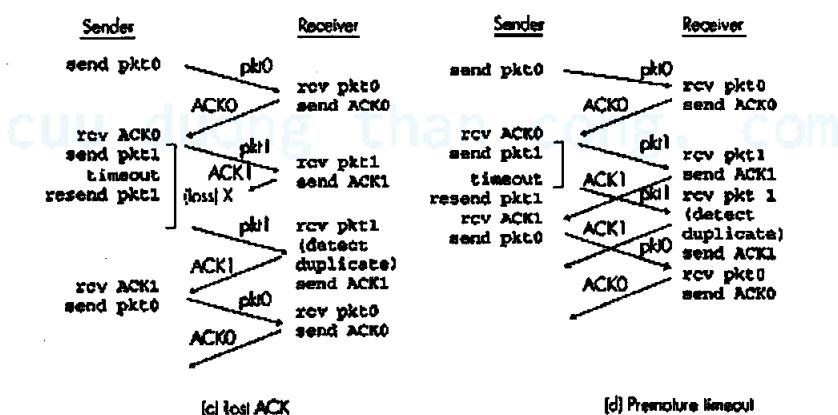
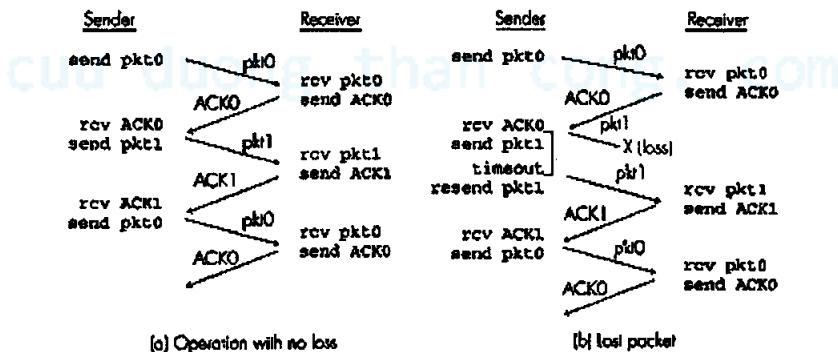
Đối với phía gửi, truyền lại là giải pháp "vạn năng". Phía gửi không xác định được gói dữ liệu bị mất, gói biên nhận ACK bị mất hay chỉ đơn giản là chúng bị trễ. Trong mọi trường hợp, hành động của nó là giống nhau, đó là truyền lại. Để thực hiện cơ chế truyền lại theo thời gian, một bộ *định thời đếm ngược* (countdown timer) được sử dụng để nhắc phía gửi thời gian đợi đã hết. Do vậy, phía gửi phải có khả năng: (1) khởi tạo timer mỗi khi gửi gói dữ liệu (gói dữ liệu gửi lần đầu hay gói dữ liệu được truyền lại); (2) phản ứng với ngắt của timer (đưa ra những hành động thích hợp) và (3) dừng timer.

Sự trùng lặp các gói dữ liệu do phía gửi tạo ra, sự mất mát các gói dữ liệu (cả gói dữ liệu lẫn gói biên nhận) gây khó khăn cho phía gửi khi xử lý các gói biên nhận ACK. Nếu nhận được ACK, làm thế nào để phía gửi biết được ACK đó là biên nhận cho gói dữ liệu gửi đi gần đây nhất, hay là ACK biên nhận cho gói dữ liệu nào đó đã gửi từ trước nhưng đến trễ? Giải pháp là ta thêm vào gói ACK trường *số thứ tự biên nhận* (acknowledge number). Giá trị của trường này do phía nhận tạo ra, là số thứ tự của chính gói dữ liệu cần được biên nhận. Bằng cách kiểm tra giá trị trường biên nhận, phía gửi có thể xác định được số thứ tự của gói dữ liệu được biên nhận.

Hình 3.15 là FSM của bên gửi trong giao thức rdt 3.0 – giao thức truyền dữ liệu trên kênh truyền có thể có lỗi hoặc bị mất dữ liệu. Hình 3.16 minh họa sự vận hành của giao thức trong một số trường hợp. Thời gian dịch chuyển theo chiều từ trên xuống. Thời điểm nhận gói dữ liệu chậm hơn thời điểm gửi gói dữ liệu, vì tính đến thời gian gói dữ liệu lan toả trên đường truyền.



Hình 3.15. FSM của bên gửi trong rdt 3.0



Hình 3.16. Ví dụ hoạt động của giao thức rdt 3.0

Trong Hình 3.16b-d, ngoặc vuông xác định thời điểm timer được thiết lập và thời điểm "timeout". Vì số thứ tự của gói dữ liệu thay đổi lần lượt giữa 0 và 1 nên đôi khi giao thức rdt 3.0 được gọi là *giao thức một bit luân chuyển* (alternate bit protocol).

Ở trên đã điểm qua các thành phần chính cho một giao thức truyền số liệu. Checksum, số thứ tự phát, bộ định thời (timer), các gói biên nhận ACK và NAK đều cực kỳ cần thiết và đóng vai trò quan trọng trong quá trình hoạt động của giao thức. Nay giờ, trình bày về một giao thức truyền dữ liệu tin cậy thực sự hoạt động được.

### 3.4.2. Giao thức truyền dữ liệu tin cậy liên tục (Pipeline)

Mặc dù hoạt động đúng, nhưng không phải ai cũng vừa lòng với hiệu suất của rdt 3.0, đặc biệt trong các mạng cao tốc ngày nay. Cốt lõi vấn đề hiệu suất của giao thức rdt 3.0 chính là hành vi dừng và chờ. Nguyên tắc của giao thức kiểu "Dừng và Chờ" là: Sau khi phát một gói dữ liệu, thiết bị phát dừng phát (stop) để chờ nhận phản hồi từ thiết bị nhận về kết quả nhận số liệu (wait). Nếu kết quả nhận tốt (biên nhận ACK), bên phát được quyền phát tiếp. Nếu kết quả nhận sai (biên nhận NAK), bên phát phải gửi lại gói dữ liệu.

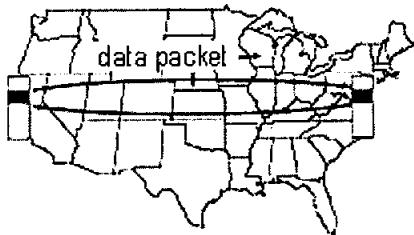
Để ước lượng hiệu suất của giao thức dừng và chờ, xét trường hợp lý tưởng, với hai thiết bị đầu/cuối, một ở bờ biển phía đông, một ở bờ biển phía tây nước Mỹ. Thời gian trễ giữa hai thiết bị (dù tín hiệu lan truyền với tốc độ ánh sáng) là  $P_{prop}$  (xấp xỉ 15μs). Giả sử rằng, hai thiết bị được kết nối bằng đường truyền tốc độ C (1 gigabit/s). Kích thước của gói dữ liệu SP là 1Kbyte/packet, thời gian cần thiết để truyền toàn bộ gói dữ liệu trên kênh truyền tốc độ 1 Gbps được tính bởi công thức:

$$T_{trans} = \frac{SP}{C} = \frac{8\text{Kbit}/\text{packet}}{1\text{Mbit/sec}} = 8 \mu\text{s}$$

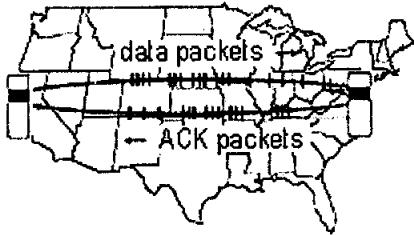
Với giao thức dừng và chờ, nếu phía gửi bắt đầu gửi gói dữ liệu tại thời điểm  $t = 0$  thì tại thời điểm  $t = 8\mu\text{s}$ , bit cuối cùng mới được bên gửi đẩy ra đường truyền. Tiếp theo phải mất 15μs để cả gói dữ liệu đi từ phía gửi sang phía nhận (Hình 3.17a). Như vậy, bit cuối cùng của gói dữ liệu đến đích tại thời điểm  $t = 15,008\mu\text{s}$ . Để đơn giản, giả thiết gói ACK có cùng độ dài với gói dữ liệu và phía nhận gửi ngay gói ACK khi nhận được bit cuối cùng của

gói dữ liệu. Như vậy, bit cuối cùng của gói ACK được truyền tới đích tại thời điểm  $t = 30,016\mu s$ . Trong khoảng thời gian  $30,016\mu s$ , phía gửi chỉ hoạt động (gửi hoặc nhận) trong  $0,016\mu s$ . Nếu định nghĩa *Hiệu suất* (utilization) của phía gửi (hay kênh truyền) là tỷ lệ thời gian phía gửi hoạt động (gửi dữ liệu trên kênh truyền), ta có hiệu suất  $U_{\text{sender}}$  cực thấp:

$$U_{\text{sender}} = \frac{0,008}{30,016} = 0,00015 = 0,015\%.$$



a) Giao thức kiểu Stop and Wait



a) Giao thức kiểu đường ống

Hình 3.17

Điều đó có nghĩa là, phía gửi chỉ hoạt động trong khoảng 0,15 phần nghìn thời gian. Theo cách tính khác, phía gửi gửi 1 Kbyte trong  $30,016\text{ ms}$  tương đương với tốc độ truyền là  $33\text{ Kbyte/s}$ , thấp hơn nhiều so với tốc độ có thể là  $1\text{ Gigabit/s}$ . Người quản trị mạng "bất hạnh" này phải trả một số tiền không lồ để thuê đường truyền  $1\text{ Gigabit/s}$ , nhưng cuối cùng chỉ nhận được một đường truyền có tốc độ  $33\text{ Kbyte/s}$ . Đây là một ví dụ sống động, minh họa việc phần mềm có thể giới hạn các khả năng của phần cứng phía dưới. Trong trường hợp này, ta đã bỏ qua thời gian xử lý của các giao thức tầng dưới ở cả phía gửi và phía nhận, cũng như thời gian xử lý và thời gian trễ của gói tin tại các router trung gian. Nếu tính cả những yếu tố này, hiệu suất hoạt động thực sự sẽ còn thấp hơn nữa.

Giải pháp cho vấn đề hiệu suất sẽ là cho phép phía gửi chuyển đồng thời nhiều gói dữ liệu, mà không cần phải đợi ACK (Hình 3.17b). Có thể hình dung các gói dữ liệu nối tiếp nhau trên đường truyền từ phía gửi đến phía nhận giống như nước chảy trong một đường ống. Vì thế, kỹ thuật gửi liên tiếp này được gọi là *kỹ thuật đường ống* (pipeline). Kỹ thuật này làm tăng hiệu suất của giao thức lên nhiều lần, tuy nhiên nó đòi hỏi những yêu cầu sau:

- ☞ Khoảng số thứ tự phải tăng, bởi vì mỗi gói dữ liệu được truyền đi (không tính các gói dữ liệu truyền lại) phải có một số thứ tự duy nhất. Trên đường truyền có thể có đồng thời nhiều gói dữ liệu đã gửi nhưng chưa được biên nhận.
- ☞ Phía gửi và phía nhận có thể phải có bộ đệm (buffer) cho nhiều gói dữ liệu. Ít nhất phía gửi có vùng đệm cho các gói dữ liệu đã được truyền đi, nhưng chưa được biên nhận. Phía nhận cũng có thể cần vùng đệm cho cả các gói dữ liệu đã nhận đúng, như trình bày dưới đây.

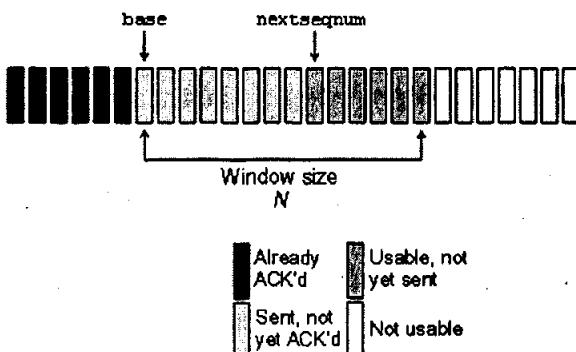
Yêu cầu về khoảng số thứ tự cần thiết cũng như về vùng đệm phụ thuộc vào cách giao thức xử lý việc mất dữ liệu, dữ liệu bị lỗi, bị trễ. Có hai cách tiếp cận chính được trình bày ở đây là *Quay lại N* (Go-back-N) và *Lặp lại có lựa chọn* (Selective Repeat).

### 3.4.3. Go-back-N (GBN)

Trong giao thức Go-back-N, phía gửi được phép truyền đi đồng thời nhiều gói dữ liệu mà không phải đợi biên nhận. Tuy nhiên, tổng số gói dữ liệu bị giới hạn bởi giá trị N – tổng số gói dữ liệu tối đa chưa được biên nhận trong đường ống. Hình 3.18 là khoảng số thứ tự trong giao thức Go-back-N. Định nghĩa *base* là số thứ tự của gói dữ liệu đã được truyền đi lâu nhất chưa được biên nhận và *nextseqnum* là số thứ tự nhỏ nhất chưa được sử dụng (là số thứ tự của gói tiếp theo sẽ gửi). Có bốn khoảng số thứ tự như sau: Khoảng  $[0, \text{base} - 1]$  ứng với số thứ tự của các gói dữ liệu đã được truyền đi và đã được biên nhận. Khoảng  $[\text{base}, \text{nextseqnum} - 1]$  ứng với các gói dữ liệu đã được gửi đi, nhưng chưa được biên nhận. Khoảng  $[\text{nextseqnum}, \text{base} + N - 1]$  có thể được sử dụng làm số thứ tự cho các gói sẽ được gửi nếu như có dữ liệu từ tầng trên chuyển xuống. Khoảng từ  $[\text{base} + N]$  trở lên chưa được sử dụng cho đến khi các gói tin đợi biên nhận được biên nhận.

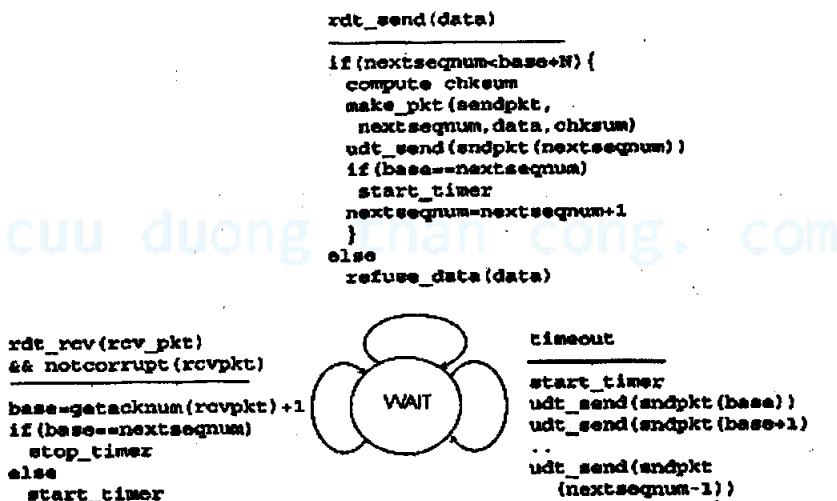
Trong Hình 3.18, khoảng số thứ tự cho phép của những gói dữ liệu đã được gửi, nhưng chưa được biên nhận có thể xem là một "cửa sổ" kích thước N nằm trong phạm vi số thứ tự. Khi giao thức vận hành, cửa sổ này có thể "trượt" trên toàn bộ khoảng số thứ tự. Vì vậy, N thường được xem là *độ lớn cửa sổ* (window size) và giao thức GBN là giao thức *cửa sổ trượt*.

(sliding – window). Tại sao ngay từ đầu phải giới hạn số lượng tối đa các gói dữ liệu được gửi mà chưa cần nhận bởi giá trị N? Tại sao không để giá trị N này là vô hạn? Trong mục 3.5, kiểm soát lưu lượng là một trong những lý do bắt buộc phải đặt giới hạn phía gửi.

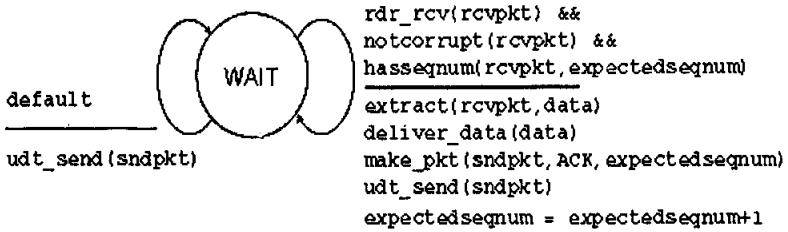


Hình 3.18. Khoảng số thứ tự của bên gửi trong giao thức Go-back-N

Trên thực tế, số thứ tự được đặt trong một trường có độ dài cố định trong tiêu đề của gói dữ liệu. Nếu k là độ lớn trường số thứ tự (tính theo bit) của gói dữ liệu, thì khoảng số thứ tự sẽ là  $[0, 2^k - 1]$ . Vì khoảng số thứ tự bị giới hạn, nên tất cả các thao tác trên số thứ tự sẽ được thực hiện theo module  $2^k$  (khoảng số thứ tự có thể xem là một vòng tròn với  $2^k$  giá trị, sau giá trị  $2^k - 1$  là giá trị 0). Giao thức rdt 3.0 chỉ sử dụng 1 bit làm số thứ tự, nên khoảng số thứ tự là  $[0, 1]$ . Trong mục 3.5, ta thấy trường số thứ tự của TCP là 32 bit, và TCP đánh số thứ tự đến từng byte, chứ không phải cho các gói.



Hình 3.19. FSM mở rộng của bên gửi trong GBN



**Hình 3.20. FSM mở rộng của bên nhận trong GBN**

Hình 3.19 và Hình 3.20 là FSM mở rộng của phía gửi và phía nhận trong giao thức GBN chỉ sử dụng ACK, không sử dụng NAK. Gọi là *FSM mở rộng* (extended FSM), vì thêm vào các biến (base và nextseqnum – giống như biến trong ngôn ngữ lập trình), các lệnh và hành động có điều kiện liên quan đến các biến này.

Trong giao thức GBN, phía gửi phải đáp ứng ba sự kiện sau:

- ☛ **Có dữ liệu từ trên chuyển xuống:** Khi `rdt_send()` được phía trên sử dụng để chuyển dữ liệu xuống, phía gửi phải kiểm tra xem cửa sổ đã đầy chưa (tức là đã có N gói dữ liệu gửi đi chưa được biên nhận). Nếu cửa sổ chưa đầy, phía gửi tạo ra và sau đó gửi gói dữ liệu đồng thời cập nhật các biến. Nếu cửa sổ đầy, phía gửi không chấp nhận dữ liệu từ tầng trên và thông báo cửa sổ đã đầy. Khi đó, tầng trên sẽ phải gửi lại. Trên thực tế, phía gửi sẽ đưa dữ liệu vào vùng đệm (nhưng chưa gửi ngay), hoặc có cơ chế đồng bộ (sử dụng semaphore hay cờ) chỉ cho phép tầng ứng dụng sử dụng `rdt_send()` khi cửa sổ chưa đầy.
- ☛ **Nhận được một ACK:** trong giao thức GBN, giá trị biên nhận mang tính tích lũy, nghĩa là, nếu biên nhận gói tin có số thứ tự n thì toàn bộ gói dữ liệu có số thứ tự nhỏ hơn hoặc bằng n đều đã được phía nhận nhận đúng. Vấn đề này sẽ được trình bày khi xem xét phía nhận trong giao thức GBN.
- ☛ **Hết thời gian đợi (timeout):** Tên giao thức "Go-back-N" bắt nguồn từ hành vi của phía gửi khi dữ liệu bị mất hay bị trễ. Giống như trong giao thức dừng và chờ, timer được sử dụng để xử lý việc mất gói dữ liệu hay gói phản hồi. Khi hết thời gian đợi (timeout), phía gửi sẽ gửi lại tất cả các gói dữ liệu đã được gửi đi trước đó, nhưng chưa được biên nhận. Trong Hình 3.19, phía gửi chỉ sử dụng duy

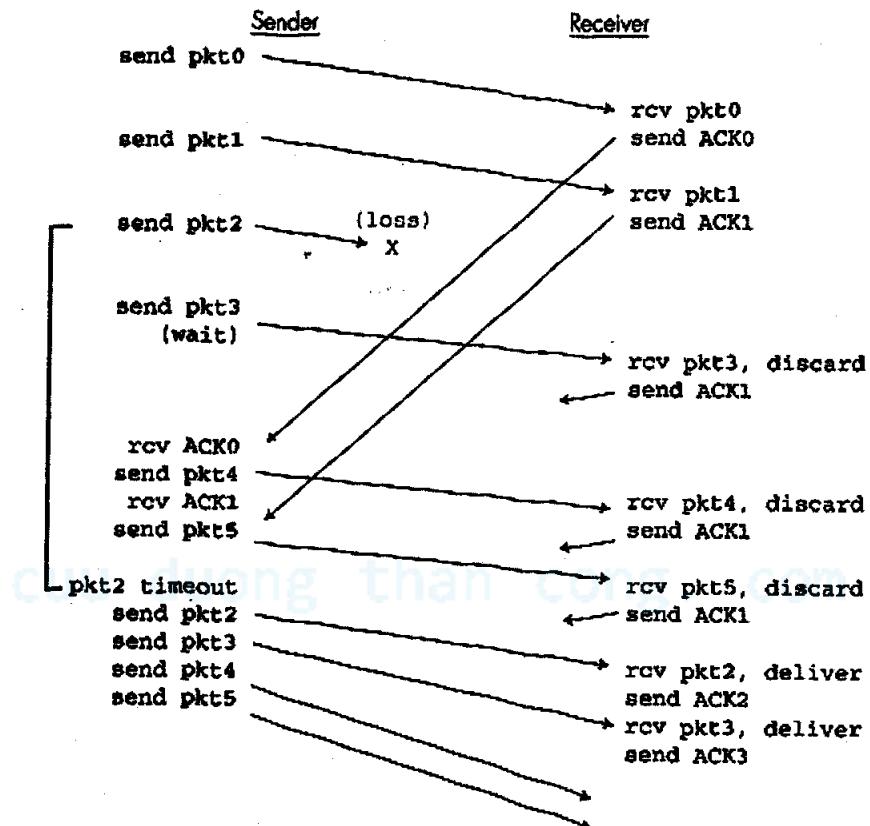
nhất một timer, có thể xem là timer của gói dữ liệu đã được truyền đi lâu nhất, nhưng chưa được nhận. Nếu ACK nào đó được nhận, nhưng vẫn còn gói dữ liệu gửi đi chưa được nhận, thì timer sẽ được khởi động lại. Nếu tất cả các gói dữ liệu đã gửi đều được nhận, thì có thể ngừng timer.

Các hành động của phía nhận trong giao thức GBN đơn giản. Nếu nhận được đúng gói dữ liệu và gói này đúng thứ tự thì phía nhận gửi ACK cho gói nhận được và chuyển dữ liệu trong gói dữ liệu này lên trên. Trong tất cả các trường hợp còn lại, phía nhận loại bỏ gói dữ liệu và gửi lại ACK cho gói dữ liệu đúng thứ tự cuối cùng nó nhận được. Chú ý rằng, gói dữ liệu được chuyển lên tầng trên một lần duy nhất, nên nếu gói dữ liệu thứ k được nhận và chuyển lên trên, thì nghĩa là tất cả các gói dữ liệu có số thứ tự nhỏ hơn k cũng đã được chuyển lên. Sử dụng ACK tích lũy là sự lựa chọn tuyệt vời cho giao thức GBN.

Trong giao thức GBN, bên nhận loại bỏ gói tin không theo thứ tự. Đường như lãng phí khi loại bỏ gói tin đã nhận đúng, nhưng không đúng thứ tự. Bên nhận phải chuyển dữ liệu lên tầng trên theo đúng thứ tự. Giả sử gói tin N đang được đợi nhận, nhưng gói tin thứ ( $N + 1$ ) lại đến trước. Trong trường hợp này, để dữ liệu chuyển lên hợp lệ, bên nhận có thể lưu tạm gói tin ( $N + 1$ ) và chỉ chuyển gói tin này lên tầng trên sau khi đã nhận đúng gói tin thứ N. Tuy nhiên, theo quy tắc truyền lại của bên gửi, nếu gói tin thứ N bị mất thì gói tin này và cả gói tin  $N + 1$  sẽ được truyền lại. Như vậy, bên nhận có thể loại bỏ gói tin  $N + 1$ . Ưu điểm của giải pháp này là bên nhận triển khai vùng đệm (buffer) đơn giản, bởi vì không cần lưu lại các gói tin không đúng thứ tự. Nếu bên gửi phải ghi nhớ các cận của cửa sổ (base, base + N) và vị trí nextseqnum trong cửa sổ, thì bên nhận chỉ phải nhớ số thứ tự của gói tin hợp lệ tiếp theo. Giá trị này được giữ trong biến expectedseqnum (số thứ tự được mong đợi) (Hình 3.20). Tất nhiên, nhược điểm của việc loại bỏ gói tin đã nhận đúng (nhưng không theo thứ tự) là khi truyền lại gói tin có thể bị mất hay lỗi, do đó phải truyền đi truyền lại nhiều lần.

Hình 3.21 là một ví dụ hoạt động của giao thức GBN trong trường hợp cửa sổ có độ lớn 4 gói tin. Với độ lớn này, bên gửi sẽ chỉ được gửi các gói tin từ 0 đến 3, nhưng sau đó phải đợi bên nhận cho các gói tin này trước khi

tiếp tục gửi tiếp. Khi nhận được các ACK liên tiếp nhau (ví dụ ACK0 và ACK1), cửa sổ sẽ trượt về phía trước, bên gửi có thể truyền gói tin mới (lần lượt là pkt4 và pkt5). Ở phía bên nhận, gói tin số 2 bị mất, do đó gói tin 3, 4, 5 gửi đến không theo đúng thứ tự và bị loại bỏ.



Hình 3.21. Giao thức Go-back-N trong quá trình hoạt động

Với GBN, có một chú ý quan trọng là triển khai GBN tương tự FSM mở rộng (Hình 3.19). Hình thức triển khai bao gồm nhiều thủ tục khác nhau, mỗi thủ tục thực hiện một nhóm các hành động nào đó để đáp lại các sự kiện khác nhau có thể xảy ra. Với lập trình hướng sự kiện (event – based programming), các thủ tục sẽ được gọi khi sự kiện tương ứng xuất hiện. Ở phía bên gửi, sự kiện có thể là: (1) thực thể tầng trên truyền dữ liệu xuống qua thủ tục `rdt_send()`; (2) ngắt khi thời gian đợi hết và (3) tầng dưới chuyển dữ liệu lên qua hàm `rdt_rcv()`.

Chú ý rằng, giao thức GBN kết hợp hầu hết các kỹ thuật mà chúng ta sẽ gặp khi nghiên cứu TCP trong mục 3.5 là số thứ tự, số biên nhận tích luỹ,

checksum, timeout và việc truyền lại. Trong thực tế, TCP là giao thức "tự" GBN. Tuy nhiên, có sự khác biệt giữa GBN và TCP. Nhiều phiên bản TCP lưu lại các segment không theo thứ tự nhận đúng. Trong phương án nâng cấp TCP, sử dụng biên nhận có lựa chọn [RFC 258] cho phép bên nhận có thể biên nhận tuỳ ý một gói tin không theo thứ tự (chứ không sử dụng giá trị biên nhận tích luỹ). Biên nhận có lựa chọn chính là lớp giao thức gửi liên tiếp thứ hai, sẽ nghiên cứu dưới đây là *lặp lại có lựa chọn* (selective repeat – SR). Có thể xem TCP là sự kết hợp của cả hai giao thức GBN và SR.

### 3.4.4. Giao thức lặp lại có lựa chọn (Selective Repeat)

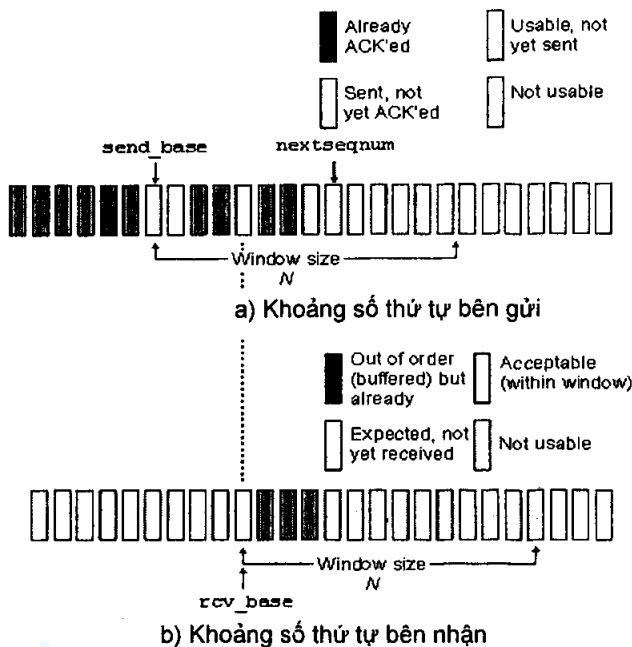
Giao thức GBN cho phép bên gửi "đổ tràn đường truyền" bằng các gói tin như trong Hình 3.17 và do đó khắc phục được hiệu suất thấp của giao thức dừng và chờ. Tuy nhiên, trong một vài tình huống, chính hiệu suất của giao thức GBN cũng cực thấp. Ví dụ, khi kích thước cửa sổ và thời gian truyền một gói tin lớn, có thể có nhiều gói tin ở trên đường truyền. Một gói tin bị lỗi có thể khiến GBN phải truyền lại nhiều gói tin, mà trong một số trường hợp là không cần thiết. Nếu trong ví dụ đọc chính tả đã trình bày ở trên, nếu mỗi từ bị lỗi phải đọc lại khoảng 1.000 từ đứng trước (kích thước cửa sổ là 1.000 từ), thì tốc độ đọc sẽ rất chậm.

Giao thức lặp lại có lựa chọn tránh việc truyền lại không cần thiết bằng cách bên gửi chỉ truyền lại các gói tin mà nó cho là có lỗi (hoặc mất). Để truyền lại từng gói tin cần thiết, bên nhận cần biên nhận cho từng gói tin nhận đúng. Giao thức này vẫn sử dụng kích thước cửa sổ là  $N$  để giới hạn tổng số gói tin chưa được biên nhận trên đường truyền. Tuy nhiên, khác với GBN, bên gửi sẽ nhận được biên nhận ACK cho một số gói tin trong cửa sổ. Hình 3.22 là không gian số thứ tự của phía gửi SR. Hình 3.23 mô tả chi tiết hành động của bên gửi trong giao thức SR.

Bên nhận Selective Repeat sẽ biên nhận cho bất kỳ gói tin nhận đúng; cho dù không theo đúng thứ tự. Gói tin không đúng thứ tự vẫn được lưu giữ lại cho đến khi tất cả các gói tin còn thiếu (gói tin có số thứ tự nhỏ hơn) được chuyển đến, khi đó tất cả các gói tin sẽ được chuyển lên tầng trên theo đúng thứ tự.

Trong Hình 3.23 tóm tắt các hoạt động khác nhau của bên nhận trong SR, đó là một ví dụ hoạt động của SR trong trường hợp mất gói tin. Trong

Hình 3.23b, bên nhận sẽ lưu giữ tạm gói tin 3, 4 và gửi chúng cùng với gói tin 2 lên tầng trên khi gói tin 2 được chuyển đến.



Hình 3.22. Khoảng số thứ tự của bên gửi và bên nhận

### 1. Dữ liệu nhận được từ phía trên

Khi nhận được dữ liệu từ phía trên, bên gửi SR kiểm tra số thứ tự sẽ gửi. Nếu số thứ tự sẽ gửi nằm trong cửa sổ gửi, dữ liệu được đóng gói và gửi đi, ngược lại thì dữ liệu được lưu giữ trong bộ đệm hoặc gửi trả lên tầng trên để gửi sau, giống GBN.

### 2. Hết thời gian đợi (Timeout)

Timer lại được sử dụng để phát hiện mất gói tin. Tuy nhiên, mỗi gói tin gửi đi có một timer riêng, bởi vì chỉ có duy nhất một gói tin được gửi lại khi hết thời gian đợi. Có thể sử dụng đồng hồ hệ thống giữ vai trò đồng bộ cho các timer.

### 3. Nhận được ACK

Nếu nhận được ACK, bên gửi đánh dấu gói tin đã được chuyển đúng. Nếu số thứ tự của gói tin vừa được nhận bằng `send_base`, cánh cửa sổ sẽ trượt tới gói tin có số thứ tự nhỏ nhất chưa được nhận. Nếu cửa sổ di chuyển và còn các gói tin chưa được truyền thì các gói tin đó sẽ được gửi.

Hình 3.23. Sự kiện và phản ứng của bên gửi

**1. Nhận đúng gói tin với số thứ tự trong khoảng [rcv\_base, rcv\_base + N – 1].** Trong trường hợp này, gói tin nhận được nằm trong cửa sổ nhận. Bên nhận gửi biên nhận cho gói tin này. Nếu gói tin đó chưa được nhận từ trước, nó sẽ được ghi lại trong bộ đệm. Nếu gói tin đó có số thứ tự bằng với cận dưới của cửa sổ nhận (rcv\_base trong Hình 3.22), thì nó cùng các gói tin có số thứ tự liên tiếp đã lưu giữ từ trước (bắt đầu từ rcv\_base) được chuyển lên tầng trên. Cửa sổ nhận sẽ trượt về phía trước một khoảng bằng với khoảng số gói tin đã chuyển lên tầng trên. Với ví dụ trên Hình 3.25 khi nhận được gói tin có số thứ tự rcv\_base = 2 thì gói tin này cùng với gói tin rcv\_base + 1 và gói tin rcv\_base + 2 được chuyển lên tầng trên.

**2. Nhận được gói tin với số thứ tự trong [rcv\_base – N, rcv\_base – 1].** Trong trường hợp này, gửi biên nhận lại cho gói tin (mặc dù đã biên nhận từ trước).

**3. Các trường hợp khác.** Bỏ qua gói tin đó.

Hình 3.24. Sự kiện và phản ứng của bên nhận

## 3.5. TCP – GIAO THỨC GIAO VẬN HƯỚNG NỐI

### 3.5.1. Kết nối TCP

Chức năng dồn kệnh, phân kệnh và phát hiện lỗi của TCP giống UDP. Tuy nhiên, TCP và UDP có nhiều điểm khác biệt. Điểm khác nhau cơ bản nhất là UDP không hướng nối, còn TCP hướng nối. UDP không hướng nối do có thể gửi dữ liệu mà không cần phải thiết lập trước đường truyền. TCP hướng nối vì trước khi tiến trình ứng dụng có thể bắt đầu gửi dữ liệu tới tiến trình khác, hai tiến trình này phải có thủ tục "bắt tay" với nhau, nghĩa là chúng phải gửi một số gói segment đặc biệt để xác định các tham số đảm bảo cho quá trình truyền dữ liệu. Trong giai đoạn thiết lập kết nối TCP, hai bên sẽ khởi tạo nhiều biến trạng thái TCP cho kết nối.

"Kết nối" TCP không phải kết nối thực sự giữa hai điểm đầu mút (end – to – end) giống như mạch TDM hay FDM trong mạng chuyển mạch kệnh. Nó cũng không phải là mạch ảo, bởi vì trạng thái kết nối nằm hoàn toàn trên hệ thống đầu/cuối. Giao thức TCP chỉ hoạt động trên thiết bị đầu/cuối và không hoạt động trên các thiết bị trung gian (switch, bridge, router). Trong thực tế, các router trung gian chỉ có thể thấy các datagram, không nhìn thấy các kết nối.

Kết nối TCP cung cấp đường truyền dữ liệu *hai hướng* (song công – full duplex). Nếu có kết nối TCP giữa tiến trình A chạy trên một máy tính

và tiến trình B chạy trên máy tính khác, khi đó dữ liệu ứng dụng có thể truyền từ A tới B cùng lúc với dữ liệu truyền từ B sang A. Kết nối TCP luôn luôn thuộc kiểu điểm nối điểm, giữa một bên gửi và một bên nhận (point – to – point). Chế độ truyền "multicasting" (tiến trình gửi có thể gửi đồng thời một thông điệp tới nhiều tiến trình nhận) không thực hiện được trong TCP.

Bây giờ hãy xem kết nối TCP được thiết lập như thế nào? Giả sử có tiến trình đang chạy trên một máy tính muốn khởi tạo đường truyền tới tiến trình trong một máy tính khác. Nhớ lại rằng, tiến trình nào khởi tạo kết nối là tiến trình khách (client) và tiến trình kia là tiến trình phục vụ (server). Đầu tiên tiến trình ứng dụng client yêu cầu thực thi TCP của mình thiết lập đường kết nối tới tiến trình nào đó trên server. Chương trình bằng ngôn ngữ Java thực hiện điều này bằng cách sử dụng mã:

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

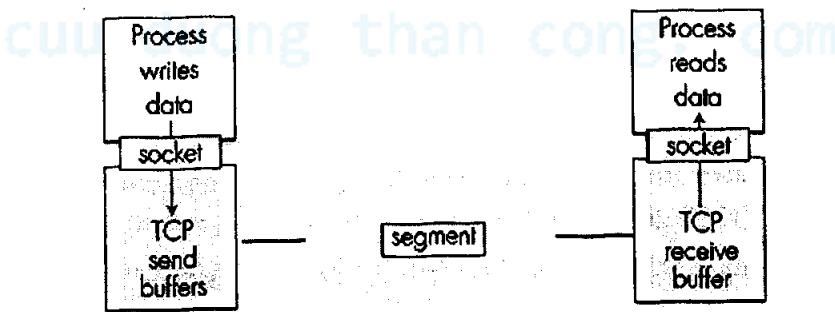
Sau đó thực thi giao vận trong máy client thiết lập kết nối TCP tới thực thi TCP trên máy phục vụ. Chi tiết thủ tục thiết lập đường truyền sẽ được trình bày ở cuối mục này. Bây giờ chỉ cần biết là đầu tiên máy khách sẽ gửi một gói tin TCP đặc biệt, máy server trả lời bằng một gói TCP đặc biệt thứ hai, và cuối cùng client trả lời lại bằng một gói TCP đặc biệt thứ ba. Hai gói TCP đầu tiên không tải, có nghĩa là không có dữ liệu thực sự từ tầng ứng dụng, chỉ bắt đầu từ gói thứ ba mới mang dữ liệu. Vì ba gói dữ liệu đặc biệt này được trao đổi giữa hai máy tính trước khi kết nối, thủ tục thiết lập kết nối này còn được gọi là giai đoạn *bắt tay ba bước* (three way handshake).

Sau khi thiết lập đường truyền TCP, hai tiến trình ứng dụng có thể trao đổi dữ liệu với nhau. TCP là kênh truyền song công nên máy tính có thể gửi và nhận đồng thời. Xét quá trình gửi dữ liệu từ tiến trình client tới tiến trình server. Tiến trình client sẽ "đổ" luồng dữ liệu qua socket ("cửa" của tiến trình). Khi đã qua cửa, tiến trình gửi sẽ không kiểm soát được dữ liệu, mà chính thực thi TCP chạy trên máy client sẽ chịu trách nhiệm kiểm soát. Trong Hình 3.25, TCP đầy dữ liệu vào bộ đệm gửi (send buffer), một trong các bộ đệm được khởi tạo trong quá trình thiết lập kết nối. Sau đó, TCP sẽ lấy và gửi dần dữ liệu trong bộ đệm gửi. Tuy nhiên, đặc tả TCP không xác định tường minh khi nào TCP phải gửi dữ liệu trong bộ đệm. Thường nó chỉ yêu cầu TCP "gửi khi thuận tiện". Lượng dữ liệu ứng dụng lớn nhất có thể đặt trong một segment giới hạn bởi MMS (maximum segement size). Giá trị

MMS phụ thuộc vào chính phần mềm triển khai TCP (thường là hệ điều hành) và có thể cấu hình được. Các giá trị MMS phổ biến thường là 1.500 byte, 536 byte hay 512 byte. (Độ lớn của segment thường được giới hạn để tránh hiện tượng phân mảnh IP, một hiện tượng sẽ được đề cập trong chương sau). Chú ý, MMS là lượng dữ liệu ứng dụng lớn nhất trong segment, chứ không phải là kích thước lớn nhất của segment TCP bao gồm cả tiêu đề (header).

Thực thể TCP gói dữ liệu cùng với TCP header trong TCP segment. TCP segment được chuyển xuống dưới tầng mạng và được đặt trong gói tin của tầng mạng (IP datagram) để gửi qua mạng. Ở phía nhận, thực thể TCP sẽ đặt dữ liệu vào bộ đệm nhận (receiver buffer) của kết nối TCP. Ứng dụng sẽ đọc dòng dữ liệu này từ bộ đệm. Mỗi kết nối đều có bộ đệm gửi và bộ đệm nhận. Bộ đệm gửi và nhận cho dữ liệu được minh họa trên Hình 3.25.

Chú ý rằng, bộ đệm, các biến trạng thái và socket tạo thành kết nối TCP chỉ nằm trên hai thiết bị đầu/cuối, chứ không nằm trên các thiết bị trung gian (router, hub, switch,...).



Hình 3.25. Bộ đệm của thực thể TCP

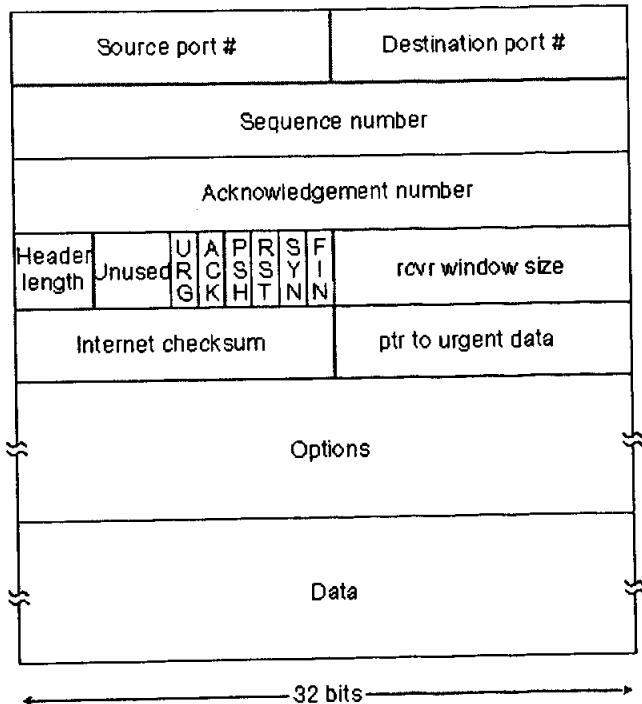
### 3.5.2. Cấu trúc TCP Segment

Sau khi đã nói qua về TCP, bây giờ sẽ xem xét cấu trúc gói dữ liệu TCP (TCP segment). TCP segment bao gồm các trường tiêu đề và trường dữ liệu. Trường dữ liệu chứa một phần dữ liệu ứng dụng. Như đã trình bày ở trên, giá trị MMS giới hạn độ lớn trường dữ liệu của segment. Khi TCP gửi một file lớn – ví dụ file ảnh trong trang Web, nó phải chia file thành các đoạn có kích thước MMS (ngoại trừ đoạn cuối cùng có độ lớn bé hơn hoặc bằng MMS). Tuy nhiên, độ lớn dữ liệu của các ứng dụng tương tác thường nhỏ hơn MMS. Ví dụ, với ứng dụng đăng nhập từ xa (Telnet), trường dữ liệu

trong TCP segment thường chỉ là 1 byte. Độ lớn trường tiêu đề của TCP là 20 byte (của UDP là 12 byte). Segment được Telnet gửi có thể chỉ có 21 byte.

Hình 3.26 minh họa cấu trúc TCP segment. Tương tự UDP, tiêu đề TCP bao gồm trường số hiệu cổng nguồn, số hiệu cổng đích để thực hiện dịch vụ dòn, phân khen dữ liệu cho các ứng dụng bên trên và trường checksum. Tuy nhiên, tiêu đề của TCP segment còn có các trường sau:

- ☞ Trường *số thứ tự* (sequence number) 32 bit và trường *số biên nhận* (acknowledge number) 32 bit được bên gửi và bên nhận sử dụng trong việc cung cấp dịch vụ truyền dữ liệu tin cậy.
- ☞ Trường *độ lớn cửa sổ* (window size) 16 bit được sử dụng để kiểm soát lưu lượng. Nó chính là số lượng dữ liệu tối đa (tính theo byte) mà bên nhận có thể chấp nhận được.
- ☞ Trường *độ dài tiêu đề* (length field) 4 bit xác định độ dài của tiêu đề TCP theo đơn vị là các từ 32 bit. Tiêu đề TCP có thể có độ dài thay đổi phụ thuộc trường *option* (Nếu trường option rỗng, thì chiều dài của tiêu đề TCP là 20 byte).
- ☞ Trường *option* là tùy chọn, có thể thay đổi tùy ý. Trường này được sử dụng khi bên gửi, bên nhận có thể thương lượng về giá trị MMS hoặc giá trị gia tăng của cửa sổ trong mạng cao tốc. Lựa chọn nhãn thời gian (timestamping) cũng được định nghĩa. Xem RFC 854 và RFC 1323 để biết thêm chi tiết.
- ☞ Trường *cờ* (flag) gồm 6 bit. Bit ACK được sử dụng để chỉ ra rằng, giá trị đặt trong trường biên nhận là đúng. Các bit RST, SYN và FIN được sử dụng trong việc thiết lập hay đóng kết nối. Khi bit PSH được bật, thì đây là dấu hiệu để yêu cầu bên nhận phải chuyển dữ liệu lên tầng trên ngay lập tức. Cuối cùng, bit URG được dùng để báo hiệu dữ liệu trong segment, được thực thi tầng trên phía gửi tạo ra là "khẩn cấp". Vị trí byte cuối cùng của dữ liệu khẩn cấp được xác định bởi con trỏ dữ liệu khẩn 16 bit (ptr to urgent data). TCP phải báo cho tầng trên biết có dữ liệu khẩn và đặt con trỏ vào cuối dữ liệu khẩn (Trong thực tế, PSH, URG và con trỏ dữ liệu khẩn không được sử dụng).

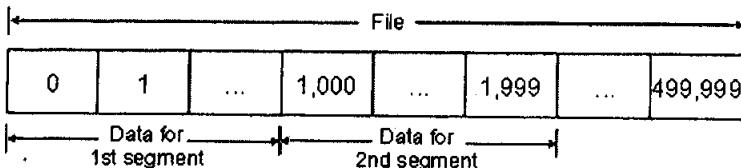


Hình 3.26. Cấu trúc gói dữ liệu TCP

### 3.5.3. Số thứ tự và số biên nhận

Hai trong số những trường quan trọng nhất của tiêu đề TCP segment là trường số thứ tự và trường số biên nhận. Trước khi nói đến những trường này được sử dụng để cung cấp đường truyền dữ liệu tin cậy như thế nào, cần nói đến những trường này nhận giá trị gì?

TCP xem dữ liệu là dòng các byte không có cấu trúc, nhưng có thứ tự và TCP sẽ đánh số thứ tự cho từng byte của dòng dữ liệu này. Mỗi segment có một số thứ tự là số thứ tự của byte đầu tiên của segment. Xét ví dụ sau: Giả sử có tiến trình trên máy A muốn gửi dòng dữ liệu tới tiến trình trên máy B thông qua kết nối TCP. Thực thể TCP trên máy A sẽ đánh số thứ tự cho từng byte trong dòng dữ liệu. Giả sử dòng dữ liệu này chứa file có kích thước 500.000 byte, giá trị MMS là 1000 byte, và byte đầu tiên của dòng dữ liệu được đánh số thứ tự 0. Trong Hình 3.27, TCP sẽ tạo ra 500 segment từ dòng dữ liệu này. Segment đầu tiên có số thứ tự 0, segment thứ hai có số thứ tự là 1000, segment thứ ba có số thứ tự là 2000,... Mỗi số thứ tự như vậy được chèn vào trường số thứ tự trong tiêu đề của TCP segment tương ứng.



Hình 3.27. Chia nhỏ file dữ liệu vào các TCP segment

Số biên nhận phức tạp hơn số thứ tự. Vì TCP là kênh truyền song công, nên A có thể nhận được dữ liệu từ B, trong khi nó gửi dữ liệu tới B (trên cùng kết nối TCP). Mỗi segment đến từ máy B có một số thứ tự cho dòng dữ liệu đi từ B sang A. *Số biên nhận mà máy A đặt trong segment của nó sẽ là số thứ tự của byte tiếp theo mà máy A đang chờ máy B gửi tới.* Để có thể hiểu rõ hơn chúng ta xét ví dụ sau: Giả sử rằng máy A đã nhận được tất cả các byte từ byte số 0 đến byte số 535 máy B gửi đến và giả sử máy A cũng gửi một segment tới máy B. Trong trường hợp này, máy A đợi byte thứ 536 và toàn bộ các byte tiếp theo trong dòng dữ liệu từ máy B. Khi đó máy A đặt giá trị 536 vào trường số biên nhận của segment mà nó gửi tới máy B.

Một ví dụ khác, giả sử rằng máy A đã nhận được một segment từ máy B bao gồm byte 0 đến byte 535 và một segment khác bao gồm byte 900 đến byte 1000. Vì lý do nào đó, máy A không nhận được byte thứ 536 đến byte 899. Trong trường hợp này, máy A sẽ vẫn đợi byte thứ 536 (và các byte tiếp theo) để tạo lại dòng dữ liệu của máy B. Do đó, trong segment tiếp theo bên A gửi cho bên B, trường số biên nhận vẫn chứa giá trị 536. TCP biên nhận tất cả các byte cho đến byte đầu tiên chưa nhận được (còn thiếu trong dòng dữ liệu), cho nên có thể nói TCP biên nhận kiểu tích luỹ (cumulative acknowledgement).

Ví dụ sau đưa ra vấn đề quan trọng, nhưng đơn giản. Bên A nhận được segment thứ ba (byte thứ 900 đến 1.000) trước khi nhận được segment thứ hai (byte thứ 536 đến byte 899). Khi đó segment thứ ba đến không theo đúng thứ tự. Vấn đề ở đây là, máy tính sẽ làm gì khi nó nhận được một segment không đúng thứ tự trong kết nối TCP? Trong khi, các đặc tả RFC TCP không đưa ra bất cứ một quy tắc nào để giải quyết. Chính người lập trình phần mềm TCP sẽ đưa ra cách giải quyết. Về cơ bản, có hai lựa chọn: (1) Bên nhận ngay lập tức loại bỏ các byte không đúng thứ tự; (2) Bên nhận giữ lại các byte không đúng thứ tự và chờ đến khi nhận được các byte thiếu, tạo thành dòng dữ liệu liên tục. Rõ ràng, giải pháp sau có hiệu quả hơn nếu

dánh giá theo hiệu suất mạng, trong khi đó giải pháp thứ nhất đơn giản hóa việc cài đặt TCP. Trong cuốn sách nhập môn này, sẽ coi TCP phía nhận loại bỏ các segment không đúng thứ tự.

### 3.5.4. Truyền dữ liệu tin cậy

Dịch vụ tầng mạng của Internet không tin cậy, do IP không đảm bảo việc chuyển datagram, không đảm bảo gửi datagram đúng thứ tự cũng như không đảm bảo tính toàn vẹn dữ liệu. Với dịch vụ IP, datagram có thể bị tràn tại bộ đệm router và do đó không bao giờ đến được đích, dữ liệu có thể đến không đúng thứ tự hay các bit trong datagram có thể bị lỗi. Bởi vì, segment của tầng giao vận được đặt trong IP datagram để truyền qua mạng, nên segment của tầng giao vận cũng có thể phải gặp những vấn đề nêu trên.

TCP tạo ra đường truyền dữ liệu tin cậy trên dịch vụ không tin cậy của IP. Dịch vụ truyền dữ liệu tin cậy của TCP đảm bảo dòng dữ liệu tới tiến trình nhận không có lỗi, liên tục, không trùng lặp dữ liệu, đúng thứ tự. Có nghĩa là dòng byte nhận được giống hệt dòng byte gửi đi. Mục này nghiên cứu cách thức cung cấp dịch vụ truyền dữ liệu tin cậy của TCP. Người ta thấy rằng, dịch vụ truyền dữ liệu tin cậy của TCP sử dụng rất nhiều nguyên lý cơ bản đã nghiên cứu trong mục 3.4.

Hình 3.28 minh họa các sự kiện chính liên quan đến việc truyền hay truyền lại dữ liệu tại phía bên gửi TCP. Để đơn giản, coi kết nối TCP giữa hai máy A và B chuyển dữ liệu từ máy A tới máy B. Tại phía gửi (máy A), thực thể TCP lấy dữ liệu của tầng ứng dụng, đóng gói trong các segment và chuyển xuống tầng mạng. Do đó, nhận dữ liệu từ tầng ứng dụng, đóng gói dữ liệu trong các segment và gửi segment đi chính là sự kiện quan trọng đầu tiên mà thực thể TCP bên gửi phải xử lý. Ngay sau khi chuyển segment cho IP, TCP khởi động timer cho segment đó. Thời gian đợi hết (timeout) gây ra một ngắt tại máy A. TCP phản ứng với sự kiện timeout, đây chính là sự kiện thứ hai mà bên gửi TCP phải xử lý bằng cách truyền lại segment gây ra ngắt thời gian.

Sự kiện thứ ba mà bên gửi TCP phải xử lý là nhận được một segment biên nhận (ACK) từ bên gửi (chính xác hơn là một segment chứa giá trị trường biên nhận ACK hợp lệ). Ở đây, thực thể TCP phía gửi phải quyết định đó là ACK lần đầu tiên nhận được (tức là biên nhận cho một segment

đã gửi nhưng chưa được biên nhận), hay chỉ là ACK trùng lặp (biên nhận lại một gói tin đã từng được biên nhận). Trong trường hợp thứ nhất, bên gửi sẽ biết rằng, tất cả các byte có số thứ tự không vượt giá trị biên nhận vừa nhận được đã được gửi thành công. Khi đó, bên gửi có thể cập nhật biến trạng thái TCP kiểm soát số thứ tự của byte cuối cùng mà nó cho rằng đã được nhận chính xác và theo đúng thứ tự tại phía bên nhận.

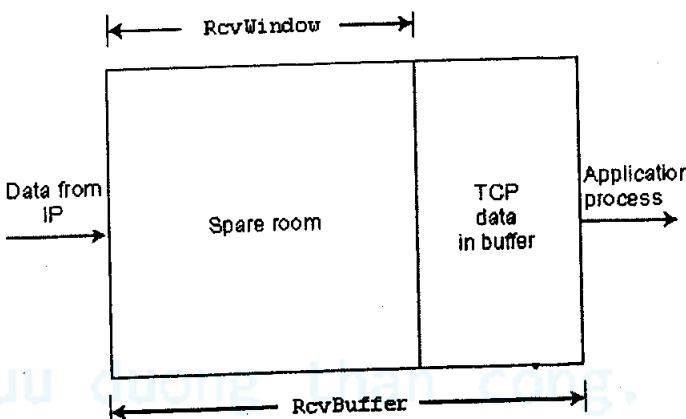
Sự kiện	Hành động tiếp nhận của TCP
Segment đến có số thứ tự là số thứ tự mong muốn. Tất cả dữ liệu đến số thứ tự mong muốn đã được biên nhận. Không có khoảng trống trong dữ liệu nhận được.	Tri hoãn ACK. Đợi segment đúng thứ tự tiếp theo trong khoảng thời gian 500ms. Nếu segment này không xuất hiện mới gửi ACK.
Segment đến có số thứ tự là số thứ tự mong muốn. Segment đến trước đang đợi gửi biên nhận. Không có khoảng trống trong dữ liệu nhận được.	Ngay lập tức gửi đi ACK tích lũy duy nhất biên nhận cho cả hai segment đúng thứ tự.
Segment không đúng thứ tự đến, có số thứ tự cao hơn số thứ tự mong muốn nhận. Phát hiện có khoảng trống dữ liệu.	Ngay lập tức gửi đi ACK trùng lặp và chỉ ra số thứ tự của byte mong muốn nhận tiếp theo.
Segment đến lấp đầy một phần hoặc toàn bộ khoảng trống trong dữ liệu nhận được.	Ngay lập tức gửi đi ACK biên nhận cho đoạn dữ liệu đúng thứ tự liên tục lớn nhất nhận được.

Hình 3.28

Để hiểu về phản ứng của bên gửi, khi nhận được ACK trùng lặp, trước tiên phải xét tại sao bên nhận gửi ACK trùng lặp. Hình 3.28 tóm tắt các chính sách chung của thực thể TCP nhận. Khi nhận được segment có số thứ tự lớn hơn số thứ tự đang được mong đợi, bên nhận phát hiện có đoạn trống trong dòng dữ liệu – nghĩa là thiếu segment. Vì TCP không sử dụng biên nhận phủ định (NAK), nên bên nhận không thể gửi biên nhận phủ định. Thay vào đó, nó biên nhận lại byte đúng thứ tự cuối cùng mà nó nhận được (tạo ra ACK trùng lặp). Nếu bên gửi TCP nhận được 3 ACK trùng lặp cho cùng một segment, nó sẽ cho rằng segment ngay sau segment được biên nhận ba lần bị mất. Trong trường hợp này, TCP thực hiện cơ chế *truyền lại nhanh* (fast retransmit) [RFC 258], gửi lại segment bị cho là mất trước khi timer của segment đó hết hạn (kết thúc).

### 3.5.5. Kiểm soát lưu lượng

Nhắc lại rằng, thiết bị đầu/cuối ở mỗi phía của kết nối TCP đều có bộ đệm dữ liệu (buffer). Khi nhận được đúng một dòng byte liên tục (đúng thứ tự), TCP sẽ đặt dòng byte này vào bộ đệm nhận (receive buffer). Tiến trình ứng dụng nhận sẽ đọc dữ liệu từ bộ đệm này, nhưng không nhất thiết phải đọc ngay khi dữ liệu đến. Có thể tiến trình ứng dụng nhận phải thực hiện nhiều tác vụ khác, nên chưa đọc ngay dữ liệu trong bộ đệm. Nếu ứng dụng đọc dữ liệu chậm, bên gửi có thể làm tràn bộ đệm nhận do dữ liệu được gửi quá nhiều và quá nhanh. Chính vì lý do này, TCP cung cấp dịch vụ *kiểm soát lưu lượng* (flow control) để tránh hiện tượng bên gửi làm tràn bộ đệm bên nhận. Kiểm soát lưu lượng là quá trình làm tương thích (matching) về tốc độ (tương thích giữa tốc độ gửi và tốc độ nhận). Như đã lưu ý ở trên, bên gửi TCP cũng bị giới hạn do tắc nghẽn trong mạng IP, đây chính là cơ chế *kiểm soát tắc nghẽn* (congestion control) của TCP. Mặc dù kiểm soát lưu lượng giống kiểm soát tắc nghẽn (hạn chế tốc độ gửi của bên gửi), tuy nhiên chúng được thực hiện với những mục đích khác nhau. Nhiều người coi hai thuật ngữ này tương đương nhau, vì vậy người đọc nên xem xét kỹ để phân biệt hai trường hợp.



Hình 3.29. Biến receive window và bộ đệm nhận

Để cung cấp cơ chế kiểm soát lưu lượng, TCP bên gửi sử dụng biến receive window. Đây là giá trị mà bên nhận báo cho bên gửi biết độ lớn vùng đệm còn rỗng của mình. Trong kết nối hai hướng, ở mỗi phía kết nối có một giá trị receive window riêng. Giá trị receive window thay đổi trong thời gian kết nối. Để hiểu rõ, xét giá trị receive window trong ví dụ truyền file.

Giả sử máy A gửi một file lớn tới máy B qua kết nối TCP. Máy B sẽ khởi tạo bộ đệm cho kết nối này với độ lớn **RcvBuffer**. Tiến trình ứng dụng trên B đọc dữ liệu từ bộ đệm. Ta định nghĩa một số biến sau:

- ☞ **LastByteread** = số thứ tự của byte cuối cùng trong dòng dữ liệu mà tiến trình ứng dụng trong máy B đọc từ buffer.
- ☞ **LastByteRcvd** = số thứ tự byte cuối cùng trong dòng dữ liệu đến từ mạng và được để trong receive buffer trên máy B

Vì TCP không cho phép tràn bộ đệm nên

$$\text{LastByteRcvd} - \text{LastByteread} \leq \text{RcvBuffer}$$

Receive window là giá trị **RcvWindow**, là độ lớn vùng đệm rồi:

$$\text{RcvWindow} = \text{RcvBuffer} - [\text{LastByteRcvd} - \text{LastByteread}]$$

Bởi vì độ lớn vùng đệm rồi thay đổi theo thời gian nên giá trị **RcvWindow** cũng biến đổi. Biến **RcvWindow** được minh họa trong Hình 3.29.

Kết nối sử dụng biến **RcvWindow** để cung cấp dịch vụ kiểm soát lưu lượng như thế nào? Máy B báo cho máy A độ lớn vùng rồi trong bộ đệm của mình bằng cách đặt giá trị **RcvWindow** hiện thời vào trong trường window của tất cả các segment gửi tới A. Ban đầu máy B thiết lập **RcvWindow = RcvBuffer**. Rõ ràng, để đạt được điều này thì máy B phải kiểm soát vài biến kết nối.

Máy A cũng có hai biến **LastByteSent** và **LastByteAcked**. Độ lệch giữa hai biến này, **LastByteSent – LastByteAcked** là số lượng dữ liệu chưa được biến nhận mà A gửi qua kết nối. Bằng cách không chế số lượng dữ liệu chưa được biến nhận nhỏ hơn giá trị **RcvWindow**, A đảm bảo không làm tràn bộ đệm tại B. Do vậy, trong suốt thời gian kết nối, A phải đảm bảo:

$$\text{LastByteSent} - \text{LastByteAcked} \leq \text{RcvWindow}$$

Một vấn đề kỹ thuật nhỏ này sinh ở đây: Giả sử bộ đệm ở máy B đầy, có nghĩa là **RcvWindow = 0**. Sau khi thông báo tới máy A là **RcvWindow = 0**, máy B không có gì để gửi tới máy A. Khi tiến trình ứng dụng ở máy B lấy dữ liệu lên làm cho bộ đệm rỗng, thì TCP không gửi segment mới cùng với giá trị **RcvWindow** mới tới máy A – TCP chỉ gửi segment tới A khi có dữ liệu hoặc ACK để gửi. Bởi vậy, máy A không bao giờ được thông báo đã có thêm khoảng trống trong bộ đệm ở máy B. Máy A bị "khoá" và không thể

truyền thêm dữ liệu. Để giải quyết vấn đề này, đặc tả TCP yêu cầu máy A tiếp tục gửi segment với một byte dữ liệu khi receive window của máy B bằng 0. Những segment này sẽ được B nhận. Khi bộ đệm bắt đầu có vùng rỗng, thì trong gói nhận sẽ có cả giá trị khác 0 của RcvWindow.

Khác với TCP, UDP không có cơ chế kiểm soát lưu lượng. Để hiểu vấn đề này, hãy xem thực thể UDP gửi các segment từ tiến trình trên máy A tới tiến trình trên máy B. UDP sẽ đặt các segment (chính xác hơn là dữ liệu trong segment) vào trong một hàng đợi có độ lớn hữu hạn ứng với socket nào đó ("cửa" của tiến trình). Tiến trình đọc lần lượt từng segment trong hàng đợi. Nếu tốc độ đọc của tiến trình không đủ nhanh thì hàng đợi sẽ tràn và các segment đến sau sẽ bị mất.

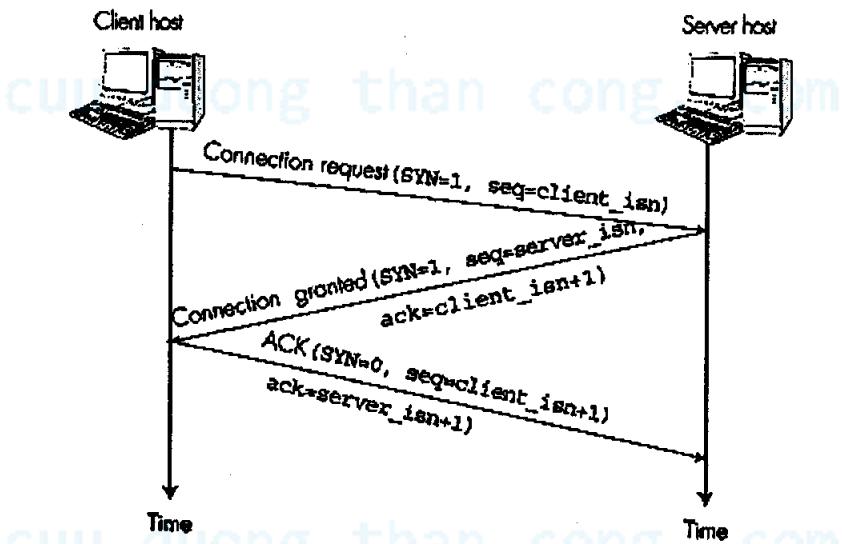
### 3.5.6. Quản lý kết nối TCP

Trong phần này sẽ trình bày một kết nối TCP được thiết lập và giải phóng như thế nào? Đây là vấn đề quan trọng, bởi vì giai đoạn thiết lập kết nối TCP ảnh hưởng lớn đến độ trễ (chẳng hạn như khi duyệt Web). Trước hết, xem một kết nối TCP được thiết lập như thế nào? Giả sử tiến trình chạy trên máy client muốn khởi tạo một kết nối tới tiến trình trên server. Đầu tiên, tiến trình ứng dụng trên client yêu cầu thực thể TCP của nó (client) thiết lập một kết nối tới một tiến trình trên server. Sau đó, thực thể TCP client khởi tạo kết nối TCP tới thực thể TCP trên server qua các bước sau:

- ☛ **Bước 1:** Phía TCP client gửi một segment đặc biệt tới TCP server. Segment đặc biệt này không chứa dữ liệu của tầng ứng dụng, nhưng cờ SYN (một bit thuộc trường cờ (flag)) trong phần tiêu đề được đặt giá trị 1. Vì thế, đôi khi segment đặc biệt này được gọi là SYN segment. Ngoài ra, TCP client chọn số thứ tự ban đầu (client\_isn) và đặt giá trị này vào trường số thứ tự của SYN segment. Segment này được đặt trong IP datagram để gửi tới server.
- ☛ **Bước 2:** Khi IP datagram chứa TCP segment đến server (nếu đến được), thì server lấy SYN segment ra khỏi datagram, phân phôi bộ đệm và các biến TCP phục vụ kết nối đồng thời gửi đi một segment đặc biệt thông báo chấp nhận kết nối từ client. Segment này cũng không chứa dữ liệu của tầng ứng dụng. Tuy nhiên, nó chứa ba

thông tin quan trọng trong phần tiêu đề. Thứ nhất bit SYN sẽ được thiết lập giá trị 1. Thứ hai, trường biên nhận trong tiêu đề nhận giá trị  $client\_isn + 1$ . Cuối cùng, server chọn số thứ tự bắt đầu của mình ( $server\_isn$ ) và đặt giá trị này vào trường số thứ tự trong tiêu đề của segment. Với segment chấp nhận kết nối, server ngũ ý "đã nhận được từ client gói SYN yêu cầu thiết lập kết nối với số thứ tự bắt đầu từ  $client\_isn$ . Chấp nhận thiết lập kết nối này. Số thứ tự của server bắt đầu từ  $server\_isn$ ". Đôi khi đây được gọi là *SYNACK segment*.

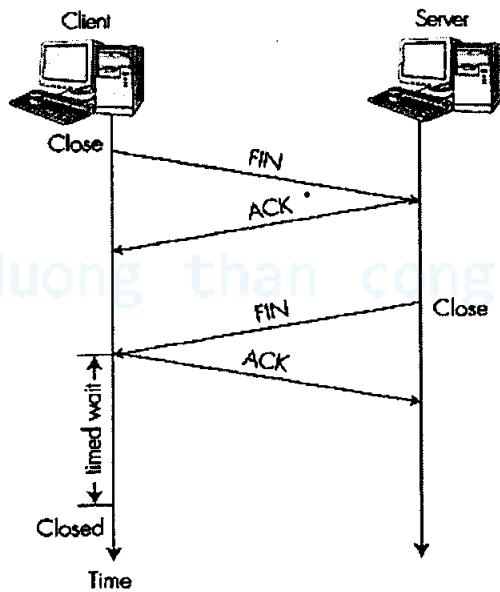
- ☞ **Bước 3:** Khi nhận được segment chấp nhận kết nối, client khởi tạo bộ đệm và các biến phục vụ kết nối. Client gửi segment thứ ba biên nhận cho segment chấp nhận kết nối của server (bằng cách đặt giá trị  $server\_isn + 1$  vào trường số biên nhận trong tiêu đề của TCP segment). Bit SYN được đặt giá trị 0, vì kết nối đã được thiết lập.



Hình 3.30. Giai đoạn "bắt tay ba bước" trong thiết lập đường truyền của TCP

Sau khi thực hiện xong ba bước này, client và server có thể trao đổi các segment chứa dữ liệu. Bit SYN trong các segment sau được đặt giá trị 0. Như vậy, để thiết lập được kết nối hai máy phải trao đổi ba segment (Hình 3.30). Vì thế thủ tục kết nối được xem là *quá trình bắt tay ba bước* (three way handshake).

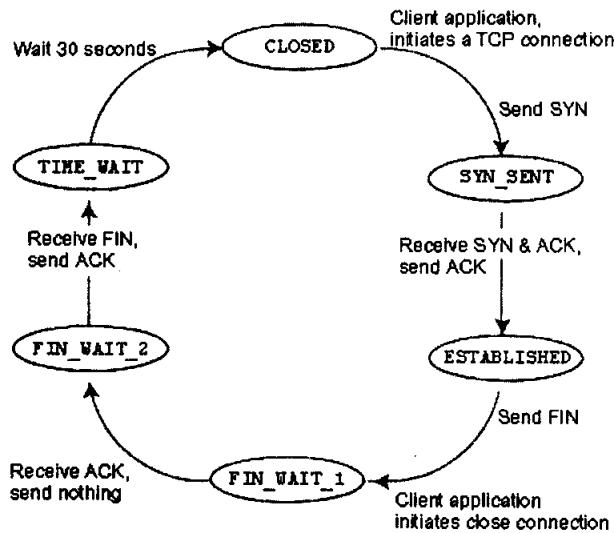
Bây giờ xét đến việc đóng kết nối TCP. Cả hai tiến trình tham gia kết nối TCP đều có thể kết thúc kết nối. Khi kết nối đóng lại, các tài nguyên phục vụ kết nối (bộ đệm và các biến TCP) trong máy được giải phóng. Ví dụ, client quyết định đóng kết nối (Hình 3.31). Tiến trình ứng dụng client sẽ đưa ra lệnh đóng. Khi đó TCP client gửi một segment TCP đặc biệt đến tiến trình server. Đây là FIN segment, vì cờ FIN trong segment này được đặt giá trị 1. Khi server nhận được segment FIN, nó sẽ gửi lại cho client một segment ACK biên nhận segment FIN của client. Tiếp theo server gửi lại một segment kết thúc FIN (có cờ FIN được đặt giá trị 1). Cuối cùng, client gửi segment ACK biên nhận segment FIN của server. Tại thời điểm này thì tất cả tài nguyên trên hai máy đều được giải phóng.



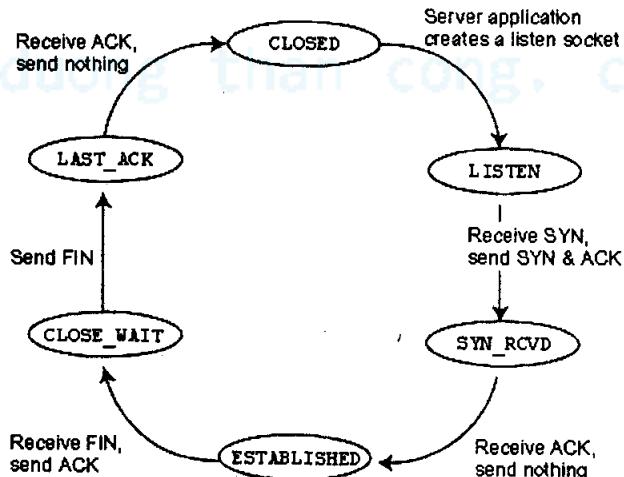
Hình 3.31. Kết thúc kết nối TCP

Trong suốt thời gian kết nối TCP, giao thức TCP chạy trên mỗi máy chuyển qua các trạng thái TCP (TCP state). Hình 3.32 minh họa quá trình thay đổi trạng thái TCP xảy ra bên phía client. TCP client bắt đầu ở trạng thái đóng (CLOSED). Ứng dụng bên phía client khởi tạo một kết nối TCP. Điều này đòi hỏi TCP client gửi SYN segment tới TCP server. Sau khi gửi SYN segment, TCP client chuyển sang trạng thái SYN\_SENT. Trong trạng thái SYN\_SENT, TCP client đợi SYNACK segment (biên nhận cho SYN segment của mình). Khi nhận được segment này, TCP client chuyển sang trạng thái ESTABLISHED. Ở trạng thái ESTABLISHED, TCP client có thể

gửi và nhận TCP segment chứa dữ liệu (là dữ liệu thực sự do ứng dụng tạo ra).



Hình 3.32. Dòng trạng thái của TCP



Hình 3.33. Các trạng thái của kết nối TCP

Giả sử ứng dụng client quyết định đóng kết nối (server tương tự). Khi đó TCP client gửi FIN segment và chuyển sang trạng thái **FIN\_WAIT\_1**. Trong trạng thái này, TCP client đợi segment biên nhận từ phía server. Sau khi nhận được segment này, TCP client chuyển sang trạng thái **FIN\_WAIT\_2**. Trong trạng thái **FIN\_WAIT\_2**, TCP client đợi FIN segment từ server. Sau khi nhận segment này, TCP client gửi segment ACK biên

nhận tới server và chuyển sang trạng thái TIME\_WAIT. Trong trạng thái TIME\_WAIT, TCP client có thể gửi lại biên nhận ACK trong trường hợp ACK trước bị mất. Thời gian đợi ở trạng thái TIME\_WAIT phụ thuộc vào phần mềm triển khai TCP, nhưng thường nhận các giá trị 30 giây, một phút, hai phút. Sau khi hết thời gian đợi, kết nối chính thức được đóng và tất cả tài nguyên phía client (bao gồm cả số hiệu cổng) được giải phóng.

Hình 3.33 minh họa quá trình thay đổi trạng thái xảy ra ở TCP server. Giả sử client yêu cầu kết thúc trước.

Trong hai sơ đồ biến đổi trạng thái vừa nghiên cứu, chúng ta đã tìm hiểu về cơ chế thiết lập và giải phóng đường truyền của TCP.

### 3.6. KIỂM SOÁT TẮC NGHẼN TRONG TCP

Trong các phần trước, chúng ta đã nghiên cứu cách thức TCP cung cấp dịch vụ truyền tin tin cậy giữa hai tiến trình chạy trên những thiết bị đầu/cuối khác nhau. Một dịch vụ cực kỳ quan trọng khác của TCP là cơ chế kiểm soát tắc nghẽn. Cơ chế này của TCP chỉ dựa vào các thiết bị đầu/cuối chứ không dựa vào cơ chế kiểm soát tắc nghẽn của tầng mạng, vì tầng IP không cung cấp cho TCP các thông tin minh bạch khi có tắc nghẽn. Trước khi đi sâu vào chi tiết, xét chung về cơ chế kiểm soát tắc nghẽn của TCP và mục tiêu kiểm soát tắc nghẽn của TCP khi nhiều kết nối TCP cùng chia sẻ một đường truyền bị tắc nghẽn.

Kết nối TCP kiểm soát tốc độ truyền của mình bằng cách giới hạn số lượng các segment đã gửi, nhưng chưa được biên nhận. Giả sử  $w$  là số lượng cho phép các segment chưa cần biên nhận, thường được coi như *kích thước cửa sổ của TCP* (TCP window). Điều lý tưởng là kết nối TCP cho phép truyền với tốc độ tối đa có thể (càng nhiều segment chưa biên nhận càng tốt) chừng nào chưa xảy ra hiện tượng mất segment do bị tắc nghẽn. Nói chung, kết nối TCP bắt đầu với giá trị  $w$  tương đối nhỏ và sau đó "thăm dò" kênh truyền còn rỗi không bằng cách tăng dần giá trị  $w$ . Kết nối TCP tiếp tục tăng  $w$  cho đến khi xảy ra mất dữ liệu (sự kiện hết thời gian đợi – timeout hay nhận được các biên nhận trùng lặp). Khi đó, TCP sẽ giảm  $w$  tới một giá trị "an toàn" và sau đó lại bắt đầu "thăm dò" kênh truyền rỗi bằng cách tăng dần giá trị  $w$ .

### a) Tổng quan về kiểm soát tắc nghẽn của TCP

Trong mục 3.5 biết rằng, mỗi kết nối TCP có bộ đệm gửi, bộ đệm nhận và một số biến (**LastByteRead**, **RcvWin**,...). Cơ chế kiểm soát tắc nghẽn của TCP bổ sung thêm hai biến nữa là congestion window (**CongWin**) (cửa sổ tắc nghẽn) và **threshold** (ngưỡng). Cửa sổ tắc nghẽn ký hiệu là **CongWin**, biểu thị số lượng dữ liệu tối đa mà người gửi có thể gửi qua kết nối. Như vậy, khối lượng dữ liệu được gửi không được vượt quá **CongWin** và **RcvWin**, tức là:

$$\text{Last ByteSent} - \text{LastByteAcked} \leq \min \{\text{CongWin}, \text{RcvWin}\}.$$

Ngưỡng ký hiệu là **threshold** sẽ ảnh hưởng tới quá trình tăng của **CongWin** như trình bày dưới đây.

Hãy xem giá trị **CongWin** tăng trong suốt kết nối TCP như thế nào? Để tập trung vào cơ chế kiểm soát tắc nghẽn (khác với kiểm soát lưu lượng), giả thiết rằng bộ đệm nhận đủ lớn để có thể bỏ qua các hạn chế của cửa sổ nhận. Trong trường hợp này, số lượng dữ liệu gửi chưa cần được biên nhận chỉ bị giới hạn bởi **CongWin**, cũng giả thiết rằng phía gửi cần gửi nhiều dữ liệu.

Sau khi thiết lập kết nối TCP giữa hai hệ thống đầu/cuối, tiến trình ứng dụng bên gửi chuyển dữ liệu tới bộ đệm gửi của TCP. TCP chia dữ liệu thành các khối với kích thước MSS, đặt các khối dữ liệu trong TCP segment, và chuyển segment xuống tầng mạng để gửi đi. Cửa sổ tắc nghẽn của TCP điều tiết số lượng segment được gửi. Ban đầu, **CongWin** nhận giá trị 1 MSS, TCP gửi segment đầu tiên và được biên nhận. Nếu segment này được biên nhận trước khi timeout, phía gửi tăng **CongWin** lên 1 MSS và gửi đi hai segment. Nếu những segment này được biên nhận trong thời gian đợi của chúng, **CongWin** lại được tăng thêm 1 MSS cho mỗi segment được biên nhận. Khi đó **CongWin** mới là 4 MSS và phía gửi gửi đi bốn segment. Thủ tục này được thực hiện liên tục cho tới khi (1) **CongWin** vượt ngưỡng (threshold) hay (2) không nhận được biên nhận trong thời gian chờ biên nhận.

Trong giai đoạn này, cửa sổ tắc nghẽn (**CongWin**) tăng theo hàm số mũ. Ban đầu nó nhận giá trị 1 MSS, sau đó tăng lên 2 MSS, 4 MSS, 8 MSS,... Đây là giai đoạn *khởi đầu chậm* (slow start) vì giá trị cửa sổ khởi đầu với giá trị nhỏ (1 MSS). Tuy vậy giá trị cửa sổ tăng khá nhanh.

Giai đoạn slow – start kết thúc khi **CongWin** vượt ngưỡng. Khi đó giá trị **CongWin** sẽ tăng tuyến tính chứ không còn tăng theo hàm số mũ. Tức là

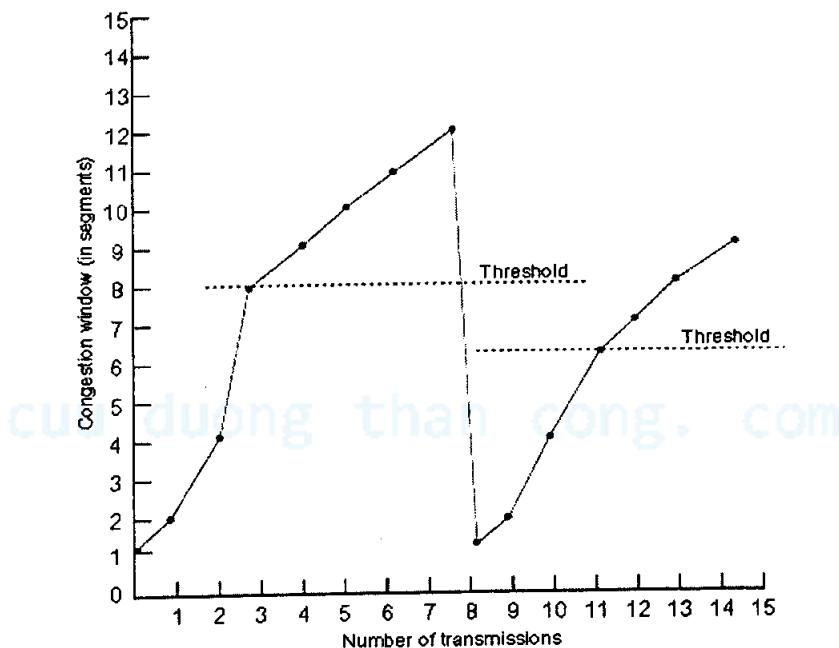
nếu **CongWin** =  $w$ , sau khi nhận được biên nhận cho  $w$  segment, giá trị **CongWin** sẽ tăng lên 1, **CongWin** =  $w + 1$ . Đây là giai đoạn *tránh tắc nghẽn* (congestion avoidance).

Giai đoạn tránh tắc nghẽn tiếp tục khi vẫn nhận được biên nhận trong thời gian đợi. Tuy nhiên, giá trị cửa sổ cũng như tốc độ gửi dữ liệu không thể tăng mãi. Đến lúc nào đó sẽ xảy ra sự cố mất gói dữ liệu ở router. Điều này dẫn đến sự kiện timeout ở phía gửi. Lúc này giá trị ngưỡng (threshold) nhận giá trị bằng một nửa **CongWin**, **CongWin** được đặt bằng 1 MMS. Bên gửi sẽ tiếp tục tăng nhanh giá trị **CongWin** theo hàm số mũ cho đến khi nó vượt ngưỡng.

*Tóm lại:* Khi cửa sổ tắc nghẽn chưa vượt ngưỡng, cửa sổ sẽ tăng theo hàm mũ. Khi cửa sổ tắc nghẽn vượt ngưỡng, cửa sổ sẽ tăng tuyến tính.

Khi hết thời gian đợi, giá trị ngưỡng bằng một nửa giá trị cửa sổ tắc nghẽn hiện thời và cửa sổ tắc nghẽn nhận giá trị 1.

Nếu bỏ qua giai đoạn slow – start, sẽ thấy TCP tăng độ lớn cửa sổ theo cấp số cộng khi mạng chưa bị tắc nghẽn và giảm độ lớn cửa sổ theo cấp số nhân (chia 2) ngay khi mạng bị tắc nghẽn. Vì vậy, TCP được coi là thuật toán AIMD (additive – increase, multiplicative – decrease).



Hình 3.34. Cửa sổ kiểm soát tắc nghẽn

Giá trị độ lớn cửa sổ tắc nghẽn của TCP được minh họa trong Hình 3.34. Trong hình này, ngưỡng ban đầu bằng 8 MSS. Cửa sổ tắc nghẽn tăng nhanh theo luỹ thừa 2 trong giai đoạn slow-start và đạt ngưỡng tại  $t = 3$ . Giá trị cửa sổ tắc nghẽn tăng tuyến tính đến khi xuất hiện mất mát dữ liệu. Giả sử khi dữ liệu bị mất, cửa sổ tắc nghẽn có giá trị 12 MMS. Ngưỡng mới được đặt bằng  $0.5 \text{ CongWin} = 6 \text{ MSS}$  và cửa sổ tắc nghẽn bằng 1 MMS. Quá trình lại được tiếp tục. Thuật toán kiểm soát tắc nghẽn này do V.Jacobson đề xuất. Hiện nay có nhiều biến thể của thuật toán Jacobson (xem RFC 2581).

### b) Tahoe, Reno và Vegas

Thuật toán kiểm soát tắc nghẽn của TCP ở đây được gọi là *Tahoe*. Một vấn đề với thuật toán Tahoe là khi một segment bị mất, người gửi có thể phải đợi trong một khoảng thời gian dài để gửi lại. Vì vậy một biến thể của Tahoe gọi là *Reno* đã được triển khai trong phần lớn các hệ điều hành. Giống Tahoe, Reno đặt độ lớn cửa sổ tắc nghẽn bằng 1 khi timeout (hết thời gian của bộ định thời). Phía gửi sẽ gửi lại gói tin đã nhận được biên nhận trùng lặp ba lần liên tiếp ngay cả khi chưa hết thời gian đợi của gói tin này. Reno cũng sử dụng cơ chế khôi phục nhanh (fast recovery). Hiện nay phần lớn thực thể TCP sử dụng thuật toán Reno. Tuy nhiên có nhiều thuật toán cải tiến đáng kể hiệu suất của Reno – như *Vegas*.

# **Chương 4**

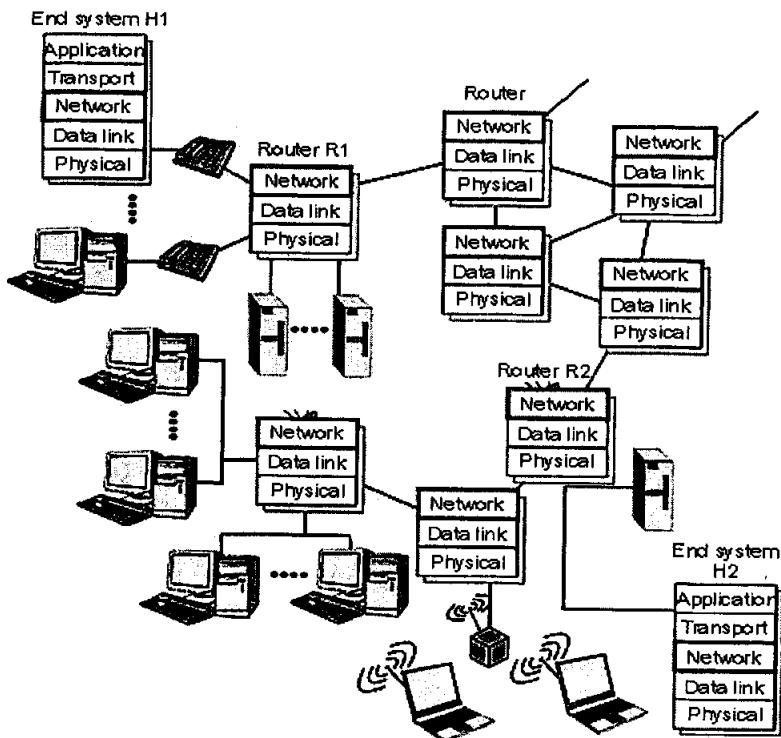
## **TẦNG MẠNG**

---

### **4.1. CÁC MÔ HÌNH DỊCH VỤ CỦA TẦNG MẠNG**

Như đã trình bày trong các chương trước, tầng giao vận cung cấp dịch vụ truyền thông giữa hai tiến trình đang chạy trên hai máy tính khác nhau. Để cung cấp được dịch vụ này, tầng giao vận phải sử dụng dịch vụ cung cấp đường truyền giữa hai máy tính của tầng mạng. Cụ thể hơn, tầng mạng chuyển gói tin (segment) của tầng giao vận từ máy tính này đến máy tính khác. Tại máy tính gửi, tất cả các segment của tầng giao vận được chuyển xuống tầng mạng. Nhiệm vụ của tầng mạng là chuyển những segment này đến máy tính đích và gửi tới thực thể nào đó ở tầng giao vận bên trên. Công việc chuyển segment từ tầng giao vận máy tính nguồn đến tầng giao vận máy tính đích của tầng mạng chính là nội dung của chương này. Trong chương này sẽ thấy rằng, không giống tầng giao vận, tầng mạng gồm nhiều máy tính và các router trung gian. Vì thế, giao thức tầng mạng là một trong những giao thức phức tạp nhất.

Hình 4.1 minh họa một mạng đơn giản với hai máy tính (H1 và H2) và một số router trên đường truyền giữa H1 và H2. Tầng mạng ở máy tính gửi thực hiện bước gửi đầu tiên trên toàn bộ hành trình của gói tin. Ví dụ, nếu H1 gửi gói tin đến H2 thì tầng mạng trên H1 sẽ truyền gói tin này đến router gần nhất (R2). Tại máy tính nhận (chẳng hạn H2), tầng mạng sẽ nhận gói tin từ router gần nó nhất (trong trường hợp này là R2) và chuyển lên cho tầng giao vận tại H2. Vai trò chính của router là chuyển gói tin từ một đầu vào nào đó (input) tới một đầu ra nào đó (output). Chú ý rằng, các tầng trên của tầng mạng (giao vận, ứng dụng) không hoạt động tại các router trong Hình 4.1, bởi vì router không cần thiết phải chạy các giao thức ở tầng giao vận hay tầng ứng dụng (ngoại trừ các mục đích kiểm soát).



Hình 4.1. Một mô hình mạng đơn giản

Vai trò của tầng mạng đơn giản, nó chỉ là chuyển gói tin từ máy tính gửi đến máy tính nhận. Vì thế, tầng mạng có ba chức năng quan trọng sau:

- ☞ **Xác định đường đi** (Path determination): Tầng mạng phải xác định các router trung gian hay tuyến đường (path) mà gói tin được truyền từ nơi gửi đến nơi nhận. Thuật toán xác định tuyến đường như vậy gọi là *thuật toán định tuyến* (routing algorithm). Thuật toán định tuyến sẽ quyết định đường đi của các gói tin từ máy tính nhận đến máy tính gửi (trong ví dụ này là máy tính H1 và máy tính H2). Trọng tâm của chương này là các thuật toán định tuyến. Trong mục 4.2, sẽ nghiên cứu lý thuyết của thuật toán định tuyến, tập trung vào hai kiểu thuật toán là *Link state* và *Distance vector*. Độ phức tạp của thuật toán định tuyến phụ thuộc vào số lượng router trên đường truyền. Điều này dẫn đến định tuyến phân cấp, một chủ đề được trình bày trong mục 4.3.
- ☞ **Chuyển mạch** (Switching): Khi gói tin đến đầu vào, router phải quyết định gửi gói tin đến đầu ra thích hợp. Ví dụ, gói tin từ máy

H1 đến router R1 sẽ được chuyển đến router kế tiếp trên đường tới H2. Trong mục 4.6 sẽ nghiên cứu hoạt động của router và quá trình chuyển một gói tin từ đầu vào đến đầu ra trong một router diễn ra như thế nào.

- ☞ **Thiết lập đường truyền** (Call setup): Trong phần trước trình bày về TCP, ta thấy rằng, hai thực thể truyền thông phải có một giai đoạn "bắt tay" trước khi trao đổi dữ liệu thực sự. Điều này cho phép bên gửi và bên nhận thiết lập các thông tin trạng thái cần thiết (ví dụ, số thứ tự khởi đầu, độ lớn cửa sổ). Với một số kiến trúc mạng khác (ví dụ như ATM) đòi hỏi các router trên tuyến đường từ nguồn đến đích phải "bắt tay" nhau trước khi bắt đầu truyền dữ liệu thực sự. Trong tầng mạng, quá trình này được gọi là *thiết lập đường truyền* (call setup).

Trước khi đi sâu vào chi tiết các khái niệm và triển khai của tầng mạng, xét tổng quát những kiểu dịch vụ khác nhau của tầng mạng.

#### 4.1.1. Mô hình dịch vụ mạng

Khi tầng giao vận ở thiết bị gửi chuyển các gói tin xuống tầng mạng, liệu tầng giao vận có thể tin cậy tầng mạng chuyển gói tin này đến đích hay không? Liệu khi gửi nhiều gói tin, chúng có được chuyển đến tầng giao vận của thiết bị nhận theo đúng thứ tự không? Vận tốc gửi các gói tin có được xác định trước không? Tầng mạng có phản hồi lại các thông tin liên quan đến tắc nghẽn không? Mô hình dịch vụ mạng sẽ trả lời những câu hỏi này và nhiều vấn đề khác.

Điểm trừu tượng quan trọng nhất mà tầng mạng che dấu các tầng trên là việc có sử dụng *mạch ảo* (Virtual Circuit – VC) hay không. Về khía cạnh nào đó, mạch ảo tương tự mạng điện thoại truyền thống (mặc dù mạng điện thoại sử dụng mạch thực). Có ba giai đoạn trong chuyển mạch ảo:

- ☞ **Thiết lập mạch ảo:** Trong cả giai đoạn thiết lập, phía gửi thông báo địa chỉ nhận với tầng mạng, yêu cầu tầng mạng thiết lập VC. Tầng mạng xác định tuyến đường giữa bên gửi và bên nhận, tức là chuỗi các cung đường (hay đường kết nối – link) cũng như các thiết bị chuyển mạch (switch – nút trung gian) mà tất cả các gói dữ

liệu sẽ đi qua. Giai đoạn này yêu cầu việc cập nhật bảng định tuyến và dự trữ tài nguyên trong mỗi thiết bị chuyển mạch.

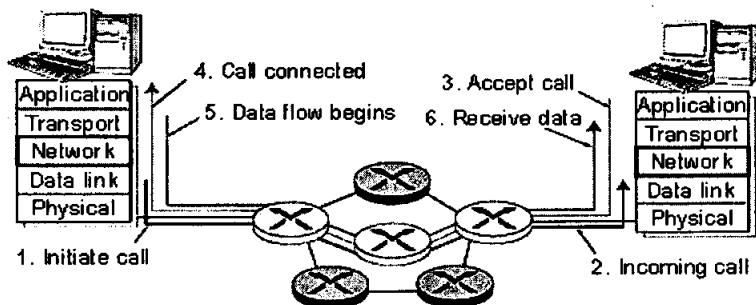
- ☞ **Truyền dữ liệu:** Sau khi thiết lập, dữ liệu có thể được truyền trong VC.
- ☞ **Giải phóng mạch ảo:** Giai đoạn này bắt đầu khi phía gửi (hoặc phía nhận) báo cho tầng mạng yêu cầu đóng VC. Tầng mạng sẽ thông báo cho thiết bị đầu/cuối bên kia cũng như các thiết bị chuyển mạch trên VC để cập nhật lại các bảng định tuyến, giải phóng tài nguyên.

Có sự khác biệt tuy nhỏ, nhưng quan trọng giữa thiết lập VC ở tầng mạng và thiết lập kết nối ở tầng giao vận (giai đoạn bắt tay 3 bước của TCP). Thiết lập kết nối ở tầng giao vận chỉ liên quan đến các thiết bị đầu/cuối ở trên hai đầu mút. Hai thiết bị đồng ý thiết lập kết nối và thỏa thuận các thông số của kết nối (ví dụ, số thứ tự khởi tạo, độ lớn cửa sổ kiểm soát lưu lượng). Hai thiết bị đầu/cuối này sẽ nhận biết được về sự kết nối ở tầng giao vận, nhưng các thiết bị chuyển mạch ở giữa thì không. Trái lại, trong tầng mạng của mạng chuyển mạch ảo, tất cả các thiết bị chuyển mạch giữa hai thiết bị đầu/cuối đều tham gia vào quá trình thiết lập mạch ảo, do đó đều nhận biết được tất cả các VC đi qua.

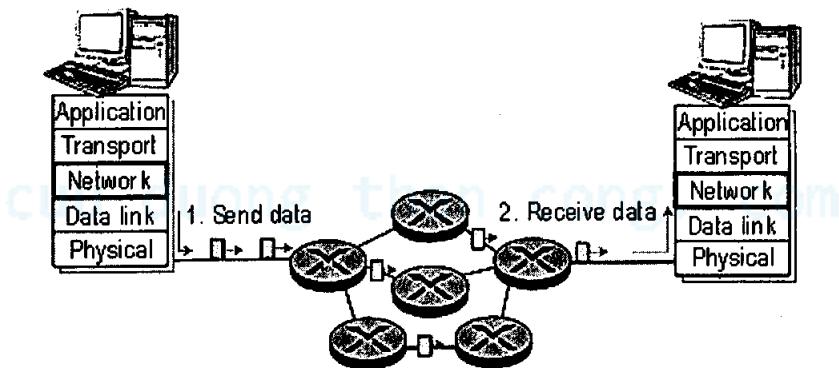
Thông điệp trao đổi giữa các thiết bị đầu/cuối yêu cầu khởi tạo, kết thúc mạch ảo, hay thông điệp trao đổi giữa các thiết bị chuyển mạch yêu cầu thiết lập VC (để cập nhật bảng định tuyến) được gọi là *thông điệp báo hiệu* (signaling message). Giao thức được sử dụng để trao đổi những thông điệp này là *giao thức báo hiệu* (signaling protocol). Quá trình thiết lập VC được minh họa trong Hình 4.2. ATM, Frame Relay và X.25 là ba kiến trúc mạng sử dụng chuyển mạch ảo.

Trong mạng chuyển mạch gói, khi muốn gửi gói tin, thiết bị đầu/cuối đặt vào gói tin địa chỉ thiết bị nhận và sau đó chuyển gói tin vào mạng. Như minh họa trong Hình 4.3, không có giai đoạn thiết lập VC nào. Những thiết bị trung chuyển trong mạng chuyển mạch gói (được gọi là *bộ định tuyến* – router trên Internet) không duy trì bất kỳ trạng thái nào về VC vì không có VC. Thiết bị trung chuyển sẽ định tuyến gói tin đến đích bằng cách xác định địa chỉ đích, tìm kiếm trên bảng định tuyến và chuyển tiếp gói tin theo hướng đến đích (giống việc chuyển thư bình thường trong

hệ thống bưu điện). Vì bảng định tuyến có thể được cập nhật liên tục, nên các gói tin được gửi từ thiết bị đầu/cuối này đến thiết bị đầu/cuối khác có thể đi theo nhiều tuyến đường khác nhau và đến đích không theo thứ tự. Mạng Internet công cộng ngày nay sử dụng dịch vụ chuyển mạch gói.



Hình 4.2. Mô hình dịch vụ chuyển mạch ảo



Hình 4.3. Mô hình chuyển mạch gói

Để chuyển gói tin của tầng giao vận, tầng mạng thường đưa ra dịch vụ chuyển mạch ảo hoặc dịch vụ chuyển mạch gói, nhưng không bao giờ cung cấp cả hai dịch vụ này. Ví dụ, dịch vụ của mạng ATM là VC trong khi mạng Internet cung cấp dịch vụ chuyển mạch gói.

Một thuật ngữ tương đương của mạch ảo (VC) và mạch gói (Datagram) tương ứng là dịch vụ *hướng nội* (connection-oriented) và dịch vụ *không hướng nội* (connectionless). Dịch vụ chuyển mạch ảo được xếp vào lớp dịch vụ hướng nội, vì phải thiết lập và kết thúc kết nối cũng như việc duy trì thông tin trạng thái của kết nối tại tất cả thiết bị chuyển mạch. Dịch vụ chuyển mạch gói được xếp vào lớp dịch vụ không hướng nội. Cả hai nhóm thuật ngữ đều có ưu điểm cũng như nhược điểm và đều được sử dụng phổ

biến trong các tài liệu về mạng. Trong cuốn sách này, sử dụng thuật ngữ "dịch vụ chuyển mạch ảo" và "dịch vụ chuyển mạch gói" cho tầng mạng và sử dụng thuật ngữ "dịch vụ hướng nối" và "dịch vụ không hướng nối" cho tầng giao vận. Điểm khác biệt này sẽ giúp người đọc hiểu được những dịch vụ khác nhau của 2 tầng này.

Bảng 4.1 tổng kết những nét chính của mô hình dịch vụ Internet và kiến trúc mạng ATM. Ở đây không đi sâu vào những khía cạnh chi tiết của mô hình dịch vụ. Kiến trúc hiện nay của Internet chỉ cung cấp duy nhất dịch vụ chuyển mạch gói, một dịch vụ theo kiểu *có gắng tối đa* (best – effort). Như minh họa trong bảng 4.1; dịch vụ này cũng giống với việc không cung cấp bất kỳ dịch vụ nào cả.

**Bảng 4.1**

Kiến trúc mạng	Mô hình dịch vụ	Đảm bảo băng thông	Đảm bảo không mất hoặc có mất	Thứ tự	Độ trễ	Kiểm soát tắc nghẽn
Internet	Có gắng tối đa	Không	Không	Không	Không	Không
ATM	CBR	Có	Có	Có	Có	Không có tắc nghẽn
ATM	VBR	Có	Có	Có	Có	Không có tắc nghẽn
ATM	ABR	Bảo đảm tốc độ nhỏ nhất	Không	Có	Có	Không có tắc nghẽn
ATM	UBR	Không	Không	Có	Có	Không

Mạng không đảm bảo thời gian gửi các gói tin giống nhau, các gói tin không được đảm bảo đến đích theo đúng thứ tự, thậm chí không đảm bảo gói tin đến được đích. Với định nghĩa này, một mạng không chuyển bất kỳ

gói tin nào đến đích cũng được phân loại theo kiểu cõ găng tối đa (ví dụ, mạng Internet công cộng hay bị tắc nghẽn).

Chuyển sang mô hình dịch vụ ATM: Ở đây, tập trung vào mô hình dịch vụ đã được Diễn đàn ATM [ATM Forum] chuẩn hóa. Kiến trúc ATM cung cấp nhiều kiểu dịch vụ khác nhau (tức là chuẩn ATM có nhiều mô hình dịch vụ). Trong phạm vi cùng một mạng, những kết nối khác nhau có thể được cung cấp những lớp dịch vụ khác nhau.

☞ **Lớp dịch vụ truyền với tốc độ cố định** – Constant bit rate (CBR):

Là mô hình dịch vụ ATM đầu tiên được chuẩn hóa, ở đây có thể thấy được vai trò của các công ty điện thoại đăng sau ATM. Dịch vụ mạng CBR là sự lựa chọn lý tưởng cho việc truyền dữ liệu đa phương tiện (ví dụ điện thoại số) theo thời gian thực, với tốc độ truyền cố định. Mục tiêu của dịch vụ CBR là làm cho kết nối mạng trông giống như một đường kết nối thực sự (bằng dây đồng hay cáp quang) giữa bên gửi và bên nhận. Trong dịch vụ CBR, các gói tin ATM (trong thuật ngữ ATM là các *té bào ATM* – ATM cell) được truyền qua mạng với một độ trễ nào đó (được gọi là cell transfer delay, CTD). Biến thiên của độ trễ ("jitter" hay cell – delay variation, CDV), tỷ lệ các cell bị mất hay đến trễ (cell – lost rate, CLR) được đảm bảo không vượt quá một giá trị ngưỡng. Tốc độ truyền tối đa của mỗi kết nối được xác định trước (pick cell rate, PCR) và bên gửi có thể gửi dữ liệu với tốc độ này. Các giá trị PCR, CTD, CDV và CLR đã được máy tính gửi và mạng ATM thỏa thuận trước trong giai đoạn thiết lập kết nối CBR.

☞ **Lớp dịch vụ truyền với tốc độ không xác định** (Unspecified bit rate – UBR). Không giống dịch vụ CBR (đảm bảo tốc độ, độ trễ, mất mát dữ liệu), UBR không đảm bảo những điều này, ngoại trừ việc gửi các cell theo đúng thứ tự. Như vậy, dịch vụ UBR giống mô hình dịch vụ cõ găng tối đa của Internet. Dịch vụ UBR không cung cấp thông tin phản hồi cho bên gửi về việc các cell có đến được đích hay không. Với mạng UBR, tính tin cậy của truyền dữ liệu được triển khai trong các giao thức ở tầng cao hơn. Dịch vụ UBR phù hợp với những ứng dụng truyền dữ liệu không cần tốc độ truyền cố định như email, newsgroup.

Nếu UBR được xem như một dịch vụ theo kiểu cố gắng tối đa, thì *dịch vụ truyền với tốc độ có sẵn* (available rate bit – ABR) có thể phân loại vào nhóm dịch vụ theo kiểu cố gắng tối đa, nhưng ưu việt hơn. Hai tính năng bổ sung quan trọng nhất của dịch vụ ABR là:

- *Tốc độ truyền cell nhỏ nhất* (MRC) được đảm bảo cho kết nối ABR. Tuy nhiên, khi tài nguyên của mạng rỗi, bên gửi có thể gửi với tốc độ cao hơn MCR.
- *Có phản hồi về tắc nghẽn từ tầng mạng*. Mạng ATM có thể cung cấp thông tin phản hồi cho bên gửi (là bit thông báo tắc nghẽn hay tốc độ gửi thấp) để bên gửi điều chỉnh tốc độ gửi.

ABR không đảm bảo một băng thông tối thiểu, nhưng cố gắng truyền dữ liệu nhanh nhất có thể. Như vậy, ABR phù hợp với các ứng dụng truyền dữ liệu yêu cầu độ trễ nhỏ (ví dụ duyệt Web).

- ☞ **Lớp dịch vụ truyền với tốc độ biến đổi** (variable bit rate – VBR): Trong dịch vụ VBR thời gian thực, tỷ lệ mất gói dữ liệu, độ trễ có thể chấp nhận được thoả thuận trước giống dịch vụ CBR. Tuy nhiên, tốc độ gửi thực sự được phép thay đổi theo các tham số do người dùng đưa vào. Điều này cho phép sử dụng tài nguyên có hiệu quả hơn, nhưng xét theo các tiêu chí về mất mát dữ liệu, độ trễ, thì VBR tương tự CBR.

Có thể tìm hiểu kỹ hơn về ATM trong ATM Forum's Traffic Management Specification 4.0 [ATM Forum 1996].

#### **4.1.2. Nguồn gốc của dịch vụ chuyển mạch gói và chuyển mạch ảo**

Lịch sử phát triển mô hình dịch vụ mạng Internet và ATM phản ánh nguồn gốc của chúng. Với khái niệm trọng tâm là mạch ảo và dịch vụ CBR là dịch vụ đầu tiên, ATM xuất phát từ mạng điện thoại truyền thống (sử dụng "mạch thực"). Các định nghĩa sau này về lớp dịch vụ UBR và ABR đã ghi nhận tầm quan trọng của việc phát triển các ứng dụng truyền dữ liệu. Với kiến trúc VC và trọng tâm hỗ trợ truyền dữ liệu theo thời gian thực cùng với sự bảo đảm về hiệu năng hệ thống (thậm chí kể cả với dịch vụ ABR), tầng mạng phức tạp hơn rất nhiều so với mô hình Internet theo kiểu cố gắng tối đa. Mạng điện thoại đặt tính phức tạp vào bên trong mạng, vì mạng này

kết nối các thiết bị đầu/cuối câm (dump device), ví dụ các máy điện thoại quay số.

Ngược lại, mạng toàn cầu Internet phát sinh từ nhu cầu kết nối các máy tính (được xem là thiết bị đầu/cuối thông minh) với nhau. Với thiết bị đầu/cuối phức tạp, kiến trúc Internet lựa chọn mô hình dịch vụ mạng đơn giản nhất có thể và đặt các chức năng phụ trợ (ví dụ như truyền dữ liệu tin cậy), cũng như các ứng dụng mạng ở tầng cao hơn triển khai trên các thiết bị đầu/cuối. Điều này ngược với mô hình mạng điện thoại và kết quả là: Mô hình dịch vụ mạng của Internet không đảm bảo bất kỳ một dịch vụ nào, do vậy có thể dễ dàng kết nối các mạng sử dụng những công nghệ kết nối rất khác nhau (ví dụ vệ tinh, Ethernet, cáp quang, sóng vô tuyến). Các công nghệ này có tốc độ và tỷ lệ mất mát dữ liệu khác nhau. Kết nối các mạng IP được trình bày chi tiết trong mục 4.4.

Như đã biết trong Chương 2, những ứng dụng như thư điện tử, Web, và thậm chí cả dịch vụ của tầng mạng như DNS được triển khai trên các máy tính (là thiết bị đầu/cuối). Các dịch vụ mới có thể nhanh chóng được sử dụng rộng rãi thông qua các giao thức tầng ứng dụng.

Tuy nhiên, có sự tranh cãi lớn trong cộng đồng Internet về việc cài tiến kiến trúc của tầng mạng để hỗ trợ các ứng dụng đa phương tiện thời gian thực.

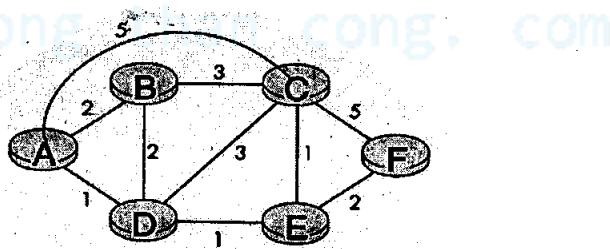
## 4.2. CÁC NGUYÊN LÝ ĐỊNH TUYẾN

Để truyền gói dữ liệu từ máy tính gửi đến máy tính nhận, tầng mạng phải quyết định đường đi hoặc các router mà gói dữ liệu phải đi qua. Dù mạng chuyển mạch gói (các gói tin khác nhau có thể đi theo các tuyến đường khác nhau), hay mạng mạch ảo (tất cả các gói tin được truyền trên cùng một tuyến đường định trước), thì tầng mạng đều phải xác định đường đi cho gói tin. Đây là công việc của các giao thức định tuyến ở tầng mạng.

"Trái tim" của giao thức định tuyến là thuật toán xác định đường đi cho gói tin – thuật toán định tuyến. Mục tiêu của thuật toán định tuyến hết sức đơn giản: Với một tập hợp router cùng với liên kết giữa các router, thuật toán định tuyến phải xác định đường đi tốt nhất từ thiết bị nguồn đến thiết bị đích. Đường đi tốt đơn giản có thể là đường đi có giá (trọng số) nhỏ nhất.

Tuy nhiên trong thực tế ta thấy, các vấn đề liên quan đến chính sách (policy) (ví dụ router X thuộc tổ chức Y không được chuyển tiếp các gói tin được tạo ra từ mạng của tổ chức Z ) có thể làm phức tạp các thuật toán của router lên nhiều.

Người ta thường sử dụng đồ thị để xây dựng các thuật toán định tuyến như minh họa trên Hình 4.3. Trong chương này, sử dụng kỹ thuật biểu diễn sơ đồ mạng dưới dạng đồ thị. Ở đây, nút (node) của đồ thị biểu diễn router – điểm quyết định việc định tuyến gói tin và những đoạn thẳng ("cung" trong lý thuyết đồ thị) nối các nút biểu diễn đường truyền vật lý thực sự giữa các router. Cung được đặc trưng bởi đại lượng "giá" (cost), là chi phí của việc gửi gói tin qua đường truyền tương ứng. Giá có thể phản ánh mức tắc nghẽn trên đường truyền (thời gian trễ trung bình), hoặc khoảng cách vật lý thực sự giữa hai router (ví dụ, đường truyền xuyên đại dương sẽ có giá cao hơn đường truyền giữa các khu vực trên đất liền). Để đơn giản, coi mỗi cung trên đồ thị có một giá và không quan tâm đến việc xác định giá bằng cách nào.



Hình 4.4. Ví dụ một mô hình mạng

Với mô hình đồ thị, vấn đề tìm kiếm tuyến đường từ nguồn đến đích có chi phí thấp nhất yêu cầu xác định chuỗi các cung sao cho:

- ☛ Cung đầu tiên trong chuỗi xuất phát từ nguồn.
- ☛ Đích của cung cuối cùng trong chuỗi là đích.
- ☛ Với mọi  $i$ , cung thứ  $i$  và  $i - 1$  cùng kết nối vào một nút.

Với đường đi có giá nhỏ nhất, tổng chi phí tất cả các cung trên tuyến đường là nhỏ nhất. Chú ý, nếu tất cả các cung có giá như nhau thì đường đi có giá nhỏ nhất cũng là đường đi ngắn nhất giữa nguồn và đích.

Ví dụ như trong Hình 4.4, đường đi có giá nhỏ nhất giữa nút A (nguồn) và nút C (đích) là đường ADEC.

Một bài tập đơn giản: Hãy trình bày cách tìm đường đi có giá thấp nhất từ A đến F. Phần lớn mọi người sẽ tìm bằng cách kiểm tra Hình 4.3, lần theo các router từ A đến F qua nhiều con đường và tự thuyết phục rằng đường đi mà mình chọn có giá nhỏ nhất so với tất cả các đường khác (Có tất cả 12 tuyến đường khác nhau nối A và F). Quá trình xác định như thế là ví dụ của thuật toán định tuyến tập trung – chạy trong bộ não của bạn với đầy đủ thông tin về mạng. Nói chung, có thể phân loại thuật toán định tuyến vào hai kiểu là toàn cục hay phân tán.

*Thuật toán định tuyến toàn cục* (global) xác định đường đi với giá thấp nhất giữa nguồn và đích, bằng cách sử dụng tất cả thông tin về tổng thể mạng. Đầu vào của thuật toán là tất cả các nút, cung và giá của các cung. Rõ ràng, router phải bằng một cách nào đó thu được các thông tin này trước khi bước vào giai đoạn tính toán thực sự. Thuật toán có thể được thực hiện tại một nơi (thuật toán định tuyến tập trung) hoặc nhiều nơi. Tuy nhiên, điểm phân biệt chính yếu là thuật toán định tuyến toàn cục phải có trước đầy đủ thông tin về đồ thị mạng. Trong thực tế, thuật toán như vậy được gọi là *thuật toán Link state*, vì thuật toán phải biết được giá của mỗi liên kết trên mạng.

Trong *thuật toán phân tán*, xác định đường đi có giá thấp nhất được thực hiện dần dần theo cách thức phân tán. Không nút nào có đầy đủ thông tin về giá của tất cả các liên kết trên mạng. Ban đầu mỗi nút chỉ biết về giá của các cung có kết nối trực tiếp với nó. Sau đó, thông qua các bước tính toán và trao đổi thông tin với các nút hàng xóm (hai nút được gọi là *hàng xóm* nếu giữa chúng có một đường kết nối vật lý trực tiếp và trong thuật ngữ đồ thị gọi là *hai đỉnh kề nhau*), nút sẽ dần dần xác định được đường đi có giá nhỏ nhất đến một tập hợp đích nào đó. Thuật toán định tuyến phân tán được gọi là *thuật toán Distance vector*, bởi vì nút không biết được đường đi cụ thể đến đích, mà chỉ biết đến nút hàng xóm trên đường đến đích và tổng giá của đường đi đến đích.

Một kiểu phân loại thuật toán thứ hai là theo tính chất tĩnh hay động. Trong thuật toán định tuyến tĩnh, tuyến đường thay đổi rất ít theo thời gian, thường là kết quả do con người tác động (ví dụ, đặt lại cấu hình cho bảng định tuyến trong router). Thuật toán định tuyến động cho phép thay đổi các tuyến đường khi lưu lượng mạng hay kiến trúc liên kết mạng bị thay đổi.

Thuật toán động có thể được chạy định kỳ, hoặc gửi thông điệp trực tiếp khi cấu trúc mạng hay giá các cung bị thay đổi. Thuật toán động có thể xử lý được khi mạng thay đổi, nhưng lại nảy sinh các vấn đề mới như định tuyến lặp (mục 4.2.2).

Thuật toán định tuyến thường được sử dụng trong Internet gồm hai kiểu chính là thuật toán Link state tĩnh và thuật toán Distance vector động. Những thuật toán này lần lượt được trình bày trong mục 4.2.1 và mục 4.2.2. Những thuật toán định tuyến khác sẽ được trình bày qua trong mục 4.2.3.

### 4.2.1. Thuật toán định tuyến link state

Trong thuật toán Link state, cấu trúc mạng và giá của tất cả các liên kết đều phải được xác định trước. Đây là dữ kiện đầu vào của thuật toán Link state. Trong thực tế, điều này được thực hiện bằng cách, mỗi nút sẽ gửi thông báo quảng bá về định danh của mình và giá các cung liên kết trực tiếp đến nó tới tất cả các router khác trên mạng. Việc quảng bá rộng rãi trạng thái liên kết có thể được thực hiện ngay khi nút không biết về đầy đủ các nút khác trên mạng. Ban đầu nút chỉ biết được thông tin về các hàng xóm của mình, cũng như giá các cung đến các hàng xóm. Nhưng sau đó nó sẽ xác định được topo của phần còn lại của mạng khi nhận những thông báo quảng bá từ các nút khác (Trong Chương 5, ta thấy các router hàng xóm trao đổi thông tin với nhau như thế nào). Kết quả của việc quảng bá trạng thái liên kết là tất cả các nút có thể đầy đủ thông tin về tổng thể mạng. Sau đó, mỗi nút đều có thể chạy thuật toán Link state và xác định đường đi có giá thấp nhất tới mọi nút.

Thuật toán Link state được trình bày ở đây là thuật toán Dijkstra (đặt theo tên của người phát minh ra). Thuật toán Dijkstra xác định đường đi có giá thấp nhất từ một nút nguồn (không mất tổng quát, giả sử là A) đến tất cả các nút khác trên mạng. Thuật toán Dijkstra có nhiều bước và sau  $k$  bước sẽ xác định được đường đi có giá thấp nhất tới  $k$  nút đích. Để hiểu rõ thuật toán, ta định nghĩa một số ký hiệu sau:

$c(i, j)$ : Giá liên kết từ nút i đến nút j. Nếu nút i và nút j không có đường kết nối trực tiếp thì  $c(i, j) = \infty$ . Để đơn giản, chúng ta coi  $c(i, j) = c(j, i)$ .

$D(v)$ : Giá hiện tại thấp nhất của tuyến đường đi từ nút nguồn đến nút v.

$P(v)$ : Nút phía trước nút v (hàng xóm của v) trên tuyến đường hiện có giá thấp nhất từ nguồn tới nút v.

$N$ : Tập hợp của các nút đã xác định được đường đi ngắn nhất.

Thuật toán Link state gồm có bước khởi tạo cho vòng lặp. Số các bước bằng tổng số nút trên mạng. Khi kết thúc, thuật toán sẽ xác định được đường đi ngắn nhất từ nút nguồn đến tất cả các nút khác trên mạng.

#### Thuật toán Link state (LS):

Khởi tạo:

$$N = \{A\}$$

for (tất cả các nút v)

if v kề A

$$\text{thì } D(V) = c(A, V)$$

else  $D(v) = \infty$

Repeat

Tìm w không ở trong N có  $D(w)$  nhỏ nhất

Bổ sung w vào N

Cập nhật  $D(v)$  cho tất cả v kề với w và không nằm trong N:

$$D(v) = \min ((D(v), D(w) + c(w, v)))$$

/\* Giá mới đến v khác giá cũ đến v hoặc biết được giá đường đi ngắn nhất đến w cộng với giá từ w đến v \*/

Until tất cả các nút nằm trong N

**Bảng 4.2.** Bảng trạng thái các bước cho mạng minh họa trên Hình 4.4

Bước	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	$\infty$	$\infty$
1	AD	2, A	4, D		2, D	$\infty$
2	ADE	2, A	3, E			4, E
3	ADEB		3, E			4, E
4	ADEBC					4, E
5	ADEBCF					

Xét đồ thị mạng trong Hình 4.4 và tính đường đi có giá thấp nhất từ A đến tất cả các nút khác. Bảng 4.2 cho thấy các kết quả tính toán của thuật

toán, mỗi dòng trong bảng ứng với trạng thái của thuật toán sau khi kết thúc một bước. Sau đây ta phân tích một số bước:

- ☞ **Bước khởi tạo (bước 0):** Giá hiện tại thấp nhất của đường đi từ A đến các nút hàng xóm B, C và D tương ứng là 2, 5 và 1. Chú ý, giá đến C được đặt là 5 (ngay sau đây sẽ thấy đây không phải là đường đi tốt nhất), đây là giá của đường nối trực tiếp từ A đến C. Giá đến E và F được đặt là vô cùng vì giữa A và E, F không có đường kết nối trực tiếp.
- ☞ **Bước thứ nhất:** Tìm kiếm trên những nút chưa được đưa vào tập N và xác định nút có giá đến thấp nhất. Đó là nút D với giá là 1 và do đó D được bổ sung vào N. Dòng 12 của thuật toán LS được thực hiện để cập nhật  $D(v)$  cho tất cả các nút v, kết quả nhận được được trình bày trong dòng thứ 2 (bước 1) trong Bảng 4.2. Giá của đường đi đến B không đổi. Giá đường đi đến C (nhận giá trị 5 trong bước khởi tạo trước) qua D có giá trị nhỏ hơn là 4. Đây là đường tốt hơn được chọn và nút phía trước của C trên đường đi ngắn nhất từ A sẽ là D. Tương tự vậy, giá đường đi đến E (qua D) được tính là 2 và bảng được cập nhật tương ứng.
- ☞ **Bước thứ hai:** Đường đi đến nút B và E đều có giá thấp nhất nên E được bổ sung vào tập N (bây giờ N chứa A, D và E). Giá đến các nút chưa nằm trong N (gồm B, C và F) được cập nhật trong dòng 12 của thuật toán LS, kết quả là dòng 3 của Bảng 4.2.

☞ ...

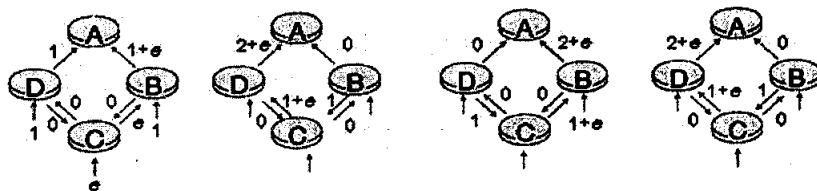
Khi thuật toán LS kết thúc, với mỗi nút xác định được nút ngay trước nó trên tuyến đường có giá thấp nhất xuất phát từ nguồn. Với mỗi nút phía trước, ta lại có nút phía trước nữa,... Cuối cùng, xác định được toàn bộ đường đi từ nguồn đến tất cả các nút đích.

Độ phức tạp tính toán của thuật toán này bằng bao nhiêu? Với  $n$  nút (không kể nút nguồn), để tìm đường đi có giá thấp nhất từ nguồn đến tất cả các đích, khối lượng tính toán là bao nhiêu trong trường hợp xấu nhất? Trong vòng lặp đầu tiên, cần kiểm tra qua tất cả  $n$  nút để xác định nút  $w$  có giá nhỏ nhất không nằm trong  $n$ ; trong vòng lặp thứ hai, cần kiểm tra  $n - 1$  nút để xác định giá thấp nhất; trong vòng lặp thứ ba cần kiểm tra  $n - 2$  nút,...

Tổng số các nút cần phải kiểm tra qua tất cả các bước là  $n(n + 1)/2$  và theo đó có thể nói rằng, thuật toán Link state có độ phức tạp là  $O(n^2)$ . (Thuật toán này có thể được cải tiến bằng cách sử dụng cấu trúc dữ liệu HEAP, độ phức tạp chỉ còn theo hàm logarit của  $n$ ).

Trước khi kết thúc trình bày về thuật toán LS, xét ví dụ minh họa một cấu hình mạng giống như trên Hình 4.5. Giá của mỗi liên kết (cung) bằng tải hiện tại trên nó (như thế giá sẽ là độ trễ mà gói tin phải chịu). Trong trường hợp này, giá không có tính chất đối xứng, nghĩa là  $c(A, B)$  chỉ bằng  $c(B, A)$  nếu tải trên cả hai hướng A và B là như nhau. Giả sử hai nút B và D gửi một đơn vị dữ liệu, nút C gửi khối lượng dữ liệu là  $e$  tới A. Định tuyến ban đầu được minh họa trên Hình 4.5(a), giá của mỗi cung ứng với tải trên cung đó.

Trong bước tiếp theo của thuật toán LS, nút C (với giá liên kết cơ bản đã được xác định trong Hình 4.5a) nhận thấy đường đi đến A theo chiều kim đồng hồ có giá là 1, trong khi theo chiều ngược lại có giá là  $1+e$ . Do đó, đường đi đến A có giá thấp nhất của C bây giờ là theo chiều kim đồng hồ. Tương tự, B nhận thấy đường đi đến A có giá thấp nhất mới cũng theo chiều kim đồng hồ, kết quả được trình bày trong Hình 4.5b. Trong bước tiếp theo, nút B, C và D nhận thấy đường đi đến A ngược chiều kim đồng hồ có giá là 0 và tất cả các nút định lại tuyến đường theo ngược chiều kim đồng hồ. Trong bước tiếp theo B, C và D lại thay đổi việc định tuyến theo chiều kim đồng hồ.



a) Hình trạng ban đầu

b) B, C phát hiện tuyến đường tới A tốt hơn là đi theo chiều kim đồng hồ

c) B, C, D phát hiện đi theo chiều ngược kim đồng hồ tới A tốt hơn

d) B, C, D phát hiện tuyến đường tới A tốt hơn là đi theo chiều kim đồng hồ

Hình 4.5. Xung đột định tuyến

Làm sao có thể ngăn ngừa sự dao động như trên (điều này luôn xuất hiện với những thuật toán chọn độ tắc nghẽn hoặc thời gian trễ làm giá cho đường truyền). Một giải pháp được đưa ra là, định giá cho đường truyền không phụ thuộc vào tải trên đường đi – một giải pháp khó có khả năng

chấp nhận, vì mục tiêu của định tuyến là tránh những đường truyền hay tắc nghẽn (có độ trễ cao). Một giải pháp khác là, làm thế nào để tất cả các router không chạy thuật toán LS tại cùng một thời điểm? Giải pháp này dường như hợp lý hơn, vì hy vọng rằng, thậm chí nếu các router có chạy thuật toán LS với cùng chu kỳ, thì thuật toán sẽ đưa ra những kết quả khác nhau tại mỗi nút.

#### 4.2.2. Thuật toán Distance vector

Nếu thuật toán LS sử dụng thông tin về toàn bộ trạng thái mạng, thuật toán Distance vector (DV) là thuật toán lặp, không đồng bộ và phân tán. Thuật toán được xem là phân tán, vì mỗi nút nhận thông tin từ những nút hàng xóm có đường kết nối trực tiếp, thực hiện các bước tính toán và phân tán kết quả tính toán tới tất cả các nút hàng xóm. Tính lặp được thể hiện ở chỗ quá trình này được thực hiện liên tục cho đến khi không còn thông tin được trao đổi giữa các cặp hàng xóm (Chúng ta sẽ thấy đây là thuật toán tự kết thúc – nó tự dừng chứ không cần một tín hiệu kết thúc). Thuật toán không đòi hỏi các nút không được hoạt động trong khi trao đổi với những nút khác – đây chính là đặc điểm không đồng bộ. Chúng ta sẽ thấy thuật toán với các đặc điểm dị bộ, lặp, tự kết thúc và phân tán thú vị và đáng quan tâm hơn so với thuật toán tập trung.

Cấu trúc dữ liệu chính trong thuật toán DV là *bảng khoảng cách* (distance table) được đặt ở mỗi nút. Trong bảng khoảng cách, mỗi hàng ứng với một nút đích trên mạng và mỗi cột ứng với một nút hàng xóm có đường kết nối trực tiếp đến. Giả sử nút X muốn định tuyến đến đích Y qua nút hàng xóm Z. Trong bảng khoảng cách của nút X,  $D^X(Y, Z)$  là tổng của giá đường liên kết trực tiếp giữa X và Z,  $c(X, Z)$  với giá đường đi bé nhất từ Z đến Y. Đó là:

$$D^X(Y, Z) = c(X, Z) + \min_w\{D^Z(Y, w)\}$$

Biểu thức  $\min_w$  trong đẳng thức trên được lấy trên tất cả các hàng xóm của Z (kể cả X).

Đẳng thức giúp ta hình dung ý tưởng của thuật toán DV – mỗi nút phải biết được giá nhỏ nhất của đường đi từ tất cả các hàng xóm của nó đến bất kỳ đích nào. Và khi giá nhỏ nhất đến đích nào đó thay đổi, nút phải báo cho tất cả các hàng xóm biết.

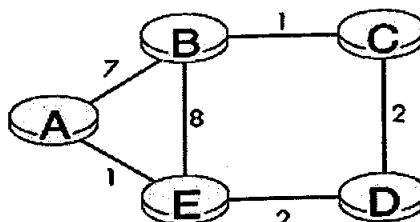
Trước khi trình bày thuật toán DV, xét kiến trúc mạng và bảng khoảng cách của nút E trên Hình 4.6. Đây là bảng khoảng cách của nút E sau khi thuật toán DV hội tụ. Nhìn vào dòng đầu tiên với đích đến là A, rõ ràng giá đường kết nối trực tiếp đến A từ E phải là 1, do đó  $D^E(A, A) = 1$ .

Bây giờ chú ý đến giá trị của  $D^E(A, D)$  – giá đường đi từ E đến A qua D. Trường này trong bảng khoảng cách là giá của đường đi từ E đến D cộng với giá đường đi nhỏ nhất từ D đến E. Chú ý rằng, giá thấp nhất của đường đi từ D đến A là 3 – đường đi qua E. Do vậy, giá thấp nhất từ E đến A qua D là 5.

Tương tự, trường trong bảng khoảng cách với đường đi qua B là  $D^E(A, B) = 14$ . Chú ý rằng, tại sao ở đây không phải là 15?

Các vòng tròn trong bảng khoảng cách là giá nhỏ nhất của đường đi đến các đích (ứng với các hàng). Cột ứng với vòng tròn xác định nút tiếp theo trên đường đi đến đích có giá thấp nhất. Từ đó có thể dễ dàng xây dựng bảng định tuyến cho mỗi nút (với một đích cụ thể nào đó, cần gửi gói tin ra theo đường ra nào).

Trong khi trình bày bảng vector cho nút E ở trên, ta vẫn có được một cái nhìn toàn cục, biết được giá của tất cả các liên kết trên mạng. Thuật toán DV phân tán được trình bày ở đây không sử dụng đến những thông tin tổng thể như vậy.



		cost to destination via		
		A	B	D
D^E()		A	B	D
	A	①	14	5
destination	B	7	8	⑤
	C	6	9	④
	D	4	11	②

Hình 4.6

### Thuật toán Distance vector (DV)

Tại mỗi nút X:

- 1: Khởi tạo:
- 2: for (tất cả các nút kề với v)
  - 3:  $D^X(*, v) = \infty$  /\* dấu "\*" là để chỉ cho tất cả các hàng\*/
  - 4:  $D^X(v, v) = c(X, v)$

```

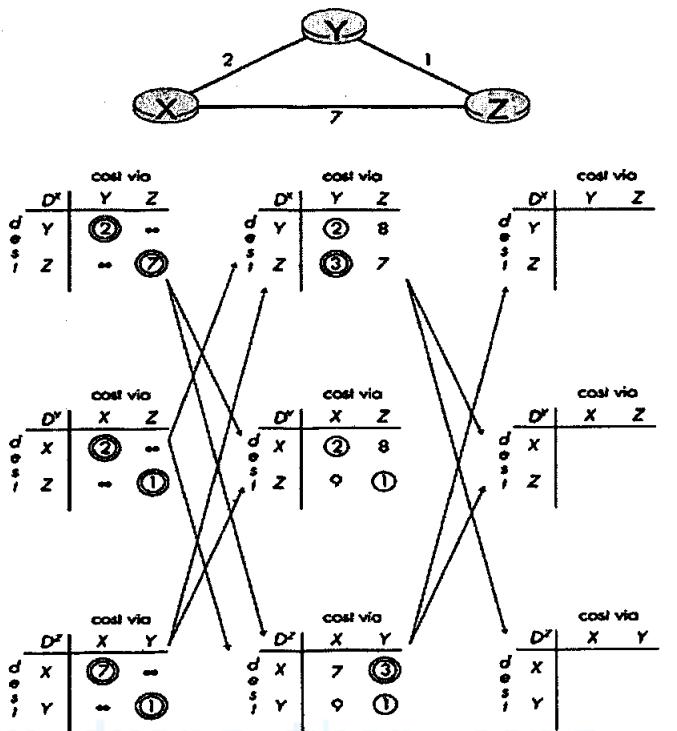
5:   for (tất cả các đích Y)
6:       gửi  $\min_w D(Y, w)$  đến mỗi hàng xóm
7:       /* w chạy trong tập các hàng xóm của X*/
8:   Lặp
9:   Đợi cho đến khi (thay giá liên kết đến hàng xóm V thay đổi hoặc nhận
   được sự cập nhật của hàng xóm V)
10:  if ( $c(X, V)$  thay đổi một lượng d)
11:      /* thay đổi giá của tất cả các đường đi đến đích qua v bằng d*/
12:      /* chú ý: d có thể dương hoặc âm*/
13:      for (tất cả các đích y:  $D^X(y, V) = D^X(y, V) + d$ 
14: else if (nhận được cập nhật từ V đến Y)
15:     /* đường đi ngắn nhất từ V đến nút nào đó Y thay đổi*/
16:     /* V đã gửi giá trị mới của  $\min_w D^V(Y, w)$ */
17:     /* gọi giá trị mới nhận được là "newval"*/
18:     for (đích y):  $D^X(Y, V) = c(X, V) + newval$ 
19:     if (chúng ta có  $\min_w D^X(Y, w)$  mới cho đích Y nào đó)
20:         gửi giá trị  $\min_w D^X(Y, w)$  mới đến tất cả các hàng xóm.

```

Thật vậy, mỗi nút sẽ chỉ biết thông tin về giá đường liên kết tới nút hàng xóm cũng như thông tin nó nhận được từ những hàng xóm này. Thuật toán Distance vector còn được gọi là *thuật toán Bellman – Ford*. Nó được áp dụng trong nhiều giao thức định tuyến trong thực tế, bao gồm: Internet BGP, ISO IDRP, Novell IPX và mạng ARPAnet.

Bước mấu chốt là ở nút cập nhật bảng khoảng cách khi nhận được sự thay đổi về giá của liên kết đến hàng xóm, hoặc nhận được thông tin cập nhật từ hàng xóm. Một bước quan trọng khác là nút gửi cập nhật đến tất cả các hàng xóm nếu đường đi có giá nhỏ nhất đến đích nào đó bị thay đổi.

Hình 4.7 minh họa hoạt động của thuật toán DV cho mạng đơn giản gồm 3 nút. Hoạt động của thuật toán được thực hiện một cách đồng bộ: Tất cả các nút đồng thời nhận được thông điệp từ hàng xóm của chúng, tính toán bảng khoảng cách mới và báo cho các hàng xóm về sự thay đổi giá đường đi ngắn nhất. Sau khi xem xét ví dụ này, có thể nghĩ rằng, thuật toán hoạt động đúng trong chế độ không đồng bộ – việc tính toán và cập nhật sự thay đổi của mỗi nút có thể diễn ra tại bất kỳ thời điểm nào.



Hình 4.7. Ví dụ về thuật toán Distance vector

Các ô khoanh tròn trong Hình 4.7 ứng với giá nhỏ nhất đến đích nào đó nằm trong hàng tương ứng. Khoanh hai vòng tròn biểu diễn giá nhỏ nhất mới được xác định. Khi đó các thông tin cập nhật sẽ được gửi đến các nút hàng xóm – ứng với mũi tên giữa các cột trong Hình 4.7.

Cột ngoài cùng bên trái trong Hình 4.7 là các bảng khoảng cách của nút X, Y và Z sau bước khởi tạo.

Bây giờ hãy xem làm thế nào để nút X tính lại bảng khoảng cách (cột giữa của Hình 4.7) sau khi nhận được thông tin cập nhật từ nút Y và Z? Khi nhận được cập nhật từ Y và Z, X thực hiện dòng 8 của thuật toán DV:

$$\begin{aligned}
 D^X(Y, Z) &= c(X, Z) + \min_w D^Z(Y, w) \\
 &= 7 + 1 \\
 &= 8
 \end{aligned}$$

$$\begin{aligned}
 D^X(Z, Y) &= c(X, Y) + \min_w D^Y(Z, w) \\
 &= 2 + 1 \\
 &= 3
 \end{aligned}$$

X biết giá trị  $\min_w D^Z(Y, w)$  và  $\min_w D^Y(Z, w)$  vì nút Z và Y gửi những giá trị này đến X (và X cũng nhận được theo dòng 9 của thuật toán DV). Việc tính lại bằng khoảng cách của Y và Z ở cột giữa trong Hình 4.7 được thực hiện tương tự.

Giá trị  $D^X(Z, Y) = 3$ , có nghĩa là giá nhỏ nhất từ X đến Z giảm từ 7 xuống 3. Do đó, X gửi cập nhật đến Y và Z để thông báo cho chúng giá thấp nhất mới đến Z. Chú ý, X không cần cập nhật cho Y, Z về giá của nó đến Y, vì giá trị này không bị thay đổi. Khi Y tính lại bằng khoảng cách không phát hiện ra thay đổi, do đó Y không gửi cập nhật đến X và Z.

Quá trình nhận cập nhật từ hàng xóm, tính lại bằng khoảng cách và cập nhật các thay đổi đến hàng xóm được thực hiện cho đến khi không còn thông điệp nào được trao đổi. Trong trường hợp này, vì không có thông tin cập nhật được gửi nên các nút không phải tính lại bằng khoảng cách và thuật toán ở trạng thái không hoạt động – tất cả các nút ở trạng thái đợi của thuật toán DV. Thuật toán DV sẽ ở trong trạng thái không hoạt động cho đến khi giá của một liên kết nào đó thay đổi.

### 4.3. ĐỊNH TUYẾN PHÂN CẤP

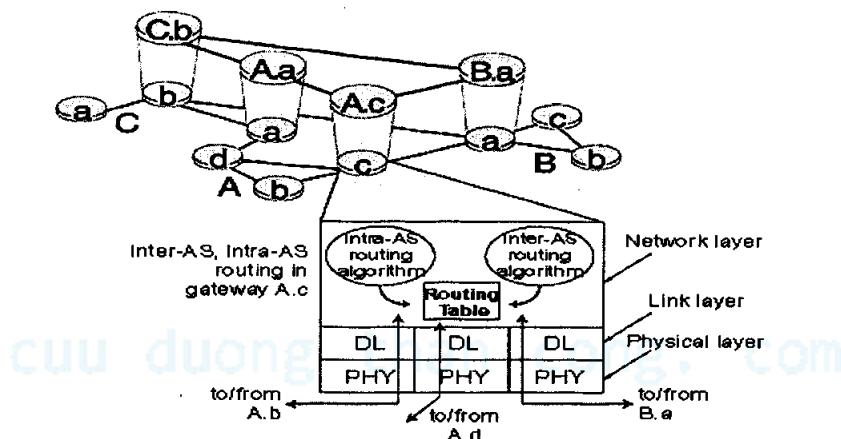
Trong các phần trước, ta thấy mạng là tập hợp các router liên kết với nhau. Tất cả các router sẽ giống nhau nếu như đều sử dụng cùng một thuật toán định tuyến để xác định đường đi trên toàn bộ hệ thống mạng.

Trong thực tế, mô hình mạng như vậy và việc xem các router giống hệt nhau cùng thực hiện một thuật toán định tuyến quá đơn giản với hai lý do sau:

- ☞ **Phạm vi (scale):** Khi số lượng các router lớn, khối lượng thông tin phải tính toán, lưu trữ và trao đổi giữa các bảng chứa thông tin định tuyến trên mỗi router (ví dụ các cập nhật về đường đi ngắn nhất) cũng trở nên cực lớn. Mạng Internet ngày nay bao gồm hàng triệu router liên kết với nhau và hơn 50 triệu máy tính. Lưu trữ thông tin về tất cả các máy tính cũng như các router đòi hỏi một lượng bộ nhớ khổng lồ. Các thông tin trao đổi cập nhật giữa các router sẽ "ngốn" toàn bộ băng thông của đường truyền. Thuật toán Distance vector trên hàng triệu router chắc chắn sẽ không bao giờ hội tụ. Do

đó, nảy sinh nhu cầu làm giảm độ phức tạp trong việc xác định đường đi trên một mạng lớn như Internet.

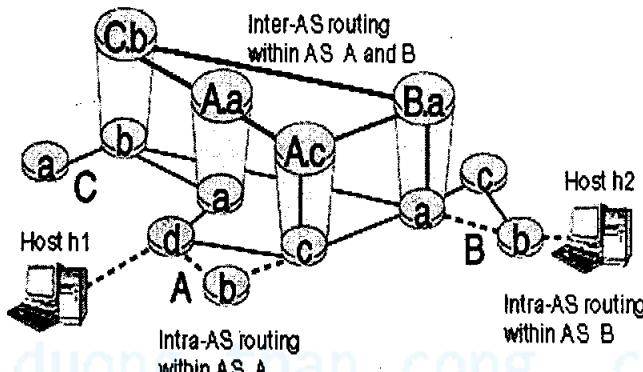
- ☞ **Quản trị** (Administrative autonomy): Mặc dù các nhà thiết kế thường bỏ qua yêu cầu của các tổ chức, chẳng hạn khả năng lựa chọn thuật toán định tuyến hay che dấu cấu trúc mạng bên trong của tổ chức với bên ngoài, nhưng trên thực tế đây là những vấn đề quan trọng. Lý tưởng mà nói, một tổ chức phải giữ khả năng quản trị và kiểm soát mạng máy tính của mình, nhưng vẫn có khả năng kết nối với các mạng bên ngoài.



Hình 4.8. Ví dụ miền tự quản

Cả hai vấn đề trên đều có thể giải quyết bằng cách nhóm các router thành các vùng hay *Miền tự quản* (Autonomous System – AS). Các router trong cùng AS sử dụng cùng một thuật toán định tuyến (ví dụ như thuật toán LS hay DV) và biết đầy đủ về nhau (giống trường hợp đã trình bày ở trên). Thuật toán định tuyến chạy trong mỗi AS được gọi là *giao thức định tuyến nội miền* (intraautonomous system routing protocol). Dĩ nhiên, cần phải kết nối các AS với nhau và vì thế một số router trong AS có thêm nhiệm vụ định tuyến gói tin ra ngoài AS (đích nằm trong một AS khác). Các router định tuyến gói tin ra phía ngoài như vậy được gọi là *giao thức định tuyến liên miền* (gateway router). Để định tuyến gói tin đi giữa các AS (có thể phải đi qua nhiều AS trên toàn bộ tuyến đường), các gateway router phải biết cách xác định đường đi giữa các AS. Thuật toán định tuyến được sử dụng tại các gateway router là *interautonomous system routing protocol*.

Nói tóm lại, vấn đề phạm vi và quản trị được giải quyết bằng cách sắp xếp router vào các miền tự quản (AS). Tất cả router trong một AS sử dụng cùng một thuật toán định tuyến. Các gateway router trên mỗi AS sử dụng thuật toán interautonomous system routing để định tuyến giữa các AS. Vấn đề phạm vi được giải quyết ở chỗ mỗi router chỉ cần biết về các router khác trong cùng AS và gateway router của AS đó (chứ không cần phải biết tất cả các router). Vấn đề quản trị được giải quyết vì tổ chức có thể lựa chọn sử dụng bất kỳ thuật toán định tuyến intra-AS nào, miễn là thuật toán inter-AS mà nó sử dụng có khả năng liên kết với các AS khác.



Hình 4.9. Ví dụ định tuyến phân cấp

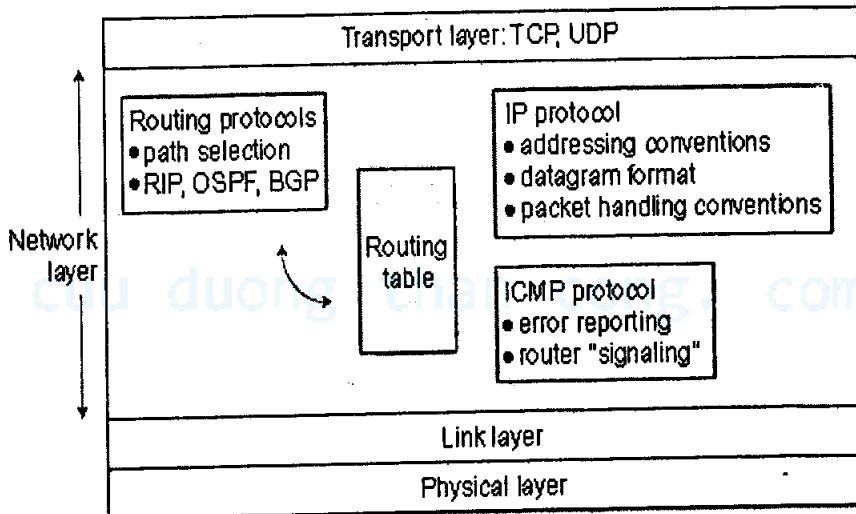
Trên Hình 4.9 có ba AS là A, B và C. Miền A có 4 router là A.a, A.b, A.c và A.d, sử dụng cùng một thuật toán intra-AS của miền A. Bốn router này đều có đầy đủ các thông tin về nhau cũng như các liên kết trong miền A. Tương tự miền B có 3 và miền C có 2 router. Các thuật toán intra-AS trong các miền A, B, C không nhất thiết giống nhau. Các gateway router là A.a, A.c, B.a và C.b. Ngoài các thuật toán intra-AS để trao đổi với các router trong miền, bốn gateway router này phải sử dụng thuật toán inter-AS để định tuyến giữa các AS. Về mặt topo, chúng sử dụng giao thức inter-AS, được minh họa bằng các đường kết nối màu đen đậm (hình vẽ) ở mức cao hơn. Các đường kết nối này có thể là đường kết nối vật lý thực sự (giữa A.c và B.a), có thể là đường ảo (ví dụ, giữa A.c và A.a). Trên hình vẽ chúng ta cũng thấy A.c sử dụng cả intra-AS để định tuyến với A.b, A.d và inter-AS để định tuyến với B.a

Giả sử máy tính h1 nối với router A.d cần gửi gói tin tới máy tính h2 trong AS B như trên Hình 4.9. Bảng định tuyến tại A.d cho biết router A.c

chịu trách nhiệm gửi gói tin ra bên ngoài AS. Gói tin từ A.d tới router đầu tiên A.c sử dụng giao thức định tuyến intra-AS của A. Có một điểm cực kỳ quan trọng cần chú ý là router A.d không cần biết gì về cấu trúc nội tại trong miền B và C và cũng như topo giữa ba miền A, B và C. Router A.c nhận gói tin, xác định đích của gói tin đó nằm ngoài miền A (miền B), bảng định tuyến của inter-AS sẽ xác định rằng, để gửi tới miền B thì phải chuyển tới B.a. Khi gói tin tới B.a, giao thức inter-AS xác định rằng, gói tin này tới máy tính nào đó trong miền B và chuyển cho giao thức intra-AS của B. Cuối cùng, router B.a chuyển gói tin đó tới máy tính đích h2 sử dụng giao thức intra-AS của B. Trong Hình 4.9, các giao thức intra-AS của A và B là các đường kẻ đứt, giao thức inter-AS giữa các miền là đường kẻ đậm.

#### 4.4. INTERNET PROTOCOL

Trong các phần trước đã trình bày các nguyên lý chung của tầng mạng, mà chưa nói đến bất kỳ một kiến trúc mạng cụ thể nào. Chúng ta cũng đã nói tới các mô hình dịch vụ khác nhau của tầng mạng, các thuật toán định tuyến xác định đường đi giữa thiết bị gửi và thiết bị nhận, phân cấp hệ thống mạng để giải quyết vấn đề phạm vi. Trong phần này, sẽ trình bày tới tầng mạng của Internet – mà một phần trong đó được xem là tầng IP. Chúng ta sẽ thấy IP chỉ là một phần (dù là phần quan trọng) trong kiến trúc tầng mạng của Internet (Hình 4.10).



Hình 4.10. Vị trí và các thành phần của tầng mạng

Như đã nói trong mục 4.1, tầng mạng của Internet sử dụng dịch vụ chuyển mạch gói (datagram), chứ không phải dịch vụ chuyển mạch ảo (VC). Tại máy gửi, khi nhận được một segment từ tầng giao vận, tầng mạng đặt segment trong gói dữ liệu IP (IP datagram) với các trường địa chỉ gửi, địa chỉ nhận,... và gửi datagram này tới router đầu tiên trên đường tới đích. Điều này tương tự như một người sau khi viết thư, bô lá thư vào phong bì, ghi địa chỉ người nhận và thả phong bì thư vào hộp thư. Tầng mạng của Internet cũng giống như hệ thống bưu điện, đều không có bất kỳ một sự liên hệ trước nào với bên nhận trước khi chuyển "bưu kiện" (datagram hay lá thư) tới phía nhận. Hơn nữa, như đã nói trong mục 4.1, tầng mạng của Internet và bưu điện sẽ chỉ cung cấp một dịch vụ kiểu "cố gắng tối đa", nghĩa là không có bất kỳ đảm bảo nào về việc gói tin có đến đích hay không, đến trong một khoảng thời gian bao lâu, hay đến theo đúng thứ tự gửi.

Trong Hình 4.10, tầng mạng trong kiểu mạng chuyển mạch gói giống như mạng Internet có ba thành phần chính:

- ☞ Thành phần thứ nhất là giao thức mạng: Xác định địa chỉ tầng mạng; ý nghĩa các trường trong datagram (là gói dữ liệu – PDU của tầng mạng); hành động của router và thiết bị đầu/cuối khi nhận được datagram. Giao thức mạng trong Internet gọi là *giao thức Internet* hay phổ biến hơn là *giao thức IP*. Hiện nay có hai phiên bản giao thức IP được sử dụng. Các mục 4.4.1 đến 4.4.4 nói về IPv4 (được sử dụng chủ yếu hiện nay) [RFC 701] và mục 4.7 nói tới IPv6 (được đưa ra để thay thế IPv4).
- ☞ Thành phần thứ hai của tầng mạng là bộ phận xác định đường đi: Xác định tuyến đường của datagram trên đường đi tới đích. Mục 4.2 nói về thuật toán tạo ra các bảng định tuyến như vậy. Các thành phần của bộ định tuyến sẽ được trình bày trong mục 4.5.
- ☞ Thành phần cuối cùng của tầng mạng là chức năng báo lỗi và khả năng trả lời một số yêu cầu về thông tin của tầng mạng. Giao thức báo lỗi của Internet – ICMP được nói đến trong mục 4.4.5.

#### 4.4.1. Địa chỉ IPv4

Trước tiên sẽ nói về địa chỉ IPv4. Mặc dù vấn đề địa chỉ tương đối đơn giản, song mối quan hệ giữa địa chỉ và giao thức tầng mạng lại quan trọng.

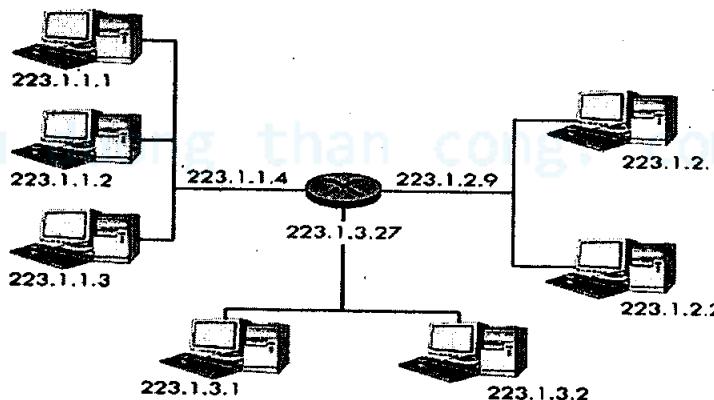
Trước khi thảo luận về địa chỉ IP, cần xem xét máy tính và router nối vào mạng như thế nào? Máy tính thường có một đường kết nối duy nhất vào hệ thống mạng, mà qua đó thực thể IP trong máy tính sẽ sử dụng khi gửi datagram. Nằm giữa máy tính và đường kết nối vật lý là một *giao diện ghép nối* (interface). Ngược lại, router khác hoàn toàn máy tính. Công việc của router là chuyển một datagram từ kết nối (incoming link) này tới kết nối khác (outgoing link). Router có thể có nhiều kết nối và bộ phận nằm giữa router với kết nối cũng được gọi là *giao diện*. Như vậy, router có nhiều giao diện, mỗi giao diện ứng với một kết nối. Vì tất cả các máy tính và router đều phải có khả năng gửi và nhận IP datagram, nên mỗi giao diện phải có một địa chỉ IP. Do đó, địa chỉ IP ứng với giao diện chứ không phải với máy tính hay router.

Địa chỉ IP có độ dài 32 bit (4 byte) và do đó không gian địa chỉ có  $2^{32}$  địa chỉ. Địa chỉ IP được viết theo ký pháp *dấu chấm thập phân* (dotted decimal notation), vì mỗi byte của địa chỉ được viết dưới dạng thập phân và phân cách với các byte khác bằng ký tự chấm (.). Xét địa chỉ IP 193.32.216.9, 193 là số thập phân ứng với nhóm 8 bit đầu của địa chỉ, 32 là số thập phân ứng với nhóm 8 bit thứ hai của địa chỉ,... Bởi vậy, địa chỉ 193.32.216.9 đổi ra hệ nhị phân sẽ là:

110000001 00100000 11011000 00001001.

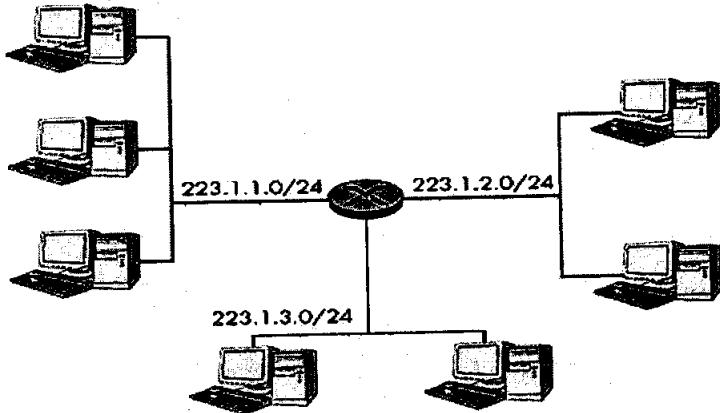
Mỗi giao diện ghép nối của máy tính hay router trên mạng toàn cầu Internet phải có một địa chỉ IP được xác định duy nhất. Địa chỉ này không thể chọn một cách tùy ý, mà phụ thuộc vào "mạng" mà nó kết nối vào. Trong ngữ cảnh này, thuật ngữ "mạng" không có ý là một cấu trúc tổng thể gồm các máy tính, router và các liên kết giữa chúng. Hiện tại, thuật ngữ này được sử dụng với ý nghĩa cụ thể hơn, có quan hệ chặt chẽ với địa chỉ IP. Hình 4.11 minh họa một router có 3 giao diện được sử dụng để kết nối 7 máy tính. Quan sát địa chỉ IP của giao diện ứng với mỗi máy tính và router, giao diện của 3 máy tính ở phần bên trái trong Hình 4.11 và router nối với chúng đều có địa chỉ IP là 223.1.1.xxx. – nghĩa là 24 bit đầu của địa chỉ

IP giống nhau. Chúng cũng được kết nối với nhau bằng một đường kết nối vật lý duy nhất (trong trường hợp này là môi trường quảng bá sử dụng cáp Ethernet) mà không cần qua bất kỳ router trung gian nào. Giao diện của những máy tính này và giao diện phía bên trái của router tạo nên *mạng IP* (IP network) hay đơn giản là *mạng*. 24 bit địa chỉ đầu giống nhau là phần mạng trong cấu trúc địa chỉ IP, 8 bit còn lại là phần máy tính (host) của địa chỉ IP. Chính mạng này cũng có một địa chỉ là 223.1.1.0/24, trong đó ký hiệu /24 là *mặt nạ mạng* (network mask) với ý nghĩa 24 bit đầu tiên của địa chỉ 32 bit xác định địa chỉ mạng. Những bit này cũng được xem là *tiền tố mạng* (network prefix). Mạng 223.1.1.0/24 gồm giao diện của 3 máy tính (223.1.1.1; 223.1.1.2; 223.1.1.3) và một giao diện của router (223.1.1.4). Bất kỳ máy tính nào nối với mạng 223.1.1.0/24 đều phải có địa chỉ dưới dạng 223.1.1.xxx. Trên Hình 4.11 còn có hai mạng khác là 223.1.2.0/24 và 223.1.3.0/24. Hình 4.12 minh họa 3 mạng IP trong Hình 4.11.

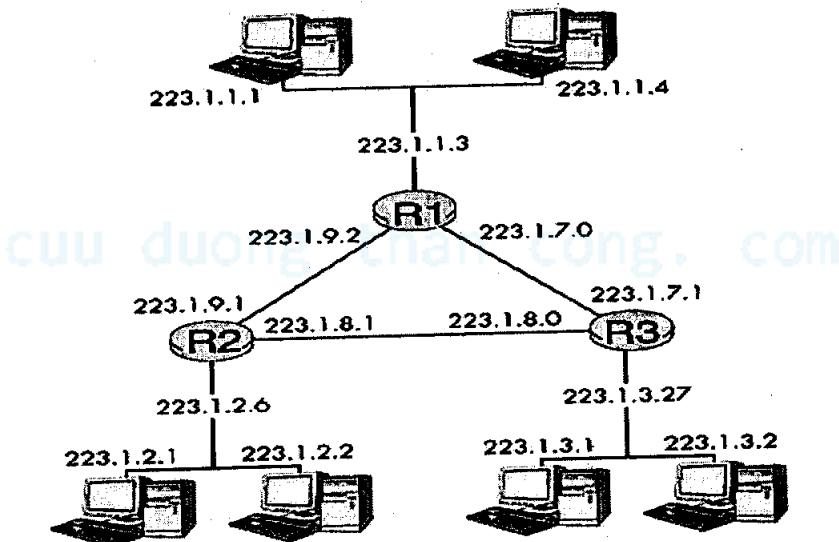


Hình 4.11

Định nghĩa IP về "mạng" không chỉ với phân đoạn mạng Ethernet nối nhiều máy tính với một router. Trên Hình 4.13, ba router đối mặt nối với nhau qua các đường liên kết điểm nối điểm (point – to – point). Mỗi router có ba giao diện, hai giao diện kết nối tới hai router kia và một giao diện dành cho kết nối quảng bá với các máy tính. Có bao nhiêu mạng IP ở đây? Ba mạng 223.1.1.0/24, 223.1.2.0/24 và 223.1.3.0/24 tương tự các mạng nói tới trong Hình 4.11. Nhưng chú ý, có thêm 3 mạng nữa trong Hình 4.13 là mạng 223.1.9.0/24 cho hai giao diện nối router R1 và R2; mạng 223.1.8.0/24 cho hai giao diện nối router R2 và R3 và mạng 223.1.7.0/24 ứng với hai giao diện nối router R3 và R1.



Hình 4.12

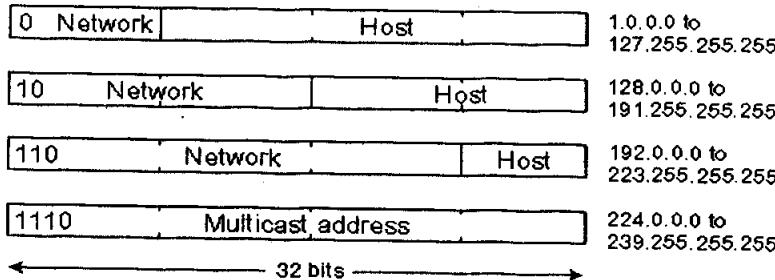


Hình 4.13. Có 6 mạng IP

Với một hệ thống liên mạng gồm nhiều router và máy tính, ta có thể sử dụng một công thức để xác định các mạng trong hệ thống. Khi loại bỏ tất cả giao diện của các máy tính và router. Khi đó sẽ tạo ra các mạng cô lập, mỗi mạng cô lập đó được coi là một mạng IP. Áp dụng cách thức này trong ví dụ trên Hình 4.13, ta có 6 mạng IP tách biệt. Mạng toàn cầu Internet gồm hàng triệu hệ thống mạng như vậy. Ký pháp mạng và địa chỉ mạng có vai trò then chốt trong kiến trúc định tuyến Internet. Sau khi đã có định nghĩa về mạng, tiếp tục trình bày chi tiết về địa chỉ IP. Kiến trúc địa chỉ Internet đầu tiên đưa ra 4 lớp địa chỉ minh họa trên Hình 4.14. Lớp địa chỉ thứ năm, bắt đầu

bằng 11110 được dự trữ cho việc sử dụng sau này. Với lớp địa chỉ A, 8 bit đầu là địa chỉ mạng và 24 bit cuối là địa chỉ cho giao diện (máy tính) trong mạng. Như thế, có  $2^7$  mạng lớp A (bit đầu tiên trong 8 bit địa chỉ mạng luôn nhận giá trị 0), mỗi mạng lớp A lại có thể có  $2^{24}$  giao diện. Lớp B có  $2^{14}$  mạng, mỗi mạng có  $2^{16}$  giao diện. Lớp địa chỉ C dùng 24 bit làm địa chỉ mạng và chỉ có 8 bit làm địa chỉ máy. Lớp D dự trữ làm địa chỉ multicast.

Bốn lớp địa chỉ trình bày trên hiện tại không còn được áp dụng trong kiến trúc địa chỉ IP nữa. Điều kiện phần mạng của địa chỉ IP có độ dài là một, hai hoặc ba byte không hợp lý khi số lượng các tổ chức với mạng cỡ nhỏ hay trung bình ngày càng tăng. Mạng lớp C (/24) chỉ có thể có  $2^8 - 2 = 254$  máy tính (2 trong số  $2^8$  địa chỉ được dự trữ cho một số mục đích khác) – một số lượng quá nhỏ với nhiều tổ chức. Tuy nhiên, mạng lớp B (/16) có thể có tới 65.634 máy tính lại là quá lớn. Một tổ chức với 2.000 máy tính phải sử dụng địa chỉ lớp B (/16). Với kiểu gán địa chỉ như vậy, không gian địa chỉ lớp B sẽ nhanh chóng bị cạn kiệt, nhưng không gian địa chỉ lại không được sử dụng hiệu quả. Ví dụ, tổ chức sử dụng địa chỉ lớp B cho 2.000 máy tính thì sẽ có khoảng 63.000 địa chỉ còn lại bị lãng phí trong khi đáng ra có thể phân phối cho các tổ chức khác.



**Hình 4.14.** Cấu trúc địa chỉ mạng với 4 lớp A, B, C và D

Năm 1993, IETF chuẩn hoá *Định tuyến liên miền không phân lớp* (Classless interdomain routing – CIDR – phát âm là "cider") [RFC 1519]. Với cơ chế không phân lớp, phần mạng của địa chỉ IP có thể có độ dài tùy ý, không nhất thiết phải là 8, 16 hay 24 bit. Khuôn dạng của một địa chỉ không phân lớp sẽ là a.b.c.d/x, trong đó x là số lượng bit dùng để làm địa chỉ mạng. Trong ví dụ trước, tổ chức cần không gian địa chỉ cho 2.000 máy tính chỉ cần một không gian 2.048 địa chỉ dưới dạng a.b.c.d/21, cho phép 63.000 địa chỉ còn lại được phân phối cho các tổ chức khác. Trong trường hợp này, 21

bit đầu tiên xác định địa chỉ mạng của tổ chức và tất cả địa chỉ IP của máy tính trong tổ chức đều phải có phần địa chỉ mạng giống nhau. 11 bit còn lại xác định cụ thể máy tính nào trong tổ chức. Trên thực tế, tổ chức có thể tiếp tục chia 11 bit đó trong thủ tục *tạo mạng con* (subnetting) [RFC 950] để tạo ra các mạng con bên trong mạng a.b.c.d/21.

### **Gán địa chỉ cho mỗi giao diện:**

Sau khi biết được địa chỉ IP, bạn có thể hỏi: Bằng cách nào máy tính nhận được địa chỉ IP? Địa chỉ IP có hai phần là phần mạng và phần máy tính. Phần máy tính của địa chỉ có thể được cấp phát theo nhiều cách như sau:

- ☞ **Cấu hình bằng tay:** Địa chỉ IP được người quản trị hệ thống cấu hình vào máy tính (thường trong file cấu hình).
- ☞ **Giao thức cấu hình địa chỉ động** (Dynamic host configuration protocol – DHCP) [RFC 2131]: DHCP là phiên bản mở rộng của giao thức BOOTP [RFC 1542]. Với DHCP, khi máy DHCP phục vụ (server) trong mạng (mạng cục bộ chẳng hạn) nhận yêu cầu DHCP từ một máy khách, nó sẽ phân phối (gán) một địa chỉ IP cho máy khách yêu cầu. DHCP được sử dụng rộng rãi trong mạng cục bộ hay truy cập Internet từ nhà (kết nối modem với ISP).

Để có địa chỉ mạng, người quản trị mạng phải liên lạc với ISP của mình. ISP đã được phân phối một không gian địa chỉ tương đối lớn và sẽ cấp một phần trong không gian địa chỉ này cho tổ chức thuê bao. Ví dụ, ISP được phân phối không gian địa chỉ 200.23.16.0/20. Đến lượt mình ISP chia không gian địa chỉ này thành 8 không gian địa chỉ nhỏ hơn và cấp phát 8 không gian này cho các tổ chức thuê bao như dưới đây (để tiện theo dõi, phần mạng trong địa chỉ được gạch chân):

#### **Không gian địa chỉ của ISP:**

11001000 00010111 00010000 00000000 200.2.3.16.0/20

#### **Tổ chức 0:**

11001000 00010111 00010000 00000000 200.23.16.0/23

#### **Tổ chức 1:**

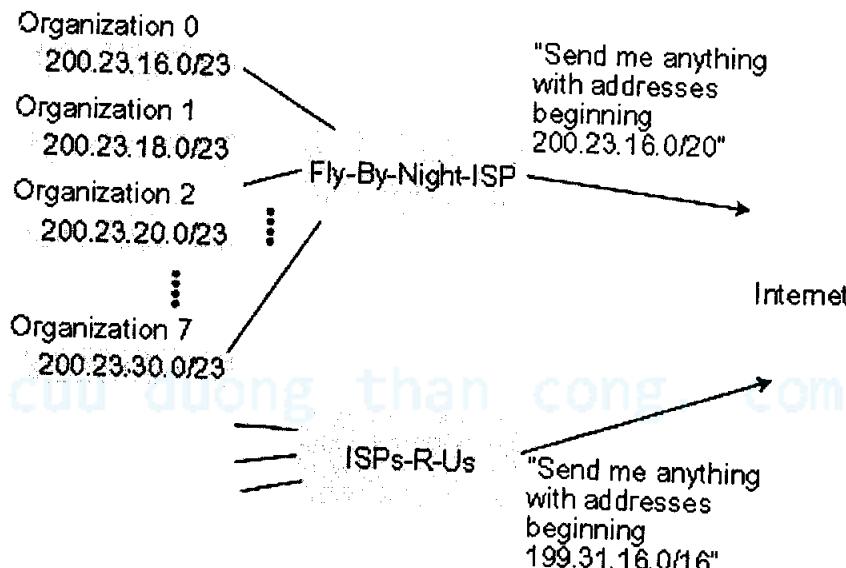
11001000 00010111 00010010 00000000 200.23.18.0/23

## Tổ chức 7:

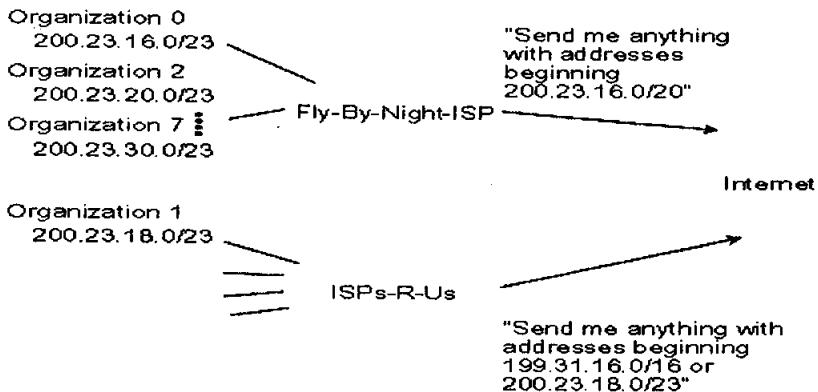
11001000 00010111 00010100 0000000 200.23.30.0/23

Vậy làm thế nào để ISP nhận được không gian địa chỉ? Địa chỉ IP được tổ chức ICANN (The Internet Corporation for Assigned Names and Numbers) [ICANN 2000] quản lý theo các nguyên tắc chỉ đạo ghi trong RFC 2050. Vai trò của tổ chức phi lợi nhuận ICANN không chỉ cấp phát địa chỉ IP mà còn quản trị các root DNS server. Nó chịu trách nhiệm đặt tên miền cũng như giải quyết các tranh chấp về tên miền. Hiện nay, việc phân phối địa chỉ IP được cơ quan đăng ký Internet cấp vùng quản lý. Giữa năm 2000, có 3 cơ quan đăng ký như vậy là American Registry for Internet Number (ARIN) cho châu Mỹ và một số phần của châu Phi; Reseaux IP Europeans (RIPE) cho châu Âu và một số nước chung quanh; Asia Pacific Network Information Center (APNIC) cho khu vực châu Á.

Các máy tính có khả năng di động (mobile) có thể thay đổi mạng theo cách động (khi di chuyển) hoặc tĩnh (theo thời gian). Khi định tuyến, phải xác định mạng trước và sau đó mới tới máy tính trong mạng, nên địa chỉ IP của máy tính di động sẽ phải thay đổi khi máy tính thay đổi mạng. Các kỹ thuật xử lý trường hợp này đang được phát triển [RFC 2002, RFC 2131].



Hình 4.15. Ví dụ 2 ISP



Hình 4.16

#### 4.4.2. Chuyển datagram từ nguồn tới đích: vấn đề địa chỉ và định tuyến

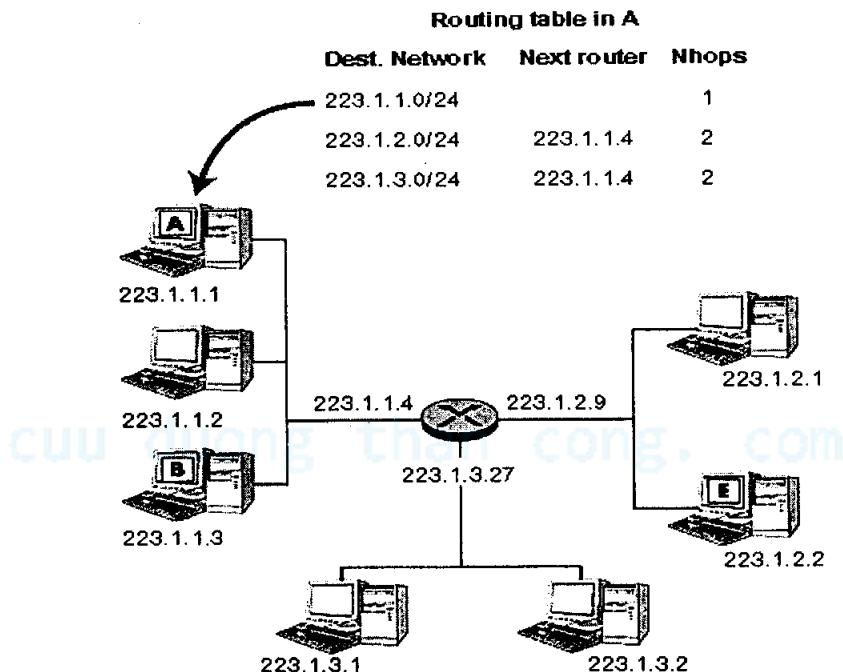
Giờ đây khi đã có định nghĩa về giao diện, mạng cũng như các hiểu biết cơ bản về địa chỉ IP, chúng ta xem máy tính và các router chuyển một gói tin IP (IP datagram) từ nơi gửi đến nơi nhận như thế nào? Để đơn giản, coi khuôn dạng chung của gói tin IP giống như trong Hình 4.17. Mỗi địa chỉ IP datagram có trường địa chỉ gửi và trường địa chỉ nhận. Máy tính gửi sẽ điền vào trường địa chỉ gửi 32 bit địa chỉ IP của mình và điền vào trường địa chỉ nhận 32 bit địa chỉ IP của máy tính nhận (giống như các trường FROM và TO trên phong bì thư). Trường dữ liệu của datagram thường là gói TCP hoặc UDP segment. Khuôn dạng gói dữ liệu IP được trình bày chi tiết trong các phần sau.

Misc fields	Source IP Address	Destination IP address	Data
-------------	-------------------	------------------------	------

Hình 4.17. Khuôn dạng chung gói dữ liệu tầng mạng

Sau khi máy tính gửi tạo ra IP datagram, tầng mạng làm thế nào để gửi datagram từ máy tính nguồn tới máy tính đích? Câu trả lời phụ thuộc vào việc máy tính nguồn và máy tính đích có nằm trong cùng một mạng hay không (thuật ngữ "mạng" sử dụng giống như trong mục 4.4.1). Đầu tiên, giả sử máy tính A muốn gửi IP datagram tới máy tính B nằm trong cùng mạng 223.1.1.0/24 với A. Điều này được thực hiện như sau: Trước hết thực thẻ IP trong máy A dò trong bảng định tuyến cục bộ của mình (Hình 4.18) và tìm

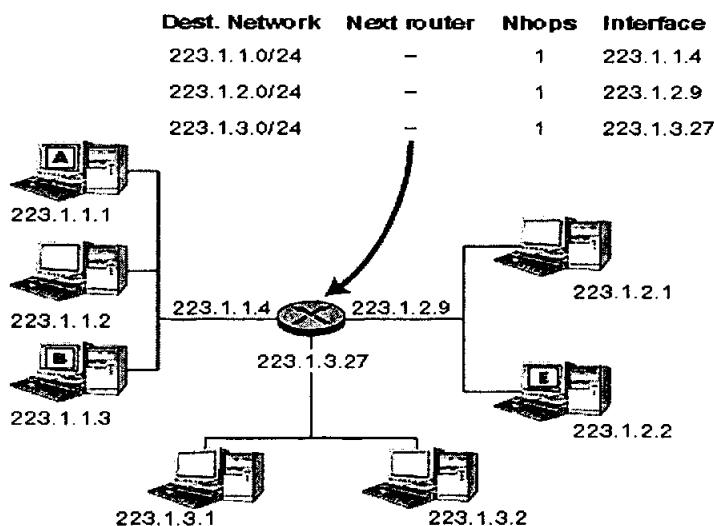
thấy hàng 223.1.1.0/24 trùng với các bit đầu (địa chỉ mạng) trong địa chỉ IP của máy B. Bảng định tuyến chỉ ra rằng, số lượng các thiết bị trung gian để đến mạng 223.1.1.0 là 1, nghĩa là B nằm trong cùng một mạng với A. Do đó, máy A có thể gửi trực tiếp đến B mà không cần qua các router trung gian. Sau đó, máy A chuyển gói dữ liệu IP cho tầng liên kết dữ liệu để chuyển gói dữ liệu đó tới B (công việc của tầng liên kết dữ liệu chuyển dữ liệu giữa hai thiết bị nối trực tiếp với nhau được trình bày trong Chương 5).



**Hình 4.18.** Ví dụ định tuyến từ A đến B

Xét tiếp trường hợp máy A gửi gói dữ liệu tới máy E nằm trên mạng khác. A dò trên bảng định tuyến của mình và thấy 223.1.1.0/24 có địa chỉ mạng trùng với phần mạng trong địa chỉ IP của E. Vì số lượng các thiết bị trung gian là 2, nên máy A biết máy đích nằm trên mạng khác và do đó sẽ phải chuyển qua router trung gian. Ngoài ra, bảng định tuyến tại A cũng cho biết để gửi tới E, đầu tiên A phải gửi gói dữ liệu tới địa chỉ IP 223.1.1.4 – là địa chỉ IP của giao diện router nằm trên cùng một mạng với A. Thực tế IP trong máy A chuyển gói dữ liệu xuống tầng liên kết dữ liệu và yêu cầu chuyển tới địa chỉ IP 223.1.1.4. Một chú ý rất quan trọng ở đây là, mặc dù gói dữ liệu IP được gửi tới giao diện của router (qua tầng liên kết dữ liệu),

địa chỉ đích trong gói dữ liệu vẫn là địa chỉ đích cuối cùng (địa chỉ E) chứ không phải là địa chỉ router trung gian.



Hình 4.19. Chuyển dữ liệu từ A đến E

Khi gói dữ liệu tới router, thì trách nhiệm chính của router là chuyển gói dữ liệu hướng tới đích cuối cùng. Như minh họa trong Hình 4.19, router tìm trên bảng định tuyến và thấy hàng 223.1.2.0/24 trùng với phần mạng địa chỉ IP của E. Như vậy, gói dữ liệu phải được chuyển đến giao diện của router có địa chỉ IP là 223.1.2.9. Vì số lượng các thiết bị trung gian giữa router và đích là 1, nên router biết rằng đích (E) nằm trên cùng một mạng với giao diện ứng với địa chỉ 223.1.2.9. Vì thế, router chuyển gói dữ liệu tới giao diện (cổng) này và sau đó gói dữ liệu được chuyển trực tiếp tới E.

Chú ý rằng, trong Hình 4.19, các hàng trong cột "next router" là rỗng, vì mỗi mạng 223.1.1.0/24, 223.1.2.0/24 và 223.1.3.0/24 được kết nối trực tiếp với router. Trong trường hợp này, chúng không cần phải đi qua các router trung gian để đến đích. Tuy nhiên, nếu máy A và máy E cách nhau 2 router, thì trong bảng định tuyến của router đầu tiên trên tuyến đường từ A tới E, dòng tương ứng với đích E sẽ chỉ ra phải qua 2 chặng nữa mới tới được đích, và phải chỉ ra địa chỉ IP của router thứ hai trên tuyến đường. Router đầu tiên sẽ chuyển gói dữ liệu tới router thứ hai nhờ vào giao thức của tầng liên kết dữ liệu giữa hai router. Kế đó, router thứ hai sẽ chuyển gói dữ liệu tới máy đích nhờ giao thức tầng liên kết dữ liệu giữa router thứ hai và máy đích.

Định tuyến cho gói dữ liệu trên mạng Internet tương tự việc người lái xe hỏi đường cảnh sát giao thông tại mỗi "bùng binh". Trên đường di chuyển từ nguồn tới đích, gói dữ liệu sẽ đi qua nhiều router. Tại mỗi router, gói tin dừng lại và "hỏi" router đi đường nào để tới đích. Trừ khi router trên cùng mạng với máy tính đích, về cơ bản, bảng định tuyến trong router sẽ nói với gói dữ liệu: "Tôi không biết chính xác đi đến đích như thế nào, nhưng tôi biết hướng tới nó là đi theo đường này". Sau đó gói dữ liệu sẽ được gửi đi trên đường kết nối này để đến một router khác, và lại tiếp tục hỏi đường.

Rõ ràng, bảng định tuyến của router đóng vai trò then chốt trong việc định tuyến gói tin qua mạng Internet. Nhưng làm thế nào để cấu hình và bảo trì bảng định tuyến trong một hệ thống liên mạng cực lớn với nhiều tuyến đường giữa đích và nguồn (như trong mạng Internet)? Rõ ràng, bảng định tuyến phải được cấu hình sao cho các gói dữ liệu được đi theo tuyến đường tốt nhất từ nguồn tới đích. Chính các thuật toán định tuyến đã nói tới trong mục 4.2 thực hiện công việc định tuyến và cập nhật bảng định tuyến. Thuật toán định tuyến của Internet được trình bày trong mục 4.5.

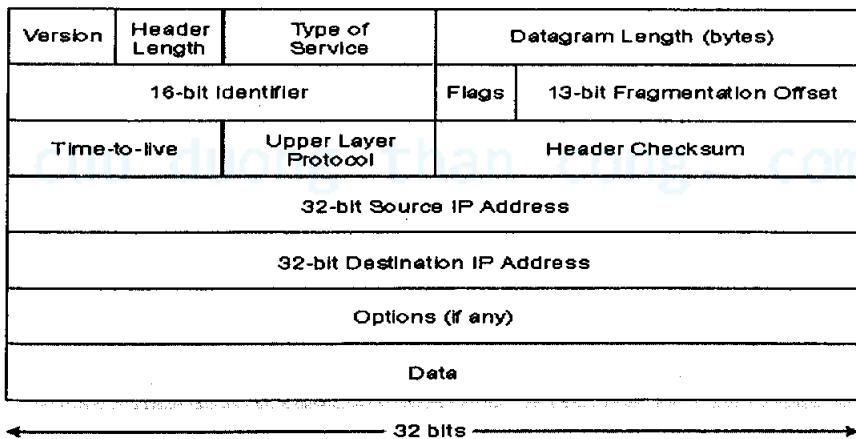
#### 4.4.3. Khuôn dạng gói dữ liệu IP

Sau đây là các trường trong gói dữ liệu IPv4:

- ☛ **Phiên bản (version):** Trường 4 bit này xác định phiên bản giao thức IP của gói dữ liệu. Qua trường phiên bản, router mới xác định được ý nghĩa các trường còn lại của gói dữ liệu IP. Các phiên bản IP khác nhau sử dụng các khuôn dạng dữ liệu khác nhau. Khuôn dạng gói dữ liệu IP hiện tại IPv4 được minh họa trong Hình 4.20. Khuôn dạng gói dữ liệu IPv6 được trình bày trong mục 4.7.
- ☛ **Độ dài tiêu đề (Header length):** Gói dữ liệu IPv4 có thể có nhiều trường mang tính lựa chọn (không bắt buộc phải có). 4 bit này được dùng để xác định vị trí bắt đầu của dữ liệu thực sự trong gói dữ liệu IP. Tuy nhiên, phần lớn gói dữ liệu IP không chứa các trường lựa chọn nên tiêu đề của gói dữ liệu thường cố định là 20 byte.
- ☛ **Kiểu dịch vụ (Type of service – TOS):** Trường kiểu dịch vụ (TOS) giúp phân biệt các kiểu khác nhau của gói dữ liệu IP, để từ đó có thể xử lý theo những cách khác nhau. Ví dụ, khi mạng quá tải, cần

phân biệt được gói dữ liệu chứa thông tin kiểm soát mạng (ICMP) với gói dữ liệu thực sự (thông điệp HTML), hay giữa datagram chứa dữ liệu thời gian thực (ứng dụng điện thoại qua Internet) với datagram không chứa dữ liệu thời gian thực (ứng dụng FTP). Gần đây, Cisco (công ty chiếm thị phần router lớn nhất) đã sử dụng 3 bit đầu tiên của trường TOS để định nghĩa các mức dịch vụ khác nhau mà router có thể cung cấp. Các mức dịch vụ cụ thể được người quản trị router thiết lập theo những tiêu chí của tổ chức.

- ☞ **Độ dài gói dữ liệu (datagram length):** Đây là tổng độ dài của gói dữ liệu IP (cả phần tiêu đề lẫn phần dữ liệu) tính theo byte. Độ dài trường này là 16 bit, nên về lý thuyết kích thước tối đa của gói dữ liệu IP là 65.535 byte. Tuy nhiên, hiếm khi kích thước gói dữ liệu vượt quá 1.500 byte và thường giới hạn là 576 byte.



**Hình 4.20.** Khuôn dạng gói dữ liệu IP

- ☞ **Định danh, cờ và vị trí phân đoạn (Identifier, Flags, Fragmentation Offset):** Ba trường này được sử dụng khi phân mảnh gói IP (fragmentation), một chủ đề sẽ xem xét chi tiết dưới đây. Chú ý, phiên bản mới của IP (IPv6) không cho phép phân mảnh gói dữ liệu tại các router.
- ☞ **Thời gian tồn tại (Time – to – live – TTL):** Trường thời gian tồn tại (TTL) được sử dụng để bảo đảm gói dữ liệu không thể lưu chuyển mãi mãi trong mạng (nguyên nhân có thể do định tuyến lặp, nên các gói tin truyền lồng vòng theo một chu trình). Trường này bị

giảm đi một (-1) mỗi lần gói tin đi qua router. Nếu trường TTL bằng 0, router sẽ loại bỏ gói tin.

- ☞ **Giao thức (Protocol):** Trường này chỉ được sử dụng khi gói dữ liệu IP đến được máy tính đích. Giá trị của trường này xác định giao thức tầng giao vận ở máy tính đích sẽ nhận được phần dữ liệu trong gói dữ liệu IP. Ví dụ, giá trị 6 có ý nghĩa phần dữ liệu cần chuyển tới thực thể TCP, giá trị 17 có ý nghĩa phần dữ liệu phải chuyển đến thực thể UDP. RFC 1700 liệt kê các giá trị này. Chú ý rằng, vai trò của trường giao thức trong gói dữ liệu IP tương tự vai trò trường số hiệu cổng trong segment của tầng giao vận. Trường giao thức được xem là điểm nối giữa tầng mạng và tầng giao vận cũng như trường số hiệu cổng là điểm nối giữa tầng giao vận với ứng dụng cụ thể. Trong Chương 5 sẽ thấy trong frame của tầng liên kết dữ liệu cũng có một trường đặc biệt để nối với tầng mạng.
- ☞ **Checksum của tiêu đề (Header checksum):** Trường checksum trong tiêu đề giúp router phát hiện lỗi trong tiêu đề gói dữ liệu IP được gửi đến. Giá trị checksum được tính bằng cách xem phần tiêu đề là một chuỗi các từ 2 byte, cộng các từ này lại và sau đó lấy bù một. Như đã trình bày trong mục 3.3, số bù một của tổng này được gọi là *Internet checksum*. Router tính lại Internet checksum cho mỗi gói dữ liệu IP nhận được và có thể phát hiện ra lỗi nếu như giá trị checksum tính lại khác giá trị checksum trong gói dữ liệu. Router thường loại bỏ những gói dữ liệu bị lỗi. Chú ý rằng, router phải tính lại checksum, trường TTL và có thể một số trường khác. RFC 1071 trình bày phương thức tính checksum nhanh. Một vấn đề thường được đặt ra ở đây là: Tại sao TCP/IP thực hiện kiểm tra lỗi ở cả tầng giao vận lẫn tầng mạng? Có nhiều nguyên nhân của việc này. Thứ nhất, các router không bắt buộc phải kiểm tra lỗi, vì vậy tầng giao vận không thể dựa vào tầng mạng để làm việc này. Thứ hai, TCP/UDP và IP không nhất thiết nằm trong cùng một nhóm giao thức. TCP có thể chạy trên giao thức khác (ví dụ ATM) và IP có thể chuyển dữ liệu không phải là TCP hay UDP segment.
- ☞ **Địa chỉ IP nguồn và đích:** Những trường này là 32 bit địa chỉ IP của máy tính gửi và máy tính nhận. Tầm quan trọng của địa chỉ

đích là rõ ràng. Trong mục 3.2 thấy rằng, địa chỉ IP máy gửi (cùng với số hiệu cổng nguồn và đích) được máy nhận sử dụng để hướng dữ liệu ứng dụng tới socket phù hợp.

- ☞ **Lựa chọn (Option):** Các trường này cho phép mở rộng tiêu đề IP. Phần lựa chọn trong tiêu đề ít khi được sử dụng. Sự tồn tại của phần lựa chọn trong tiêu đề làm phức tạp việc xử lý các gói tin, vì tiêu đề của gói dữ liệu có phần lựa chọn không có độ dài cố định, do đó không xác định được vị trí bắt đầu của dữ liệu thực sự. Như vậy, thời gian xử lý gói dữ liệu IP tại mỗi router có thể khác nhau. Đây là nhược điểm của các mạng hiệu suất cao. Vì thế, IPv6 sẽ loại bỏ các trường lựa chọn.
- ☞ **Dữ liệu (payload):** Là trường quan trọng nhất, thông thường trường dữ liệu của gói IP là gói dữ liệu của tầng giao vận (TCP hay UDP segment). Tuy nhiên, trường dữ liệu có thể là các kiểu dữ liệu khác, ví dụ thông điệp ICMP (sẽ được trình bày trong mục 4.4.5).

#### 4.4.4. Phân mảnh (Fragmentation) và hợp nhát (Reassembly) gói tin IP

Trong Chương 5 sẽ thấy, không phải tất cả các giao thức của tầng liên kết dữ liệu đều có khả năng truyền các gói tin (packet) có cùng độ lớn. Một vài giao thức có khả năng gửi những gói tin lớn, trong khi một vài giao thức chỉ có thể gửi những gói tin nhỏ. Ví dụ, gói tin của mạng Ethernet có độ lớn lên tới 1.500 byte, trong khi gói tin trên những liên kết ở mạng diện rộng có độ lớn không được vượt quá 576 byte. Số lượng dữ liệu tối đa của gói tin trên một đường truyền vật lý được định nghĩa là MTU (maximum transfer unit). Gói dữ liệu IP được đặt trong gói dữ liệu của tầng liên kết dữ liệu giữa hai router kế tiếp nhau trên đường truyền. Vì thế, giá trị MTU của giao thức ở tầng liên kết dữ liệu giới hạn độ dài gói tin IP. Giới hạn kích thước gói tin IP không phải là vấn đề lớn. Vấn đề ở đây là kết nối giữa các router dọc theo tuyến đường từ nơi gửi đến nơi nhận có thể sử dụng các giao thức liên kết dữ liệu khác nhau có giá trị MTU khác nhau.

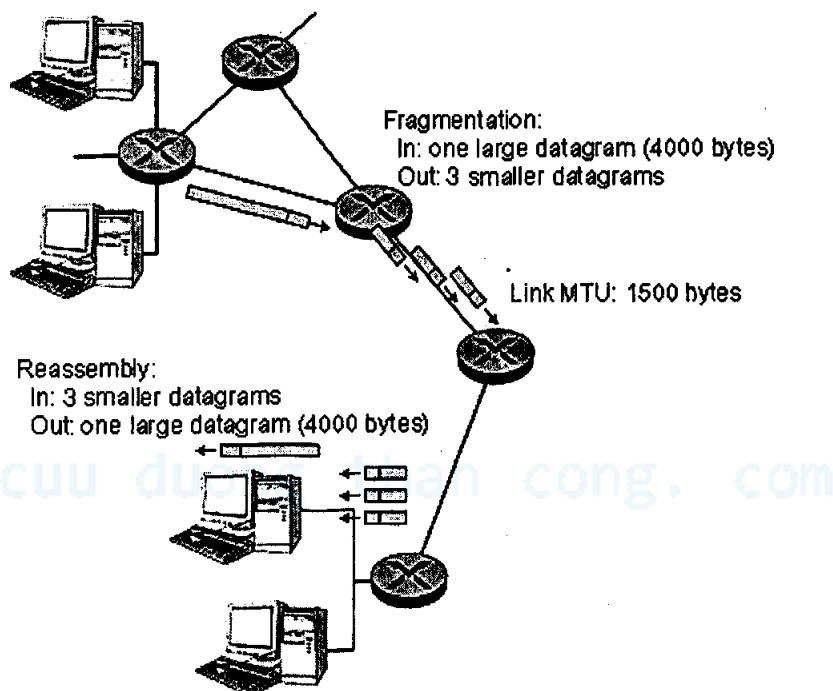
Để hiểu vấn đề rõ hơn, xét router với nhiều kết nối, mỗi kết nối có một giao thức liên kết dữ liệu khác nhau với giá trị MTU khác nhau. Giả sử khi nhận được gói dữ liệu đến từ kết nối nào đó, căn cứ vào địa chỉ đích, router

kiểm tra bảng định tuyến để xác định cần gửi gói tin đi ra theo kết nối nào. Tuy nhiên, đường kết nối ra ngoài này có giá trị MTU nhỏ hơn độ dài gói dữ liệu IP. Làm thế nào router có thể đặt gói tin IP lớn trong gói tin của tầng liên kết dữ liệu có kích thước nhỏ hơn? Giải pháp cho vấn đề này là phân mảnh (fragmentation) dữ liệu trong gói dữ liệu IP thành nhiều gói dữ liệu IP nhỏ hơn, và sau đó gửi những gói dữ liệu nhỏ hơn này trên đường kết nối. Mỗi gói dữ liệu IP nhỏ này được coi là một mảnh (fragment).

Các mảnh tách rời này cần được ráp lại trước khi chuyển lên tầng giao vận tại máy tính nhận. Rõ ràng là, cả TCP và UDP đều mong muốn nhận được một segment đầy đủ, không bị phân mảnh từ tầng mạng. Việc hợp nhất các gói dữ liệu tại các router sẽ làm giao thức phức tạp lên nhiều và làm giảm hiệu suất của router. Giữ vững nguyên tắc thiết kế tầng mạng đơn giản nhất có thể, IPv4 quyết định việc hợp nhất các mảnh dữ liệu được thực hiện tại thiết bị đầu/cuối chứ không phải là tại router.

Khi nhận được nhiều gói dữ liệu đến từ cùng một nguồn, máy tính cần phải xác định liệu đây là những gói dữ liệu độc lập hay là các mảnh của một gói dữ liệu lớn ban đầu. Trong trường hợp thứ hai, nó phải tiếp tục xác định liệu đã nhận được đầy đủ các mảnh chưa và làm sao để ráp các mảnh này lại theo trật tự ban đầu để tạo ra gói dữ liệu nguyên thuỷ. Máy tính đích sẽ sử dụng các trường identification, flag và fragmentation để thực hiện công việc hợp nhất này. Khi tạo ra một gói dữ liệu IP, ngoài địa chỉ gửi và địa chỉ nhận, máy tính gửi sẽ đặt vào trường identification một số định danh. Tại máy tính gửi, giá trị của số định danh sẽ tăng dần. Khi router cần chia nhỏ gói dữ liệu, tất cả các gói dữ liệu con được tạo ra đều có địa chỉ nguồn, địa chỉ đích và giá trị trường định danh giống hệt gói dữ liệu ban đầu. Khi máy tính đích nhận được một loạt các gói dữ liệu từ cùng một nơi gửi đến, nó có thể kiểm tra giá trị định danh để xác định liệu những gói dữ liệu đó có phải là các mảnh của một gói dữ liệu lớn hay không. Vì dịch vụ IP không tin cậy, nên một số mảnh có thể không đến được đích. Để máy tính nhận có thể chắc chắn là đã nhận được mảnh cuối cùng của gói dữ liệu ban đầu, thì trường cờ của mảnh cuối cùng phải có giá trị 0, trong khi trường cờ của các mảnh khác có giá trị 1. Tương tự, để máy nhận xác định được liệu có mất mảnh nào không (và để ghép các mảnh theo đúng thứ tự), trường offset được sử dụng để xác định vị trí của mảnh trong gói dữ liệu IP ban đầu.

Xét ví dụ trên Hình 4.21. Một gói dữ liệu có độ lớn 4.000 byte đến router và phải gửi qua đường liên kết có MTU là 1.500 byte. Điều này có nghĩa rằng, 3.980 byte dữ liệu trong gói dữ liệu ban đầu phải được tách ra thành ba mảnh phân biệt (mỗi mảnh trở thành một gói dữ liệu IP độc lập). Giá trị trong gói dữ liệu ban đầu giá trị trường định danh là 777. Giá trị các trường trong 3 phân mảnh này được chỉ ra trong Bảng 4.3.



Hình 4.21. Phân mảnh và hợp nhất gói dữ liệu IP

Dữ liệu của gói IP chỉ được chuyển lên tầng giao vận tại máy tính nhận khi tầng IP "tái tạo" hoàn chỉnh gói dữ liệu IP ban đầu. Nếu một số mảnh dữ liệu bị mất, không đến được đích, thì toàn bộ gói dữ liệu sẽ bị loại bỏ và không được chuyển lên tầng giao vận. Nhưng, như đã trình bày trong chương trước, nếu sử dụng TCP ở tầng giao vận, thì thực thể TCP sẽ khắc phục mất mát do phía gửi sẽ gửi lại gói dữ liệu ban đầu.

Phân mảnh và hợp nhất khiến nhiệm vụ xử lý gói tin tại router (tạo ra các mảnh) và thiết bị nhận (hợp nhất các mảnh) phức tạp hơn. Vì thế, người ta cố gắng giảm thiểu việc phân mảnh dữ liệu. Điều này thường được thực hiện bằng cách giới hạn độ lớn gói dữ liệu của tầng giao vận (TCP hay UDP

segment) bởi một giá trị tương đối nhỏ. Khi đó, việc phân mảnh trở nên không cần thiết. Vì phần lớn các giao thức liên kết dữ liệu hỗ trợ IP có MTU tối thiểu là 536 byte, có thể loại bỏ hoàn toàn việc phân mảnh nếu đặt giá trị MSS là 536 byte với 20 byte tiêu đề của gói TCP và 20 byte tiêu đề của gói IP. Đây chính là lý do hầu hết các gói TCP khi truyền khối lượng lớn dữ liệu (chẳng hạn FTP) có độ dài từ 512 đến 536 byte (Khi duyệt WEB, bạn sẽ thấy thường khoảng 500 byte dữ liệu đến cùng một lần).

**Bảng 4.3. Ví dụ phân mảnh gói tin**

Fragment	Bytes	ID	Offset	Flag
1	1480 byte trong trường dữ liệu	identification=777	offset=0 (dữ liệu bắt đầu từ byte thứ 0)	flag=1 (còn mảnh nữa)
2	1480 byte trong trường dữ liệu	identification=777	offset=1480 (dữ liệu bắt đầu từ byte thứ 1480)	flag=1 (còn mảnh nữa)
3	1020 byte (=3980 – 1480 – 1480) trong trường dữ liệu	identification=777	offset=2960 (dữ liệu bắt đầu từ byte thứ 2960)	flag=0 (đây là mảnh cuối cùng)

Nếu gói TCP được lồng trong gói IP và cả hai gói TCP và IP đều không có trường tùy chọn (option), thì gói dữ liệu IP sẽ có 40 byte tiêu đề, phần còn lại là dữ liệu ứng dụng.

#### 4.4.5. Giao thức kiểm soát lỗi ICMP (Internet Control Message Protocol)

ICMP được các máy tính đầu/cuối, router và các cổng (gateway) sử dụng để trao đổi các thông tin tầng mạng với nhau. ICMP được đặc tả trong RFC 792. ICMP được sử dụng chủ yếu cho việc báo lỗi. Ví dụ, khi chạy

một phiên Telnet, FTP hoặc HTTP, bạn có thể gặp một thông điệp như "Destination network unreachable" (*Không đến được mạng đích*). Thông điệp này do thực thể ICMP ở router tạo ra. Khi không tìm được đường dẫn đến máy tính đích, router sẽ tạo ra và gửi thông báo ICMP **kiểu 3** tới máy tính của bạn với mục đích thông báo lỗi. Máy tính nhận được thông báo lỗi ICMP sẽ trả lại mã lỗi cho thực thể TCP đang cố gắng kết nối tới máy tính đích. Đến lượt mình, TCP trả lại mã lỗi cho ứng dụng (là phiên làm việc FTP, HTTP,...).

ICMP Type	Code	Description
0	0	echo reply (to ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unknown
4	0	source quench (congestion control)
8	0	echo request
9	0	router advertisement
10	0	router discovery
11	0	TTL expired
12	0	IP header bad

**Hình 4.22.** Các kiểu thông điệp ICMP hay gặp

ICMP thường được coi là một phần của IP, nhưng về mặt kiến trúc lại nằm trên IP, bởi vì thông báo ICMP được đặt trong gói IP, giống như TCP

hay UDP segment nằm trong trường dữ liệu (payload) của gói dữ liệu IP. Tương tự, khi nhận được một gói tin IP với trường protocol xác định giao thức ICMP, tầng mạng của máy tính nhận sẽ chuyển phần dữ liệu (là thông điệp ICMP) lên thực thể ICMP, giống như đã làm với TCP hay UDP.

Thông báo ICMP có trường kiểu (type) và trường mã (code), chứa 8 byte đầu tiên của gói dữ liệu IP gây ra lỗi (nguyên nhân để tạo ra thông báo ICMP). Do đó, phía gửi có thể xác định được gói tin nào gây ra lỗi. Một số kiểu thông điệp ICMP tiêu biểu được minh họa trên Hình 4.22. Chú ý rằng, thông báo ICMP không chỉ được sử dụng để báo lỗi. Chương trình Ping rất thông dụng gửi thông báo ICMP kiểu 8, mã 0 tới máy tính nào đó, máy tính nhận được yêu cầu ICMP sẽ gửi lại một thông báo ICMP phản hồi với kiểu 0, mã 0.

Chương trình Traceroute cho phép ta xác định tất cả các router trên một tuyến đường giữa bất kỳ hai thiết bị đầu/cuối nào. Chương trình Traceroute cũng sử dụng các thông báo của ICMP để xác định tên và địa chỉ của các router giữa nguồn và đích. Chương trình Traceroute trong máy tính nguồn sẽ gửi đi một loạt các gói dữ liệu IP tới máy tính đích. Gói IP đầu tiên có trường TTL nhận giá trị 1, gói thứ hai là 2, gói thứ ba là 3,... Máy tính nguồn đặt timer cho mỗi gói IP gửi đi. Khi gói IP thứ n đến router thứ n, router này thấy trường TTL của gói dữ liệu nhận giá trị 0, nên theo nguyên tắc của giao thức IP, router sẽ loại bỏ gói dữ liệu và gửi thông điệp cảnh báo ICMP (kiểu 11 mã 0). Trong thông điệp cảnh báo này có tên và địa chỉ IP của router. Khi nhận được thông báo ICMP, máy tính nguồn xác định được thời gian khứ hồi đến router thứ n (nhờ timer) cũng như tên và địa chỉ IP của router đó.

## 4.5. ĐỊNH TUYẾN TRÊN INTERNET

Sau khi đã nghiên cứu về địa chỉ Internet và giao thức IP, bây giờ nói tới các giao thức định tuyến của Internet – là các giao thức xác định tuyến đường đi từ nguồn tới đích. Ta thấy rằng, các giao thức định tuyến của Internet được triển khai dựa trên những nguyên tắc đã nói tới như link state và distance vector trong mục 4.2, miền tự trị (AS) trong mục 4.3.

Mạng Internet toàn cầu ngày nay là sự kết hợp lỏng lẻo của nhiều mạng, bao gồm các ISP khu vực, quốc gia và quốc tế. Trong mục 4.3 ta thấy rằng, tập hợp các router cùng nằm dưới một sự quản trị – ít nhất về mặt kỹ thuật, tạo thành miền tự trị (AS). Mỗi AS lại có thể bao gồm nhiều mạng (ở đây sẽ sử dụng thuật ngữ "mạng" với ý nghĩa giống trong mục 4.4). Điểm phân biệt quan trọng nhất giữa các giao thức định tuyến của Internet là liệu chúng được sử dụng để định tuyến trong một miền hay giữa các miền với nhau. Mục 4.5.1 giới thiệu giao thức định tuyến trong một miền và mục 4.5.2 giới thiệu các giao thức định tuyến giữa các miền.

#### **4.5.1. Định tuyến trong một miền (Intra-AS routing) (Định tuyến nội miền)**

Giao thức định tuyến Intra-AS được sử dụng để cấu hình và duy trì bảng định tuyến trong tất cả các router thuộc cùng một miền. Những giao thức định tuyến kiểu này được gọi là *giao thức định tuyến nội miền* (interior gateway protocol). Trên Internet có 3 giao thức định tuyến nội miền được sử dụng rộng rãi là RIP (Routing Information Protocol), OSPF (Open Shortest Path First) và EIGRP (Cisco's proprietary Enhanced Interior Gateway Routing Protocol).

##### **a) RIP – Routing Information Protocol**

RIP là một trong những giao thức định tuyến nội miền đầu tiên. Nó được triển khai trong một chương trình được gọi là *routed* trong phần lớn các hệ thống UNIX. RIP có một số đặc điểm sau:

- ☛ **Định tuyến nội miền:** Cho phép các router trong một miền trao đổi thông tin với nhau.
- ☛ **Đo khoảng cách bằng chặng:** Giá đường đi giữa hai thiết bị đầu/cuối được xác định bằng số lượng các router trung gian trên đường đi đó. Độ dài tối đa của một tuyến đường là 15, nghĩa là "đường kính" tối đa của một miền là 15 router.
- ☛ **Truyền thông không tin cậy:** RIP sử dụng UDP để truyền thông điệp.
- ☛ **Gửi quảng bá (broadcast) và multicast:** RIP được sử dụng chủ yếu trên mạng cục bộ (LAN) hỗ trợ công nghệ truyền quảng bá

(mạng Ethernet). RIP v1 sử dụng cách truyền quảng bá khi truyền giữa hai router. RIP v2 cho phép truyền theo chế độ multicast.

- ☞ **Thuật toán Distance vector:** RIP sử dụng thuật toán Distance vector. Các router hàng xóm trao đổi bảng định tuyến cho nhau 30s một lần trong các thông điệp RIP (RIP response message, RIP advertisement), mỗi thông điệp chứa tối đa 25 địa chỉ đích tới.
- ☞ **Các máy tính có thể thu nhận thông tin từ các router:** RIP cho phép các thiết bị đầu/cuối (chủ yếu là máy tính) "lắng nghe" và cập nhật bảng định tuyến. Điều này đặc biệt hữu dụng với các mạng có nhiều router. Khi đó, máy tính trong mạng có thể dễ dàng xác định được router cần chuyển tới.

Chú ý rằng, router gửi một thông điệp RIP liệt kê các mạng mà nó có thể kết nối tới. Khi nhận được một quảng bá như vậy, thực thể RIP (phần mềm) trên router sử dụng những thông tin này để cập nhật lại bảng định tuyến của mình. Mỗi một trường trong thông điệp bá là một cặp:

(địa chỉ mạng đích  $n$ , khoảng cách  $r$ )

trong đó, khoảng cách  $r$  là số lượng các router trung gian từ router gửi thông điệp tới đích có địa chỉ mạng là  $n$ .

Khi nhận được một thông điệp, giả sử router nhận không có đường đi tới đích được quảng bá trong thông điệp, hoặc có đường đi đến đích nhưng giá lớn hơn, router sẽ cập nhật bảng định tuyến để sử dụng tuyến đường vừa mới nhận được quảng bá (điểm đầu tiên trên tuyến đường này chính là router gửi quảng bá).

Ưu điểm chính của RIP là tính đơn giản. RIP không đòi hỏi cấu hình nhiều. Người quản trị chỉ cần bật máy lên, cho phép router trao đổi thông tin với nhau, sau một thời gian ngắn, router sẽ tự xây dựng được bảng định tuyến cho mình.

Tổ chức có thể lựa chọn một router trong miền làm router ngầm định, thường là router nối với ISP. Sau đó, RIP sẽ thực hiện việc quảng bá cho router ngầm định này. Tiếp theo, các gói tin gửi ra phía ngoài sẽ được gửi qua router ngầm định tới ISP.

Hình 4.23 minh họa khuôn dạng thông điệp cập nhật RIP. Mỗi trường trong thông điệp ứng với một địa chỉ đích, mặt nạ mạng của địa chỉ đích (do

đó có thể sử dụng địa chỉ không phân lớp CIDR), khoảng cách tới đích và nút kế tiếp trên đường tới đích.

0	8	16	24	31
<b>COMMAND (1-5)</b>	<b>VERSION (2)</b>	<b>MUST BE ZERO</b>		
		<b>FAMILY OF NET 1</b>	<b>ROUTE TAG FOR NET 1</b>	
			<b>IP ADDRESS OF NET 1</b>	
			<b>SUBNET MASK FOR NET 1</b>	
			<b>NEXT HOP FOR NET 1</b>	
			<b>DISTANCE TO NET 1</b>	
		<b>FAMILY OF NET 2</b>	<b>ROUTE TAG FOR NET 2</b>	
			<b>IP ADDRESS OF NET 2</b>	
			<b>SUBNET MASK FOR NET 2</b>	
			<b>NEXT HOP FOR NET 2</b>	
			<b>DISTANCE TO NET 2</b>	
				...

Hình 4.23. Khuôn dạng gói dữ liệu RIP

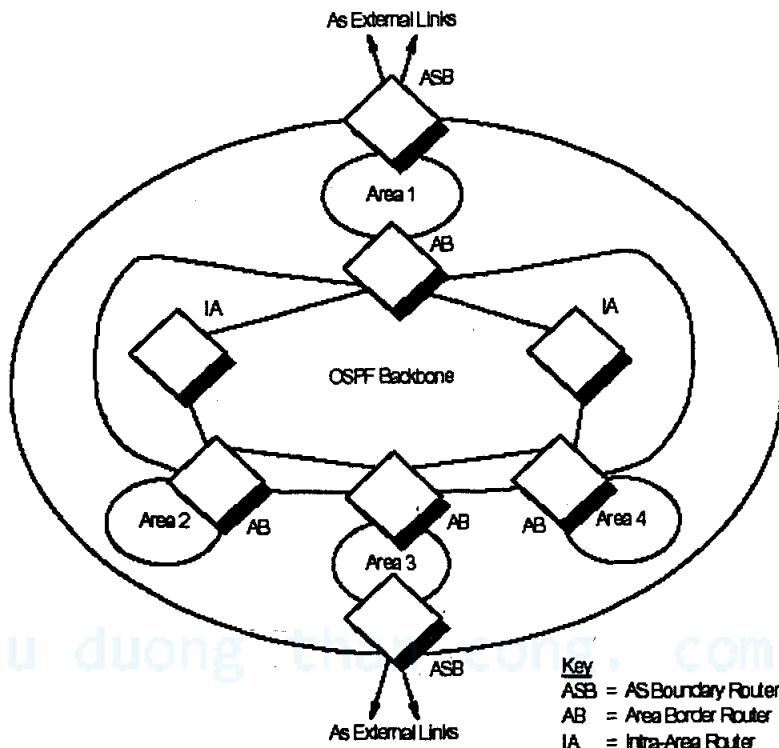
#### b) OSPF – Open Shortest Path First

RIP có một số nhược điểm của thuật toán Distance vector. Độ dài của thông điệp có thể lớn do phải liệt kê toàn bộ danh sách các địa chỉ đích và khoảng cách tới đó. Khi nhận được thông điệp, router nhận phải lây ra từng trường, so sánh trong bảng định tuyến. Như vậy, thời gian xử lý thông điệp trong mỗi router lớn, gây ra một độ trễ nào đó. Do vậy, RIP chỉ phù hợp với các mạng có kích cỡ nhỏ.

Khi một tổ chức mạng tương đối lớn, người ta cần phải đưa ra giao thức phù hợp hơn. IETF đưa ra OSPF với các đặc điểm sau:

- ☛ **Định tuyến nội miền:** Cho phép trao đổi thông tin giữa các router trong một miền.
- ☛ **Hỗ trợ phân mạng và CIDR:** Bên cạnh địa chỉ IP 32 bit là mặt nạ 32 bit. Do đó, OSPF hỗ trợ việc phân mạng, chia mạng lớn thành các mạng con.
- ☛ **Trao đổi các thông tin đã được kiểm chứng:** Hai router trao đổi thông điệp OSPF với nhau có thể tiến hành thủ tục kiểm tra để xác định mình nhận được thông điệp từ đúng phía bên kia. Điều này ngăn ngừa được tin tặc tiến hành các cuộc tấn công bằng phương pháp giả mạo.
- ☛ **Sử dụng thuật toán Link state.**

- ☞ **Hỗ trợ phân cấp trong miền:** Ưu điểm chính của OSPF là cho phép tiếp tục phân một miền thành nhiều miền con.



Hình 4.24. OSPF cho phép chia miền con

#### 4.5.2. Định tuyến giữa các miền (Inter-AS routing) (Định tuyến liên miền)

Giao thức BGP (Border Gateway Protocol v4) (xem đặc tả RFC 1771, 1772, 1773) được xem là một chuẩn ngầm định *de facto* trong định tuyến liên miền trên Internet ngày nay. Nhiệm vụ của nó là định tuyến giữa các miền được quản trị độc lập với nhau.

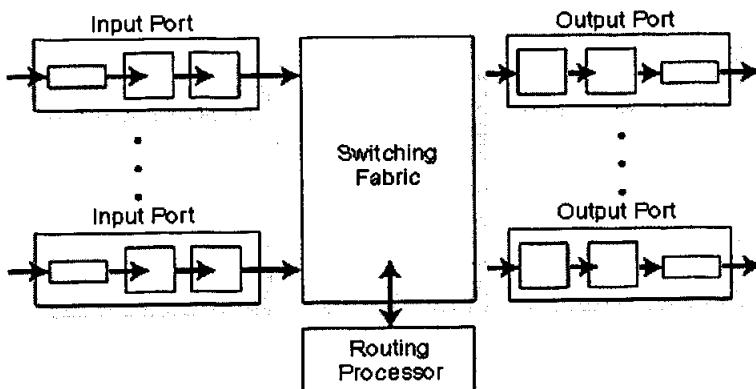
##### **BGP – Border Gateway Protocol:**

BGP là giao thức liên miền chủ yếu được sử dụng hiện nay. BGP có những đặc điểm sau:

- ☞ **Định tuyến liên miền:** BGP cho phép cung cấp các thông tin định tuyến giữa các miền. Mỗi tuyến đường được xem là một chuỗi các AS liên tiếp nhau.

- ☞ **Hỗ trợ việc thiết lập chính sách (policy):** Người quản trị có thể áp dụng những chính sách nào đó, ví dụ, hạn chế việc quảng bá ra phía ngoài.
- ☞ **Truyền thông tin cậy:** Hai thực thể BGP sử dụng kết nối TCP để trao đổi thông điệp.

## 4.6. CẤU TẠO CỦA THIẾT BỊ ĐỊNH TUYẾN (ROUTER)



**Hình 4.25. Kiến trúc bộ định tuyến**

Các phần trước trình bày về các mô hình dịch vụ của tầng mạng, các thuật toán định tuyến xác định đường đi cho các gói tin trên mạng cũng như các giao thức gắn với các thuật toán ấy. Trong phần này, sẽ nói đến một chủ đề quan trọng khác là chức năng chuyển mạch của bộ định tuyến – công việc thực sự để chuyển một datagram từ liên kết này tới liên kết kia. Nghiên cứu các khía cạnh về mặt kiểm soát và dịch vụ của tầng mạng cũng giống như nghiên cứu một công ty mà chỉ tìm hiểu cơ chế quản lý của công ty và các quan hệ với bên ngoài. Để hiểu rõ về công ty, người ta phải xem xét đến công nhân. Trong tầng mạng, công việc thực sự của việc truyền gói tin chính là việc chuyển gói tin từ liên kết này tới liên kết kia của router. Trong mục này sẽ trình bày router thực hiện công việc này như thế nào.

Một cách tổng thể, kiến trúc chung của router được minh họa trong Hình 4.25. Bốn thành phần chính của router được xác định như sau:

- ☞ **Cổng vào (Input port):** Cổng vào của router thực hiện một số chức năng như chức năng của tầng vật lý, tầng liên kết dữ liệu (là các hộp ở giữa đối với cả đường vào và đường ra) cần thiết để làm việc

được với tầng liên kết dữ liệu ở đầu bên kia của kết nối; chức năng tìm kiếm và chuyển (hộp trong cùng của cổng vào và cổng ra). Gói tin từ cổng vào sẽ đi qua kết cấu chuyển để tới cổng ra phù hợp. Các gói tin chứa thông tin điều khiển (chứa thông tin điều khiển của các giao thức RIP, OSPF, BGP) sẽ được chuyển từ cổng vào đến bộ xử lý của router.

- ☞ **Kết cấu chuyển** (Switching fabric): Kết cấu chuyển nối cổng vào của router tới cổng ra. Kết cấu chuyển nằm hoàn toàn trong router – là một mạng chuyển mạch nằm bên trong router mạng.
- ☞ **Cổng ra** (output port): Cổng ra nhận những gói dữ liệu gửi tới nó qua kết cấu chuyển, sau đó truyền gói dữ liệu này trên đường nối ra ngoài. Nó cũng thực hiện chức năng của tầng liên kết dữ liệu và tầng vật lý.
- ☞ **Bộ xử lý** (Routing Processor): Bộ xử lý router thực hiện các giao thức định tuyến (ví dụ các giao thức đã nói trong mục 4.5), duy trì bảng định tuyến và thực hiện một số chức năng quản trị mạng.

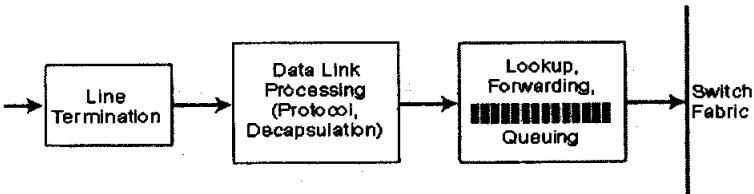
Dưới đây sẽ trình bày các thành phần này một cách kỹ lưỡng hơn.

#### 4.6.1. Cổng vào (Input port)

Trong Hình 4.26 minh họa chi tiết các chức năng của cổng vào. Như đã nói ở trên, chức năng kết thúc tín hiệu trên đường truyền (line termination) và xử lý liên kết dữ liệu ứng với tầng vật lý và tầng liên kết dữ liệu của một đường truyền vật lý thực sự với router. Chức năng tìm kiếm và chuyển tiếp của cổng vào đóng vai trò trung tâm trong việc chuyển của bộ định tuyến. Trong nhiều router, cổng vào chính là nơi xác định cổng ra cho một gói dữ liệu. Cổng ra được xác định nhờ các thông tin lưu trong bảng định tuyến. Mặc dù bảng định tuyến được bộ xử lý tạo ra, song mỗi cổng vào đều có một bảng sao chép bảng định tuyến và cập nhật khi cần thiết. Nhờ vậy, quyết định chuyển đến cổng ra nào có thể được thực hiện cục bộ ở tại cổng vào, mà không cần đến bộ xử lý trung tâm. Điều này khiến bộ xử lý sẽ không trở thành một nút cỗ chai của router.

Với những router mà khả năng xử lý ở cổng vào còn hạn chế, cổng vào sẽ chuyển gói dữ liệu tới bộ xử lý. Bộ xử lý sẽ tìm kiếm trên bảng định tuyến để xác định cổng ra thích hợp. Người ta thường áp dụng giải pháp này

trong trường hợp router là một trạm làm việc hay một máy tính. Khi đó, bộ xử lý của router là bộ xử lý của trạm làm việc hay của máy tính. Cổng vào sẽ là card mạng (ví dụ card Ethernet).



Hình 4.26. Cấu trúc cổng vào của Router

Với bảng định tuyến xác định trước, tìm kiếm trên bảng định tuyến tương đối đơn giản. Duyệt toàn bộ bảng định tuyến để xác định hàng nào có địa chỉ phù hợp nhất với địa chỉ đích của gói dữ liệu, trong trường hợp không tìm thấy, thì sử dụng cổng mặc định (Trong mục 4.4.1 ta thấy rằng, địa chỉ phù hợp nhất là địa chỉ có tiền tố mạng dài nhất trùng với phần mạng của địa chỉ đích). Tuy nhiên, triển khai trong thực tế lại không đơn giản như thế. Điều phức tạp nhất là router trên các trục chính (backbone router) phải hoạt động ở tốc độ cao, có khả năng thực hiện hàng triệu phép tra cứu mỗi giây. Thực sự người ta mong muốn tốc độ xử lý ở cổng vào phải ngang với tốc độ đường truyền (line speed), có nghĩa là tốc độ xử lý phải nhỏ hơn tốc độ đến của các gói tin. Như vậy, gói tin có thể được xử lý xong trước khi gói tin kế tiếp đến.

Để hoạt động ở tốc độ cao, không thể sử dụng phương pháp tìm kiếm tuyến tính trên bảng định tuyến, hợp lý hơn là lưu giữ giá trị của bảng định tuyến trong cấu trúc dữ liệu dạng cây. Mỗi mức trong cây ứng với vị trí một bit trong địa chỉ đích. Để tìm kiếm một địa chỉ, bắt đầu từ "gốc" của cây. Nếu bit địa chỉ đầu tiên là 0 thì địa chỉ cần tìm nằm trong cây con trái, ngược lại nằm trong cây con phải. Tiếp tục duyệt cây con bằng cách sử dụng các bit còn lại. Bit kế tiếp bằng 0, tìm trên cây con trái, bit kế tiếp bằng 1 tìm trên cây con phải. Người ta sẽ tìm kiếm bảng định tuyến trong N bước, N là số lượng bit trong địa chỉ.

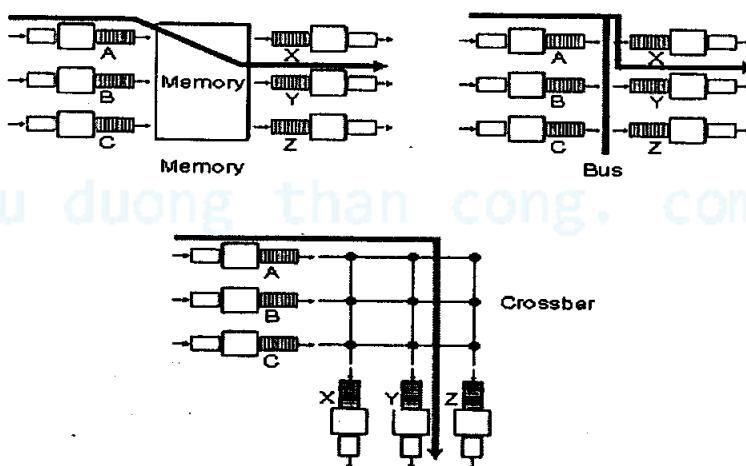
Khi xác định được cổng ra, gói dữ liệu sẽ được chuyển đến qua kết cấu chuyển. Tuy nhiên, gói dữ liệu tạm thời có thể bị "phong toả" chưa được chuyển qua kết cấu (có thể do các gói dữ liệu khác đang sử dụng kết cấu chuyển). Một gói dữ liệu bị phong toả phải xếp hàng ở cổng vào và đợi được lên lịch chuyển qua kết cấu tại một thời điểm nào đó.

#### 4.6.2. Kết cấu chuyển (Switching fabric)

Kết cấu chuyển nằm ở trung tâm của router. Gói dữ liệu chuyển từ cổng vào đến cổng ra qua kết cấu chuyển. Việc chuyển được thực hiện bằng nhiều cách như minh họa trên Hình 4.27.

- ☞ **Chuyển qua bộ nhớ:** Các router đơn giản nhất thuộc thế hệ đầu tiên, thường là các máy tính truyền thông, việc chuyển từ cổng vào tới cổng ra được thực hiện dưới sự điều khiển trực tiếp của CPU. Cổng vào và cổng ra chỉ là các thiết bị vào ra truyền thông trong hệ điều hành. Khi nhận được một gói dữ liệu, cổng vào sẽ sử dụng ngắn để báo cho CPU. Sau đó, gói dữ liệu được sao chép vào bộ nhớ của bộ vi xử lý. Bộ vi xử lý lấy địa chỉ đích từ tiêu đề gói tin, tìm cổng ra trong bảng định tuyến và sao chép gói dữ liệu vào bộ đệm của cổng ra.

Nhiều router hiện đại cũng thực hiện việc chuyển qua bộ nhớ. Điểm khác biệt chính là ở chỗ, việc xác định địa chỉ đích và việc lưu trữ (chuyển) gói dữ liệu vào vị trí phù hợp trong bộ nhớ được thực hiện bởi bộ xử lý trên mạch cổng vào. Router chuyển qua bộ nhớ, theo một cách nào đó giống hệ thống nhiều bộ xử lý với bộ nhớ dùng chung, bộ xử lý trên cổng vào sẽ đặt gói dữ liệu vào bộ nhớ của một cổng ra thích hợp. Dòng router Cisco Catalyst 8500 và dòng Bay Network Accelar 1200 chuyển gói dữ liệu qua bộ nhớ dùng chung.



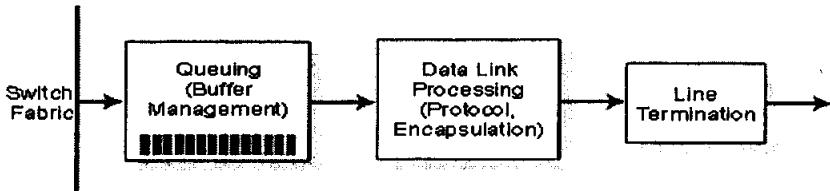
Hình 4.27. Kết cấu chuyển

- ☞ **Chuyển qua bus:** Cổng vào chuyển thẳng gói tin tới cổng ra qua một đường bus dùng chung mà không cần bộ xử lý của router can thiệp (chú ý là, khi chuyển qua bộ nhớ, gói tin phải qua bus hệ thống để đến hay đi khỏi bộ nhớ). Mặc dù bộ xử lý của router không liên quan đến việc chuyển trên bus, song do bus dùng chung, tại một thời điểm chỉ cho phép một gói tin được truyền dẫn trên bus. Một gói tin đến cổng vào và thấy bus đang bị chiếm dụng bởi gói tin khác sẽ tạm thời bị chặn lại, đưa vào hàng đợi ở cổng vào. Vì tất cả các gói tin đều phải truyền qua một bus duy nhất, tốc độ chuyển của router bị giới hạn bởi tốc độ bus. Với công nghệ băng thông của bus vượt qua 1Gbit/s, chuyển mạch qua bus đủ hiệu quả với các router hoạt động ở mức tổ chức (ví dụ mạng cục bộ). Dòng Cisco 1900 chuyển mạch các gói qua bus 1Gbit/s.
- ☞ **Chuyển mạch qua một liên mạng:** Một cách khắc phục hạn chế của bus dùng chung duy nhất là sử dụng một mạng liên kết phức tạp, giống các kỹ thuật được sử dụng để kết nối những bộ xử lý trong hệ thống đa bộ xử lý.

Hình 4.27 minh họa một mạng liên kết sử dụng 2N bus để nối N cổng vào với N cổng ra. Một gói tin đến từ một cổng vào sẽ được chuyển dọc theo bus nằm ngang gắn với cổng vào cho tới khi gặp giao điểm với bus nằm dọc gắn với cổng ra tương ứng. Nếu bus nằm dọc đó rỗi, gói tin sẽ được chuyển trên bus dọc đó tới cổng ra cần đến. Trong trường hợp ngược lại, gói tin tạm thời bị chặn lại và xếp hàng tại cổng vào. Dòng Cisco 12000 Family sử dụng công nghệ này.

#### 4.6.3. Cổng ra (Output port)

Quá trình xử lý tại cổng ra được minh họa trên Hình 4.28: Lấy gói dữ liệu đã được lưu trữ trong bộ đệm của cổng ra và truyền qua đường liên kết ra. Các chức năng xử lý giao thức liên kết dữ liệu và kết thúc đường truyền là chức năng tầng liên kết dữ liệu và tầng vật lý để làm việc với đầu vào bên kia của đường truyền vật lý. Chức năng quản lý vùng đệm và hàng đợi được sử dụng khi tốc độ dữ liệu mà kết cấu chuyển tiếp nhanh hơn tốc độ gửi đi của cổng ra.

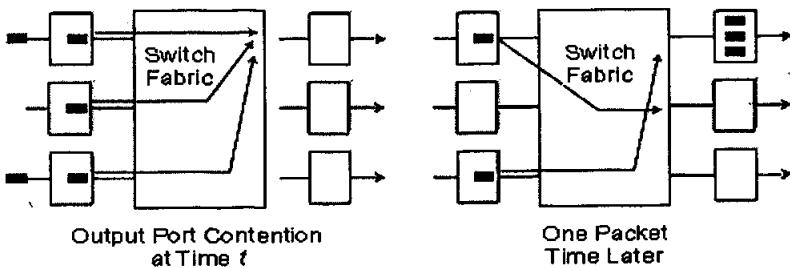


Hình 4.28. Cấu trúc cồng ra

#### 4.6.4. Hàng đợi ở router

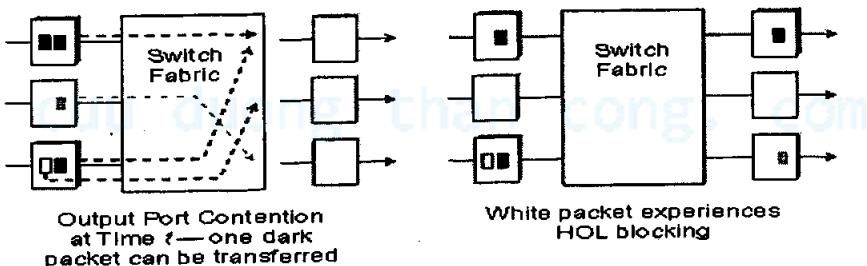
Nếu nhìn vào chức năng, cấu hình của cổng vào, cổng ra trong Hình 4.27, rõ ràng hàng đợi của các gói tin có thể được hình thành tại cả cổng vào và cổng ra. Vì khi hàng đợi lớn, bộ đệm của router sẽ đầy lên và xảy ra hiện tượng mất gói tin (tràn bộ đệm). Trong các phần trước, đã nói rằng, gói tin bị mất đâu đó "trong mạng" hay "tại router". Chính tại các hàng đợi của router, gói tin bị mất. Trên thực tế, vị trí gói tin bị mất (cổng vào hay cổng ra) phụ thuộc vào tài của mạng, tốc độ tương đối giữa kết cấu chuyển và đường truyền.

Giả sử tốc độ đường truyền vào và ra bằng nhau, và có  $n$  cổng vào,  $n$  cổng ra. Nếu tốc độ chuyển của kết cấu chuyển lớn hơn tốc độ đường truyền  $n$  lần, thì chắc chắn không có hàng đợi tại cổng vào. Bởi trong tình huống xấu nhất, tất cả  $n$  đường vào cùng nhận được gói tin, kết cấu chuyển có khả năng truyền  $n$  gói tin từ cổng vào tới cổng ra. Nhưng điều gì có thể xảy ra tại cổng ra? Giả sử tốc độ kết cấu chuyển vẫn nhanh hơn tốc độ đường truyền  $n$  lần. Trong trường hợp xấu nhất, tất cả gói tin từ  $n$  cổng vào cùng đến một cổng ra. Mỗi lần cổng ra chỉ có thể gửi đi một gói tin duy nhất, do đó  $n$  gói tin sẽ phải xếp hàng tại cổng ra để đợi truyền. Khi số lượng gói tin xếp hàng vượt qua độ lớn bộ đệm, các gói tin đến sau sẽ bị mất. Hàng đợi tại cổng ra được minh họa trong Hình 4.29. Tại thời điểm  $t$ , tất cả các cổng vào đều nhận được một gói tin và phải chuyển tới cổng ra trên cùng bên phải. Giả sử tốc độ ba đường truyền là như nhau, tốc độ kết cấu chuyển nhanh hơn tốc độ đường truyền 3 lần. Sau một đơn vị thời gian (thời gian cần thiết để nhận hay gửi một gói tin), cả 3 gói tin được chuyển tới cổng ra và xếp hàng để đợi truyền đi. Trong một đơn vị thời gian tiếp theo, một gói tin được truyền đi trên đường truyền ra. Cùng lúc đấy, lại có thêm 2 gói tin khác chuyển tới cổng ra trên cùng và do đó sẽ phải xếp hàng.



Hình 4.29. Mất gói tin tại Router

Bộ điều phối (scheduler) tại cổng ra phải chọn một gói tin trong hàng đợi để truyền đi. Có thể sử dụng một cơ chế đơn giản như First – Come – First – Served (Người đến trước được phục vụ trước – FCFS), hay cơ chế hàng đợi có trọng số (WFQ – weighted fair queuing) phức tạp hơn, cho phép chia sẻ một cách tương đối công bằng đường truyền ra giữa các kết nối đầu/cuối khác nhau. Bộ điều phối có vai trò quyết định trong việc bảo đảm chất lượng dịch vụ (quality – of – service).



Hình 4.30. Ví dụ về tắc nghẽn

Nếu kết cấu chuyển không đủ nhanh (so với tốc độ cổng vào) để chuyển ngay lập tức tất cả các gói tin qua, hàng đợi sẽ xuất hiện tại cổng vào, vì khi đó các gói tin sẽ phải xếp hàng đợi đến lượt chuyển qua kết cấu chuyển tới cổng ra. Để minh họa, xét chuyển mạch qua một liên mạng trong Hình 4.30 và giả sử (1) tốc độ của tất cả các liên kết bằng nhau; (2) thời gian gói tin chuyển từ cổng vào tới cổng ra bằng thời gian cổng vào nhận được một gói tin; (3) các gói tin được chuyển từ cổng vào tới cổng ra theo thứ tự đến (FCFS). Nhiều gói tin có thể được truyền đồng thời, miễn là cổng ra của chúng khác nhau. Tuy nhiên, nếu hai gói tin xếp đầu hàng đợi trên hai cổng vào khác nhau cùng hướng tới một cổng ra, thì một gói tin sẽ bị chặn lại tại cổng vào (phải xếp hàng trong hàng đợi), vì tại một thời điểm kết cấu chuyển chỉ có thể chuyển đi một gói tin.

Trong Hình 4.30 ta thấy hai gói tin (tô đen) đứng đầu hai hàng đợi cùng hướng tới cổng ra phía trên bên phải. Giả sử kết cấu chuyển sê chuyển gói tin từ hàng đợi phía trên bên trái trước. Khi đó, gói tin màu đen ở hàng đợi phía dưới bên trái sẽ phải đợi. Nhưng không chỉ có gói tin này phải đợi mà còn gói tin màu trắng xếp hàng sau nó cũng phải đợi – mặc dù các gói tin này hướng tới cổng ra khác. Đây là hiện tượng head – of – the – line (HOL) blocking. [Karol 1997] đã chứng minh rằng, hiện tượng HOL có thể khiến hàng đợi tăng lên vô hạn ngay cả khi tốc độ đến của các gói tin trên cổng vào bằng 58% tốc độ tối đa. [McKeown 1997b] đưa ra nhiều giải pháp ngăn chặn HOL.

## 4.7. IPv6

Đầu những năm 1990, Internet Engineering Task Force bắt đầu nỗ lực phát triển giao thức mạng thay thế IPv4. Nguyên nhân đầu tiên cho nỗ lực này là không gian địa chỉ IP 32 bit đã bắt đầu cạn kiệt trong khi số lượng những mạng mới và những nút mạng được kết nối vào Internet (cần cấp phát một địa chỉ IP duy nhất) tăng lên đáng kể. Để giải quyết nhu cầu có không gian địa chỉ IP lớn hơn, giao thức mạng IPv6 đã được phát triển. Những người thiết kế IPv6 cũng chọn lọc các tính năng, cải tiến nhiều đặc điểm khác của IPv4 dựa trên cơ sở những kinh nghiệm thực tế của IPv4.

Người ta chưa thống nhất được khi nào địa chỉ IPv4 cạn kiệt (khi đó không thể kết nối thêm bất kỳ máy tính nào vào mạng). Căn cứ trên xu hướng cấp địa chỉ IP hiện tại, hai nhóm làm việc trong IETF's Address Life time Expectations đưa ra hai thời điểm khác nhau là năm 2008 và năm 2018 [Solenky 1996]. Trong năm 1996, American Registry for Internet Number (ARIN) thông báo tất cả các địa chỉ IPv4 lớp A, 62% lớp B và 37% lớp C đã được phân phối [ARIN 1996]. Mặc dù những đánh giá và các con số dự đoán trên cho thấy còn có nhiều thời gian cho tới khi không gian địa chỉ IPv4 hết, song đã đến lúc triển khai một công nghệ mới trên một quy mô rộng lớn, và do đó "Next Generation IP" (thế hệ IP mới) [Brander 1996: RFC1752] đã bắt đầu triển khai.

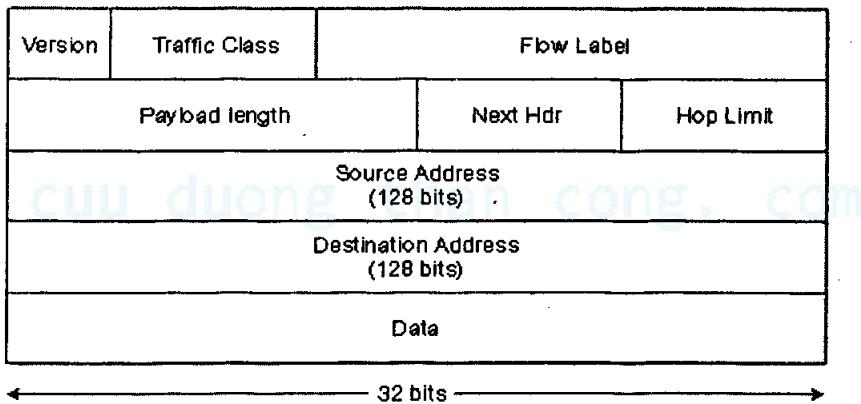
Khuôn dạng gói dữ liệu IPv6 được minh họa trên hình Hình 4.31. Điểm thay đổi quan trọng nhất của IPv6 chính là khuôn dạng gói tin.

### a) Mở rộng khả năng đánh địa chỉ

IPv6 tăng kích thước địa chỉ IP từ 32 bit lên 128 bit. Nó đảm bảo khả năng không bị thiếu địa chỉ IP. Với không gian 128 bit, có thể đánh địa chỉ cho đến từng hạt cát có trên Trái Đất. Bên cạnh địa chỉ duy nhất (unicast) và địa chỉ đa đích (multicast), IPv6 còn có một dạng địa chỉ mới gọi là "anycast address", cho phép một gói tin với địa chỉ đích thuộc kiểu "anycast address" có thể được chuyển tới một nhóm các máy tính (đặc điểm này sẽ được sử dụng ví dụ khi gửi thông điệp HTTP GET tới nhiều site phụ chứa cùng một tài liệu nào đấy).

### b) Tiêu đề có độ dài cố định 40 byte

Một số trường IPv4 mang tính chất tùy chọn. Tổng độ dài tiêu đề cố định cho phép xử lý các gói dữ liệu IPv6 nhanh hơn.



Hình 4.31. Khuôn dạng địa chỉ IPv6

### c) Gắn nhãn luồng (flow label) và độ ưu tiên (priority)

IPv6 không có định nghĩa cho "flow" một cách rõ ràng. Các khuyến nghị RFC 1752 và RFC 2460 cho phép gắn nhãn cho các gói tin thuộc về cùng một "flow". Các gói tin này đòi hỏi được xử lý một cách đặc biệt, như các dịch vụ thời gian thực với chất lượng tốt hơn. Ví dụ, các dữ liệu đa phương tiện có thể xem như một luồng liên tục. Dữ liệu các ứng dụng truyền thống, như truyền file, e-mail không được xem như một luồng. Có thể dữ liệu của những người có độ ưu tiên cao (ví dụ người trả phí cao hơn) cũng có thể coi như một luồng. Rõ ràng ở đây, những người thiết kế IPv6 đã dự đoán được nhu cầu phân biệt giữa các luồng dữ liệu ngay cả khi chưa định nghĩa chính xác được luồng là gì. Tiêu đề IPv6 cũng có trường Traffic

Class 8 bit. Trường này giống trường TOS (Type of Service) trong IPv4 có thể được sử dụng cho những gói tin có quyền ưu tiên trong một luồng, hoặc cho những ứng dụng có độ ưu tiên cao (ví dụ gói tin ICMP).

So sánh khuôn dạng gói dữ liệu IPv4 (Hình 4.20) và IPv6 (Hình 4.31), ta thấy gói IPv6 có cấu trúc đơn giản hơn. Sau đây là một số trường trong gói dữ liệu IPv6:

- ☞ **Phiên bản (version):** Trường 4 bit này xác định phiên bản IP của gói dữ liệu. Rõ ràng gói IPv6 có giá trị "6" trong trường này. Chú ý, không phải đặt giá trị "4" trong trường này thì gói dữ liệu là IPv4.
- ☞ **Traffic class:** Trường 8 bit này giống trường TOS trong IPv4.
- ☞ **Nhãn luồng (Flow label):** Trường 20 bit này xác định một luồng chứa gói dữ liệu.
- ☞ **Độ lớn dữ liệu (Payload length):** Độ lớn (tính theo byte) của phần dữ liệu không tính tiêu đề.
- ☞ **Next header:** Trường này xác định giao thức ở tầng phía trên sẽ nhận dữ liệu (ví dụ tới TCP hoặc UDP). Trường này giống trường Protocol của IPv4.
- ☞ **Hop limit:** Giá trị của trường này giảm đi 1 khi đi qua mỗi router. Nếu giá trị này bằng 0, gói dữ liệu bị loại bỏ.
- ☞ **Địa chỉ nguồn và đích (source and destination addresss):** Khuôn dạng 128 bit địa chỉ IPv6 được đặc tả trong RFC 2373.
- ☞ **Dữ liệu (data):** Khi gói tin IPv6 tới đích, các tiêu đề sẽ bị loại bỏ và phần dữ liệu này sẽ được chuyển đến thực thể ở tầng phía trên. Có một số trường trong IPv4 không xuất hiện trong IPv6.
- ☞ **Phân mảnh, hợp nhất gói tin:** IPv6 không cho phép phân mảnh và hợp nhất gói tin tại các router trung gian. Nếu một gói dữ liệu IPv6 quá lớn để có thể gửi đi trên một đường liên kết ra của router, router sẽ loại bỏ gói tin này và gửi một thông báo lỗi ICMP "Packet Too Big" tới bên gửi. Sau đó bên gửi gửi lại dữ liệu, sử dụng các gói dữ liệu có kích thước nhỏ hơn. Việc phân mảnh và hợp nhất các gói tin IP chiếm nhiều thời gian xử lý của router. Thực hiện những

công việc này tại các thiết bị đầu/cuối sẽ làm tăng tốc độ truyền trên mạng.

- ☞ **Checksum:** Do tầng giao vận (ví dụ, TCP và UDP) và các giao thức liên kết dữ liệu (ví dụ Ethenet) đã thực hiện kiểm tra lỗi, chức năng này không cần thiết trong tầng mạng, nên được bỏ đi. Vấn đề xử lý nhanh các gói tin IP cực kỳ quan trọng. Khi nói về IPv4 trong mục 4.4.1, ta đã thấy giá trị trường TTL trong tiêu đề của IPv4 giảm đi 1 khi đi qua mỗi router, nên giá trị trường checksum trong tiêu đề IPv4 cần phải được tính lại tại các router. Như vậy, giống như phân mảnh và hợp nhất, việc này khiến thời gian xử lý gói IPv4 lâu hơn.

Trong mục 4.4, giao thức ICMP được sử dụng để thông báo lỗi và cung cấp một số các thông tin hạn chế tới thiết bị đầu/cuối (ví dụ lệnh ping). Một phiên bản mới của ICMP được đặc tả cho IPv6 trong khuyến nghị RFC 2463. Bên cạnh các kiểu và mã cũ, ICMPv6 cũng đưa thêm vào nhiều kiểu và mã mới. Ví dụ, kiểu mã lỗi "Packet Too Big" hay "Unrecognized IPv6 option".

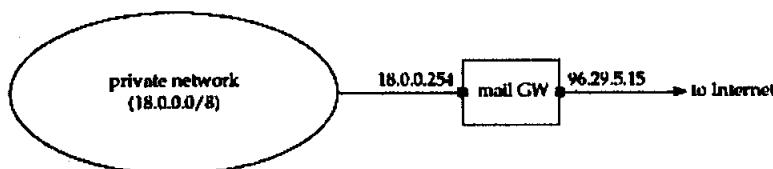
## 4.8. CƠ CHẾ DỊCH CHUYÊN ĐỊA CHỈ (NAT)

Trong trường hợp tổ chức có nhu cầu kết nối nhiều máy tính vào mạng, nhưng lại chỉ được cấp phát một lượng nhỏ địa chỉ IP chính thức, vậy làm thế nào để tất cả các máy tính trong mạng có thể truy cập Internet? NAT là một trong các giải pháp để thực hiện điều này. RFC 1918 xác định một số địa chỉ IP đặc biệt gọi là địa chỉ riêng (Hình 4.34). Gói tin có địa chỉ trong dải địa chỉ này sẽ không được router chuyển ra phía ngoài.

Khối	Dải địa chỉ
10.0.0.0/8	10.0.0.1 – 10.255.255.255
172.16.0.0/12	172.16.0.0 – 172.31.255.255
192.168.0.0/16	192.168.0.0 – 192.168.255.255

Hình 4.32. Các địa chỉ riêng

Đĩ nhiên, nếu tổ chức không có nhu cầu kết nối ra Internet, thì sử dụng bất kỳ địa chỉ IP nào cũng được. Trước khi có RFC 1918, mọi tổ chức đều có khuynh hướng chọn một lớp địa chỉ bất kỳ. Trong Hình 4.35, tổ chức chọn địa chỉ 18.0.0.0/8. Tuy nhiên, giả sử sau đó tổ chức muốn cài đặt Mail server để máy tính ở trong mạng nội bộ có thể gửi và nhận thư với bên ngoài. Để kết nối với các máy tính bên trong mạng nội bộ, mail server sử dụng địa chỉ 18.0.0.254. Để kết nối ra ngoài Internet, mail server sử dụng địa chỉ 96.29.5.15 thực xác định duy nhất trên Internet.



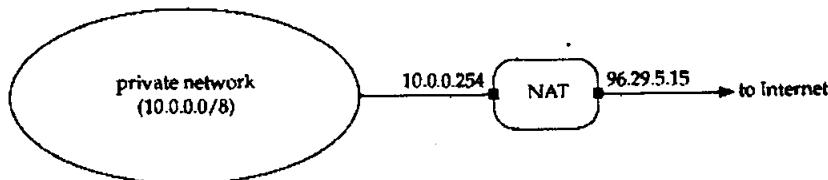
**Hình 4.33. Ví dụ một tổ chức dùng Gateway Mail Server**

Bây giờ giả sử người trong cơ quan muốn gửi email đến ai đó ở MIT. Vấn đề ở đây là địa chỉ thật của MIT là 18.0.0.0/8. Do đó, mail gateway sẽ chuyển ngược thư vào lại mạng nội bộ, chứ không chuyển tới miền mit.edu. Vấn đề này sẽ không còn nữa nếu cơ quan sử dụng địa chỉ riêng.

Một ưu điểm khác của NAT là cho phép cấu hình mạng của cơ quan không phụ thuộc vào bất kỳ ISP nào cả.

Trước khi đi sâu vào chi tiết, chúng ta tìm hiểu qua cơ chế *Dịch chuyển địa chỉ mạng NAT* (Network Address Translation). Cơ chế này được cài đặt trên router nối ra ngoài của mạng. Nếu thay thế mail gateway trên Hình 4.35 bằng một router có hỗ trợ cơ chế NAT và đánh lại địa chỉ cho các máy tính bên trong là địa chỉ 10.0.0.0/8, ta được Hình 4.36. Quay lại ví dụ trên, giả sử máy 10.0.0.1 bên trong mạng riêng muốn gửi email đến địa chỉ 18.7.7.76. Kết nối TCP này không thể được thiết lập do mail server ở MIT không thể gửi lại gói tin phản hồi đến địa chỉ 10.0.0.1.

Về mặt lý thuyết, gói tin gửi từ 10.0.0.1 sẽ đến được MIT server. Nhưng trên thực tế, các router và mail server thường loại bỏ các gói tin có địa chỉ gửi là địa chỉ riêng với mục đích phòng ngừa tấn công từ chối dịch vụ (DoS) hay các thư rác.

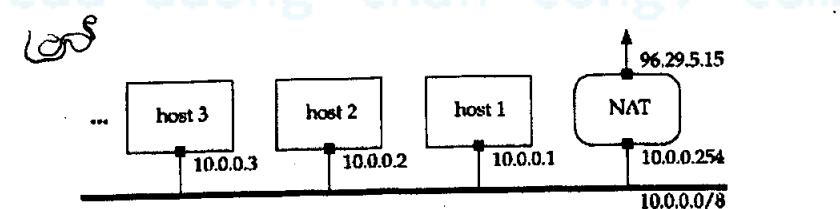


**Hình 4.34.** Mạng riêng với Router có hỗ trợ cơ chế NAT

Chính vì vậy, gói tin trước khi rời mạng riêng, địa chỉ gửi trong gói tin phải được thay đổi để bên kia có thể gửi thông điệp phản hồi.

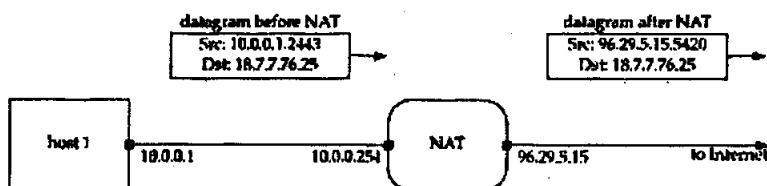
NAT hoạt động theo một trong ba chế độ sau:

- ☞ **Cơ chế tĩnh:** Mỗi máy tính trong mạng riêng khi truy cập đến Internet sẽ có một địa chỉ thật. NAT chỉ thực hiện việc ánh xạ từ địa chỉ ảo vào địa chỉ thực và ngược lại.
- ☞ **Cơ chế vòng:** Tất cả các máy tính trong mạng riêng sử dụng chung một nhóm địa chỉ chính thức. Ví dụ, mạng riêng với 300 máy tính được cấp phát một dải 32 địa chỉ thật. Khi một máy tính trong mạng riêng muốn kết nối với một máy tính bên ngoài, nó sẽ được gán cho một địa chỉ thực tạm thời. Sau khi kết nối kết thúc, địa chỉ vừa được cấp phát sẽ được thu hồi để cấp cho máy tính có nhu cầu khác. Hiển nhiên, khi hết các địa chỉ thật, các máy khác có nhu cầu cũng không được kết nối.
- ☞ **Cơ chế chuyển địa chỉ theo cổng (PAT):** Đây là cơ chế được sử dụng rộng rãi nhất. Cơ chế này được sử dụng khi có một địa chỉ thật duy nhất dùng chung cho cả tổ chức. Khi đó địa chỉ port gửi của mỗi gói tin được thay bằng một giá trị xác định duy nhất. Giá trị này sẽ được sử dụng để khi nhận một gói tin, router biết phải chuyển tiếp nó cho máy tính nào trong mạng riêng. Để hiểu rõ cơ chế hoạt động của PAT, ta minh họa mạng riêng ở Hình 4.36 thành Hình 4.37.



**Hình 4.35.** Ví dụ về mạng sử dụng cơ chế PAT

Giả sử Host 1 muốn thiết lập kết nối với email server của MIT có địa chỉ 18.7.7.76 thông qua proxy 2443. Hình 4.38 minh họa đường đi của gói tin IP chứa TCP SYN segment gửi tới MIT email server qua hai chặng: chặng thứ nhất từ Host 1 đến Router và chặng thứ hai từ router đến MIT server. Cơ chế PAT được sử dụng để biến đổi gói tin này giữa hai chặng

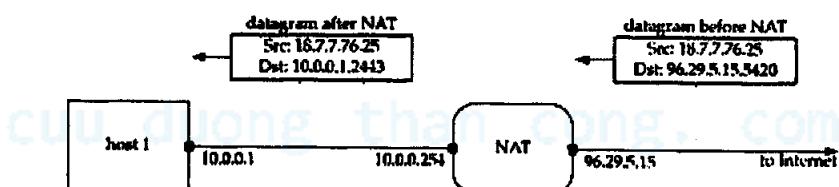


**Hình 4.36.** Gói tin từ Host 1 đến router và được chuyển tiếp

Đầu tiên gói tin được gửi từ Host 1 có địa chỉ gửi là địa chỉ riêng (10.0.0.1) và port gửi là 2443. Sau khi đi qua router, địa chỉ gửi của gói tin là 96.29.5.15 và port gửi là 5420.

MIT's mail server thấy kết nối này đến từ 96.29.5.15:5420 và sẽ gửi thông điệp trả lời đến socket này.

Chúng ta thấy gói tin trả lời trong Hình 4.39. Chú ý rằng, khi đến router, gói tin này có địa chỉ nhận là 96.29.5.15:5420, còn địa chỉ nguồn là phía mail server của MIT. Router sẽ kiểm tra địa chỉ 96.29.5.15:5420 trong bảng PAT và thấy gói tin này phải được chuyển tiếp tới 10.0.0.1:2443. Phía bên trái của Hình 4.39 minh họa gói tin sau khi đã được router biến đổi địa chỉ gửi theo cơ chế PAT.



**Hình 4.37.** Gói tin trả lời được NAT biến đổi

Giả sử tại cùng thời điểm host 1 gửi, một máy tính khác trong mạng riêng, chẳng hạn host 2 với địa chỉ 10.0.0.2 cũng gửi email tới MIT, khi đó NAT sẽ ánh xạ địa chỉ gửi của host 2 vào 96.29.5.15 nhưng sẽ sử dụng một

địa chỉ cổng khác, ví dụ 7322. Do vậy, khi nhận được gói tin có địa chỉ đích 96.29.5.15:7322, NAT biết rằng cần chuyển gói tin này đến host 2 (10.0.0.2).

Tuy nhiên, NAT cũng có nhiều nhược điểm. Trong phần trên thấy rằng, đường như NAT chỉ là sự ánh xạ rất đơn giản giữa địa chỉ riêng và địa chỉ thật. Tuy nhiên, có khá nhiều công việc trong quá trình ánh xạ. Thứ nhất, địa chỉ IP gửi bị thay đổi trong trường tiêu đề nên giá trị trong trường checksum cũng phải tính lại. Thứ hai, router phải kiểm tra xem gói tin chứa TCP segment hay UDP segment, và sau đó phải thay đổi số hiệu cổng và checksum trong tiêu đề của TCP (hay UDP) segment. Như vậy, router hỗ trợ NAT đã "vi phạm" cơ chế phân tầng do đã kiểm tra bên trong gói dữ liệu IP.

Có rất nhiều ứng dụng, chẳng hạn FTP sử dụng nhiều kết nối. FTP client gửi lệnh PORT x y tới FTP server để yêu cầu FTP server mở một kết nối tới socket địa chỉ IP x và cổng y. Địa chỉ và số hiệu cổng này được chuyển dưới dạng mã ASCII bên trong thông điệp ở tầng ứng dụng. Như vậy, NAT phải biết điều này vì hai lý do. Thứ nhất, NAT phải ánh xạ lại địa chỉ riêng của client gửi đến server và NAT phải ghi nhớ (để có thể ánh xạ lại trong trường hợp có xung đột với cổng NAT đã gán khác). Thứ hai, NAT phải xác định chính xác socket (IP/port) của máy tính trong mạng riêng đang đợi kết nối từ phía FTP server ở bên ngoài.

Để thực hiện điều này, NAT router phải kiểm tra gói tin TCP để xem nó có chứa lệnh **PORT** của FTP không. Nếu có, không những phải thay đổi tiêu đề TCP segment mà còn phải thay đổi cả độ dài gói tin (đổi 10.0.0.1 thành 96.29.5.15, khiến độ dài trường dữ liệu trong TCP segment tăng thêm 2 byte). Khi độ dài trường dữ liệu thay đổi, các giá trị biên nhận (ACK) cũng phải thay đổi theo. Chẳng hạn trong ví dụ trên, nếu byte cuối cùng mà client (bên trong mạng riêng) gửi có số thứ tự là n. Tuy nhiên, do NAT router bổ sung thêm dữ liệu ở trong TCP segment nên byte cuối cùng của gói tin này có số thứ tự là  $n + 2$ . Phía server sẽ trả lại giá trị biên nhận ACK là  $n + 3$  (hy vọng byte kế tiếp nhận được có số thứ tự là  $n + 3$ ). Khi đó NAT router phải đổi giá trị biên nhận ACK này thành  $n + 1$  khi chuyển tiếp cho client. Không chỉ thế, NAT router phải ghi nhớ sự biến đổi này trong suốt phiên kết nối.

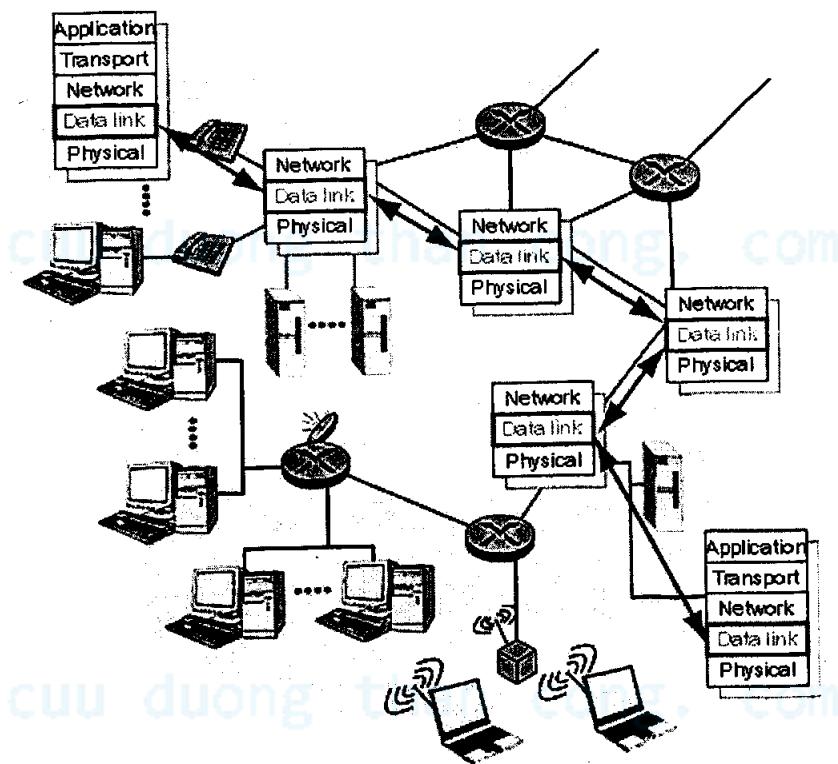
Chế độ thụ động của FTP có thể khắc phục vấn đề này. Trong chế độ này, phía client sẽ khởi tạo cả kết nối điều khiển lẫn kết nối dữ liệu. Khi đó, NAT router không cần biến đổi dữ liệu bên trong gói tin. Khi client muốn mở một kết nối dữ liệu tới server, nó sẽ gửi lệnh PASV yêu cầu server "lắng nghe" trên một cổng nào đó mà client chỉ định.

NAT có thể xử lý với FTP và một số rất ít các giao thức thông dụng khác, nhưng có thể NAT làm hỏng các giao thức ở tầng ứng dụng do người dùng đặt ra nếu những giao thức này đặt địa chỉ cổng bên trong trường dữ liệu ứng dụng.

# Chương 5

## TẦNG LIÊN KẾT DỮ LIỆU

### 5.1. CÁC KHÁI NIỆM CHUNG, DỊCH VỤ CỦA TẦNG DATA LINK



Hình 5.1. Vị trí của tầng liên kết dữ liệu

Ở chương trước chúng ta đã biết tầng mạng cung cấp dịch vụ truyền thông giữa hai máy tính. Ví dụ, trong Hình 5.1 đường truyền thông này bắt đầu từ máy nguồn qua lần lượt các router và kết thúc ở máy đích. Để thuận tiện, ta coi cả máy tính và router là các *nút* (node), vì ở đây nút là router hay

máy tính không phải là vấn đề quan trọng và kênh truyền thông kết nối giữa hai nút liền kề trên toàn bộ đường truyền thông được gọi là *đường liên kết*, hay *đường truyền* (link). Để gói dữ liệu (datagram) đi từ máy tính nguồn tới máy tính đích, gói dữ liệu phải được chuyển trên mỗi đường liên kết. Chương này tập trung vào tầng liên kết dữ liệu với nhiệm vụ truyền gói dữ liệu trên một đường liên kết (đường truyền vật lý). Trước tiên sẽ phân loại và nghiên cứu những dịch vụ của tầng liên kết dữ liệu. Từ mục 5.2 đến mục 5.4 là các nguyên lý quan trọng của những giao thức cung cấp các dịch vụ này (ví dụ, nhận lỗi và sửa lỗi, giao thức đa truy cập được sử dụng để chia sẻ đường truyền vật lý duy nhất giữa các nút và địa chỉ mức liên kết dữ liệu). Chúng ta sẽ thấy nhiều kiểu công nghệ kết nối khác nhau được sử dụng để nối hai nút, từ mục 5.5 đến mục 5.8 sẽ trình bày chi tiết về kiến trúc và giao thức của các kiểu kết nối.

### 5.1.1. Những dịch vụ của tầng liên kết dữ liệu

Giao thức tầng liên kết dữ liệu được sử dụng để truyền gói dữ liệu trên một môi trường vật lý. Giao thức tầng liên kết dữ liệu định nghĩa khuôn dạng đơn vị dữ liệu trao đổi giữa các nút ở mỗi đầu của đường truyền, cũng như những công việc các nút thực hiện khi nhận và gửi những đơn vị dữ liệu này. Trong Chương 1 đã biết rằng, đơn vị dữ liệu của tầng liên kết dữ liệu là frame và mỗi frame tầng liên kết dữ liệu chứa một gói dữ liệu tầng mạng. Công việc của giao thức tầng liên kết dữ liệu khi gửi và nhận frame gồm: phát hiện lỗi, truyền lại, điều khiển lưu lượng và truy cập ngẫu nhiên. Giao thức tầng liên kết dữ liệu gồm: Ethernet, token ring, FDDI và PPP; đôi khi ATM và frame relay có thể cũng được coi là giao thức tầng liên kết dữ liệu.

Nếu nhiệm vụ của tầng mạng là chuyển gói dữ liệu của tầng giao vận từ máy gửi tới máy nhận, thì giao thức của tầng liên kết dữ liệu có nhiệm vụ chuyển gói dữ liệu tầng mạng giữa hai nút kế tiếp trên đường truyền. Một đặc điểm quan trọng của tầng liên kết dữ liệu là gói dữ liệu tầng mạng có thể được xử lý bởi các giao thức liên kết dữ liệu khác nhau trên đường truyền. Ví dụ, gói dữ liệu này có thể được chuyển bởi giao thức Ethernet trên đường truyền đầu tiên, bởi PPP ở đường truyền cuối cùng và frame relay trên các

đường truyền ở giữa. Chú ý, quan trọng là các giao thức liên kết dữ liệu khác nhau có thể cung cấp những dịch vụ khác nhau. Giao thức tầng liên kết dữ liệu không nhất thiết phải cung cấp dịch vụ truyền tin cậy. Vì vậy, tầng mạng phải tính đến khả năng hoạt động trên nhiều kiểu dịch vụ liên kết dữ liệu khác nhau.

Để hiểu rõ quan hệ của tầng liên kết dữ liệu với tầng mạng như thế nào, xét ví dụ sau. Giả sử một đại lý du lịch phải sắp xếp cho khách du lịch đi từ Khách sạn Hilton (Hà Nội) đến Ngọ Môn (Huế). Đầu tiên khách du lịch sẽ đi xe buýt đến sân bay Nội Bài, bay bằng máy bay của Hàng không Việt Nam đến Huế và đi taxi từ sân bay Phú Bài (Huế) đến Ngọ Môn. Sau khi đại lý du lịch tiến hành đặt chỗ, thì chính công ty xe buýt chịu trách nhiệm đưa khách du lịch từ Khách sạn Hilton đến Nội Bài; Hàng không Việt Nam chịu trách nhiệm từ Nội Bài đến sân bay Phú Bài; hãng taxi từ sân bay Huế đến Ngọ Môn. Mỗi một trong ba chặng này được xem là một lần chuyển trực tiếp giữa các "nút" kế tiếp. Rõ ràng, ba chặng đường sẽ được các công ty khác nhau phụ trách. Trong ví dụ trên, có thể xem khách du lịch là datagram (gói tin của tầng mạng), chặng đường tương ứng với một đường truyền vật lý, phương tiện đi lại trên chặng đường là giao thức cho đường truyền vật lý và đại lý du lịch chính là giao thức định tuyến ở tầng mạng.

Dịch vụ cơ bản của bất kỳ tầng liên kết dữ liệu nào là chuyển gói dữ liệu của tầng mạng giữa hai nút kế tiếp, song cụ thể dịch vụ này được thực hiện như thế nào lại phụ thuộc vào giao thức tầng liên kết dữ liệu sử dụng trên đường truyền đó. Nói chung, giao thức tầng liên kết dữ liệu có thể cung cấp những dịch vụ sau:

- ☞ **Đóng gói dữ liệu (frame) và truy cập đường truyền (link access):** Phần lớn các giao thức tầng liên kết dữ liệu đặt gói dữ liệu tầng mạng vào trong gói dữ liệu tầng liên kết dữ liệu (frame) trước khi gửi đi. Frame gồm trường dữ liệu là gói dữ liệu của tầng mạng cùng với một số trường tiêu đề khác. Chú ý, frame có thể có cả trường tiêu đề đầu và cuối (header và trailer). Giao thức tầng liên kết dữ liệu xác định khuôn dạng của frame cũng như giao thức truy cập kênh truyền (cách thức truyền).
- ☞ **Dịch vụ truyền tin cậy:** Nếu cung cấp dịch vụ truyền tin cậy, giao thức tầng liên kết dữ liệu bảo đảm chuyển chính xác gói dữ liệu

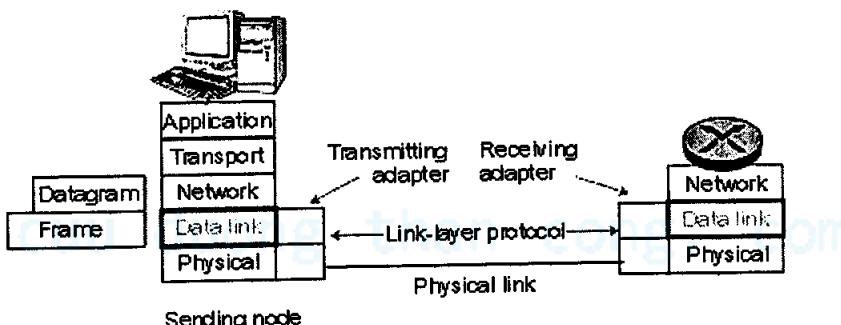
tầng mạng trên một đường truyền. Dịch vụ truyền tin cậy ở tầng liên kết dữ liệu thường được sử dụng trên đường truyền có tỷ lệ lỗi cao (ví dụ, trên đường truyền không dây). Mục đích là sửa lỗi ngay trên đường truyền bị lỗi, chứ không phải truyền lại dữ liệu từ thiết bị gửi tới thiết bị nhận bởi giao thức tầng giao vận hoặc tầng ứng dụng. Tuy nhiên, tầng liên kết dữ liệu không cần cung cấp dịch vụ truyền tin cậy cho các đường truyền ít lỗi (ví dụ cáp quang). Vì vậy, phần lớn các giao thức tầng liên kết dữ liệu phổ biến không cung cấp dịch vụ truyền tin cậy.

- ☞ **Kiểm soát lưu lượng:** Khả năng lưu trữ tạm thời (buffer) các frame tại các nút trên mỗi phía của đường truyền không phải là vô hạn. Đây sẽ là vấn đề khi tốc độ tới của các frame nhanh hơn tốc độ nút nhận có thể xử lý được. Nếu không kiểm soát lưu lượng, bộ đệm phía nhận có thể bị tràn và frame sẽ bị mất. Giống như tầng giao vận, tầng liên kết dữ liệu cung cấp cơ chế kiểm soát lưu lượng để ngăn chặn phía phát gửi quá khả năng nhận của phía thu.
- ☞ **Phát hiện lỗi:** Nút nhận có thể nhận bit 0 trong khi phía gửi gửi bit 1 hay ngược lại. Nguyên nhân bit bị lỗi có thể do tín hiệu bị suy hao hay nhiễu điện từ. Nhiều giao thức tầng liên kết dữ liệu cung cấp cơ chế phát hiện lỗi. Điều này được thực hiện bằng cách phía gửi sẽ thiết lập một số bit phát hiện lỗi trong frame và phía nhận thực hiện việc kiểm tra lỗi. Dịch vụ phát hiện lỗi rất phổ biến trong nhiều giao thức tầng liên kết dữ liệu. Trong Chương 3 và 4 ta thấy tầng giao vận và tầng mạng trên Internet cũng có khả năng phát hiện lỗi. Tuy nhiên, phát hiện lỗi trong tầng liên kết dữ liệu phức tạp hơn rất nhiều, và do vậy thường được triển khai bằng phần cứng.
- ☞ **Sửa lỗi:** Sửa lỗi cũng tương tự phát hiện lỗi. Tuy nhiên, không chỉ có khả năng phát hiện được lỗi mà phía nhận còn có khả năng xác định chính xác vị trí lỗi xuất hiện trong frame (và do đó có thể sửa được những lỗi này). Phát hiện và sửa lỗi được trình bày trong mục 5.2.
- ☞ **Bán song công và song công (Half duplex, full duplex):** Trong chế độ truyền song công, hai phía của đường truyền có thể đồng thời

truyền dữ liệu. Trong chế độ truyền bán song công, tại một thời điểm thiết bị không thể cùng truyền và nhận.

Như đã nói ở trên, nhiều dịch vụ của tầng liên kết dữ liệu giống dịch vụ của tầng giao vận. Chẳng hạn, cả hai tầng đều có dịch vụ truyền tin cậy. Mặc dù cơ chế thực hiện dịch vụ truyền tin cậy ở cả hai tầng giống nhau (xem mục 3.4), nhưng hai dịch vụ truyền tin cậy này không giống nhau. Giao thức tầng giao vận cung cấp dịch vụ truyền tin cậy giữa hai tiến trình trên cơ sở đầu/cuối (end – to – end). Giao thức tầng liên kết dữ liệu cung cấp dịch vụ truyền tin cậy giữa hai nút có một đường truyền vật lý trực tiếp. Tương tự với dịch vụ kiểm soát lưu lượng và phát hiện lỗi.

### 5.1.2. Bộ điều hợp (Adapter)

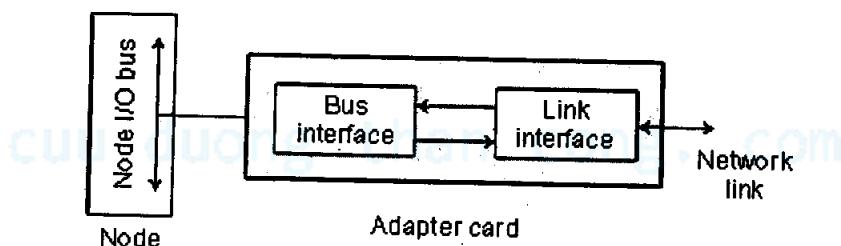


Hình 5.2. Tầng liên kết dữ liệu được triển khai tại adapter

Với phần lớn đường truyền, giao thức tầng liên kết dữ liệu được triển khai trên adapter. Adapter là bo mạch (hoặc card PCMCIA) có RAM, chip DSP, giao diện ghép nối (interface) với bus máy tính và giao diện ghép nối với đường truyền. Adapter cũng thường được coi là card giao tiếp mạng hay gọi tắt là card mạng (NIC – Network Interface Card). Như minh họa trên Hình 5.2, tầng mạng trên nút gửi (máy tính hoặc router) chuyển gói dữ liệu tầng mạng (datagram) xuống adapter để gửi sang phía kia đường truyền. Adapter đặt datagram trong frame, sau đó truyền frame qua đường truyền. Adapter phía bên kia (phía nhận) nhận frame, lấy và chuyển datagram lên tầng mạng. Nếu giao thức tầng liên kết dữ liệu cung cấp dịch vụ phát hiện lỗi thì adapter gửi thêm một số bit phát hiện lỗi và adapter nhận thực hiện việc kiểm tra lỗi. Nếu giao thức tầng liên kết dữ liệu cung cấp dịch vụ truyền tin cậy, thì những kỹ thuật cho dịch vụ truyền tin cậy (ví dụ số thứ tự,

bộ định thời, biên nhận) được cài đặt ngay trên adapter. Nếu giao thức tầng liên kết dữ liệu cung cấp dịch vụ truy cập ngẫu nhiên (mục 5.3), thì giao thức này cũng được triển khai trên adapter.

Adapter là đơn vị bán tự trị. Ví dụ, adapter có thể nhận frame, xác định liệu frame có bị lỗi không và nếu có thì loại bỏ frame mà không thông báo cho thiết bị. Khi nhận được frame đến từ môi trường vật lý (chẳng hạn card mạng), adapter chỉ thông báo cho thiết bị (chính xác là CPU của thiết bị) khi adapter muốn chuyển gói dữ liệu datagram trong frame tới tầng mạng của thiết bị. Việc này sẽ được thực hiện thông qua ngắt phần cứng (interrupt). Tương tự, khi nút gửi gói dữ liệu đến adapter, có thể coi adapter được ủy quyền để chuyển gói dữ liệu sang nút kế tiếp. Mặt khác, adapter không là đơn vị tự trị hoàn toàn. Mặc dù coi adapter là "hộp đen" như minh họa trên Hình 5.3, adapter vẫn nằm trong máy tính, dùng chung nguồn điện và bus, do đó vẫn nằm dưới sự điều khiển của máy tính.



Hình 5.3. Kiến trúc Adapter

Thành phần chính của adapter là giao diện ghép nối bus và giao diện ghép nối đường truyền. Giao diện bus chịu trách nhiệm truyền thông với nút chứa adapter (chính xác hơn là với CPU). Nó truyền dữ liệu cùng thông tin điều khiển giữa nút và card mạng. Giao diện đường truyền (link interface) có trách nhiệm triển khai giao thức tầng liên kết dữ liệu. Bên cạnh chức năng đóng gói (framing) và bóc tách (de-framing) gói dữ liệu, card mạng có thể cung cấp dịch vụ phát hiện lỗi, truy cập ngẫu nhiên và các chức năng khác của tầng liên kết dữ liệu. Nó chứa các mạch truyền và nhận (circuitry). Với những công nghệ phổ biến ở tầng liên kết dữ liệu (ví dụ như Ethernet), giao diện đường truyền được triển khai trên các chip. Chip được sản xuất đại trà và bán rộng rãi, do đó, adapter Ethernet rẻ - thường không quá 10 USD.

## 5.2. KỸ THUẬT PHÁT HIỆN VÀ SỬA LỖI

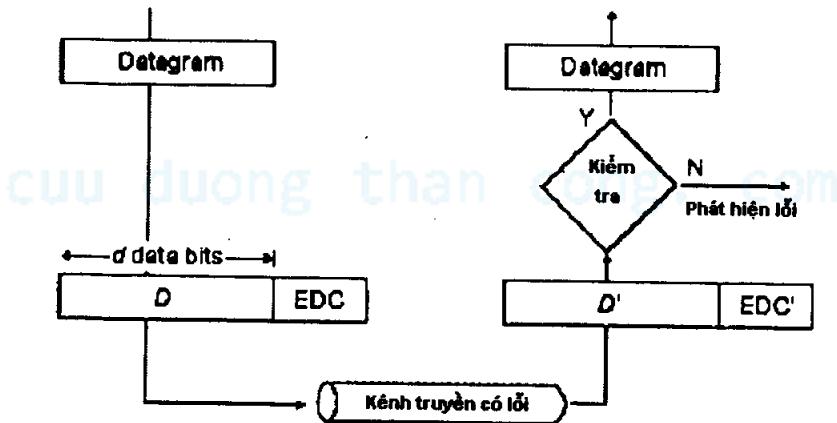
Các đường điện thoại bình thường (truyền tín hiệu tương tự trên đường dây đồng) có tỷ lệ lỗi tương đối cao. Việc thay thế hệ thống này rất khó thực thi vì quá tốn kém. Bên cạnh đó, mạng không dây có tỷ lệ lỗi cao hơn rất nhiều so với mạng có dây. Do vậy, nếu cơ sở truyền thông vẫn dựa trên mạng điện thoại bình thường, cũng như việc mạng không dây ngày càng phổ biến, thì các nhà nghiên cứu vẫn phải nghiên cứu các phương thức để xử lý lỗi trên đường truyền.

Về đặc điểm xuất hiện lỗi, người ta thấy rằng, lỗi có xu hướng xuất hiện theo cụm chứ không phải các lỗi riêng lẻ. Tại sao đặc điểm này quan trọng khi tìm giải pháp phòng chống lỗi? Xét ví dụ: Khi truyền, dữ liệu thường được gửi theo khối. Giả sử kích thước trung bình của khối là 1.000 bit và tỷ lệ lỗi là 0,001. Nếu lỗi xuất hiện độc lập với nhau thì trung bình mỗi khối đều có lỗi. Nhưng nếu lỗi xuất hiện thành cụm 100 bit thì trung bình trong 100 khối sẽ có từ 1 đến 2 khối bị lỗi. Nói chung, lỗi theo cụm khó sửa hơn lỗi riêng lẻ.

Một trong các dịch vụ mà tầng liên kết dữ liệu cung cấp là *phát hiện* (detection) và *sửa* (correct) lỗi ở mức bit – cho phép phát hiện và trong một số trường hợp có thể sửa các bit bị lỗi trong frame của tầng liên kết dữ liệu gửi giữa hai nút kế tiếp. Trong Chương 3 ta thấy dịch vụ phát hiện lỗi cũng được triển khai ở tầng giao vận. Trong phần này, sẽ nghiên cứu một vài kỹ thuật đơn giản nhất trong việc phát hiện và sửa lỗi bit. Ở đây, chỉ trình bày sơ lược, với mục tiêu giúp độc giả hình dung ra một vài kỹ thuật được áp dụng trong một số giao thức tầng liên kết dữ liệu thông dụng. Hình 5.4 minh họa ý tưởng của việc triển khai dịch vụ phát hiện lỗi. Tại nút gửi, dữ liệu D được sử dụng để xác định các bit phát hiện và sửa lỗi EDC. Dữ liệu cần bảo vệ không chỉ gồm gói dữ liệu datagram tầng mạng chuyển xuống mà còn các trường khác trong tiêu đề frame như trường địa chỉ, trường số thứ tự,... Cả D và EDC cùng được gửi đến nút nhận trong cùng một frame. Bên kia sẽ nhận được chuỗi bit là D' và EDC'. Chú ý rằng, D' và EDC' có thể không giống D và EDC vì trên kênh truyền có thể có lỗi.

Phía nhận phải xác định liệu D' có chính là D được phía bên kia gửi hay không trong trường hợp nhận được cả D' và EDC'. Tại phía nhận, kết quả

kiểm tra frame có lỗi không rất quan trọng (Hình 5.4). Chú ý rằng, ở đây xác định có *phát hiện được lỗi* không chứ không phải là *có lỗi hay không*). Kỹ thuật phát hiện và sửa lỗi không phải luôn luôn cho phép phía nhận xác định được có lỗi hay không. Sẽ có những trường hợp sử dụng bit phát hiện lỗi nhưng không có khả năng phát hiện khi xuất hiện lỗi, nghĩa là phía nhận sẽ không biết rằng thông tin nhận được bị lỗi. Hậu quả là phía nhận có thể chuyển gói dữ liệu bị lỗi lên tầng mạng, hoặc không biết rằng nội dung của một vài trường nào đó trong tiêu đề frame bị lỗi. Vì thế, mong muốn lựa chọn được phương pháp có xác suất không phát hiện được lỗi nhỏ nhất có thể. Nói chung, các kỹ thuật phát hiện và sửa lỗi tinh vi (xác suất không phát hiện được lỗi khi có lỗi rất nhỏ) thường đòi hỏi nhiều thời gian tính toán (xác định EDC từ D), số lượng bit dư thừa lớn (EDC lớn) và tại phía nhận việc kiểm tra mất nhiều thời gian.



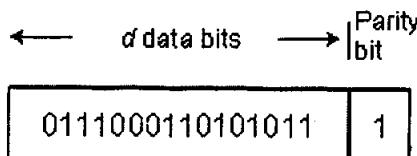
Hình 5.4. Kiểm tra và phát hiện lỗi

Bây giờ xem xét một số kỹ thuật phát hiện lỗi đơn giản như bit chẵn – lẻ (để minh họa ý tưởng của công việc); tính tổng (checksum) (được sử dụng ở tầng giao vận) và bit dư thừa vòng (được sử dụng ở tầng liên kết dữ liệu). Kế tiếp, trình bày về mã Hamming, một phương pháp đơn giản nhưng có khả năng sửa được lỗi.

### 5.2.1. Kiểm tra tính chẵn lẻ

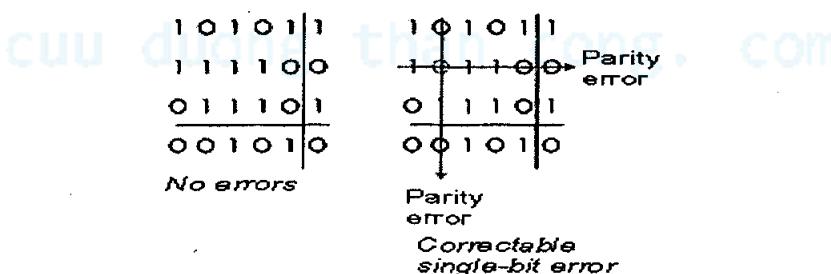
Phương pháp đơn giản nhất để phát hiện lỗi là sử dụng một *bit chẵn – lẻ* (parity bit). Giả sử thông tin D được gửi trong Hình 5.5 có d bit. Nếu sử dụng bit chẵn – lẻ chẵn, bên gửi bổ sung một bit và giá trị bit này được chọn

sao cho tổng số số bit 1 trong  $d + 1$  bit (d bit thông tin gốc D và 1 bit chẵn – lẻ) là chẵn. Với bit chẵn – lẻ lẻ, giá trị bit chẵn – lẻ được chọn sao cho tổng số số bit 1 là một số lẻ. Hình 5.5 minh họa bit chẵn – lẻ chẵn và bit chẵn – lẻ (EDC) được đặt trong một trường khác với trường dữ liệu.



Hình 5.5. Kiểm tra tính chẵn – lẻ

				Row parity
$d_{1,1}$	$\dots$	$d_{1,j}$	$d_{1,j+1}$	
$d_{2,1}$	$\dots$	$d_{2,j}$	$d_{2,j+1}$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$d_{i,1}$	$\dots$	$d_{i,j}$	$d_{i,j+1}$	
$d_{i+1,1}$	$\dots$	$d_{i+1,j}$	$d_{i+1,j+1}$	



Hình 5.6. Kiểm tra tính chẵn – lẻ hai chiều

Việc kiểm tra tại phía nhận cũng tương đối đơn giản. Phía nhận chỉ cần đếm số bit 1 trong  $d + 1$  bit nhận được. Nếu tổng số đếm được là một số lẻ trong khi sử dụng số chẵn – lẻ chẵn thì phía nhận biết rằng ít nhất đã xuất hiện một bit bị lỗi. Chính xác hơn, tổng số các bit bị lỗi là một số lẻ.

Điều gì sẽ xảy nếu số bit bị lỗi là một số chẵn? Trong trường hợp này, rõ ràng bên nhận không phát hiện được lỗi. Nếu xác suất bit bị lỗi thấp và giả định các bit bị lỗi xuất hiện độc lập với nhau, thì khả năng một gói dữ liệu có nhiều bit bị lỗi sẽ cực kỳ thấp. Trong trường hợp này, có thể chỉ cần sử dụng một bit chẵn – lẻ. Tuy nhiên, các thực nghiệm đã chỉ ra rằng, lỗi không xuất hiện độc lập mà thường theo từng "cụm" (burst). Khi đó xác suất

lỗi không bị phát hiện trong frame sử dụng một bit chẵn – lẻ là 50%. Rõ ràng, cần phải có phương pháp khác mạnh hơn. Nhưng trước khi nghiên cứu các phương pháp phát hiện lỗi được sử dụng trong thực tế, xét một phương pháp khác dựa trên bit chẵn – lẻ có khả năng sửa được lỗi.

Hình 5.6 minh họa việc cải tiến phương pháp bit chẵn – lẻ thông qua mảng hai chiều. Ở đây  $d$  bit trong  $D$  được sắp xếp vào bảng  $i$  dòng và  $j$  cột. Sau đó, bên gửi xác định giá trị bit chẵn – lẻ cho tất cả các dòng và cột.  $(i + j + 1)$  bit chẵn – lẻ được tạo ra này sẽ là các bit phát hiện và sửa lỗi của frame.

Bây giờ giả sử rằng, trên đường truyền có một bit duy nhất trong  $d$  bit thông tin gốc bị lỗi. Với phương pháp chẵn – lẻ hai chiều, sẽ xuất hiện mâu thuẫn trong cả hàng và cột chứa bit bị lỗi. Khi đó, phía nhận không những chỉ phát hiện có lỗi mà còn có thể xác định được vị trí bit bị lỗi (và do đó sửa được) thông qua chỉ số hàng và cột của bit đó.

Trên Hình 5.6 chỉ ra ví dụ, bit có giá trị 1 tại vị trí  $(1, 1)$  bị lỗi và bị chuyển thành 0, lỗi mà sẽ được phía nhận phát hiện và sửa lại. Mặc dù, ở đây chỉ quan tâm đến  $d$  bit thông tin, nhưng một bit lỗi trong các bit chẵn – lẻ kiểm tra cũng sẽ bị phát hiện. Phương pháp này cũng có thể phát hiện được (nhưng không sửa được) khi có 2 bit bị lỗi trong gói dữ liệu.

Khả năng phía nhận vừa phát hiện vừa sửa được lỗi được gọi là FEC (Forward Error Correction). Những kỹ thuật này thường được sử dụng phổ biến trong các thiết bị lưu trữ âm thanh như đĩa CD nhạc. Trong môi trường mạng, kỹ thuật FEC có thể được sử dụng một mình, hoặc cùng với kỹ thuật ARQ đã được trình bày trong Chương 3. Ưu điểm của kỹ thuật FEC là cho phép phía gửi không phải truyền lại (do phía nhận sửa được các thông tin bị lỗi). Có lẽ quan trọng hơn là phía nhận sửa được lỗi ngay khi nhận được. Điều này tránh thời gian đợi khi phía gửi phải truyền lại gói dữ liệu. Ưu điểm này được áp dụng trong các ứng dụng thời gian thực.

### **5.2.2. Phương pháp tính tổng kiểm tra (checksum)**

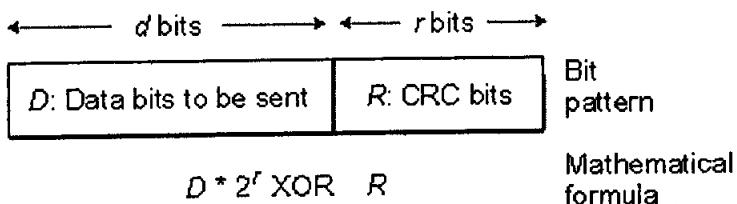
Trong kỹ thuật checksum,  $d$  bit dữ liệu trong Hình 5.4 được xem là dãy liên tiếp các số nguyên có độ dài  $k$  bit. Trong kỹ thuật checksum đơn giản,

người ta tính tổng tất cả các số nguyên k bit này và sử dụng kết quả tính được làm các bit phát hiện lỗi. Phương pháp Internet checksum dựa trên hướng tiếp cận này – luồng dữ liệu được coi là dãy liên tiếp các số nguyên 16 bit và Internet checksum là giá trị bù một của tổng các số nguyên 16 bit. Như đã trình bày trong mục 3.3.2, phía nhận tính lại checksum trên dữ liệu nhận được và kiểm tra xem nó có trùng với checksum trong gói dữ liệu nhận được hay không. RFC 1071 nêu chi tiết thuật toán Internet checksum cũng như phương thức triển khai. Trong giao thức TCP/IP, Internet checksum được tính toán trên tất cả các trường (kể cả trường tiêu đề và dữ liệu). Trong những giao thức khác, ví dụ như XTP [Strayer 1992], có cả checksum cho các trường tiêu đề và checksum cho toàn bộ gói dữ liệu.

McAuley [McAuley 1994] và Feldmeier [Feldmeier 1995] mô tả cách triển khai bằng phần mềm việc tính toán checksum.

### 5.2.3. Kiểm tra dư thừa vòng (CRC)

Kỹ thuật phát hiện lỗi được sử dụng rộng rãi trên mạng máy tính ngày nay dựa trên mã CRC. Mã CRC còn được gọi là *mã đa thức*, vì có thể xem dãy các bit được gửi như một đa thức với hệ số nhận giá trị 0 hoặc 1 và thao tác trên dãy bit này giống như thực hiện phép toán trên đa thức.



Hình 5.7. Mã dư thừa vòng

Mã CRC hoạt động như sau: Giả sử phía gửi muốn gửi d bit dữ liệu D. Đầu tiên hai bên phải thống nhất trước *đa thức sinh* (generator) ký hiệu là G có  $(r + 1)$  bit. Bit có trọng số cao nhất của G phải nhận giá trị 1. Ý tưởng chính của mã CRC được minh họa trên Hình 5.7. Căn cứ vào dữ liệu nguyên thuỷ D, bên gửi sẽ xác định dữ liệu dư thừa R gồm r bit. Sau đó, ghép R với D thu được  $(d + r)$  bit. R được chọn sao cho đa thức ứng với  $(d + r)$  bit này chia hết cho G. Phép chia ở đây được thực hiện theo module 2. Phía nhận thực hiện quá trình kiểm tra lỗi CRC khá đơn giản – chia  $d + r$  bit nhận

được cho G. Nếu phần dư khác 0 thì phía nhận xác định xuất hiện lỗi, nếu không dữ liệu được chấp nhận là đúng.

Tất cả các tính toán trên CRC được thực hiện theo module 2, nhưng không nhớ trong phép cộng và không mượn trong phép trừ. Điều này có nghĩa là, phép cộng giống phép trừ và cả hai tương đương với việc thực hiện phép XOR trên các toán hạng. Ví dụ:

$$1011 \text{ XOR } 0101 = 1110$$

$$1001 \text{ XOR } 1101 = 0100$$

Tương tự, chúng ta cũng có:

$$1011 - 0101 = 1110$$

$$1001 - 1101 = 0100$$

Phép nhân và phép chia cũng giống như thuật toán trên cơ số 2, nhưng các phép cộng và trừ đều không nhớ. Giống như khi thao tác trên các số nhị phân, phép nhân với  $2^k$  là dịch sang trái k vị trí. Do vậy, với D và R, kết quả  $D \times 2^r \text{ XOR } R$  là d + r bit (minh họa trên Hình 5.7). Chúng ta sẽ sử dụng các đặc điểm đại số của mẫu d + r bit này trong phần trình bày dưới đây.

Vấn đề quan trọng là làm sao bên gửi xác định được R. Muốn xác định R sao cho tồn tại n thoả mãn:

$$D \times 2^r \text{ XOR } R = nG$$

Nghĩa là, muốn chọn R sao cho G chia cho  $D \times 2^r \text{ XOR } R$  không có số dư. Nếu thực hiện phép XOR R (là phép cộng theo module 2 không nhớ) cả hai vế của phương trình trên, ta nhận được:

$$D \times 2^r = nG \text{ XOR } R$$

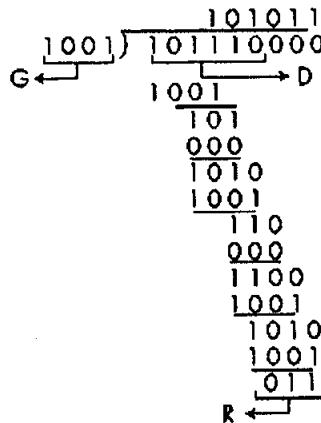
Đẳng thức này cho biết, nếu chia  $D \times 2^r$  cho G, giá trị phần dư chính là R. Nói cách khác, có thể xác định R như sau:

$$R = \text{số dư } D \times 2^r / G$$

Hình 5.8 minh họa kết quả tính toán cho trường hợp  $D = 101110$ ,  $d = 6$ ,  $G = 1001$ ,  $r = 3$ . 9 bit được truyền trong trường hợp này là 101 110 011.

Các chuẩn quốc tế định nghĩa các đa thức sinh (G) 8, 12, 16, 32 bit. CRC 8 bit được sử dụng trong 5 byte tiêu đề của ATM. CRC-32 trong nhiều giao thức sử dụng đa thức sinh sau:

$$G_{\text{CRC-32}} = 100000100110000010001110110110111$$



Hình 5.8. Ví dụ tính toán CRC

Các chuẩn CRC có thể phát hiện lỗi cụm với độ lớn nhỏ hơn  $r + 1$  bit, hay số các bit bị lỗi là một số lẻ. Hơn nữa, với một số giả định nào đó, lỗi cụm lớn hơn  $r + 1$  bit có thể được phát hiện với xác suất  $1 - 0,5^r$ . Lý thuyết mã CRC và thậm chí những mã có tính năng mạnh hơn vượt quá phạm vi của cuốn sách này.

### 5.3. GIAO THỨC ĐA TRUY CẬP VÀ MẠNG CỤC BỘ

Trong phần mở đầu của chương ta thấy rằng, có hai kiểu kết nối mạng là kiểu truyền điểm nối điểm (point – to – point) và kiểu truyền *quảng bá* (broadcast). Trên đường truyền điểm nối điểm có duy nhất một bên gửi và một bên nhận. Nhiều giao thức tầng liên kết dữ liệu được thiết kế cho đường truyền điểm nối điểm như PPP (Point – to – Point Protocol) và HDLC. Kiểu truyền thứ hai – kiểu quảng bá cho phép có nhiều nút gửi và nút nhận cùng kết nối đến kênh truyền dùng chung duy nhất. Khi bất kỳ nút nào đó truyền đi một frame, kênh truyền sẽ quảng bá frame này và tất cả các nút khác đều nhận được bản sao của frame. Ethernet – công nghệ quảng bá được sử dụng rộng rãi nhất sẽ được nghiên cứu chi tiết trong mục 5.5. Trong phần này trình bày một trong những vấn đề quan trọng nhất của tầng liên kết dữ liệu: Làm thế nào để điều phối việc truy cập vào kênh truyền chung của nhiều nút – vấn đề *đa truy cập* (multiple access problem)? Kênh truyền quảng bá thường được sử dụng trên mạng cục bộ – là mạng giới hạn trong một khu vực địa lý.

Người ta đã quá quen thuộc với công nghệ quảng bá trong hệ thống phát thanh truyền hình. Nhưng hệ thống này chỉ phát quảng bá một chiều (nghĩa là một nút cố định – là ăngten – truyền dữ liệu đến nhiều nút nhận), trong khi các nút trên kênh truyền quảng bá trong mạng máy tính có thể vừa gửi vừa nhận. Có thể xét ví dụ tương tự trong một bữa tiệc, mọi người cùng nhau tụ họp trong một đại sảnh (không khí cung cấp môi trường quảng bá) để nói chuyện với nhau. Ví dụ thứ hai là lớp học – nơi giáo viên và sinh viên cùng nhau chia sẻ môi trường quảng bá duy nhất. Vấn đề chính trong cả hai trường hợp này là việc quyết định ai sẽ là người được nói (nghĩa là được truyền trên kênh truyền). Với con người, đã có những quy tắc giao tiếp theo phép lịch sự để chia sẻ kênh truyền chung:

"Mỗi người đều có cơ hội nói"

"Im lặng cho đến khi bạn được quyền nói"

"Không được quyền nói suốt"

"Giơ tay yêu cầu nếu muốn nói"

"Đừng ngắt lời ai đó đang nói"

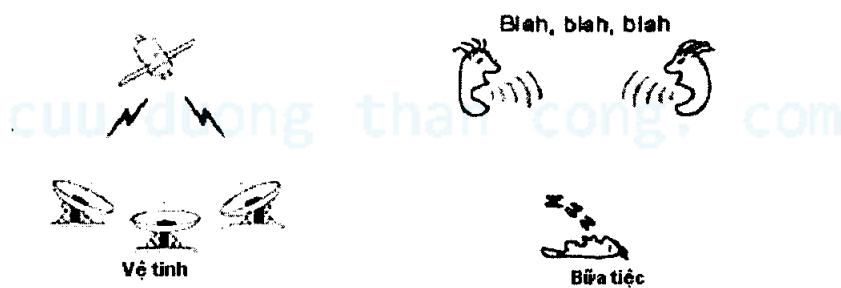
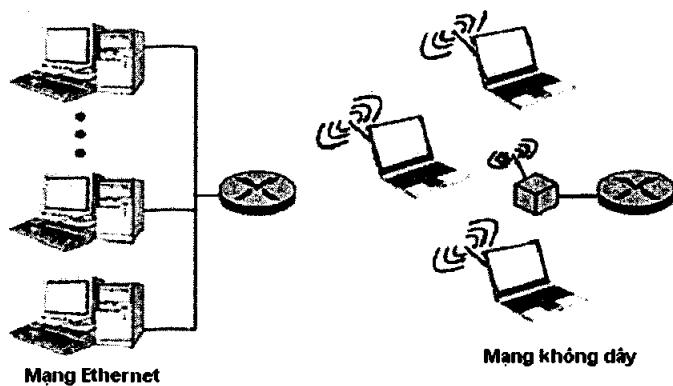
"Đừng ngủ khi ai đó đang nói"

Tương tự như vậy, mạng máy tính cũng có những giao thức – gọi là *giao thức đa truy cập* (multiple access protocol) cho phép các nút điều chỉnh việc truyền thông của mình trên kênh truyền quảng bá dùng chung. Như minh họa trên Hình 5.9, giao thức đa truy cập rất cần thiết trong nhiều kiểu môi trường mạng, như mạng không dây, mạng có dây và mạng vệ tinh.

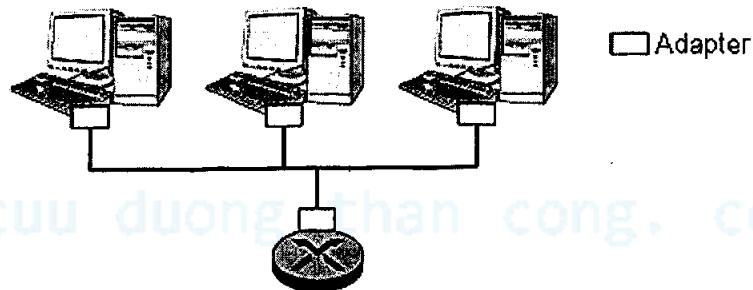
Trên Hình 5.10 là ví dụ một kênh truyền quảng bá chia sẻ kênh truyền dùng chung. Mặc dù mỗi máy tính truy cập kênh truyền qua adapter, trong phần này xem máy tính như một nút đóng vai trò vừa nhận lẫn gửi. Trên thực tế, hàng trăm hoặc thậm chí hàng nghìn nút có thể trực tiếp truyền thông trên một kênh truyền quảng bá.

Vì tất cả các nút đều có khả năng truyền frame, nên rất có thể nhiều nút truyền frame tại cùng một thời điểm. Khi đó, tất cả các nút cùng lúc nhận được nhiều frame, nghĩa là các frame được truyền sẽ *xung đột* (collide) với nhau tại tất cả các nút nhận. Thông thường khi xung đột xảy ra, không nút nào có thể nhận chính xác bất kỳ frame nào, vì tín hiệu trong các frame đan xen vào nhau hoàn toàn. Vì thế, tất cả các frame liên quan đến xung đột đều

bị mất, và có thể coi kênh truyền dùng chung không được sử dụng trong khoảng thời gian xảy ra xung đột. Rõ ràng, khi nhiều nút thường xuyên muốn truyền frame, xác suất xảy ra xung đột sẽ tăng và phần lớn băng thông của kênh truyền bị lãng phí.



Hình 5.9. Chia sẻ kênh truyền dùng chung



Hình 5.10. Mạng Ethernet quảng bá

Để bảo đảm hiệu suất của kênh truyền quảng bá đạt giá trị tối đa khi nhiều nút muốn gửi dữ liệu, bằng cách nào đó phải có cơ chế phối hợp giữa những nút có nhu cầu truyền. Cơ chế phối hợp này chính là trách nhiệm của giao thức đa truy cập. Trong ba mươi năm qua, hàng nghìn bài báo và hàng

trăm luận án tiến sĩ đã nghiên cứu về giao thức đa truy cập. Nhiều kiểu giao thức khác nhau đã được triển khai trên các công nghệ tầng liên kết dữ liệu. Tuy nhiên, người ta có thể phân loại các giao thức đa truy cập vào ba lớp là giao thức *phân chia kênh truyền*, giao thức *truy cập ngẫu nhiên* và giao thức *truy cập lần lượt*. Các lớp này trong sẽ được trình bày lần lượt dưới đây. Trên kênh truyền quảng bá với tốc độ  $R$  bit/s, giao thức đa truy cập lý tưởng sẽ có những đặc điểm sau:

- ☛ Khi chỉ có một nút có dữ liệu gửi đi, nút đó được gửi với thông lượng  $R$  bps.
- ☛ Khi  $M$  nút có dữ liệu gửi đi, mỗi nút được gửi với thông lượng  $R/M$  bps. Yêu cầu này không có nghĩa rằng, mỗi nút trong  $M$  nút luôn luôn truyền với tốc độ tức thời  $R/M$ , mà đây chỉ là tốc độ trung bình xác định trong một khoảng thời gian.
- ☛ Giao thức được triển khai phân tán, nghĩa là không có một nút đóng vai trò điều phối (nếu không toàn bộ hệ thống sẽ sụp đổ nếu nút điều phối bị hỏng).
- ☛ Giao thức phải đơn giản để chi phí cài đặt không cao.

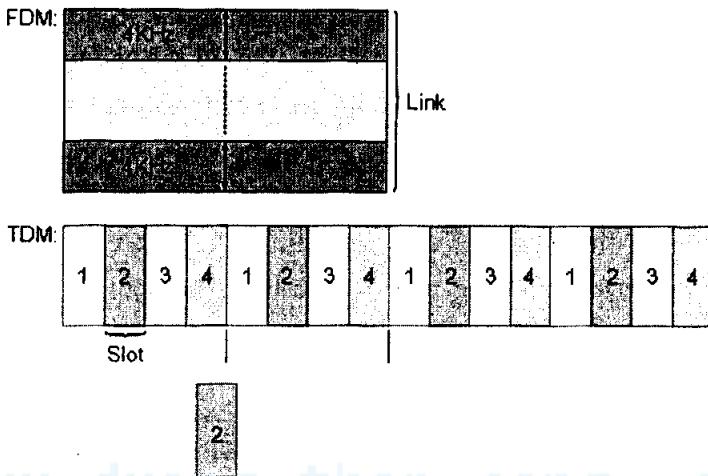
### 5.3.1. Giao thức phân chia kênh truyền (channel partitioning)

Phân kênh theo thời gian (TDM) và theo tần số (FDM) là hai kỹ thuật có thể được sử dụng để phân chia băng thông kênh truyền cho các nút tham gia truyền thông. Giả sử kênh truyền có  $N$  nút và tốc độ truyền của kênh là  $R$  bps. TMD chia thời gian thành các khoảng (time frame) (độc lập với đơn vị dữ liệu frame ở tầng data – link) và sau đó lại chia mỗi khoảng thời gian thành  $N$  khe thời gian (time slot). Mỗi khe thời gian được cấp phát cho một nút. Khi có dữ liệu cần gửi, nút truyền các bit dữ liệu của mình trong khe thời gian đã được cấp phát. Thường khoảng thời gian được chọn sao cho một frame dữ liệu có thể truyền trọn vẹn trong một khe thời gian.

Hình 5.11 minh họa TDM đơn giản cho 4 nút. Trong bữa tiệc, điều này tương tự như quy định mỗi người chỉ được nói trong từng khoảng thời gian quy định. Vì thế ai cũng có cơ hội để nói.

Ưu điểm chính của TDM là loại trừ xung đột và đảm bảo công bằng: mỗi nút có được tốc độ truyền riêng  $R/N$  bps trong mỗi khoảng thời gian.

Tuy nhiên, xuất hiện hai nhược điểm là tốc độ truyền trung bình của mỗi nút bị giới hạn bởi R/N bps và nút chỉ được truyền trong khoảng thời gian của mình ngay cả khi nó là nút duy nhất có nhu cầu gửi (Trong ví dụ bữa tiệc, mọi người đều muốn nghe một vị khách nào đó nói, vị khách đó chỉ được nói trong khoảng thời gian được cấp phát của mình). Rõ ràng, TDM không phải là một giao thức đa truy cập tốt cho bữa tiệc kiểu này.



Hình 5.11. Ví dụ TDM cho 4 nút

Nếu TDM chia kênh truyền theo thời gian, thì FDM chia kênh truyền R bps ra các tần số khác nhau (mỗi tần số có băng thông R/N) và mỗi nút được cấp phát một dải tần số. Vì thế, FDM tạo ra N kênh truyền nhỏ R/N bps từ kênh truyền lớn R bps. Ưu điểm của FDM giống TDM, tức là cũng loại bỏ được xung đột và phân chia công bằng dải tần giữa N nút. Tuy nhiên, nhược điểm là tốc độ gửi của nút bị giới hạn ngay cả khi chỉ có duy nhất một nút có nhu cầu gửi dữ liệu.

Giao thức phân chia kênh truyền thứ ba là *chia mã* (CDMA – Code Division Multiple Access). Nếu TDM và FDM cấp phát khoảng thời gian và tần số cho các nút, thì CDMA cấp phát cho mỗi nút một mã khác nhau. Sau đó nút sử dụng mã duy nhất này để mã hoá dữ liệu gửi đi. CDMA cho phép nhiều nút gửi đồng thời và các nút nhận tương ứng nhận đúng dữ liệu gửi cho mình (miễn là nó biết được mã của nút gửi). CDMA đã được sử dụng trong hệ thống quốc phòng nhờ đặc tính chống nhiễu và bây giờ được áp dụng phổ biến cho mục đích dân sự, đặc biệt trong đa truy cập kênh truyền không dây.

### 5.3.2. Giao thức truy cập ngẫu nhiên (random access)

Kiểu giao thức đa truy cập thứ hai là truy cập ngẫu nhiên. Trong giao thức truy cập ngẫu nhiên, nút truyền luôn luôn truyền dữ liệu với tốc độ cao nhất của kênh truyền R bps. Khi có xung đột, các nút liên quan đến xung đột sẽ truyền lại frame cho đến khi frame đến đích an toàn. Nhưng khi biết có xung đột, nút không cần thiết truyền lại frame ngay lập tức, mà đợi một thời gian ngẫu nhiên nào đó trước khi truyền lại. Mỗi nút liên quan đến xung đột chọn thời gian đợi ngẫu nhiên một cách độc lập, vì thế sau mỗi xung đột xác suất hai nút cùng truyền lại cùng một lúc (lại xảy ra xung đột) sẽ giảm.

Có đến hàng trăm giao thức truy cập ngẫu nhiên, tuy nhiên trong phần này chỉ trình bày một vài giao thức truy cập ngẫu nhiên được sử dụng phổ biến nhất – giao thức ALOHA và giao thức đa truy cập cảm nhận sóng mang (CSMA). Sau đó, trong mục 5.5 sẽ trình bày chi tiết về Ethernet – công nghệ sử dụng CSMA cực kỳ phổ biến.

#### a) Slotted ALOHA

Một trong số những giao thức đa truy cập đơn giản nhất là slotted ALOHA. Giả định như sau:

- ☛ Tất cả frame có chính xác L bit.
- ☛ Thời gian được chia thành các khoảng  $L/R$  s (phải là khoảng thời gian đủ để truyền một frame).
- ☛ Nút bắt đầu truyền frame tại đầu mỗi khoảng thời gian.
- ☛ Tất cả các nút được đồng bộ hóa sao cho mỗi nút đều xác định được khi nào là đầu của khoảng thời gian.
- ☛ Nếu có nhiều frame xung đột trong khoảng thời gian nào đó thì tất cả các nút đều phát hiện sự kiện xung đột ngay trong khoảng thời gian đó.

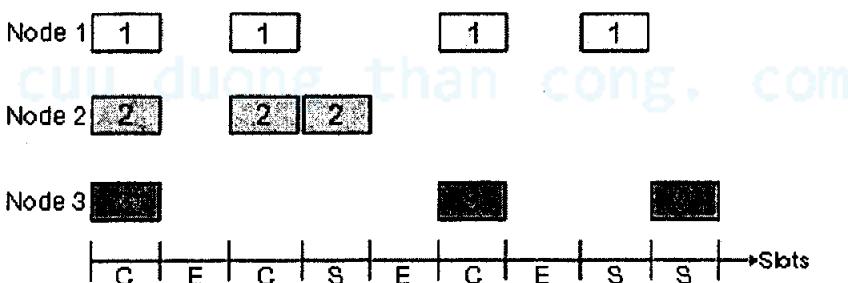
Gọi  $p$  là xác suất gửi ( $0 \leq p \leq 1$ ). Hoạt động của slotted ALOHA trong mỗi nút như sau:

- ☛ Khi có frame mới cần gửi, nút sẽ đợi đến thời điểm đầu của khoảng thời gian kế tiếp và gửi toàn bộ frame trong khoảng thời gian đó.
- ☛ Nếu không xảy ra xung đột, nút truyền thành công frame và vì vậy không cần thiết phải truyền lại (nút có thể chuẩn bị frame mới để truyền, nếu có).

- Nếu có xung đột, nút phát hiện xung đột ngay trong khoảng thời gian và sẽ truyền lại frame trong khoảng thời gian tiếp theo với xác suất  $p$  cho đến khi frame được truyền thành công.

Truyền lại với xác suất  $p$  giống như việc tung đồng xu: biến cố mặt ngửa ứng với việc truyền lại xảy ra với xác suất  $p$ . Biến cố mặt sấp ứng với việc "bỏ qua" khoảng thời gian này và tung lại đồng xu trong khoảng thời gian kế tiếp" xảy ra với xác suất  $(1 - p)$ . Mỗi nút liên quan đến xung đột tung đồng xu độc lập với nhau.

Slotted ALOHA có nhiều ưu điểm. Không giống phân chia kênh truyền, nút tích cực duy nhất (nghĩa là nút có nhu cầu gửi dữ liệu) liên tục gửi frame ở tốc độ cao nhất của kênh truyền. Slotted ALOHA là một thuật toán phân tán, vì mỗi nút khi phát hiện ra xung đột sẽ quyết định khi nào truyền lại một cách độc lập (Tuy nhiên, slotted ALOHA đòi hỏi phải có cơ chế đồng bộ trên tất cả các nút).



Hình 5.12. Ví dụ giao thức Slotted ALOHA

Slotted ALOHA hoạt động tốt khi chỉ có một nút ở trạng thái tích cực. Nhưng hiệu suất của nó bằng bao nhiêu khi có nhiều nút tích cực? Có hai yếu tố phải tính đến ở đây. Thứ nhất, như trong Hình 5.12 khi có nhiều nút ở trạng thái tích cực sẽ xuất hiện nhiều khoảng thời gian xung đột, và do đó kênh truyền bị lãng phí. Thứ hai, sẽ có một số khoảng thời gian "rỗng" vì trong khoảng thời gian này tất cả các nút tích cực đều dừng lại đợi (kết quả của cơ chế truyền theo xác suất). Chỉ trong những khoảng thời gian "không bị lãng phí" sẽ có duy nhất một nút truyền thành công. Khoảng thời gian này gọi là khoảng thời gian thành công. Hiệu suất của giao thức được định nghĩa là tỷ lệ với khoảng thời gian truyền thành công trong trường hợp có nhiều nút tích cực, mỗi nút cần gửi đi nhiều frame. Rõ ràng rằng, nếu không có cơ

chế điều khiển truy cập và nút truyền lại ngay sau mỗi lần xung đột, hiệu suất sẽ bằng 0. Slotted ALOHA có hiệu suất lớn hơn 0, nhưng bao nhiêu?

Bây giờ sẽ xác định hiệu suất tối đa của slotted ALOHA. Để đơn giản, thay đổi giao thức một chút và giả thiết rằng mỗi nút truyền frame trong mỗi khoảng thời gian với xác suất  $p$  (tức là mỗi nút luôn có một frame để gửi đi và frame này được gửi đi với xác suất  $p$  cho dù đây là frame mới hay frame phải gửi lại). Đầu tiên giả sử có  $N$  nút. Xác suất thành công của một khoảng thời gian nào đó là xác suất chỉ có một nút duy nhất truyền và  $N - 1$  nút còn lại không truyền. Xác suất một nút nào đó truyền là  $p$ ; xác suất mà các nút còn lại không truyền là  $(1 - p)^{N - 1}$ . Do vậy, xác suất để một nút nào đó truyền trong khi các nút khác không truyền là  $p(1 - p)^{N - 1}$ . Vì có  $N$  nút, nên xác suất để có khoảng thời gian thành công bằng  $N \times p \times (1 - p)^{N - 1}$ .

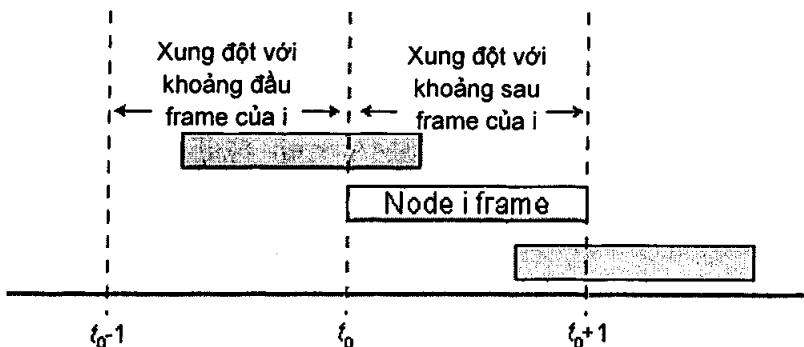
Do đó, khi có  $N$  nút tích cực, hiệu suất của slotted ALOHA là  $N \times p \times (1 - p)^{N - 1}$ . Để đạt được hiệu suất lớn nhất, phải xác định  $p^*$  sao cho biểu thức này đạt giá trị lớn nhất. Và để đạt được hiệu suất lớn nhất khi có nhiều nút tích cực, phải tính giới hạn của  $N \times p^* \times (1 - p^*)^{N - 1}$  khi  $N$  tiến tới vô cùng. Áp dụng các công cụ toán học, sẽ xác định được hiệu suất lớn nhất của giao thức là  $1/e = 0,37$ . Nghĩa là, khi nhiều nút cùng ở trạng thái tích cực, thì trong điều kiện tốt nhất chỉ 37% thời gian đường truyền được sử dụng có ích. Vì vậy, tốc độ truyền hiệu quả của kênh truyền không phải là  $R$  bps mà chỉ là  $0,37R$  bps. Phân tích tương tự chỉ ra rằng, 37% thời gian đường truyền không được sử dụng và 26% thời gian xảy ra xung đột trên đường truyền. Như vậy, một frame nào đó có thể được truyền với tốc độ tối đa  $R$ , nhưng về tổng thể thông lượng truyền thành công của toàn bộ kênh truyền không vượt qua  $0,37R$ .

### b) ALOHA

Giao thức slotted ALOHA đòi hỏi tất cả các nút đồng bộ việc truyền tại đầu mỗi khoảng thời gian. Giao thức ALOHA đầu tiên thực sự là giao thức không chia khoảng thời gian, hoàn toàn phân tán. Trong giao thức này, khi có dữ liệu cần gửi đi, ngay lập tức nút truyền toàn bộ frame vào kênh truyền dùng chung. Nếu frame được truyền xung đột với frame từ các nút khác, thì ngay sau khi truyền cho xong frame, nút sẽ ngay lập tức truyền lại frame với

xác suất  $p$ . Ngược lại, nút đợi trong một khoảng thời gian truyền frame. Sau quá trình chờ đợi, nút truyền frame với xác suất là  $p$ , hoặc đợi (không làm gì cả) trong khoảng thời gian truyền frame với xác suất  $(1 - p)$ .

Để xác định được hiệu suất cực đại của ALOHA, xét một nút duy nhất và cũng giả định như trong trường hợp slotted ALOHA thời gian truyền frame là một đơn vị thời gian. Tại bất kỳ thời gian nào, xác suất để nút truyền frame là  $p$ . Giả sử frame này bắt đầu truyền tại thời điểm  $t_0$ . Như minh họa trong Hình 5.13, để frame này được truyền thành công thì không nút nào được bắt đầu truyền trong khoảng thời gian  $[t_0 - 1, t_0]$ . Nếu không tín hiệu của những frame này sẽ xung đột với các tín hiệu đầu tiên của frame đang xét. Xác suất để tất cả các nút khác không được bắt đầu truyền trong khoảng thời gian này là  $(1 - p)^{N-1}$ . Tương tự, không nút nào được bắt đầu truyền trong khi nút đang xét đang truyền. Xác suất của điều này cũng là  $(1 - p)^{N-1}$ . Vì vậy, xác suất nút nào đó truyền thành công là  $p \times (1 - p)^{2(N-1)}$ . Bằng cách lấy giới hạn như trong trường hợp slotted ALOHA, ta thấy rằng, hiệu suất lớn nhất của giao thức ALOHA là  $1/2e$  – bằng một nửa của slotted ALOHA. Đây là cái giá phải trả cho giao thức ALOHA hoàn toàn phân tán.



Hình 5.13. Các frame đan xen vào nhau trong ALOHA

### c) CSMA – Đa truy cập cảm nhận sóng mang

Trong cả hai giao thức ALOHA và slotted ALOHA, quyết định truyền của nút được đưa ra độc lập với các nút khác. Cụ thể hơn, một nút không để ý tới việc liệu có nút khác đang truyền khi nó bắt đầu truyền hay không và nút cứ truyền kể cả khi có nút khác truyền (gây xung đột). Tương tự trong ví dụ buổi tiệc, giao thức ALOHA giống như hành vi của vị khách bất lịch sự

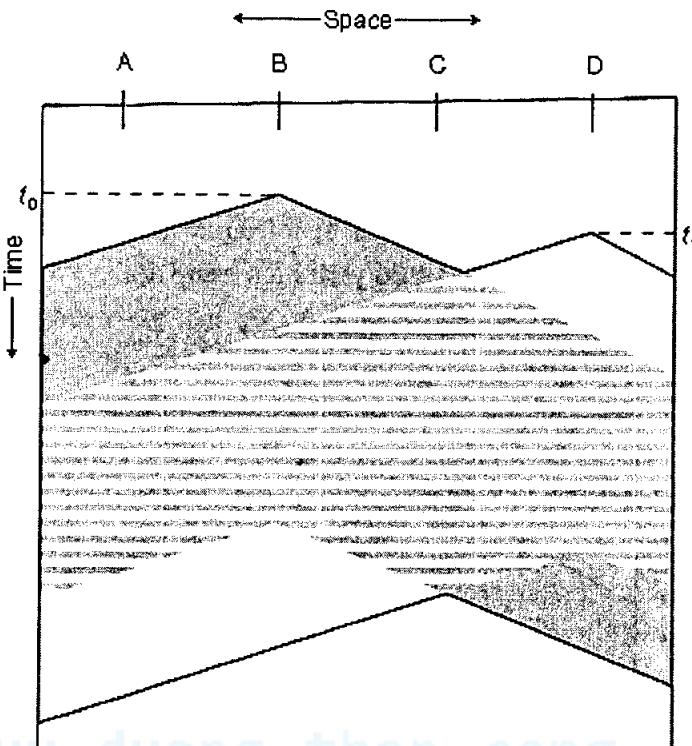
cứ liên tục nói bất chấp việc có người đang nói hay không. Xã hội loài người có những quy tắc ứng xử cho phép xử sự một cách lịch sự và làm giảm "xung đột" giữa nhiều người nói chuyện. Đặc biệt, có hai quy tắc quan trọng cho một cuộc đối thoại của người lịch sự:

- ☛ **Nghe trước khi nói:** Nếu có ai đang nói, hãy đợi đến khi họ nói xong. Trong mạng máy tính, điều này được gọi là *cảm nhận sóng mang* (carrier sense) – nút "lắng nghe" kênh truyền trước khi truyền. Nếu có frame đang được truyền trên kênh truyền thì nút sẽ chờ (backs off) một khoảng thời gian ngẫu nhiên và lại tiếp tục "lắng nghe" kênh truyền. Lúc này nếu kênh truyền được cảm nhận là rỗi thì nút bắt đầu việc truyền frame. Trong trường hợp ngược lại, nút lại đợi một khoảng thời gian ngẫu nhiên khác và lặp lại quá trình này.
- ☛ **Nếu có ai đó bắt đầu nói cùng lúc thì hãy tạm ngừng (Nghe trong khi nói):** Trong mạng máy tính điều này được gọi là *phát hiện xung đột* (collision detection) – nút đang truyền vẫn phải tiếp tục "lắng nghe" kênh truyền trong khi đang truyền. Nếu phát hiện có nút khác truyền xen vào, nút sẽ dừng truyền và sử dụng giao thức nào đó để quyết định khi nào nên thử truyền tiếp.

Hai quy tắc này là ý tưởng chủ đạo của giao thức CSMA (Carrier Sense Multiple Access) và CSMA/CD (CSMA with Collision Detection). Có nhiều biến thể của CDMA và CDMA/CD với việc thực hiện các chiến lược truyền lại khác nhau. CDMA/CD – sử dụng trong mạng Ethernet sẽ được trình bày chi tiết trong mục 5.5. Ở đây, chỉ xét một số đặc trưng cơ bản quan trọng nhất của CSMA và CSMA/CD.

Nếu tất cả các nút thực hiện cảm nhận sóng mang thì tại sao xung đột có khả năng xuất hiện? Xét cho cùng, một nút sẽ "tự kiềm chế" không truyền khi nó cảm thấy nút khác đang truyền. Vấn đề này được giải quyết bằng biểu đồ thời gian.

Hình 5.14 minh họa biểu đồ thời gian của 4 nút (A, B, C và D) với bus dùng chung. Trục hoành biểu thị vị trí của nút trong không gian, trục tung mô tả thời gian.



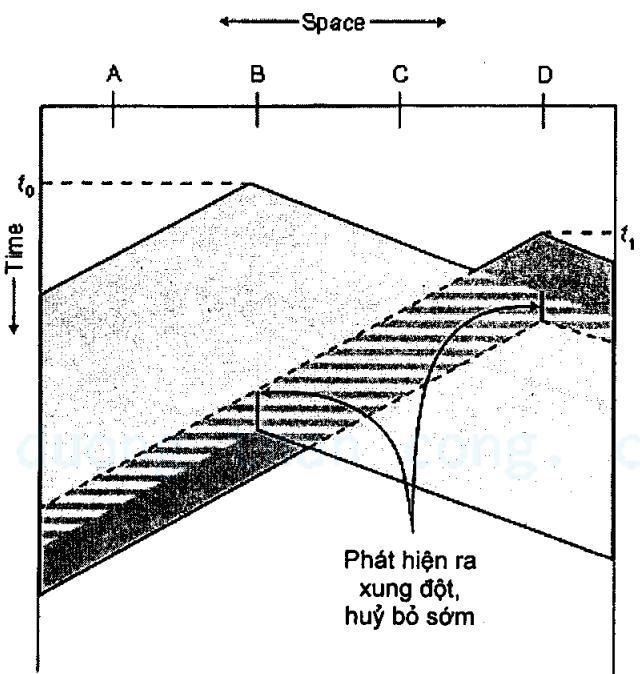
Hình 5.14. Biểu đồ thời gian của 4 nút sử dụng CSMA

Tại thời điểm  $t_0$ , nút B nhận thấy kênh truyền rõ (lúc này không có nút nào đang truyền). Do đó nút B bắt đầu truyền và tín hiệu do B truyền lan toả theo cả hai hướng của môi trường dùng chung. Trong thực tế, cho dù vận tốc truyền xấp xỉ tốc độ ánh sáng thì tín hiệu từ B lan truyền đến một điểm nào đó cũng phải mất một khoảng thời gian. Tại thời điểm  $t_1$  ( $t_1 > t_0$ ) nút D có nhu cầu gửi dữ liệu. Mặc dù tại thời điểm  $t_1$ , B đang truyền, song các tín hiệu từ B chưa lan toả tới D và vì vậy D cảm thấy kênh truyền rõ vào thời điểm  $t_1$ . Do đó, theo đúng giao thức CDMA, D bắt đầu truyền dữ liệu. Ngay sau đó, tín hiệu từ B xung đột với tín hiệu từ D. Hiển nhiên rằng, độ trễ lan toả (propagation delay) giữa hai đầu mút của kênh truyền dùng chung – thời gian để tín hiệu lan truyền từ đầu này đến đầu kia kênh truyền đóng vai trò quyết định trong hiệu suất hoạt động của kênh truyền. Thời gian trễ này càng lớn, xác suất một nút không phát hiện được có nút khác đang truyền cũng càng lớn.

Trong Hình 5.14 các nút không thực hiện phát hiện xung đột, cả nút B và D tiếp tục truyền toàn bộ frame ngay cả khi có xung đột. Nếu thực hiện

công việc phát hiện xung đột, nút sẽ ngừng truyền ngay khi phát hiện xung đột.

Hình 5.15 tương tự như Hình 5.14, chỉ khác là cả hai nút ngừng truyền ngay sau khi phát hiện có xung đột. Rõ ràng đưa khả năng phát hiện xung đột vào giao thức đa truy cập sẽ làm tăng hiệu suất của giao thức do các nút không cố gắng tiếp tục gửi frame đã bị lỗi. Giao thức Ethernet trình bày trong mục 5.5 là giao thức CSMA có phát hiện xung đột.



Hình 5.15. CSMA có phát hiện xung đột

### 5.3.3. Giao thức truy cập lần lượt (Taking – turns)

Hai tính chất mà tất cả các giao thức đa truy cập muốn có là (1) khi chỉ có một nút tích cực, nút này có thể chiếm toàn bộ đường truyền, nghĩa là truyền với băng thông tối đa  $R$  bps; (2) khi  $M$  nút tích cực, mỗi nút có băng thông trung bình  $R/M$  bps.

Giao thức ALOHA và CSMA có tính chất đầu tiên, nhưng không có tính chất thứ hai. Điều này đã thúc đẩy các nhà nghiên cứu xây dựng một lớp các giao thức khác – kiểu lần lượt. Giống như kiểu truy cập ngẫu nhiên, có rất nhiều giao thức kiểu lần lượt, và mỗi giao thức này lại có nhiều biến thể. Ở đây, sẽ trình bày hai giao thức quan trọng nhất. Đầu tiên là *kiểu hỏi*

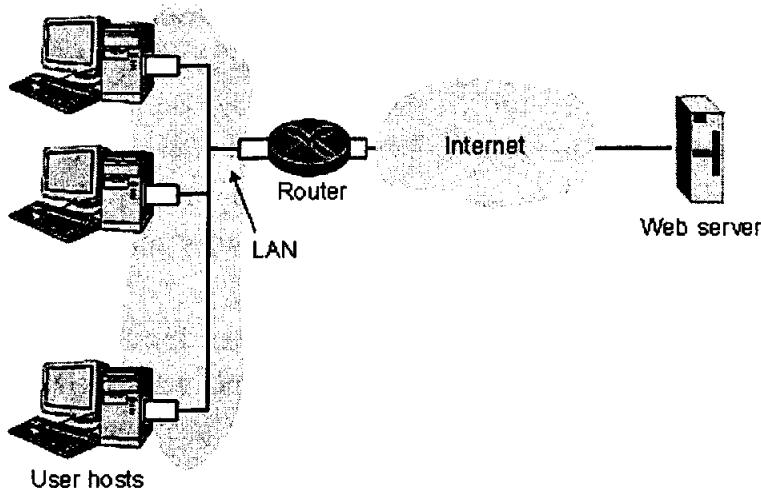
vòng (polling). Với kiểu giao thức này, một nút được chọn đóng vai trò điều phối. Nút điều phối lần lượt hỏi từng nút theo thứ tự vòng tròn. Đầu tiên nút điều phối gửi thông điệp tới nút thứ nhất, thông báo nút thứ nhất có thể truyền một lượng dữ liệu nào đó. Sau khi nút thứ nhất truyền, nút điều phối thông báo cho phép nút thứ hai có thể truyền một lượng dữ liệu nào đó,... Nút điều phối có thể xác định nút nào đó kết thúc quá trình dữ liệu khi không có tín hiệu lan truyền trên kênh truyền.

Giao thức hỏi vòng loại trừ sự xung đột hay các khoảng thời gian không được sử dụng như trong kiểu giao thức truy cập ngẫu nhiên. Điều này khiến hiệu suất của nó cao hơn nhiều. Nhưng không phải giao thức hỏi vòng không có nhược điểm. Nhược điểm trước tiên là có độ trễ vòng – lượng thời gian cần thiết để nút điều phối báo cho nút nào đó có thể truyền. Ví dụ, nếu chỉ có một nút tích cực thì nút sẽ truyền với tốc độ nhỏ hơn R bps, vì nút điều phối phải lần lượt hỏi vòng tất cả các nút, trong mỗi vòng nút tích cực chỉ được phép gửi một lượng dữ liệu hạn chế. Nhược điểm thứ hai nghiêm trọng hơn nhiều, là nếu nút điều phối gặp sự cố thì toàn bộ hệ thống cũng bị sụp đổ.

Kiểu giao thức thứ hai là giao thức *thẻ bài* (token – passing). Trong giao thức này không có nút điều phối. Một frame đặc biệt được gọi là *thẻ bài* (token) được trao đổi giữa các nút theo một thứ tự định trước. Ví dụ, nút thứ nhất gửi thẻ bài tới nút thứ hai, nút thứ hai gửi thẻ bài tới nút thứ ba,... nút thứ N gửi thẻ bài tới nút thứ nhất. Khi nút nhận được thẻ bài, nó chỉ giữ thẻ bài khi có dữ liệu cần truyền, nếu không nó sẽ ngay lập tức chuyển thẻ bài tới nút kế tiếp. Nếu nút có frame để truyền, khi nhận được thẻ bài, nó gửi đi lượng dữ liệu được phép và sau đó chuyển thẻ bài tới nút kế tiếp. Giao thức thẻ bài được triển khai phân tán và có hiệu suất cao. Nhưng nó cũng có nhiều vấn đề cần giải quyết. Ví dụ, một nút gặp sự cố có thể làm toàn bộ hệ thống sụp đổ. Hoặc nếu một nút tinh cờ không chuyển tiếp hay làm mất thẻ bài thì cần có cơ chế đưa thẻ bài mới vào lưu thông.

### 5.3.4. Mạng cục bộ LAN (Local Area Network)

Những giao thức đa truy cập được sử dụng trên nhiều loại kênh truyền quảng bá khác nhau. Chúng được sử dụng cho kênh truyền vệ tinh hay môi trường không dây (các nút truyền trên cùng một dải tần số).



**Hình 5.16.** Máy tính người dùng truy cập máy dịch vụ Web Internet thông qua LAN. Kênh truyền quảng bá giữa máy tính người dùng và router gồm một "đường truyền"

Mạng cục bộ LAN là mạng máy tính giới hạn trong một khu vực địa lý, ví dụ trong một toà nhà hoặc trong khuôn viên trường đại học. Thông thường, khi truy cập Internet từ trường đại học hay cơ quan, hầu hết mọi người truy cập thông qua mạng LAN. Khi đó máy tính của người dùng là một nút trong mạng LAN và mạng LAN cung cấp khả năng truy cập tới Internet thông qua router, như minh họa trong Hình 5.16. Mạng LAN là kênh truyền duy nhất giữa tất cả các máy tính và router; do đó nó cần tới giao thức tầng liên kết dữ liệu và giao thức đa truy cập. Tốc độ truyền R của hầu hết các mạng LAN rất cao. Tốc độ mạng LAN trước năm 1980 là 10Mbps, ngày nay là 100Mbps và trong tương lai sẽ là 1Gbps.

Vào những năm 80 đầu những năm 90 của thế kỷ XX, có hai kiểu công nghệ mạng LAN phổ biến trên thị trường. Công nghệ thứ nhất là mạng cục bộ Ethernet (được biết đến là 802.3 LAN [IEEE 802.3]) sử dụng giao thức truy cập ngẫu nhiên. Công nghệ thứ hai dựa trên công nghệ thẻ bài gồm công nghệ token – ring và FDDI. Công nghệ Ethernet được trình bày chi tiết trong mục 5.4, vì đây là công nghệ chủ đạo ngày nay.

Trong mạng token – ring, N nút của mạng (máy tính hoặc router) được kết nối vào vòng (ring) bằng đường truyền trực tiếp. Topo mạng xác định thứ tự chuyển thẻ bài. Khi nút nhận được thẻ bài và có nhu cầu gửi dữ liệu, dữ liệu (frame) được gửi đi sẽ lan toả trên toàn bộ vòng. Nút gửi sẽ chịu

trách nhiệm loại bỏ frame trên vòng. FDDI được thiết kế cho mạng nội bộ cho một khu vực lớn (vài km<sup>2</sup>). Do vậy, sẽ không hiệu quả nếu frame phải lan toả ngược lại phía gửi sau khi đã đến đích. Trong công nghệ FDDI, chính nút nhận phải loại bỏ frame ra khỏi vòng (thực sự FDDI không phải là kênh truyền quảng bá thuần tuý vì không phải nút nào cũng nhận được bất kỳ frame nào được truyền). Có thể đọc thêm về công nghệ token – ring và FDDI trong [3Com 1999].

## 5.4. ĐỊA CHỈ LAN VÀ ARP

Như đã nói trong phần trước, các nút trong mạng LAN gửi frame cho nhau trên kênh truyền quảng bá dùng chung. Điều này nghĩa là, khi một nút trong mạng LAN truyền frame, mọi nút khác kết nối tới mạng LAN đều có khả năng nhận được frame. Tuy nhiên, mỗi nút trong mạng LAN không muốn gửi frame tới tất cả các nút khác mà chỉ muốn gửi tới một nút cụ thể nào đó trong mạng LAN. Để thực hiện điều này, các nút trong mạng LAN phải có khả năng xác định địa chỉ của nhau khi gửi frame, nghĩa là mỗi nút cần có một địa chỉ trong mạng LAN và trong gói tin tầng liên kết dữ liệu (frame) cần có trường chứa địa chỉ nút đích. Như vậy, khi nhận được frame, nút có thể xác định liệu frame đó có phải gửi cho mình không.

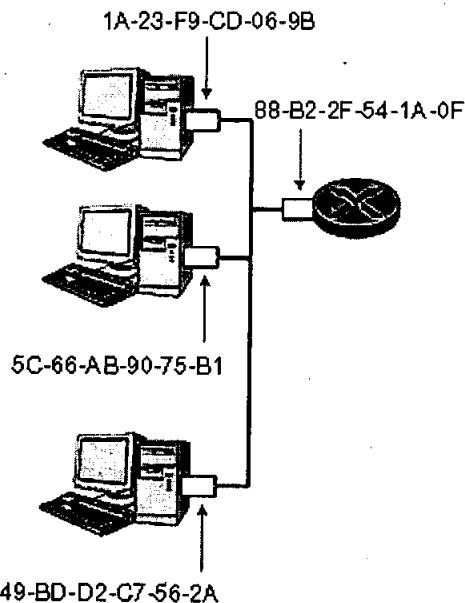
Nếu địa chỉ đích của frame trùng với địa chỉ LAN của mình, nút nhận sẽ lấy gói dữ liệu tầng mạng từ frame tầng liên kết dữ liệu và chuyển gói dữ liệu này lên tầng mạng phía trên.

Nếu địa chỉ đích không trùng với địa chỉ nút nhận, đơn giản nút sẽ loại bỏ frame nhận được.

### 5.4.1. Địa chỉ LAN

Thực sự, không phải nút (máy tính) có địa chỉ LAN mà chính adapter mới có địa chỉ LAN. Điều này được minh họa trong Hình 5.17. Địa chỉ LAN có nhiều tên gọi khác nhau như: địa chỉ vật lý (physical address), địa chỉ Ethernet, địa chỉ MAC (media access control). Với hầu hết các mạng LAN (kể cả mạng Ethernet và thẻ bài), địa chỉ LAN có độ dài 6 byte (có thể có  $2^{48}$  địa chỉ). 6 byte địa chỉ này được biểu diễn dưới dạng thập lục phân (co số 16), mỗi byte ứng với một cặp số thập lục phân. Địa chỉ mạng LAN

của adapter mang giá trị cố định, được ghi cứng vào ROM của adapter trong quá trình sản xuất.



Hình 5.17. Mỗi adapter có một địa chỉ LAN duy nhất

Một điểm thú vị là hai adapter bất kỳ có địa chỉ LAN khác nhau. Thoạt tiên điều này dường như khó có thể thực hiện được, vì nhiều công ty khác nhau có thể sản xuất adapter. Làm thế nào để địa chỉ adapter sản xuất ở Đài Loan khác với địa chỉ adapter sản xuất tại Bi? Điều này có được là do IEEE quản lý không gian địa chỉ vật lý. Khi muốn sản xuất adapter, công ty phải mua một phần không gian địa chỉ gồm  $2^{24}$  địa chỉ với một mức phí nào đó. IEEE cấp từng khối  $2^{24}$  địa chỉ bằng cách cố định 24 bit đầu của địa chỉ vật lý và công ty có thể tự ý gán 24 bit sau cho bất kỳ sản phẩm nào của mình.

Địa chỉ vật lý của adapter có cấu trúc phẳng (đối lập với cấu trúc phân cấp) và không thay đổi cho dù có mang adapter đi đâu chăng nữa. Mỗi máy tính xách tay với một card Ethernet luôn có cùng một địa chỉ vật lý, cho dù ở bất kỳ nơi đâu. Điều này ngược với địa chỉ IP có cấu trúc phân cấp (gồm địa chỉ mạng và địa chỉ máy tính). Địa chỉ IP của nút sẽ thay đổi khi chuyển sang mạng khác. Địa chỉ vật lý của adapter giống như số chứng minh thư nhân dân của một người – không phân cấp và luôn luôn không thay đổi bất kể người đó ở đâu.

Như đã trình bày, khi adapter muốn gửi frame đến adapter nào đó nằm trên cùng mạng LAN, adapter gửi sẽ đặt địa chỉ nhận vào trong frame. Khi nhận được frame, adapter đích loại bỏ các tiêu đề của frame và gửi dữ liệu lên tầng phía trên. Tất cả các adapter khác trong LAN cũng đều nhận được frame. Tuy nhiên, các adapter này sẽ loại bỏ frame (không gửi gói dữ liệu lên trên). Như vậy, các adapter sẽ không làm gián đoạn CPU trung tâm khi chúng nhận được dữ liệu dành cho nút khác. Tuy nhiên, đôi khi adapter muốn gửi tới tất cả các adapter khác trên mạng LAN. Khi đó, adapter sử dụng một địa chỉ đặc biệt là địa chỉ quảng bá trong frame gửi đi. Đối với mạng cục bộ sử dụng địa chỉ 6 byte (như là LAN Ethernet và token-passing) thì địa chỉ quảng bá là chuỗi 48 bit 1 (FF–FF–FF–FF–FF–FF trong hệ 16).

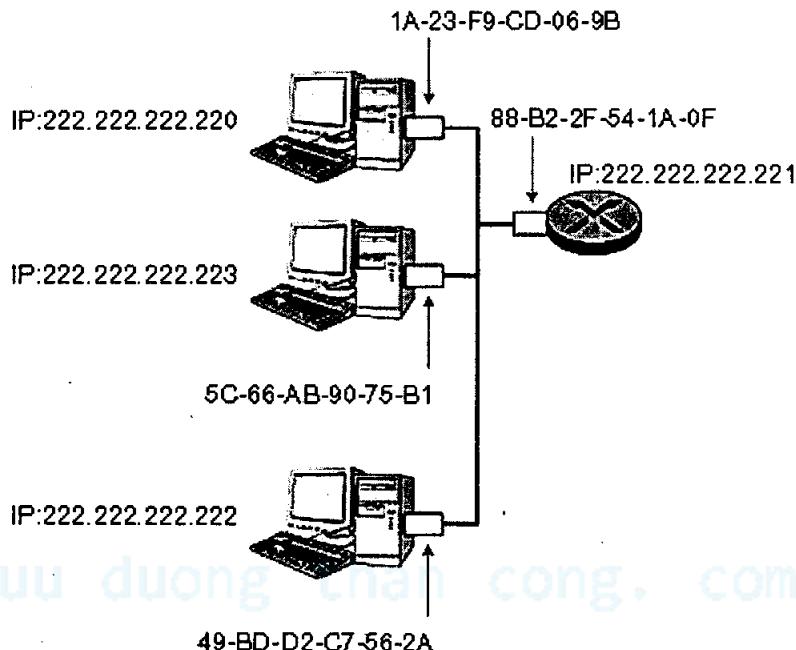
#### 5.4.2. Giao thức giải mã địa chỉ (ARP)

Do tồn tại cả hai kiểu địa chỉ là địa chỉ tầng mạng (chẳng hạn địa chỉ IP) và địa chỉ tầng liên kết dữ liệu (địa chỉ vật lý), nên chắc chắn cần phải có một phương thức biến đổi giữa chúng. Đối với Internet, đây là công việc của giao thức giải mã địa chỉ ARP [RFC 826]. Tất cả máy tính và router trên LAN đều có module ARP.

Để hiểu rõ hơn về ARP, xét mạng minh họa trên Hình 5.18. Mỗi nút trong mạng có địa chỉ IP duy nhất và adapter của nút có một địa chỉ vật lý. Địa chỉ IP được viết dưới dạng ký pháp dấu chấm thập phân và địa chỉ LAN được viết dưới dạng ký pháp thập lục phân. Bây giờ, giả sử rằng, nút có địa chỉ IP là 222.222.222.220 muốn gửi gói dữ liệu IP đến nút có địa chỉ IP là 222.222.222.222. Để thực hiện công việc này, nút gửi phải chuyển cho adapter của nó không chỉ gói dữ liệu IP mà cả địa chỉ vật lý của nút nhận (222.222.222.222). Khi nhận được gói dữ liệu IP và địa chỉ LAN, adapter của nút gửi sẽ tạo ra frame tầng liên kết dữ liệu chứa địa chỉ vật lý của nút nhận và gửi frame đó trên LAN. Nhưng làm thế nào để nút gửi xác định địa chỉ vật lý của nút có địa chỉ IP là 222.222.222.222? Nó sẽ chuyển cho module ARP địa chỉ IP 222.222.222.222, sau đó module ARP trả lại địa chỉ vật lý tương ứng với địa chỉ IP được hỏi là 49–DB–D2–C7–56–2A.

Do vậy thấy rằng, ARP đã xác định địa chỉ vật lý từ địa chỉ IP. Trong khía cạnh nào đó, chức năng này tương tự DNS (đã nghiên cứu trong mục 2.5), DNS xác định địa chỉ IP từ tên máy tính. Tuy nhiên, sự khác biệt rất

lớn giữa hai dịch vụ này là DNS chuyển đổi tên mọi máy tính trên Internet. Ngược lại, ARP chỉ chuyển đổi địa chỉ IP cho những nút trên cùng mạng LAN. Nếu một nút ở Hà Nội có gắng dùng module ARP để xác định địa chỉ vật lý của một nút ở Huế, thì chắc chắn module ARP sẽ trả lại một mã lỗi.



**Hình 5.18.** Mỗi nút trong mạng LAN có một địa chỉ IP, mỗi adapter của nút có một địa chỉ mạng LAN

Bây giờ xét xem module ARP làm việc như thế nào? Module ARP trong mỗi nút chứa một bảng ARP trong RAM của mình. Mỗi hàng của bảng là một ánh xạ giữa địa chỉ IP và địa chỉ vật lý.

Hình 5.19 minh họa bảng ARP của nút 222.222.222.220. Với mỗi ánh xạ trong bảng ARP có trường "thời gian sống" (TTL) cho chính ánh xạ đó, xác định thời gian tồn tại của ánh xạ trong bảng. Chú ý rằng, bảng này không nhất thiết phải chứa tất cả các ánh xạ cho mọi nút trên LAN, ánh xạ có thể dần dần được thêm vào bảng. Thời gian sống của một ánh xạ thường là 20 phút kể từ khi ánh xạ được đưa vào bảng ARP.

Bây giờ, giả sử nút 222.222.222.220 muốn gửi gói dữ liệu IP đến một nút trên LAN. Nút gửi cần xác định địa chỉ vật lý từ địa chỉ IP của nút nhận. Công việc này đơn giản nếu bảng ARP của nút gửi chứa ánh xạ của nút nhận. Nhưng nếu bảng ARP đó không chứa ánh xạ tương ứng cho nút nhận

thì sao? Giả sử nút có địa chỉ IP là 222.222.222.220 muốn gửi gói dữ liệu tới nút 222.222.222.222. Khi đó nút gửi phải sử dụng giao thức ARP để giải mã địa chỉ. Đầu tiên, nút gửi tạo ra một gói đặc biệt gọi là *gói ARP* (ARP packet). Trong gói ARP có trường chứa địa chỉ IP và địa chỉ vật lý của nút gửi, nút nhận. Cả gói truy vấn và trả lời ARP đều có chung khuôn dạng. Mục đích của gói truy vấn ARP là hỏi tất cả các nút khác trên LAN để xác định địa chỉ vật lý ứng với địa chỉ IP.

IP address	LAN address	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

Hình 5.19. Bảng ARP của nút 222.222.222.220

Trở lại ví dụ trên, nút 222.222.222.220 gửi gói truy vấn ARP đến adapter và yêu cầu adapter gửi tới tất cả các nút trên mạng LAN, có nghĩa là sử dụng địa chỉ quảng bá FF-FF-FF-FF-FF-FF. Adapter đặt gói ARP trong frame tầng liên kết dữ liệu, với địa chỉ đích của frame là địa chỉ quảng bá và gửi frame vào mạng LAN. Nó cũng giống như việc giáo viên vào lớp hỏi to "Sinh viên nào có tên là Trần Văn X hãy báo số thẻ sinh viên". Câu nói này lan toả khắp lớp học (được quảng bá), tất cả các sinh viên đều nghe thấy, nhưng chỉ có sinh viên Trần Văn X mới trả lời. Frame chứa truy vấn ARP được tất cả các adapter trên LAN nhận (do sử dụng địa chỉ quảng bá) và mỗi adapter gửi gói ARP trong frame lên bộ xử lý trung tâm của mình. Sau đó, mỗi nút tự kiểm tra xem địa chỉ IP của mình có giống với địa chỉ IP đích trong gói ARP không. Chỉ có duy nhất nút phù hợp mới gửi gói ARP trả lời chứa ánh xạ yêu cầu. Sau đó, nút gửi truy vấn (222.222.222.220) có thể cập nhật bảng ARP và gửi đi gói dữ liệu IP.

Có hai điểm cần chú ý trong giao thức ARP. Thứ nhất, thông điệp truy vấn ARP được gửi quảng bá trong khi thông điệp trả lời ARP được gửi trong frame bình thường. Thứ hai, ARP hoạt động theo kiểu "cắm vào là chạy" (plug and play) vì bảng ARP của nút được xây dựng tự động, không cần người quản trị thiết lập cấu hình. Và nếu một nút dừng kết nối với LAN

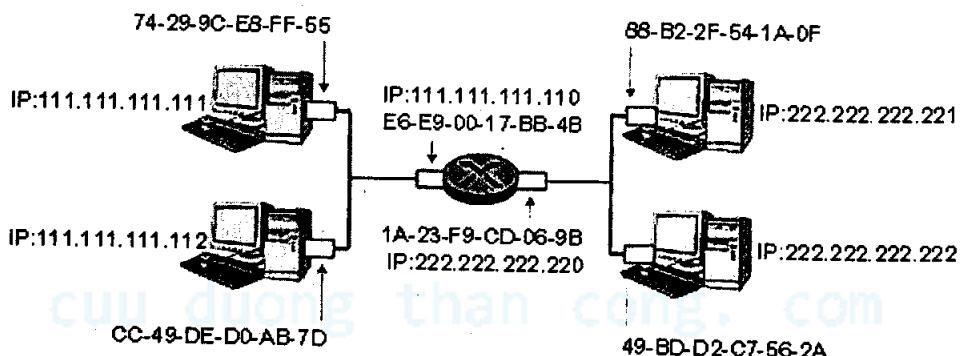
thì ánh xạ tương ứng của nó cũng bị xoá khỏi bảng sau một khoảng thời gian nào đấy.

### *Gửi gói dữ liệu đến nút không nằm trong LAN:*

Chúng ta đã hình dung rõ ràng hoạt động của ARP khi một nút gửi gói dữ liệu đến nút khác nằm trong cùng mạng LAN. Bây giờ xét tình huống phức tạp hơn, khi nút muốn gửi dữ liệu tới nút nằm ngoài mạng LAN. Xét ví dụ minh họa trên Hình 5.20 gồm hai mạng LAN kết nối với nhau qua router.

Có một số điểm cần chú ý trong Hình 5.20. Đầu tiên nút chia ra làm hai kiểu là máy tính và router. Mỗi máy tính có duy nhất một địa chỉ IP và một adapter. Như đã trình bày trong mục 4.4, mỗi giao diện ghép nối của router có một địa chỉ IP riêng. Mỗi giao diện của router cũng có module ARP (trong router) và adapter. Router trong Hình 5.20 có hai giao diện, nên có hai địa chỉ IP, hai module ARP và hai adapter. Dĩ nhiên, mỗi adapter có một địa chỉ vật lý riêng.

Cũng chú ý rằng, tất cả giao diện kết nối vào mạng LAN 1 có địa chỉ IP dạng 111.111.111.xxx và kết nối vào mạng LAN 2 có địa chỉ IP dạng 222.222.222.xxx. Trong ví dụ này, 3 byte đầu tiên của địa chỉ IP xác định địa chỉ mạng, trong khi đó byte cuối cùng xác định nút cụ thể nào trong mạng (chính xác hơn là adapter).



**Hình 5.20. Hai mạng LAN kết nối với nhau qua router**

Bây giờ giả sử rằng, máy tính 111.111.111.111 muốn gửi gói dữ liệu IP đến máy tính 222.222.222.222. Như thường lệ, máy tính gửi sẽ chuyển gói dữ liệu xuống adapter của mình. Nhưng máy tính gửi cũng cần phải chỉ cho adapter biết địa chỉ vật lý đích thích hợp. Adapter sẽ dùng địa chỉ vật lý nào? Có

phải là địa chỉ vật lý của máy tính 222.222.222.222 là 49–BD–D2–C7–56–2A không? Hiện nhiên nếu adapter gửi sử dụng địa chỉ vật lý này thì chắc chắn không một adapter nào của LAN 1 sẽ gửi gói dữ liệu IP lên tầng mạng của mình, vì địa chỉ đích của frame không phù hợp với địa chỉ vật lý của bất kỳ adapter nào trên LAN 1, do đó gói dữ liệu sẽ mất.

Nếu quan sát trên Hình 5.20, ta thấy rằng, để gửi gói dữ liệu từ nút 111.111.111.111 đến nút khác trên LAN 2, đầu tiên gói dữ liệu phải được gửi đến giao diện router 111.111.111.110. Như đã thảo luận trong mục 4.4, bảng định tuyến của máy tính 111.111.111.111 sẽ chỉ ra rằng, để đi đến máy tính 222.222.222.222, thì đầu tiên gói dữ liệu cần được gửi tới router (chính xác hơn là adapter của router) có địa chỉ 111.111.111.110. Như vậy, địa chỉ vật lý đích của frame là địa chỉ vật lý của giao diện router có địa chỉ 111.111.111.110, tức là E6–E–00–17–BB–4B. Làm thế nào máy tính gửi xác định được địa chỉ vật lý của 111.111.111.110? Tất nhiên là, bằng cách sử dụng ARP. Sau khi xác định được địa chỉ vật lý, nút gửi sẽ tạo ra frame và gửi frame vào LAN 1. Adapter của router trên LAN 1 sẽ nhận frame tầng liên kết dữ liệu gửi cho nó và sẽ chuyển gói IP lên tầng mạng của router. Gói dữ liệu IP đã được chuyển từ máy tính nguồn đến router. Bây giờ, router phải xác định giao diện để gửi dữ liệu đi tiếp. Như đã nêu trong mục 4.4, công việc này được thực hiện bằng cách tra cứu bảng định tuyến của router. Bảng định tuyến của router cho biết gói dữ liệu cần gửi qua giao diện 222.222.222.220. Sau đó, giao diện này sẽ gửi gói dữ liệu đến adapter của nó, adapter này đặt gói dữ liệu trong một frame mới và gửi vào LAN 2. Lúc này, địa chỉ vật lý đích trong frame gửi đi là địa chỉ vật lý của đích cuối cùng (nút 222.222.222.222). Làm thế nào router có được địa chỉ vật lý đích này? Tất nhiên là dựa vào ARP.

ARP cho Ethernet được đặc tả trong RFC 826 và có thể tham khảo thêm trong RFC 1180.

## 5.5. ETHERNET

Hiện nay Ethernet thống trị thị trường mạng cục bộ. Mới chỉ đầu những năm 1980 đến đầu những năm 1990, Ethernet còn có nhiều đối thủ khác như FDDI, token – ring, ATM. Một số công nghệ đã thành công trong việc

chiếm lĩnh một thị phần nào đó trong vài năm. Nhưng từ khi ra đời vào giữa những năm 70 của thế kỷ XX, Ethernet liên tục phát triển để rồi dần dần chiếm lĩnh phần lớn thị phần. Ngày nay, Ethernet là công nghệ vượt xa các công nghệ LAN khác và khả năng này khó có thể bị đảo lộn trong tương lai gần.

Có rất nhiều lý do dẫn đến sự thành công của Ethernet. Thứ nhất, Ethernet là mạng cục bộ tốc độ cao được triển khai rộng rãi đầu tiên. Được triển khai tương đối sớm nên các nhà quản trị mạng lập tức trở nên quen thuộc với Ethernet (ưu điểm, các đặc tính,...) và ngại chuyển sang những công nghệ LAN mới. Thứ hai, token – ring, FDDI và ATM phức tạp và đắt hơn Ethernet. Thứ ba, lý do chính đáng nhất để sử dụng các công nghệ LAN khác (FDDI hay ATM) là do công nghệ mới có tốc độ cao hơn, tuy nhiên Ethernet luôn luôn "phản công" lại bằng cách liên tục nâng cấp tốc độ. Ethernet dạng chuyển mạch được phát minh vào đầu những năm 90 của thế kỷ XX, với tốc độ rất cao. Cuối cùng, vì Ethernet quá phổ biến, nên phần cứng Ethernet (đặc biệt là card mạng) cũng hết sức phổ biến và rẻ. Giá cả thấp này cũng do giao thức đa truy cập của Ethernet – CSMA/CD hoàn toàn phân tán nên có thiết kế đơn giản.

Kiến trúc Ethernet (Hình 5.21) được Bob Metcalf và David Boggs đưa ra vào khoảng giữa những năm 70 của thế kỷ XX.

### 5.5.1. Những khái niệm cơ bản của Ethernet

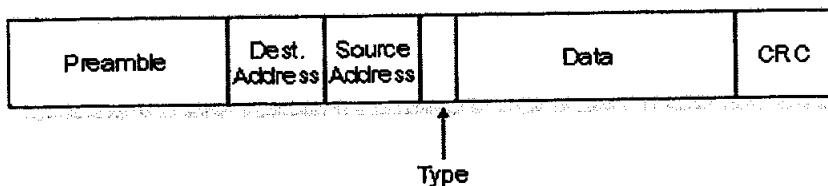
Ngày nay Ethernet xuất hiện dưới nhiều hình thức. Mạng cục bộ Ethernet có thể có topo dạng bus hay dạng sao. Mạng cục bộ Ethernet có thể sử dụng cáp đồng trực hay cáp quang. Hơn nữa, Ethernet có thể truyền dữ liệu với các tốc độ khác nhau như: 10Mbps, 100Mbps hay 1Gbps. Nhưng dù là Ethernet kiểu nào, tất cả các công nghệ Ethernet đều có một số đặc trưng quan trọng sau đây:

#### a) Cấu trúc frame Ethernet

Các công nghệ Ethernet khác nhau đều có chung cấu trúc frame. Cho dù công nghệ Ethernet sử dụng cáp đồng trực hay cáp quang, chạy với tốc độ 10Mbps, 100Mbps hay 1Gbps thì cấu trúc frame đều như nhau.

Frame Ethernet được minh họa trong Hình 5.21. Kiến thức về khuôn dạng frame Ethernet sẽ giúp hiểu kỹ về Ethernet. Xét việc gửi gói dữ liệu IP

giữa hai máy tính trên cùng một mạng cục bộ Ethernet (Chú ý rằng, Ethernet cũng có thể mang các gói dữ liệu tầng mạng khác IP). Giả sử A là adapter gửi có địa chỉ vật lý AA-AA-AA-AA-AA-AA và adapter nhận B có địa chỉ vật lý là BB-BB-BB-BB-BB-BB.



Hình 5.21. Khuôn dạng gói tin Ethernet

Adapter gửi đặt gói dữ liệu (IP datagram) trong frame Ethernet và gửi frame này xuống tầng vật lý. Adapter nhận sẽ nhận frame từ tầng vật lý, lấy ra gói dữ liệu IP và chuyển lên tầng mạng phía trên. Ở đây chỉ nghiên cứu 6 trường trong frame Ethernet:

- ☞ **Trường dữ liệu (từ 46 đến 1.500 byte):** trường này chứa gói dữ liệu IP, MTU (Maximum Transfer Unit) của Ethernet là 1.500 byte. Điều này có nghĩa là, nếu gói dữ liệu IP vượt quá 1.500 byte thì máy tính phải chia nhỏ gói dữ liệu ra (xem mục 4.4.4). Kích thước tối thiểu của trường này là 46 byte. Điều này có nghĩa là, nếu gói dữ liệu nhỏ hơn 46 byte, trường dữ liệu phải được "chèn" thêm một số dữ liệu giả cho đủ 46 byte. Khi bên gửi chèn thêm dữ liệu vào thì tầng mạng, ở bên nhận cũng nhận được cả gói dữ liệu IP lẫn dữ liệu được chèn thêm vào, khi đó nó phải sử dụng trường độ dài trong gói dữ liệu IP để loại bỏ phần thêm vào.
- ☞ **Địa chỉ đích (6 byte):** Trường này chứa địa chỉ vật lý của adapter nhận (chẳng hạn, BB-BB-BB-BB-BB-BB). Khi adapter B nhận bất kỳ frame nào, nó sẽ kiểm tra địa chỉ đích của frame. Nếu địa chỉ đích là BB-BB-BB-BB-BB-BB (địa chỉ của chính nó), hoặc địa chỉ quảng bá LAN (FF-FF-FF-FF-FF-FF) thì adapter mới chuyển gói tin datagram trong trường dữ liệu lên tầng mạng. Nếu không, adapter sẽ loại bỏ frame (trong trường hợp này frame được gửi tới một adapter khác).
- ☞ **Địa chỉ nguồn (6 byte):** Trường này chứa địa chỉ vật lý của adapter gửi frame, trong ví dụ này là AA-AA-AA-AA-AA-AA.

- ☞ **Trường kiểu (2 byte):** Trường này cho phép Ethernet hỗ trợ nhiều giao thức tầng mạng khác nhau. Cần chú ý rằng, máy tính có thể sử dụng nhiều giao thức tầng mạng (không chỉ có IP). Trên thực tế, máy tính nào đó có thể hỗ trợ nhiều giao thức tầng mạng và sử dụng các giao thức khác nhau cho những ứng dụng khác nhau. Vì thế, khi nhận được một frame Ethernet, adapter B cần xác định giao thức tầng mạng nào sẽ nhận nội dung của trường dữ liệu. Những giao thức tầng mạng như IP, Novell IPX hoặc AppleTalk đều có một mã định danh (là một số) đã được chuẩn hóa. Hơn nữa, giao thức ARP cũng có một định danh. Trường kiểu tương tự trường protocol trong gói dữ liệu IP hay trường số hiệu cổng trong tầng giao vận. Mục đích của tất cả các trường này là kết hợp giao thức ở tầng dưới với giao thức ở tầng trên nó.
- ☞ **Mã kiểm tra dư thừa vòng (Cyclic Redundancy Check –CRC) (4 byte):** Như đã nêu trong mục 5.2.3, mục đích của trường CRC là cho phép adapter phát hiện liệu có lỗi nào trong frame nhận được hay không. Có nhiều nguyên nhân lỗi bit, chẳng hạn, suy hao năng lượng điện tử của tín hiệu; tỏa nhiệt trong card Ethernet hay cáp mạng. Việc phát hiện lỗi được thực hiện như sau: Khi tạo ra frame Ethernet, máy tính A tính giá trị trường CRC dựa trên trường dữ liệu thực sự. Công việc kiểm tra tại B xem dữ liệu thực sự và CRC có mâu thuẫn không được gọi là *CRC check*. Nếu việc kiểm tra CRC thất bại (nghĩa là nếu giá trị trường CRC không phù hợp với phần dữ liệu), thì máy tính B xác định trong frame có lỗi.
- ☞ **Lời mở đầu (preamble) (8 byte):** Frame Ethernet bắt đầu với trường preamble 8 byte, trong đó 7 byte đầu tiên có giá trị là 10101010; byte thứ tám có giá trị là 10101011. 7 byte đầu tiên của phần mở đầu làm nhiệm vụ "đánh thức" adapter nhận và đồng bộ hóa đồng hồ bên gửi với đồng hồ bên nhận. Tại sao các đồng hồ lại không đồng bộ hóa? Chú ý rằng, adapter A truyền frame với tốc độ 10Mbps, 100Mbps hay 1Gbps phụ thuộc vào kiểu Ethernet. Tuy nhiên, adapter A chưa chắc đã truyền frame với tốc độ xác định mà với tốc độ nào đó. Adapter nhận có thể chốt đồng hồ của adapter A bằng cách chốt tất cả các bit trong 7 byte đầu tiên. Hai bit cuối

cùng trong byte thứ 8 (hai bit 1 liên tiếp nhau) báo cho adapter B biết rằng, "dữ liệu quan trọng" chuẩn bị đến. Khi máy tính B thấy hai bit 1 liên tiếp nhau, nó biết rằng 6 byte tiếp theo là địa chỉ đích. Adapter có thể phát hiện frame đã được truyền xong khi không thấy dòng điện.

### b) Dịch vụ không hướng nối, không tin cậy

Tất cả công nghệ Ethernet cung cấp cho tầng mạng *dịch vụ không hướng nối*. Nghĩa là, khi adapter A muốn gửi gói dữ liệu đến adapter B, adapter A sẽ đặt gói dữ liệu trong frame và gửi frame đó vào LAN mà không cần "bắt tay" trước với adapter B. Dịch vụ không kết nối ở tầng 2 này tương tự với dịch vụ IP ở tầng 3 và dịch vụ UDP ở tầng 4.

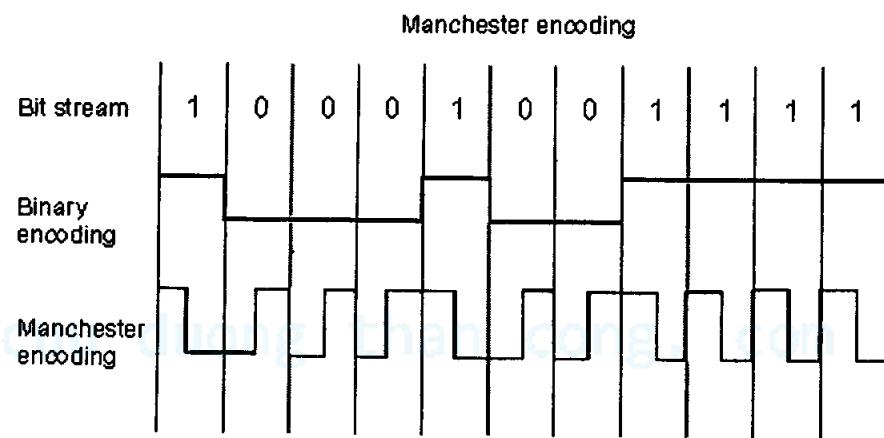
Công nghệ Ethernet cung cấp dịch vụ không tin cậy cho tầng mạng. Cụ thể, khi nhận được frame từ adapter A, adapter B sẽ không gửi phản hồi cho A. Adapter A không thể xác định liệu frame nó truyền đi có được nhận đúng hay không. Nếu phát hiện lỗi khi kiểm tra CRC, adapter B sẽ loại bỏ frame. Chính điều này giúp Ethernet đơn giản và rẻ. Nhưng dòng dữ liệu chuyển tới tầng mạng có thể bị gián đoạn.

Nếu có sự gián đoạn do một số frame Ethernet bị loại bỏ, giao thức tầng ứng dụng tại máy B có phát hiện được sự gián đoạn đó hay không? Như đã trình bày trong Chương 3, điều này phụ thuộc việc ứng dụng sử dụng UDP hay TCP. Nếu ứng dụng dùng UDP thì giao thức tầng ứng dụng trong máy B sẽ không phát hiện được gián đoạn trong dữ liệu. Mặt khác, nếu ứng dụng dùng TCP thì thực thể TCP trong máy B sẽ không gửi biên nhận cho những dữ liệu đã bị loại bỏ, do vậy, thực thể TCP trong máy A sẽ gửi lại. Chú ý rằng, khi TCP gửi lại dữ liệu, thì cuối cùng dữ liệu cũng sẽ đi qua các adapter Ethernet. Và như vậy, Ethernet truyền lại dữ liệu. Tuy nhiên, Ethernet không biết rằng nó đang truyền lại mà coi rằng đó là một gói dữ liệu mới.

### c) Giải tần cơ sở và mã hoá Manchester

Ethernet sử dụng băng tần cơ sở (baseband), nghĩa là adapter gửi tín hiệu số trực tiếp vào kênh truyền dùng chung. Card giao diện không dịch chuyển tín hiệu sang dải tần số khác như trong ASDL và các hệ thống cáp modem. Ethernet sử dụng mã hoá Manchester (Hình 5.22). Trong phương

pháp mã hoá Manchester, mỗi bit ứng với một quá trình chuyển trạng thái (transition) – bit 1 chuyển từ trên xuống dưới, bit 0 chuyển từ dưới lên trên. Lý do sử dụng mã hoá Manchester là đồng hồ của adapter gửi và nhận không đồng bộ hoàn toàn với nhau. Khi xuất hiện sự chuyển trong phần giữa mỗi bit, máy tính nhận có thể đồng bộ đồng hồ của nó với đồng hồ của máy tính gửi. Sau khi đồng hồ của adapter nhận được đồng bộ hoá, phía nhận có thể thu được tín hiệu của mỗi bit và xác định nó là 0 hay 1. Mã hoá Manchester được sử dụng nhiều trong tầng vật lý chứ không phải trong tầng liên kết dữ liệu.



Hình 5.22. Mã hoá Manchester

### 5.5.2. CSMA/CD – Giao thức đa truy cập của Ethernet

Các nút trên mạng cục bộ Ethernet được kết nối qua một kênh truyền quang bá dùng chung, vì vậy, khi adapter gửi đi một frame, tất cả các adapter trên LAN đều nhận được frame. Như đã đề cập trong mục 5.3, Ethernet dùng thuật toán đa truy cập CSMA/CD. Chú ý rằng, CSMA/CD sử dụng các cơ chế sau:

- ☞ Adapter có thể bắt đầu truyền tại bất kỳ thời điểm nào, nghĩa là không chia khoảng thời gian.
- ☞ Adapter không bao giờ truyền frame khi thấy có adapter khác đang truyền (cảm nhận sóng mang).
- ☞ Adapter đang truyền chấm dứt truyền ngay khi phát hiện ra adapter khác cũng đang truyền (phát hiện xung đột).

- ☞ Trước khi cỗ găng thử truyền lại, adapter đợi một khoảng thời gian ngẫu nhiên tương đối nhỏ.

Những cơ chế này giúp hiệu suất của CSMA/CD được cải thiện đáng kể so với slotted ALOHA khi vận hành trong môi trường LAN. Trong thực tế, nếu thời gian để tín hiệu lan truyền giữa hai nút là rất nhỏ, thì hiệu suất của CSMA/CD có thể đạt tới 100%. Nhưng chú ý rằng, cơ chế thứ hai và thứ ba kể trên yêu cầu adapter Ethernet có khả năng: (1) cảm nhận được khi nào thì có một adapter khác đang truyền; (2) phát hiện xung đột trong khi truyền. Adapter Ethernet thực hiện hai nhiệm vụ này bằng việc đo mức điện áp trước và trong khi truyền.

Adapter dùng giao thức CSMA/CD không cần kết hợp với adapter khác trên Ethernet. Trên một adapter, giao thức CSMA/CD làm việc như sau:

- ☞ Adapter nhận PDU tầng mạng, tạo ra frame Ethernet và đặt frame vào trong bộ đệm của adapter.
- ☞ Nếu adapter cảm nhận kênh truyền rỗi (không có năng lượng tín hiệu trên kênh truyền) thì adapter bắt đầu truyền. Nếu adapter thấy kênh truyền bận, nó sẽ đợi cho đến khi không phát hiện được năng lượng tín hiệu và sau đó bắt đầu truyền.
- ☞ Trong khi truyền, adapter kiểm tra xem có năng lượng tín hiệu đến từ adapter khác hay không. Nếu sau khi đã truyền xong frame mà không phát hiện được năng lượng trên đường truyền thì có thể xem frame được truyền thành công.
- ☞ Nếu adapter phát hiện năng lượng tín hiệu từ adapter khác trong khi đang truyền thì lập tức nó dừng lại không truyền và gửi đi tín hiệu báo nhiễu 48 bit (jam signal).

Sau khi dừng phát và gửi tín hiệu báo nhiễu, adapter sẽ thực hiện thuật toán *exponential backoff*. Khi truyền frame nào đó, nếu thấy frame đó bị xung đột n lần liên tiếp, adapter chọn một giá trị ngẫu nhiên K trong khoảng  $(0, 1, 2, \dots, 2^m - 1)$ , với  $m = \min(n, 10)$ . Sau đó adapter sẽ đợi  $K \times 512$  trước khi quay lại bước 2.

Sau đây sẽ giải thích về giao thức CSMA/CD. Mục đích của tín hiệu báo nhiễu là bảo đảm tất cả các adapter đang truyền khác đều phát hiện ra xung đột. Xét ví dụ sau: Giả sử adapter A bắt đầu truyền đi một frame và

ngay trước khi tín hiệu từ A tới được adapter B, adapter B bắt đầu truyền. Do vậy, B chỉ truyền được vài bit trước khi dừng lại không truyền tiếp. Vài bit này sẽ lan toả được đến A, nhưng chúng không tạo đủ năng lượng để A có thể phát hiện xung đột. Để đảm bảo A phát hiện được xung đột, B phải truyền thêm tín hiệu báo nhiễu dài khoảng 48 bit.

Tiếp theo, xét thuật toán exponential backoff. Cần chú ý rằng, thời gian để truyền đi một bit rất nhỏ – với tốc độ Ethernet 10Mbps, thời gian này là  $0,1\mu s$ . Xét ví dụ sau: Giả sử adapter lần đầu tiên truyền đi một frame và trong khi truyền phát hiện có xung đột. Sau đó, adapter sẽ chọn  $K = 0$  với xác suất 0,5 và chọn  $K = 1$  với xác suất 0,5. Nếu adapter chọn  $K = 0$  thì ngay lập tức nó sẽ nhảy đến bước 2 sau khi truyền đi tín hiệu báo nhiễu. Nếu adapter chọn  $K = 1$  thì nó sẽ đợi 51,2ms trước khi quay lại bước 2. Sau xung đột lần thứ hai, K được chọn ngẫu nhiên giữa các giá trị (0, 1, 2, 3) với xác suất bằng nhau, sau ba xung đột K sẽ được chọn ngẫu nhiên giữa các giá trị (0, 1, 2,..., 7) với xác suất bằng nhau, sau trên 10 xung đột K được chọn ngẫu nhiên giữa các giá trị (0, 1, 2,..., 1023) với xác suất bằng nhau. Như vậy, tổng số giá trị mà K có thể lựa chọn tăng theo luỹ thừa cơ số 2 với số mũ là số lần xung đột cho (cho đến khi  $N = 10$ ).

Chuẩn Ethernet ánh định giới hạn khoảng cách giữa hai nút bất kỳ. Giới hạn này bảo đảm rằng, nếu adapter A chọn giá trị K thấp hơn giá trị K của tất cả các adapter khác liên quan đến xung đột trong pha trước, thì A có thể truyền đi frame mà không bị xung đột nữa.

Tại sao lại sử dụng thuật toán exponential backoff? Tại sao không chọn K trong khoảng  $\{0, 1, 2, 3, 4, 5, 6, 7\}$  sau mọi xung đột. Lý do là, sau khi adapter gấp xung đột lần đầu tiên, nó không hình dung được có bao nhiêu adapter liên quan đến xung đột đó. Nếu chỉ có một số lượng nhỏ adapter thì chắc chắn K sẽ được chọn trong một tập hợp hạn chế. Ngược lại, nếu có nhiều adapter liên quan, thì K được chọn trong một tập hợp lớn hơn. Bằng cách tăng kích cỡ của tập hợp sau mỗi xung đột, adapter sẽ thích nghi được với nhiều hoàn cảnh.

Chú ý rằng, mỗi lần adapter chuẩn bị frame mới để gửi đi, nó sử dụng thuật toán CSMA/CD nói trên. Cụ thể, adapter không quan tâm tới bất kỳ xung đột nào trước đó. Do vậy, rất có khả năng adapter với frame mới có

thể truyền xen vào, trong khi một vài adapter khác đang trong trạng thái exponential backoff.

### **Hiệu suất Ethernet:**

Khi chỉ có một nút có frame để truyền, nút đó có thể truyền với tốc độ tối đa (10Mbps, 100Mbps hoặc 1Gbps). Tuy nhiên, nếu nhiều nút cùng truyền thì tốc độ truyền thành công (effective rate) của kênh truyền có thể giảm đi đáng kể. Ta định nghĩa *hiệu suất* (efficiency) của Ethernet là tỷ lệ thời gian không có xung đột trên kênh truyền khi có nhiều nút tích cực, mỗi nút cần truyền nhiều frame trong một khoảng thời gian dài. Để xác định hiệu suất gần đúng của Ethernet, giả sử  $t_{prop}$  là thời gian lớn nhất năng lượng tín hiệu lan toả giữa hai adapter. Giả sử  $t_{trans}$  là thời gian để truyền đi một frame Ethernet với độ lớn cực đại (xấp xỉ 1,2ms với Ethernet 10 Mbps). Có thể xem [Lam 1980] và [Bertsekas 1991] để xác định công thức tính. Ở đây sử dụng công thức:

$$\text{Efficiency} = \frac{1}{1 + 5t_{prop}/t_{trans}}$$

Từ công thức này ta thấy, nếu  $t_{prop}$  đạt tới 0 thì hiệu suất đạt tới 1. Điều này rất hợp lý: Nếu thời gian trễ là 0 thì các nút xung đột sẽ lập tức bỏ dở mà không lãng phí kênh truyền. Khi  $t_{trans}$  trở lên rất lớn, hiệu suất đạt tới 1. Điều này cũng hiển nhiên, vì khi frame có được kênh truyền nó sẽ chiếm dụng kênh truyền trong khoảng thời gian dài, như vậy, kênh truyền hầu như lúc nào cũng trong trạng thái làm việc.

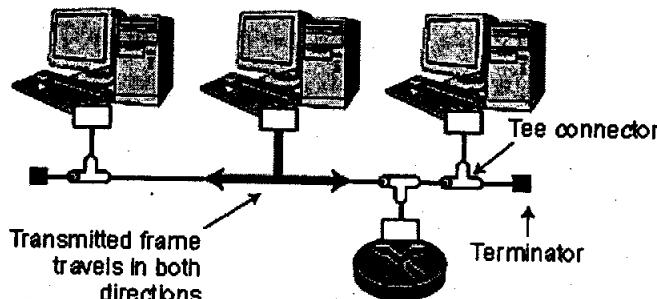
### **5.5.3. Ethernet**

Phần lớn Ethernet phổ biến ngày nay là *10Base2*, sử dụng cáp đồng trục gầy (thin coaxial cable), có topo dạng bus, tốc độ truyền là 10Mbps; *10BaseT* sử dụng cáp đồng trục, topo hình sao, tốc độ truyền là 10 Mbps; *100BaseT* sử dụng dây đồng xoắn, topo hình sao, tốc độ truyền là 100Mbps; Gigabyte Ethernet sử dụng cá sợi quang hay dây đồng xoắn, truyền với tốc độ 1Gbps. Những công nghệ Ethernet này được chuẩn hoá bởi IEEE 802.3. Vì thế, LAN Ethernet thường được gọi là 802.3 LAN.

Trước khi tiếp tục, cần nói về *bộ tiếp sức* (repeater) – được sử dụng phổ biến trong mạng LAN cũng như các đường truyền trên khoảng cách xa.

Repeater là thiết bị tầng vật lý xử lý trên từng bit riêng lẻ chứ không phải trên frame. Khi tín hiệu (biểu diễn bit 0 hoặc 1) đến từ một cổng, repeater thường tái tạo lại tín hiệu này bằng cách tăng cường độ năng lượng của tín hiệu và gửi tín hiệu đó qua tất cả các cổng còn lại. Repeater được sử dụng rộng rãi trong LAN để mở rộng phạm vi địa lý. Cần chú ý rằng, trong Ethernet, repeater không có khả năng cảm nhận sóng mang hay thực hiện bất kỳ một chức năng nào của CSMA/CD, repeater chỉ tái tạo và gửi tín hiệu đến từ một cổng ra tất cả các cổng khác, kể cả trong trường hợp các cổng kia cũng đang có tín hiệu để truyền.

### a) 10Base2



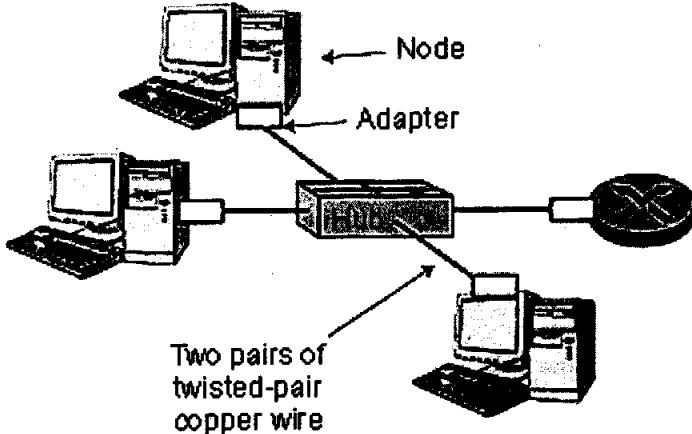
Hình 5.23. Ethernet 10Base2

Ethernet 10Base2 rất phổ biến. "10" trong 10Base2 có nghĩa là tốc độ truyền là 10Mbps. "2" có ý nghĩa là khoảng cách tối đa giữa hai trạm không có repeater ở giữa không vượt quá 200m. Hình 5.23 minh họa mạng Ethernet 10Base2. 10Base2 có topo dạng bus, các nút (chính xác hơn là các adapter) được kết nối trực tiếp vào một môi trường dùng chung – cáp đồng trực gập (là loại cáp tương tự như cáp truyền hình nhưng mỏng và nhẹ hơn). Khi adapter gửi đi một frame, frame sẽ được truyền qua đầu nối chữ T (T connector). Sau đó frame sẽ lan tỏa theo hai hướng của dây dẫn. Trên đường đi, mỗi adapter sẽ nhận được một bản sao của frame (chính xác hơn là các adapter thu được các tín hiệu của frame). Khi đến điểm cuối cùng của dây dẫn, tất cả các tín hiệu sẽ bị terminator hấp thụ (triệt tiêu). Chú ý rằng, do tất cả adapter đều có khả năng nhận mọi frame nên 10Base2 rõ ràng là môi trường quảng bá.

Nếu không có repeater, độ dài tối đa của bus là 185m. Nếu bus có độ dài lớn hơn, suy hao tín hiệu sẽ làm hệ thống hoạt động không chính xác.

Ngoài ra, nếu không có repeater, số lượng tối đa các nút là 30. Người ta sử dụng repeater để nối các đoạn 10Base2 liên tiếp nhau, mỗi đoạn có thể có 30 máy và dài 185m. Chỉ có thể sử dụng tối đa 4 repeater, do đó có 5 đoạn 10Base2.

### b) 10BaseT và 100BaseT



Hình 5.24. Topo dạng sao 10BaseT và 100BaseT

10BaseT và 100BaseT là hai Topo tương tự nhau. Điểm khác biệt quan trọng nhất là tốc độ truyền của 10BaseT là 10Mbps trong khi tốc độ truyền của Ethernet 100BaseT là 100Mbps. 10BaseT và 100BaseT là công nghệ được sử dụng rất phổ biến hiện nay. Chúng có topo dạng sao, như minh họa trên Hình 5.24. Trong topo hình sao có một thiết bị trung tâm được gọi là *hub* (đôi khi gọi là *bộ tập trung* – concentrator). Adapter trên mỗi nút có kết nối trực tiếp đến hub. Kết nối này gồm hai cặp dây đồng xoắn đôi, một để truyền và một để nhận. Tại mỗi đầu của kết nối có một connector (bộ nối) RJ-45 giống như connector RJ-45 được sử dụng cho điện thoại thông thường. Chữ "T" trong 10BaseT và 100BaseT là viết tắt của "Twisted pair". Đôi với 10BaseT và 100BaseT, khoảng cách tối đa giữa adapter và hub là 100m, vì vậy, độ dài lớn nhất giữa hai nút là 200m. Trong phần sau ta thấy, khoảng cách này có thể được tăng nếu sử dụng các thiết bị như hub, bridge, switch.

Về bản chất, hub là repeater, vì khi nhận được tín hiệu từ adapter, hub sẽ gửi tín hiệu đó đến tất cả các adapter khác. Theo cách này, mỗi adapter có thể: (1) cảm nhận kênh truyền để xác định liệu kênh truyền có rỗi không;

(2) phát hiện xung đột trong khi đang truyền dữ liệu. Nhưng hub được dùng phổ biến do có khả năng trợ giúp việc quản trị mạng. Ví dụ, nếu adapter trực trặc và tiếp tục gửi frame Ethernet (gọi là *jabbering adapter*), thì mạng 10Base2 Ethernet sẽ sụp đổ vì không adapter nào có khả năng truyền thông nữa. Nhưng điều này không xảy ra với mạng 10BaseT, vì hub sẽ phát hiện vấn đề và ngưng kết nối tới adapter đang trực trặc. Tính năng này có tính chất tự động, có nghĩa là, không cần sự can thiệp bằng tay của người quản trị mạng. Hơn thế nữa, hầu hết hub có thể thu thập và báo cáo thông tin đến máy tính có kết nối trực tiếp đến hub. Máy tính kiểm tra này sẽ sử dụng giao diện đồ họa hiển thị các thông tin trạng thái của hub như băng thông, tỷ lệ xung đột, kích thước trung bình của frame,... Người quản trị mạng có thể sử dụng thông tin này không chỉ để kiểm tra và khắc phục lỗi, mà còn để lập kế hoạch phát triển LAN trong tương lai.

Nhiều adapter Ethernet ngày nay là adapter 10/100Mbps. Tức là, có thể sử dụng được cả hai kiểu Ethernet là 10BaseT và 100BaseT. Đặc trưng của 100BaseT là sử dụng loại cáp xoắn kiểu 5 (loại cáp chất lượng cao với nhiều vỏ). Khác 10Base2 và 10BaseT, 100BaseT không sử dụng phương pháp mã hoá Manchester mà sử dụng phương pháp 4B5B có hiệu suất cao hơn, mỗi nhóm 5 chu kỳ đồng hồ được sử dụng để mã hoá 4 bit và cung cấp đủ thông tin cho phép đồng bộ hoá đồng hồ.

Chú ý rằng, cả hai công nghệ 10BaseT và 100BaseT đều có thể sử dụng cáp quang, cáp quang thường được sử dụng để kết nối đến hub. Giá thành của cáp quang cao do giá thành của connector, nhưng ưu điểm là khả năng chống nhiễu rất tốt. Chuẩn IEEE 802 cho phép LAN có thể trải rộng trên vùng địa lý lớn nếu sử dụng cáp quang để nối các nút nằm trên trục chính (backbone).

### c) Gigabit Ethernet

Gigabit Ethernet là sự mở rộng của chuẩn Ethernet 10BaseT và 100BaseT. Với tốc độ truyền dữ liệu dạng thô là 1.000Mbps, Gigabit Ethernet vẫn duy trì khả năng tương thích hoàn toàn với các thiết bị Ethernet kiểu cũ. Chuẩn Gigabit Ethernet (IEEE802.3x) thực hiện các công việc sau:

- ☛ Sử dụng khuôn dạng frame Ethernet chuẩn (Hình 5.24), tương thích với công nghệ 10BaseT và 100BaseT. Điều này cho phép dễ dàng tích hợp Gigabit Ethernet vào các cơ sở đã cài đặt các thiết bị Ethernet.

- Cho phép đường truyền điểm nối điểm cũng như kênh truyền quảng bá dùng chung. Đường truyền điểm nối điểm dùng switch (xem mục 5.6), trong khi kênh truyền quảng bá sử dụng hub giống 10BaseT và 100BaseT. Trong thuật ngữ Gigabit Ethernet, hub được gọi là "buffered distributors".
- Sử dụng CSMA/CD cho kênh truyền quảng bá dùng chung. Để đạt được hiệu suất mong muốn, khoảng cách lớn nhất giữa các nút bị hạn chế.
- Kênh truyền điểm nối điểm có đặc tính song công, mỗi hướng truyền với tốc độ 1 Gbps.
- Giống 10BaseT và 100BaseT, Ethernet Gigabit có topo dạng sao với hub hoặc switch ở trung tâm (switch Ethernet được trình bày trong mục 5.6). Gigabit Ethernet thường được sử dụng trên các trực chính (backbone) kết nối nhiều mạng cục bộ Ethernet 10BaseT và 100BaseT. Gigabit Ethernet có thể sử dụng loại cáp 5UTP hoặc cáp quang.

## 5.6. HUB, BRIDGE VÀ SWITCH

Cơ quan – các công ty, trường đại học có đặc điểm gồm nhiều bộ phận con, mỗi bộ phận có mạng cục bộ Ethernet riêng. Tất nhiên, cơ quan muốn kết nối mạng cục bộ của các bộ phận. Trong mục, trình bày một số hướng tiếp cận để kết nối các LAN với nhau bằng ba hướng tiếp cận là sử dụng hub, bridge và switch.

### 5.6.1. Hub

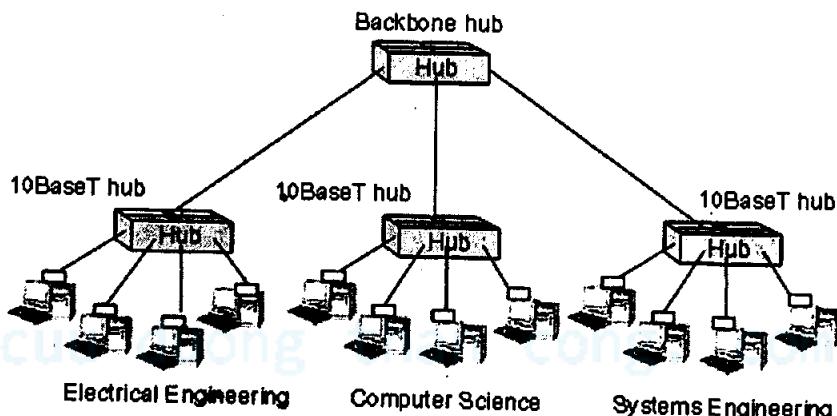
Cách đơn giản nhất để kết nối LAN là sử dụng hub. Hub là thiết bị đơn giản, sao chép tín hiệu đến từ một cổng ra tất cả các cổng còn lại. Bản chất của hub là repeater, thao tác trên bit, vì thế chúng là thiết bị ở tầng vật lý. Khi bit đi vào một cổng, hub sẽ truyền bit này qua tất cả các cổng khác.

Hình 5.25 minh họa kết nối mạng LAN của ba khoa trong một trường đại học qua hub. Mỗi khoa có một mạng Ethernet 10BaseT để cán bộ và sinh viên của khoa sử dụng. Mỗi máy tính của khoa kết nối điểm nối điểm đến hub. Hub thứ tư, được gọi là *backbone hub* (hub trực chính) có kết nối

điểm nối điểm đến các hub của khoa được sử dụng để liên kết LAN của ba khoa. Thiết kế được chi ra trong Hình 5.25 là *thiết kế hub nhiều tầng* (multi – tier hub design), vì các hub được tổ chức trong hệ thống phân cấp. Có thể tạo thiết kế nhiều tầng, ví dụ, một tầng dành cho cấp khoa, một tầng dành cho các trường trong trường đại học lớn (ví dụ, Trường công nghệ, Trường Kinh tế,...) và một tầng ứng với mức cao nhất của trường.

Trong thiết kế nhiều tầng, coi toàn bộ mạng liên kết với nhau là mạng cục bộ LAN và coi mỗi phân mạng LAN ứng với một khoa (nghĩa là hub của khoa và các máy tính nối tới hub đó) là *LAN segment*. Chú ý rằng, tất cả LAN segment trong Hình 5.25 thuộc về cùng một vùng xung đột, nghĩa là, bất cứ lúc nào nhiều nút trên LAN truyền dữ liệu tại cùng một thời điểm thì sẽ phát sinh xung đột và tất cả những nút liên quan bắt đầu quá trình "exponential backoff".

Mạng cục bộ cấp khoa liên kết tới hub trực chính có nhiều ưu điểm. Đầu tiên và quan trọng nhất là, nó cung cấp môi trường truyền thông giữa các khoa với nhau. Thứ hai, nó mở rộng khoảng cách tối đa giữa bất cứ cặp nút nào trên LAN. Ví dụ, với 10BaseT khoảng cách lớn nhất giữa nút và hub là 100m; vì thế, trong LAN segment, khoảng cách lớn nhất giữa hai nút lên tới 200m.



Hình 5.25. Kết nối qua hub

Nếu kết nối qua hub, khoảng cách tối đa này có thể được mở rộng vì khoảng cách giữa các hub kết nối trực tiếp với nhau có thể là 100m khi sử dụng cáp xoắn đôi (và khoảng cách này sẽ tăng khi dùng cáp quang). Ưu điểm thứ ba là thiết kế nhiều tầng giảm nguy cơ sụp đổ của toàn hệ thống.

Giả sử nếu bất kỳ hub của khoa nào đó bị trục trặc, hub trục chính có thể phát hiện vấn đề và phong tỏa kết nối tới hub khoa đó. Như vậy, các khoa còn lại vẫn có thể tiếp tục hoạt động và truyền thông trong khi hub bị lỗi không hoạt động.

Tuy vậy, hub cũng có nhược điểm. Đầu tiên và có lẽ quan trọng nhất là khi sử dụng hub trung tâm, miền xung đột của mạng cục bộ của từng khoa trở thành miền xung đột chung của toàn bộ hệ thống. Xét ví dụ minh họa trên Hình 5.25. Trước khi kết nối ba khoa, mạng cục bộ mỗi khoa có băng thông cực đại là 10Mbps. Vì vậy, thông lượng toàn bộ tối đa của 3 LAN là 30Mbps. Nhưng khi mạng LAN của ba khoa được kết nối vào hub trung tâm, tất cả máy tính của ba khoa thuộc về cùng một miền xung đột, nên thông lượng bị giảm xuống 10Mbps.

Hạn chế thứ hai là, nếu các khoa khác nhau sử dụng các công nghệ Ethernet khác nhau, thì không có khả năng để kết nối chúng vào hub trung tâm. Ví dụ, nếu một khoa sử dụng 10BaseT và các khoa còn lại sử dụng 100BaseT, thì không thể kết nối chúng với nhau vì hub về bản chất là repeater.

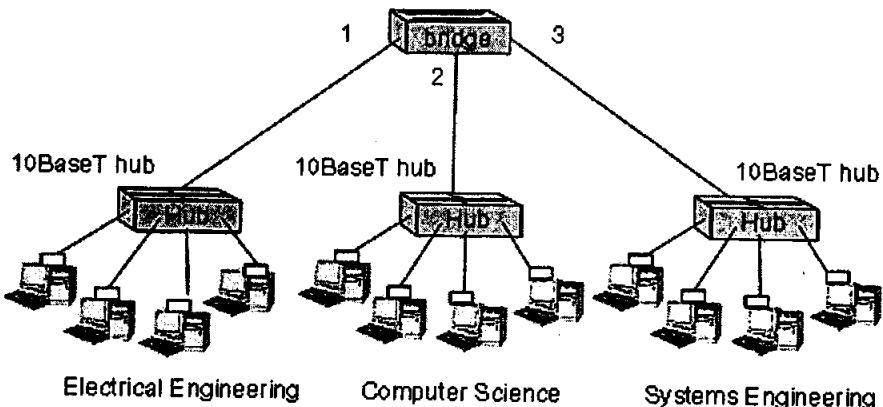
Hạn chế thứ ba là, mỗi công nghệ Ethernet (10Base2, 10BaseT, 100BaseT,...) có giới hạn về số nút, khoảng cách tối đa giữa hai máy tính trong miền xung đột và số tầng tối đa trong thiết kế nhiều tầng. Những hạn chế này hạn chế tổng số máy tính có thể kết nối đến mạng cục bộ cũng như phạm vi địa lý của mạng cục bộ nhiều tầng.

## 5.6.2. Bridge

Khác với hub (là thiết bị tầng vật lý), bridge có thể xử lý trên frame Ethernet, vì vậy nó là thiết bị tầng 2. Thực tế, bridge chính là thiết bị chuyên mạch gói thực hiện việc chuyển và lọc frame căn cứ trên địa chỉ vật lý. Khi frame đến từ một cổng nào đó của bridge, bridge không gửi frame đến tất cả các cổng khác. Bridge sẽ xác định địa chỉ đích tầng 2 (địa chỉ vật lý) của frame và chuyển frame đến cổng duy nhất dẫn về đích.

Hình 5.26 minh họa ba khoa trong ví dụ trước kết nối tới bridge. Ba chữ số bên cạnh bridge là số thứ tự các cổng của bridge. Khi các khoa được kết nối qua bridge như trong hình 5.26, chúng ta vẫn coi mạng kết nối toàn bộ là LAN và mạng của mỗi khoa là LAN segment giống như ở trên. Nhưng khác

với thiết kế hub nhiều tầng trong hình 5.26, mỗi LAN segment bây giờ là một miền xung đột đã được cô lập.



Hình 5.26. Kết nối bằng Bridge

Bridge có thể khắc phục nhiều vấn đề của hub. Thứ nhất, bridge cho phép truyền thông giữa các khoa trong khi cô lập miền xung đột của mỗi khoa. Thứ hai, bridge có thể kết nối các công nghệ LAN khác nhau (Ethernet 10Mbps và 100Mbps chẳng hạn). Thứ ba, không bị giới hạn về khoảng cách tối đa trong mạng cục bộ khi sử dụng bridge để kết nối các LAN segment. Về lý thuyết mà nói, sử dụng bridge có thể xây dựng một mạng LAN trải rộng trên toàn thế giới.

#### a) Bridge Forwarding và Filtering (chuyển tiếp và lọc)

*Lọc (Filtering)* là khả năng xác định liệu sẽ chuyển tiếp frame đến cổng nào đó hay loại bỏ luôn frame. *Chuyển tiếp (Forwarding)* là khả năng xác định cổng kế tiếp để chuyển frame đi. Bridge thực hiện hai chức năng này nhờ *bảng bridge* (bridge table). Mỗi hàng trong bảng ứng với một nút đích trên mạng LAN. Tuy vậy, bảng bridge không nhất thiết phải chứa tất cả các hàng cho mọi nút trong mạng. Mỗi hàng trong bảng bridge gồm có: (1) địa chỉ vật lý của nút; (2) cổng bridge có thể dẫn đến nút đó; (3) thời điểm thiết lập hàng đó trong bảng. Ví dụ về bảng bridge cho LAN trong Hình 5.26 được chỉ ra trong Hình 5.27. Mặc dù quá trình chuyển frame có vẻ tương tự quá trình chuyển gói dữ liệu datagram trong Chương 4, nhưng sẽ thấy ngay chúng hoàn toàn khác nhau. Ở đây chú ý rằng, địa chỉ bridge sử dụng là địa chỉ vật lý chứ không phải là địa chỉ của tầng mạng (IP); bảng bridge được xây dựng khác với bảng định tuyến.

Address	Interface	Time
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
...	...	...

Hình 5.27. Một phần bảng bridge cho LAN trong hình 5.26

Để hiểu chức năng lọc và chuyển tiếp làm việc như thế nào? Giả sử frame với địa chỉ đích DD-DD-DD-DD-DD-DD đến bridge từ cổng x. Bridge tìm kiếm trên bảng lọc hàng ứng với địa chỉ vật lý DD-DD-DD-DD-DD-DD để tìm ra cổng y tương ứng – là cổng sẽ dẫn đến nút có địa chỉ đích DD-DD-DD-DD-DD-DD. Điều gì sẽ xảy ra nếu không có giao diện y như thế trong bảng bridge?

- ☞ Nếu  $x = y$ , thì frame đến từ segment chứa adapter DD-DD-DD-DD-DD-DD. Không cần chuyển frame đến bất kỳ cổng nào khác, bridge thực hiện chức năng lọc bằng cách loại bỏ frame.
- ☞ Nếu  $x \neq y$ , thì frame cần được gửi đến segment nào đó qua cổng y. Bridge thực hiện chức năng chuyển tiếp bằng cách đặt frame vào bộ đệm ra của cổng y.

Những quy tắc đơn giản này cho phép bridge cô lập các miền xung đột của các LAN segment khác nhau. Những quy tắc này cũng cho phép hai cặp thiết bị trên hai segment khác nhau truyền đồng thời mà không bị xung đột.

Xét những quy tắc này cho mạng được minh họa trên Hình 5.26 và bảng bridge tương ứng trên Hình 5.27. Giả sử frame với địa chỉ đích là 62-EF-F7-11-89-A3 được gửi đến bridge qua cổng 1. Bridge kiểm tra bảng và thấy rằng đích nằm trên LAN segment được kết nối đến cổng 1 (là mạng LAN của khoa Electrical Engineering). Điều này có nghĩa là frame thực sự đã được quảng bá trên LAN segment này. Do vậy, bridge sẽ lọc

frame (nghĩa là, loại bỏ frame – vì thực sự máy tính đích cũng đã nhận được frame này rồi). Giả sử frame với địa chỉ đích như vậy đến từ cổng 2. Bridge lại kiểm tra bảng và thấy rằng đích nằm ở trên hướng ứng với cổng 1, do đó bridge chuyển frame ra cổng 1. Rõ ràng rằng, nếu bảng bridge đầy đủ và chính xác, bridge cho phép truyền thông giữa các khoa, nhưng có lập các miền xung đột.

Khi có frame để gửi chuyển tiếp, hub (hoặc repeater) gửi frame lên trên đường truyền mà không quan tâm xem có thiết bị nào khác đang chiếm dụng đường truyền không. Trái lại, bridge sẽ sử dụng thuật toán CSMA/CD trong mục 5.3 khi cần gửi đi frame. Tức là bridge sẽ không truyền ngay nếu như có nút khác trên LAN segment cũng đang truyền. Hơn nữa, bridge cũng sử dụng thuật toán exponential backoff khi có xung đột. Vì vậy, cổng của bridge hoạt động giống như adapter của nút. Nhưng về mặt kỹ thuật mà nói, bridge không phải là adapter, vì chúng không có địa chỉ vật lý. Chú ý rằng, adapter của nút luôn luôn chèn địa chỉ vật lý của nó vào trường địa chỉ nguồn trong tất cả các frame nó gửi đi. Điều này cũng đúng cho adapter của router. Ngược lại, bridge không thay đổi địa chỉ nguồn của frame.

Một tính năng quan trọng của bridge là khả năng kết nối các LAN segment sử dụng những công nghệ Ethernet khác nhau. Ví dụ, nếu trong Hình 5.26, Khoa Electrical Engineering sử dụng Ethernet 10BaseT, Khoa Computer Science sử dụng Ethernet 100BaseT và Khoa System Engineering sử dụng Ethernet 10BaseT thì bridge có thể kết nối cả 3 segment trên. Với bridge Ethernet Gigabit có thể sử dụng đường truyền 1 Gbps nối tới router. Như đã đề cập ban đầu, hub không có tính năng có thể kết nối các công nghệ với tốc độ truyền khác nhau.

Khi sử dụng bridge làm thiết bị kết nối, về lý thuyết, LAN không bị giới hạn bởi phạm vi địa lý. Trên lý thuyết, có thể xây dựng mạng LAN trải rộng toàn cầu bằng kết nối các hub qua bridge. Theo thiết kế này, mỗi hub là một miền xung đột và do đó, LAN không bị giới hạn. Tuy nhiên, sau đây sẽ thấy rằng, trong mạng lớn, người ta sẽ kết nối qua router, chứ không sử dụng bridge.

### b) *Tự học (Self – Learning)*

Đặc tính tuyệt vời (nhất là đối với những người quản trị mạng) của bridge là khả năng tự học. Bảng lọc của bridge được xây dựng tự động mà

không cần bất cứ sự can thiệp nào từ phía người quản trị. Nói cách khác, bridge có khả năng tự học. Khả năng này được thực hiện như sau:

- ☞ Bảng bridge khởi đầu là rỗng.
- ☞ Khi frame đến cổng nào đó và địa chỉ đích của frame không có trong bảng, thì bridge sẽ chuyển frame đến bộ đệm ra của tất cả các cổng còn lại (tại mỗi cổng, frame được truyền lên LAN segment nhờ CSMA/CD).
- ☞ Khi nhận được frame, bridge lưu trữ: (1) địa chỉ vật lý trong trường địa chỉ nguồn của frame; (2) cổng nhận được frame; (3) thời gian hiện tại. Như vậy, bridge ghi nhớ được vị trí LAN segment của nút gửi. Nếu nút nào đó trong LAN gửi frame qua bridge thì bridge sẽ xác định được cổng để đi đến nút đó.
- ☞ Khi địa chỉ đích của frame có trong bảng, thì bridge chuyển frame đến cổng thích hợp.
- ☞ Bridge sẽ xoá địa chỉ trong bảng nếu adapter có địa chỉ đó không tiếp tục gửi frame trong khoảng thời gian xác định. Theo cách này, nếu PC được thay thế bởi PC khác (với adapter khác) thì địa chỉ vật lý của PC trước sẽ bị bridge xoá.

Xét quá trình tự học của bridge trong Hình 5.26 và bảng bridge tương ứng trong Hình 5.27. Giả sử rằng tại thời điểm 9:39 bridge nhận được một frame có địa chỉ gửi là 01-12-23-45-56 đến cổng 2. Giả sử, địa chỉ này chưa có trong bảng, bridge sẽ bổ sung một hàng mới trong bảng như chi ra trên Hình 5.28.

Tiếp tục với ví dụ này, giả sử rằng "tuổi thọ" của mỗi hàng trong bảng là 60 phút và máy tính với địa chỉ 62-FE-F7-11-89-A3 không gửi đi bất kỳ frame nào qua bridge trong khoảng thời gian từ 9:32 đến 10:32 thì lúc 10:32, bridge sẽ xoá địa chỉ này khỏi bảng.

Bridge là thiết bị theo kiểu "*cắm vào là chạy*" (plug and play), bởi vì nó không cần sự can thiệp của người quản trị mạng. Người quản trị mạng chỉ cần cắm connector vào các cổng của bridge. Người quản trị mạng không cần thiết lập cấu hình cho bảng bridge trong thời gian cài đặt hay khi máy tính

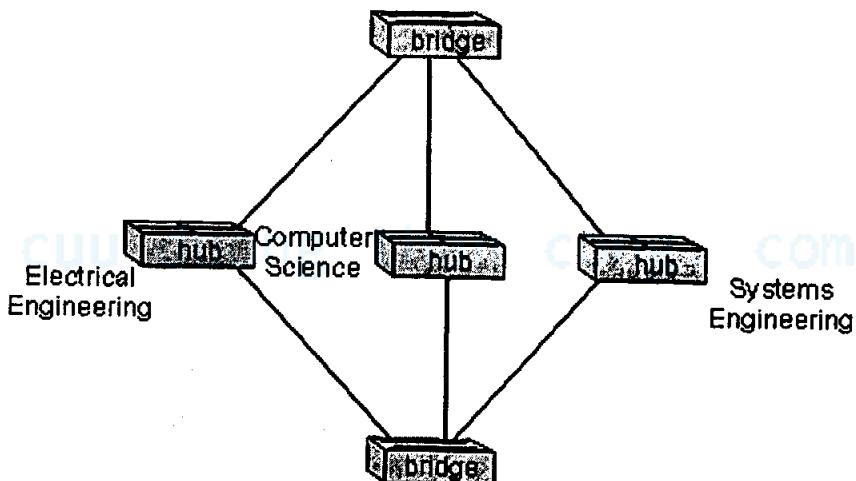
tách khỏi LAN segment. Do đặc tính này, bridge được xem là trong suốt (transparent).

Địa chỉ MAC	Interface	Time
01-12-23-34-45-56	2	9:39
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
...	...	...

Hình 5.28. Bảng lọc của bridge

### c) Spanning tree

Nếu hoàn toàn kết nối LAN segment theo kiểu phân cấp, thì khi hub hoặc bridge gần đỉnh bị hỏng, thì một phần lớn LAN sẽ không được kết nối. Chính vì lý do này, người ta thường xây dựng mạng với nhiều đường nối giữa các LAN segment. Một ví dụ về mạng như thế được minh họa trong Hình 5.29.



Hình 5.29. Kết nối dư thừa

Nhiều đường dư thừa giữa các LAN segment làm giảm khả năng sụp đổ của toàn bộ hệ thống. Nhưng có nhiều đường dẫn giữa các segment cũng sẽ phát sinh ra nhiều vấn đề – một frames có thể di chuyển vòng quanh hay được nhân bản lên nhiều lần trong mạng cục bộ. Để hình dung ra điều này, giả sử bảng bridge trong hình 5.26 rỗng và máy tính trong Khoa Electrical Engineering gửi frame đến máy tính trong Khoa Computer Science. Khi frame đến hub Electrical Engineering, hub sẽ sinh ra hai bản sao của frame và gửi mỗi bản đến hai bridge. Khi mỗi bridge nhận được frame, nó sẽ tạo ra 2 bản sao của frame, một bản gửi đến hub của Khoa System Engineering và bản kia đến hub của Khoa Computer Science. Vì cả hai bridge cùng làm như vậy, sẽ có 4 frame giống hệt nhau trong LAN. Frame có thể được nhân bản liên tục nếu bridge không biết nút nhận nằm ở đâu (chú ý rằng, để địa chỉ vật lý của máy tính nhận có trong bảng lọc, thì trước đó máy tính nhận phải gửi đi một frame qua bridge). Trong trường hợp này, số bản sao của frame gốc tăng theo hàm số mũ, làm tràn ngập toàn bộ mạng.

Để ngăn ngừa những tình huống nêu trên, bridge sử dụng giao thức *spanning tree*. Trong giao thức spanning tree, bridge truyền thông với bridge khác trên LAN để xác định spanning tree, nghĩa là, một tập con của topo ban đầu không có vòng lặp. Sau khi xác định được spanning tree, bridge chỉ kết nối với các cổng phù hợp để tạo spanning tree từ topo ban đầu. Ví dụ, trong hình 5.26, spanning tree được hình thành bằng cách bridge phía trên phong tỏa kết nối cổng kết nối đến Eletrical Engineering và bridge phía dưới phong tỏa cổng kết nối đến System Engineering. Với các cổng bị phong tỏa và loại bỏ được các vòng lặp, frame sẽ không lặp và nhân bản. Nếu khi nào đó một liên kết trong spanning tree bị lỗi, bridge có thể kết nối lại giao diện đã bị phong tỏa, kích hoạt thuật toán spanning tree lần nữa và xác định spanning tree mới.

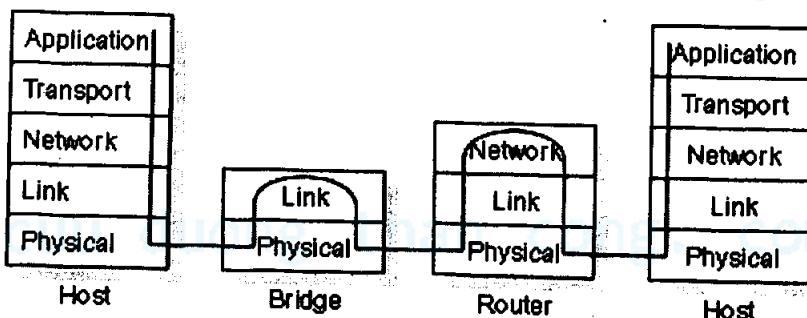
#### d) Phân biệt Bridge và Router

Như đã trình bày trong Chương 4, router là thiết bị chuyển mạch gói theo kiểu store – and – forward và chuyển gói tin theo địa chỉ tầng mạng. Mặc dù cũng là thiết bị chuyển mạch kiểu store – and – forward, nhưng điểm khác biệt cơ bản giữa bridge với router là bridge sử dụng địa chỉ vật lý.

Như vậy, router là thiết bị chuyển mạch gói ở tầng 3, trong khi bridge là thiết bị chuyển mạch gói ở tầng 2.

Người quản trị mạng sẽ phải lựa chọn giữa bridge và router khi cài đặt thiết bị kết nối. Ví dụ, với hệ thống mạng trong Hình 5.26, người quản trị mạng có thể lựa chọn router thay vì lựa chọn bridge. Thật vậy, router cũng sẽ có lập ba miền xung đột trong khi vẫn cho phép truyền thông giữa các khoa. Như vậy, cả bridge và router đều có thể làm thiết bị kết nối. Vậy ưu và nhược điểm giữa chúng là gì?

Trước tiên xét về bridge. Như đã đề cập trên, bridge là thiết bị kiểu "cắm vào là chạy" – tính năng được tất cả các nhà quản trị mạng ưa thích. Bridge cũng có tốc độ lọc và chuyển gói dữ liệu cao – như minh họa trên Hình 5.30, bridge chỉ phải xử lý gói dữ liệu của tầng 2 trong khi router phải xử lý gói dữ liệu của tầng 3. Mặt khác, giao thức spanning tree hạn chế topo của toàn bộ mạng. Điều này có nghĩa là, tất cả các frame chỉ được chuyển trên spanning tree, thậm chí khi có nhiều đường dẫn trực tiếp (nhưng bị phong tỏa) giữa nguồn và đích. Sự hạn chế của spanning tree cũng tập trung vào khả năng tải trên đường truyền spanning tree khi nó có thể đã được lan truyền đến tất cả các đường truyền khác của mạng cũ. Hơn nữa, bridge không đưa ra bất cứ sự bảo vệ nào để chống lại sự phát ra hàng loạt – nếu máy tính bị lỗi và truyền đi một luồng frame Ethernet liên tục, bridge sẽ chuyển tất cả những frame này, khiến toàn bộ mạng có thể bị sụp đổ.



Hình 5.30. Luồng đi của gói dữ liệu trong máy tính, bridge và router

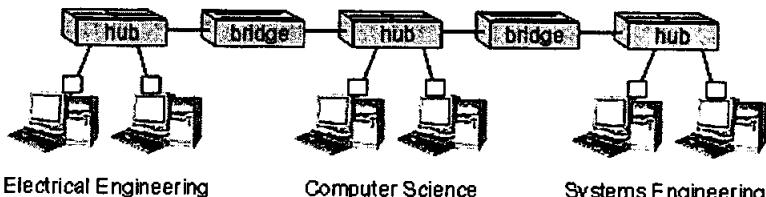
Bây giờ xét đến ưu và nhược điểm của router. Nói chung địa chỉ mạng thường phân cấp (không phẳng như địa chỉ vật lý), gói dữ liệu chắc chắn không quay vòng lại qua router ngay cả khi có nhiều đường đi (Thực sự gói

dữ liệu có thể quay vòng nếu cấu hình bảng định tuyến của router bị đặt sai, nhưng như đã biết trong Chương 4, IP sử dụng trường TTL trong tiêu đề gói dữ liệu để hạn chế chuyện này). Vì vậy, gói dữ liệu không bị giới hạn chuyển trong spanning tree, nó có thể sử dụng đường dẫn tốt nhất giữa nguồn và đích. Các router không bị giới hạn trong spanning tree, nên Internet có thể có topo rất phong phú, ví dụ, nhiều đường truyền giữa châu Âu và Bắc Mỹ. Một đặc tính quan trọng khác của router là tạo ra "firewall" (bức tường lửa) chống lại sự phát tán liên tục (quảng bá storm) ở tầng 2. Có lẽ yếu điểm duy nhất của router là không có khả năng "cắm vào là chạy" – cần cấu hình địa chỉ IP cho chúng và các máy tính kết nối đến chúng. Hơn nữa, thời gian xử lý gói tin của router thường lâu hơn bridge, vì chúng phải xử lý các trường tiêu đề của tầng 3.

Với cả ưu và nhược điểm như đã trình bày ở trên, vậy khi nào mạng sử dụng bridge, khi nào sử dụng router? Thông thường, một mạng nhỏ gồm vài trăm máy tính nằm trên một số LAN segment chỉ cần sử dụng bridge. Nhưng với những mạng lớn, gồm hàng nghìn máy tính sẽ cần tới nhiều router bên trong mạng (bên cạnh bridge). Những router này cung cấp khả năng cô lập mạnh hơn, kiểm soát việc gửi tràn ngập.

#### e) Kết nối LAN segment qua các trục chính (backbone)

Lại xét ví dụ kết nối mạng Ethernet trong 3 khoa trên Hình 5.26 với bridge. Thiết kế này sử dụng hai bridge, mỗi bridge có hai cổng. Bridge thứ nhất kết nối hai khoa Electrical Engineering và Computer Science. Bridge kia kết nối Khoa Computer Science với Systems Engineering. Mặc dù, bridge hai cổng rất phổ biến do giá rẻ và đơn giản, nhưng mô hình thiết kế trong Hình 5.31 không được ưa chuộng. Có hai lý do, thứ nhất, nếu hub của Computer Science bị hỏng thì máy tính ở hai khoa Electrical Engineering và Systems Engineering không thể trao đổi được với nhau. Thứ hai, truyền thông giữa hai khoa Electrical Engineering và System Engineering phải thông qua Computer Science, dễ gây xung đột trong LAN segment của Computer Science.



**Hình 5.31. Kết nối không có backbone**

Một nguyên tắc quan trọng định hướng dẫn việc kết nối các LAN segment khác là sử dụng đường trực chính (backbone) – là một mạng có kết nối trực tiếp đến tất cả các LAN segment. Khi LAN có một trực chính, thì mỗi cặp LAN segment có thể truyền thông mà không cần thông qua LAN segment thứ ba. Trong thiết kế ở hình 5.31, bridge ba cồng đóng vai trò một backbone.

### 5.6.3. Switch

Cho đến giữa những năm 90 của thế kỷ XX, ba loại thiết bị kết nối mạng cục bộ được sử dụng chủ yếu là hub (repeater), bridge và router. Gần đây, một thiết bị trở nên rất thông dụng là switch Ethernet. Switch Ethernet được hỗ trợ bởi ngành công nghiệp sản xuất thiết bị mạng. Về thực chất, switch là bridge nhiều cổng có hiệu suất cao. Giống bridge, switch chuyển và lọc frame căn cứ vào địa chỉ vật lý đích, tự động xây dựng bảng lọc khi có frame đi qua. Điểm khác biệt quan trọng nhất giữa bridge và switch là bridge có ít cổng (từ 2 đến 4) trong khi switch có thể có nhiều cổng hơn.

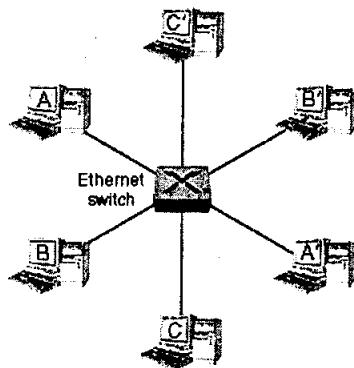
Có thể mua switch có các cổng với tốc độ khác nhau 10 Mbps, 100 Mbps và 1 Gbps. Ví dụ, một người có thể mua switch có bốn cổng 100 Mbps, hai mươi cổng 10 Mbps hoặc switch có bốn cổng 100 Mbps và một cổng 1 Gbps. Dĩ nhiên, nhiều cổng và tốc độ truyền của cổng càng cao thì switch càng đắt. Nhiều switch vận hành trong chế độ song công hoàn toàn, nghĩa là chúng có thể gửi và nhận frame tại cùng một thời điểm trên cùng một cổng. Với switch song công (cùng với bộ card mạng Ethernet song công trên các máy tính), máy tính A có thể gửi file đến máy tính B trong khi máy tính B gửi file đến máy tính A.

Ưu điểm của switch nhiều cổng là ở chỗ, dễ dàng kết nối trực tiếp giữa các máy tính với switch. Khi có đường kết nối trực tiếp song công với

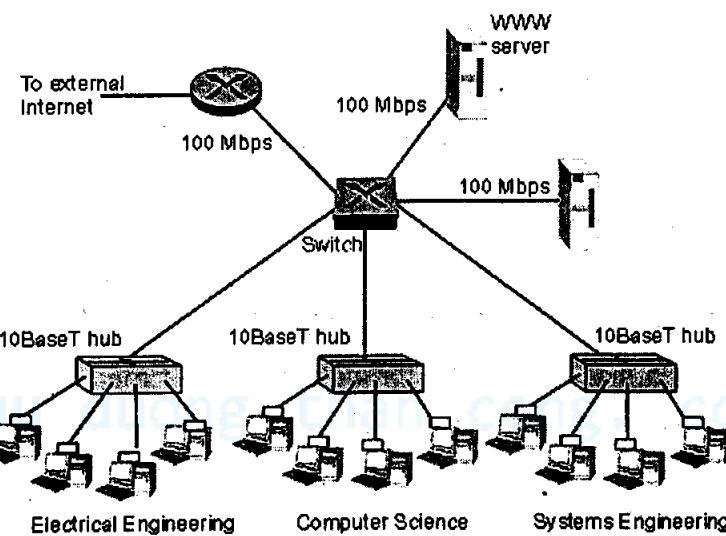
switch, máy tính có thể truyền (và nhận) frame ở tốc độ truyền tối đa của adapter, đặc biệt adapter máy tính luôn cảm nhận kênh truyền rồi và không bao giờ bị xung đột. Trong trường hợp này, máy tính được xem có *đường dùng riêng* (dedicated link).

Trên Hình 5.32 switch

Ethernet cung cấp đường dùng riêng cho 6 máy tính. Các đường truy cập dùng riêng này cho phép đồng thời A gửi file đến A', trong khi B đang gửi file đến B' và C đang gửi đến C'. Nếu mỗi máy tính sử dụng adapter card 10 Mbps, thì toàn bộ băng thông của hệ thống là 30 Mbps. Nếu A và A' có adapter 100 Mbps và những máy tính còn lại có adapter 10 Mbps, thì băng thông có thể lên tới 120 Mbps.



**Hình 5.32.** Switch Ethernet cung cấp truy cập Ethernet dành riêng đến 6 máy tính



**Hình 5.33.** Mạng cơ quan sử dụng phoi hợp hub, switch và router

Hình 5.33 minh họa cách kết nối một trường đại học với nhiều khoa và một số server quan trọng qua hub, switch và router. Trên Hình 5.33, mỗi khoa là một LAN segment sử dụng hub 10 Mbps. Vì mỗi hub có kết nối

đến switch nên các khoa hoàn toàn có khả năng trao đổi dữ liệu với nhau. Server cho dịch vụ Web và email đều có đường dùng riêng 100 Mbps đến switch. Cuối cùng, router sẽ kết nối toàn bộ hệ thống ra Internet và router cũng có đường dùng riêng 100 Mbps đến switch. Chú ý, switch này có ít nhất 3 cổng 10 Mbps và 3 cổng 100 Mbps.

### ***Chuyển mạch xuyên suốt (Cut – Through):***

Ngoài việc có nhiều cổng, hỗ trợ nhiều môi trường và tốc độ truyền khác nhau, có chức năng quản trị mạng, các nhà sản xuất switch Ethernet thường quảng bá sản phẩm của mình có khả năng gửi *xuyên suốt* (cut – through), chứ không phải kiểu store – and – forward như router và bridge. Khác biệt giữa store – and – forward và cut – through không lớn. Xét một gói dữ liệu chuyển qua thiết bị chuyển mạch (có thể là router, bridge hoặc switch). Gói tin đến switch từ một cổng nào đó và cần chuyển ra một cổng nào đó. Tại bộ đệm ở cổng ra của gói tin có thể có nhiều gói tin khác đang chờ chuyển. Khi đó store – and – forward và cut – through giống nhau. Hai công nghệ chuyển mạch này chỉ khác nhau khi bộ đệm cổng ra rỗng.

Khi gửi gói tin tới thiết bị chuyển kiểu store – and – forward, thiết bị sẽ thu và lưu trữ toàn bộ gói tin trước khi chuyển lên đường ra. Trong trường hợp bộ đệm ở cổng ra rỗng, phải thu thập toàn bộ gói tin rồi mới được chuyển đi (store and forward), việc này sẽ góp phần làm tăng thời gian trễ. Giới hạn độ trễ này là  $L/R$ , trong đó  $L$  là độ dài của gói tin và  $R$  là tốc độ truyền của cổng đến. Chú ý rằng, gói tin chỉ chịu độ trễ này nếu bộ đệm cổng ra rỗng trước khi toàn bộ gói tin đến switch.

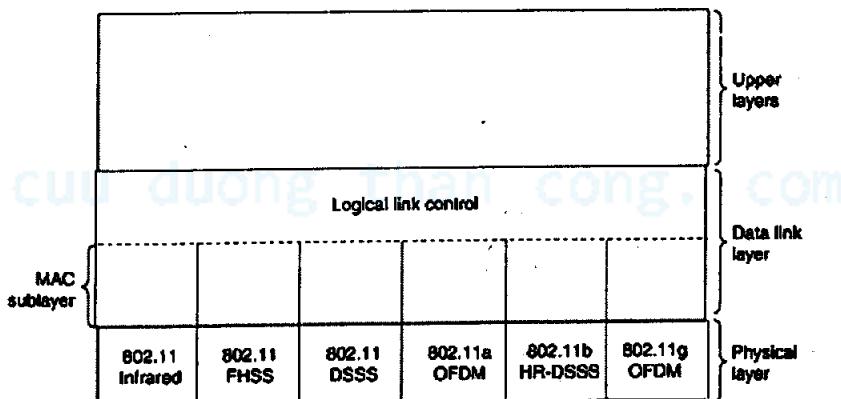
Với chuyển mạch kiểu cut – through, nếu buffer rỗng trước khi toàn bộ gói tin đến, switch có thể bắt đầu gửi đi phần trước trong khi đang nhận phần sau của gói tin. Tất nhiên, trước khi truyền gói tin trên cổng ra, phải xác định được trường địa chỉ đích (Thời gian trễ này không thể tránh khỏi đối với tất cả các loại chuyển mạch, vì switch phải xác định cổng ra thích hợp). Tóm lại, trong chuyển mạch kiểu cut – through, gói tin không cần được "lưu trữ" đầy đủ trước khi chuyển tiếp đi, mà gói tin sẽ được chuyển ngay khi cổng ra rỗng. Nếu cổng ra nối với mạng đa truy cập có chung môi

trường truyền với những máy tính khác (ví dụ nối đến hub), thì switch phải "lắng nghe" để kiểm tra kênh truyền có rỗi không trước khi chuyển.

## 5.7. MẠNG LAN KHÔNG DÂY

### 5.7.1. Giới thiệu chung

Các đặc tả về mạng LAN không dây ứng với tầng vật lý và tầng liên kết dữ liệu trong mô hình OSI. Thường tầng vật lý ở đây xác định việc sử dụng kênh truyền nào cũng như độ lớn của tín hiệu. Các chuẩn WLAN được IEEE đặc tả trong họ giao thức IEEE 802.11 chủ yếu sử dụng tia hồng ngoại, hoặc dải phổ tự do. Dải phổ này còn được gọi là *dải ISM* (Industry, Science, Medicine) dùng trong các ứng dụng công nghiệp, khoa học và y tế.



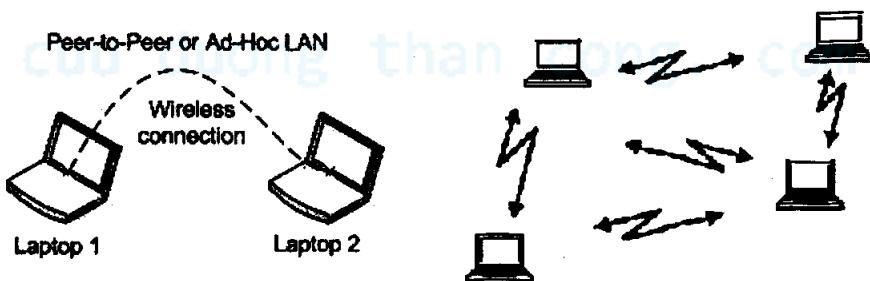
Hình 5.34. Vị trí các tầng con trong mạng LAN không dây

Chuẩn 802.11 đưa ra năm 1997, xác định ba kỹ thuật truyền được sử dụng ở tầng vật lý. Kỹ thuật hồng ngoại (giống như trong các thiết bị điều khiển tivi từ xa). Hai kỹ thuật còn lại sử dụng sóng radio có bước sóng ngắn là công nghệ FHSS và DSSS. Băng tần hai công nghệ này là băng tần tự do (2.4GHz ISM). Tốc độ truyền theo những công nghệ này tương đối thấp (1 – 2 Mbps) và có phạm vi phủ sóng khá bé (để giảm thiểu xung đột). Đến năm 1999, hai kỹ thuật mới có tốc độ cao hơn được đưa ra là OFDM và HR-DSSS, với tốc độ 54 Mbps, 11 Mbps tương ứng. Chú ý rằng, các kỹ thuật này ứng với tầng vật lý (Hình 5.34).

Nhiệm vụ của tầng liên kết dữ liệu ở đây là tổ chức việc truy cập đường truyền, đồng bộ hóa frame, kiểm soát tài nguyên. Cơ chế truy cập thực hiện chức năng điều phối phân tán (DCF – Distributed Coordination Function) được cài đặt ở tất cả các trạm (máy tính) có thiết bị thu phát không dây. Có hai kiểu thiết bị chính là trạm di động (mobile station, chẳng hạn laptop) và điểm truy cập (Access Point – AP, có nhiệm vụ chuyển tiếp dữ liệu giữa các trạm di động hay giữa trạm di động với mạng LAN cố định). Có hai kiểu kết nối của mạng LAN không dây là kiểu có cơ sở hạ tầng và kiểu không có cơ sở hạ tầng (adhoc).

#### a) Adhoc WLAN

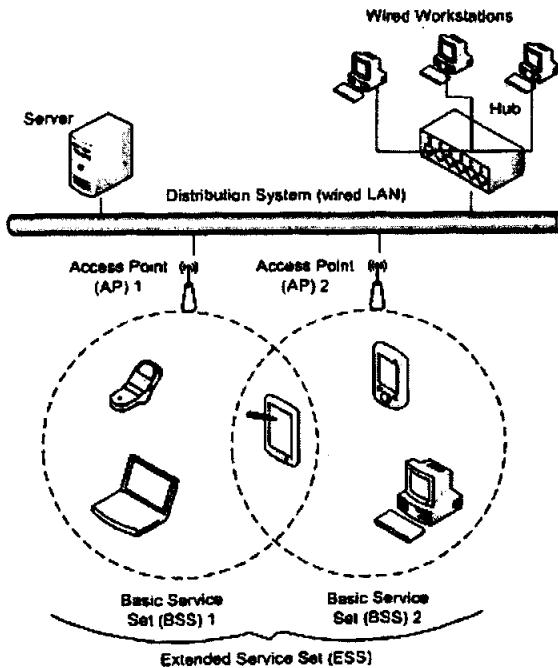
Trong kiểu kết nối adhoc, hai trạm kết nối trực tiếp với nhau không qua AP (Hình 5.35). Trong kiểu kết nối này, các trạm làm việc có thể đóng vai trò server để chuyển tiếp dữ liệu. Hai trạm làm việc vẫn có thể trao đổi dữ liệu ngay cả khi chúng không nhận tín hiệu trực tiếp từ nhau.



Hình 5.35. Mạng Adhoc LAN

#### b) WLAN có cơ sở hạ tầng

Trong kiểu kết nối này (Hình 5.36), hai trạm chỉ có thể trao đổi dữ liệu với nhau qua AP. Thông thường, AP được đính vào tường (trong một văn phòng) và có kết nối với mạng LAN cố định. Bên cạnh chức năng chính là trung chuyển các gói tin giữa các trạm và giữa trạm với mạng LAN có dây, AP có thể thực hiện chức năng điều phối điểm (point coordinating function – PCF), cho phép truyền thông giữa các nút dựa trên việc cấp phát tài nguyên (thời gian, băng thông,...).



**Hình 5.36. Mạng LAN có cấu trúc**

Đa phần tầng MAC trong họ giao thức IEEE WLAN có cơ chế truy cập tương tự mạng Ethernet. Ethernet sử dụng đa truy cập sóng mang có phát hiện xung đột (CSMA/CD). Tuy nhiên, đối với mạng không dây, việc phát hiện xung đột không hiệu quả do độ suy hao năng lượng của tín hiệu theo khoảng cách rất lớn. Thay vì sử dụng cơ chế phát hiện, đa phần các mạng không dây sử dụng cơ chế tránh xung đột (collision avoidance). Ý tưởng chính của cơ chế này là đảm bảo giữa hai lần truyền gói tin liên tiếp phải có một khoảng thời gian tối thiểu.

### 5.7.2. Lớp giao thức IEEE 802.11

Trong họ giao thức 802.11, tầng MAC xác định cách thức phân phối kênh truyền – tức là, tại một thời điểm, nút nào được quyền truyền dữ liệu. Tầng phía trên của MAC – tầng LLC (Logical Link Control) có nhiệm vụ "che dấu" các đặc điểm khác nhau của tầng MAC với tầng mạng.

#### a) *Tầng con Medium Access Control (MAC)*

Chức năng chính của mạng LAN là chia sẻ tài nguyên giữa các máy tính trong một khu vực nhỏ, do vậy, giống như Ethernet, nhiệm vụ của tầng

MAC là xác định cụ thể từng giao diện của các máy tính trên mạng. Bên cạnh chức năng chính tổ chức truy cập kênh truyền và cơ chế điều phối đa truy cập, MAC còn có một số chức năng chính sau đây:

- ☞ **Cấu hình Mạng:** Đối với mạng LAN có dây, cấu hình mạng chính là topo mạng và đã được xác lập bằng cách nối dây. Tuy nhiên, đối với mạng không dây, đây lại là việc quan trọng, vì phải ghép nối các máy tính thành các "nhóm" có thể định danh được.
- ☞ **Truy cập kênh truyền:** Đối với mạng LAN, cơ chế truy cập môi trường truyền nhanh, tin cậy và công bằng hết sức cần thiết.

#### b) *Đặc tả tầng MAC theo IEEE 802.11*

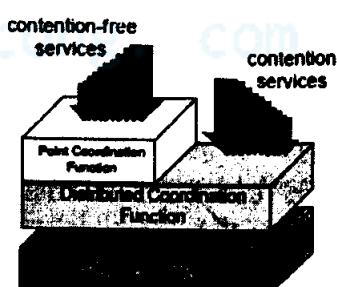
Chức năng truy cập kênh truyền của mạng LAN không dây khá tương đồng với mạng Ethernet. Đặc tả này có một vài chức năng để đảm bảo tính an ninh và sẽ được bổ sung thêm một vài tính năng để sau này có thể tích hợp được với mạng điện thoại không dây.

#### c) *Cấu hình mạng*

Chuẩn IEEE 802.11 định nghĩa BSA (Basic Service Area) là một khu vực có thể có nhiều máy tính (trạm) di động. Các máy tính trong BSA được gọi là *Service Set* (BSS). Các trạm làm việc trong BSS được kết nối với Access Point (AP). Nhiều BSS kết nối với nhau thành Distributed System (SS) để tạo thành Extended Service Set (ESS). Tất cả các trạm trong cùng BSS sử dụng chung một tốc độ truyền (Basic Rate Set) và một cấu trúc gói dữ liệu PDU (xem minh họa trên Hình 5.36).

#### d) *Truy cập kênh truyền trong IEEE 802.11*

Chuẩn IEEE 802.11 định nghĩa cả tầng MAC lẫn tầng vật lý (Hình 5.347). Tầng MAC cung cấp hai dịch vụ là *dịch vụ truy cập có tranh chấp* (Distributed Coordination Function – DCF) và *dịch vụ truy cập không tranh chấp* (cài đặt ở Point Coordination Function – PCF). Những dịch vụ này được triển khai trên

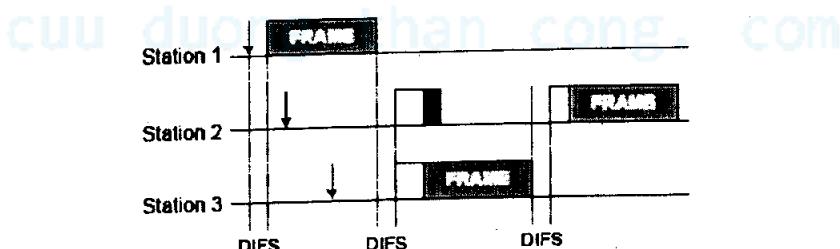


Hình 5.37. Cơ chế điều phối DCF

nền tầng vật lý, che dấu hết các đặc tính kỹ thuật của tầng vật lý đối với tầng mạng.

DCF là phương pháp chủ yếu trong họ IEEE 802.11 và sử dụng cơ chế *đa truy cập kênh truyền có tránh tắc nghẽn* (CSMA/CA). PCF được cài đặt trên nền DCF và dựa trên cơ chế hỏi vòng. Cơ chế này cho phép một trạm đóng vai trò trung tâm lần lượt "hỏi" các trạm trong mạng có muốn truyền dữ liệu không.

Trước khi truyền frame, trạm phải "lắng nghe" kênh truyền để xem có trạm khác chiếm dụng kênh truyền hay không. Nếu kênh truyền rỗi trong một khoảng thời gian lớn hơn Distributed InterFrame Space (DIFS), thì trạm được phép truyền (Hình 5.38). Nếu ngược lại (môi trường truyền bị chiếm dụng), việc truyền của trạm bị trì hoãn đến phiên truyền tiếp theo. Khoảng thời gian trì hoãn này (backoff time) được lựa chọn một cách ngẫu nhiên và được sử dụng để khởi tạo cho backoff timer. Giá trị bộ định thời này giảm khi kênh truyền được cảm nhận là rỗi, sẽ giữ nguyên trong trường hợp kênh truyền bận và sẽ được giảm về 0 (reset) khi kênh truyền rỗi trong khoảng thời gian lớn hơn DIFS. Ví dụ trên Hình 5.38 ta thấy rằng, backoff timer của trạm 2 bị dừng lại trong khi trạm 3 đang truyền. Giá trị timer này sẽ được đặt về 0 sau khi trạm 3 truyền xong một khoảng thời gian DIFS. Trạm 2 sẽ được phép truyền khi backoff timer của mình nhận giá trị 0. Giá trị backoff time cũng được chia khoảng. Cụ thể hơn, giá trị này là một số nguyên ngẫu nhiên nằm trong khoảng  $(0, CW - 1)$ , trong đó CW (Contention Window) là độ lớn cửa sổ backoff (Backoff Window). Trong lần thử truyền đầu tiên,  $CW = CW_{min}$  và cứ sau mỗi lần truyền thử, giá trị này tăng gấp đôi cho đến khi đạt đến giá trị  $CW_{max}$ .

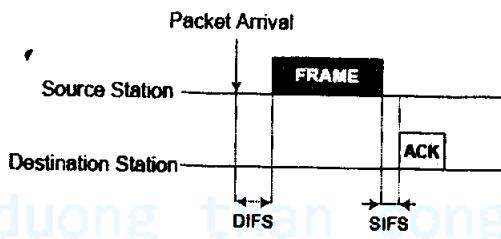


Hình 5.38. Cơ chế tránh tắc nghẽn

Theo chuẩn, giá trị của  $CW_{min}$  và  $CW_{max}$  phụ thuộc vào tầng vật lý phía dưới. Ví dụ, với công nghệ FHSS ở tầng vật lý, giá trị của  $CW_{min}$  và  $CW_{max}$  tương ứng là 16 và 1024.

Rõ ràng, hoàn toàn có khả năng tại cùng một thời điểm có nhiều trạm có nhu cầu truyền tin, khi đó xung đột chắc chắn xuất hiện. Với CSMA/CA trạm không có khả năng phát hiện xung đột bằng cách "lắng nghe" môi trường truyền giống như trong cơ chế CSMA/CD của mạng Ethernet. Do đó, để chắc chắn truyền thành công, phía nhận phải gửi phản hồi tích cực. Sau khi nhận đúng một frame, phía nhận đợi một khoảng thời gian Short InterFrame Space (SIFS) và gửi lại frame biên nhận ACK. Để ưu tiên việc gửi frame biên nhận hơn việc gửi frame dữ liệu bình thường khác, giá trị SIFS bé hơn DIFS (Hình 5.39). Nếu phía nhận không nhận được ACK, gói tin gửi đi sẽ bị coi là mất và sau một khoảng thời gian sẽ được gửi lại. Để phát hiện lỗi, người ta cũng sử dụng cơ chế CRC đã được trình bày ở trên.

Sau khi phát hiện được frame bị lỗi (do xung đột hay lỗi trên môi trường truyền), trạm phải đợi ít nhất một khoảng thời gian bằng *Extended InterFrame Space* (EIFS) trước khi khởi động thuật toán backoff. Cụ thể, DCF phải sử dụng giá trị EIFS mỗi khi tầng vật lý báo cho tầng MAC frame bắt đầu truyền không thể gửi thành công. Giá trị EIFS cũng được sử dụng để đồng bộ trạm làm việc với trạng thái của môi trường truyền (bận hay rỗng).

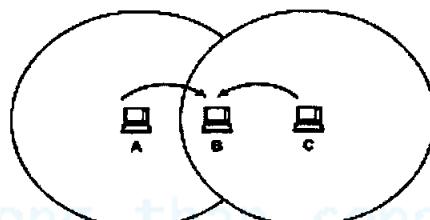


Hình 5.39. Gửi biên nhận khi nhận được frame dữ liệu

### 5.7.3. Một số vấn đề hay gặp đối với mạng không dây

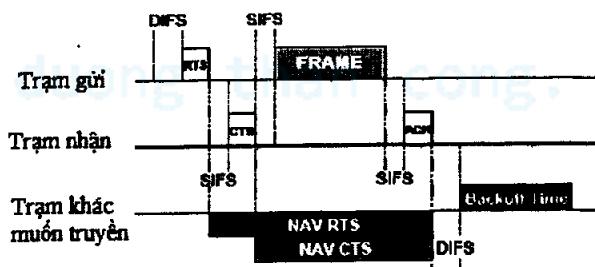
Trong phần này sẽ nêu một số vấn đề này sinh trong mạng không dây, đây là những đặc điểm cơ bản khác với mạng có dây:

- ☞ Không có tiêu chí cụ thể để xác định phạm vi môi trường truyền. Có nghĩa là, không thể xác định được phạm vi mà nút nằm trong đó chắc chắn nhận được frame, nằm ngoài phạm vi đó sẽ không nhận được.
- ☞ Môi trường truyền có thể bị nhiễu vì tín hiệu bên ngoài.
- ☞ Độ tin cậy kém môi trường có dây.
- ☞ Độ trễ trên môi trường truyền không cố định.
- ☞ Trong môi trường không dây, cơ chế truy cập ngẫu nhiên dựa vào cảm nhận kênh truyền, những đặc tính nêu trên có thể gây nên những hiện tượng phức tạp, chẳng hạn "trạm ẩn" (hidden station) hay "trạm lộ" (exposed-station).



Hình 5.40. Vấn đề trạm ẩn

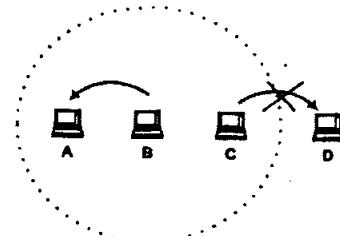
Hình 5.40 minh họa tình huống "trạm ẩn": trạm B nằm giữa miền phủ sóng của A và C. Nhưng A và C không gửi trực tiếp dữ liệu được tới nhau. Giả sử trong lúc A bắt đầu gửi dữ liệu tới B thì C cũng có dữ liệu gửi tới B. Khi C thực hiện cảm nhận môi trường truyền, thì tất nhiên không phát hiện dữ liệu do A gửi. Vì vậy, C coi kênh truyền rõ và bắt đầu truyền. Tín hiệu từ A và C khi đến B sẽ bị xung đột



Hình 5.41. Cơ chế báo gửi và báo nhận

Cả hai frame RTS và CTS đều được truyền trong những khoảng thời gian đủ để truyền đi một frame, nên tất cả các trạm trong vùng phủ sóng đều có thể nhận được các frame kiểu này và cập nhật giá trị *Network Allocation Vector* (NAV) tương ứng. Khi NVA dương, trạm không được gửi dữ liệu. Bằng cách sử dụng cơ chế RTS/CTS, trạm có thể nhận biết được sự truyền của các "trạm ẩn" cũng như khoảng thời gian truyền này.

Hình 5.42 minh họa tình huống "trạm lộ". Giả sử trạm A và C đều có thể nghe trạm B, nhưng trạm A không nghe được trạm C. Giả sử lúc B đang truyền cho A thì C nhận được một frame để truyền tới D. Rõ ràng, C cảm nhận kênh truyền và thấy kênh truyền bận (do B truyền tới A), C sẽ trì hoãn việc truyền tới D, mặc dù việc này sẽ không gây xung đột tại A, điều này làm giảm thông lượng hệ thống.



Hình 5.42. Trạm lộ

## 5.8. PPP – GIAO THỨC ĐIỀM NỐI ĐIỀM

Phần trước đã nói về kênh truyền quảng bá. Phần này sẽ trình bày giao thức liên kết dữ liệu cho kênh truyền điểm nối điểm (point – to – point) – giao thức PPP. PPP là giao thức được sử dụng chủ yếu khi người dùng truy cập Internet từ nhà thông qua đường điện thoại quay số, do đó PPP là một trong những giao thức tầng liên kết dữ liệu được sử dụng nhiều nhất hiện nay. Giao thức quan trọng thứ hai là HDLC (High Level Data – Link Control [Spragins 1991]). Giao thức PPP được trình bày tương đối đơn giản, với mục đích khảo sát một số tính năng quan trọng nhất của lớp giao thức điểm nối điểm ở tầng liên kết dữ liệu.

Giao thức PPP [RFC 1661; RFC 2153] là giao thức tầng liên kết dữ liệu trên kênh truyền nối trực tiếp giữa hai nút – mỗi nút ở một đầu của đường truyền. Đường truyền PPP có thể là đường điện thoại quay số (ví dụ, kết nối modem 56K), đường truyền SONET/SHD, kết nối X.25 hoặc mạch ISDN. Như đã nói trên, PPP chủ yếu được lựa chọn để kết nối máy tính gia đình

đến ISP thông qua đường dây điện thoại. Trước khi đi sâu vào chi tiết của PPP, hãy điểm qua một số quy tắc chính mà IETF đã đặt ra cho mọi thiết kế của PPP [RFC 1547]:

- ☞ **Đóng gói gói tin (Framing):** Phía gửi trong giao thức PPP phải có khả năng lấy gói tin ở tầng mạng, đặt nó trong frame tầng liên kết dữ liệu. Phía nhận xác định được vị trí bắt đầu và kết thúc của frame cũng như vị trí gói tin tầng mạng trong frame.
- ☞ **Tính trong suốt:** Giao thức PPP không được đặt ra bất kỳ hạn chế nào trên gói dữ liệu tầng mạng. Tức là nó có khả năng chuyển đi bất kỳ gói dữ liệu tầng mạng nào.
- ☞ **Hỗ trợ nhiều giao thức tầng mạng:** Giao thức PPP phải có khả năng hỗ trợ nhiều giao thức tầng mạng (ví dụ, IP và DECnet) trên cùng đường truyền vật lý tại cùng một thời điểm. Điều này cũng giống như giao thức IP có khả năng phân kênh cho nhiều giao thức giao vận khác nhau (ví dụ, TCP và UDP). Như vậy, PPP cũng cần có một cơ chế để thực thể PPP phía nhận xác định được cần chuyển gói dữ liệu trong frame cho thực thể tầng mạng nào.
- ☞ **Hỗ trợ nhiều kiểu đường truyền:** Ngoài khả năng hỗ trợ nhiều giao thức ở tầng cao hơn, PPP phải có khả năng vận hành trên nhiều kiểu đường truyền khác nhau, bao gồm đường truyền tuần tự (truyền lần lượt từng bit một), hoặc song song (truyền nhiều bit cùng một lần), đồng bộ (truyền tín hiệu đồng hồ cùng với bit dữ liệu), hoặc dị bộ (truyền với tốc độ chậm hoặc cao, tín hiệu điện tử hoặc quang học).
- ☞ **Phát hiện lỗi:** PPP phía nhận có khả năng phát hiện liệu có lỗi bit trong frame nhận được hay không.
- ☞ **Thời gian kết nối:** PPP phải có khả năng phát hiện đường truyền bị lỗi ở mức link (ví dụ, không có khả năng để truyền dữ liệu từ phía gửi sang phía nhận) và phải thông báo tình trạng lỗi này cho tầng mạng.

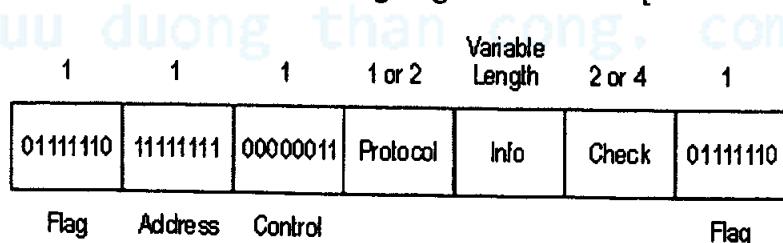
- ☞ **Thoả thuận địa chỉ tầng mạng:** PPP phải cung cấp cơ chế cho phép hai thực thể tầng mạng tham gia truyền thông (IP) có thể "học" hay đặt cấu hình địa chỉ tầng mạng cho nhau.
- ☞ **Đơn giản:** Người ta đòi hỏi PPP đáp ứng nhiều yêu cầu ngoài những yêu cầu nêu trên. Một trong những yêu cầu quan trọng nhất là "tính đơn giản". Hiện nay hơn 50 RFC định nghĩa những khía cạnh "đơn giản" của giao thức này.

Tuy vậy các đặc tả trong thiết kế PPP không yêu cầu:

- ☞ **Sửa lỗi:** PPP cần phát hiện được lỗi bit nhưng không cần phải sửa lỗi.
- ☞ **Kiểm soát lưu lượng:** PPP phía nhận được hy vọng có khả năng nhận frame với tốc độ cao nhất của tầng vật lý phía dưới. Nếu tầng mạng không thể nhận với tốc độ này, thì việc loại bỏ gói tin hay yêu cầu bên kia truyền chậm lại là trách nhiệm của các tầng cao hơn. Khi đó, tầng cao hơn sẽ yêu cầu thực thể tương đương phía bên kia giảm tốc độ tạo ra dữ liệu gửi cho PPP.
- ☞ **Đánh số thứ tự:** PPP không được yêu cầu chuyển frame đến phía nhận theo đúng thứ tự gửi.
- ☞ **Đường truyền đa điểm:** PPP vận hành trên những đường truyền với một phía gửi và một phía nhận duy nhất. Một số giao thức tầng liên kết dữ liệu khác (ví dụ, HDLC) cho phép nhiều nút nhận trên cùng một đường truyền (giống Ethernet).

### 5.8.1. Khuôn dạng gói dữ liệu (Frame PPP)

Hình 5.43 minh họa frame PPP giống frame HDLC [RFC 1662].



Hình 5.43. Khuôn dạng frame dữ liệu PPP

Frame PPP bao gồm các trường sau:

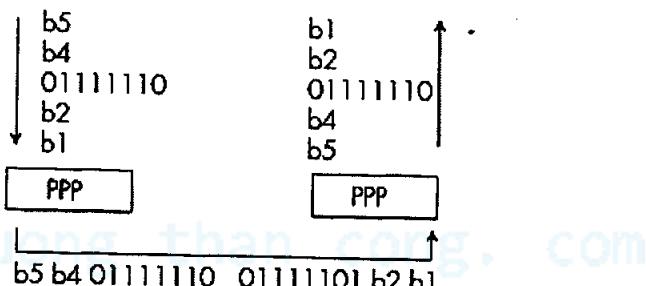
- ☛ **Trường cờ:** Mọi frame PPP bắt đầu và kết thúc bằng một byte cờ có giá trị 01111110.
- ☛ **Trường địa chỉ:** Giá trị duy nhất của trường này là 11111111.
- ☛ **Trường điều khiển:** Giá trị duy nhất của trường này là 00000011. Bởi vì, cả hai trường địa chỉ và điều khiển đều mang những giá trị cố định, điều này giải thích vì sao những trường này được định nghĩa đầu tiên. Khuyến nghị PPP [RFC 1662] nói rõ rằng, những giá trị này "có thể được định nghĩa sau này". Bởi vì, những trường này mang giá trị cố định, PPP cho phép phía gửi không cần gửi byte địa chỉ và byte điều khiển, do đó tiết kiệm được hai byte tiêu đề trong frame PPP.
- ☛ **Trường giao thức (protocol):** Trường giao thức cho PPP xác định giao thức tầng trên sẽ nhận dữ liệu trong frame PPP. Khi nhận được frame PPP, bên nhận sẽ kiểm tra xem frame có lỗi không và sau đó chuyển phần dữ liệu trong gói tin cho giao thức thích hợp. RFC 1700 định nghĩa mã 16 bit cho các giao thức được sử dụng cùng với PPP. Giao thức IP (dữ liệu trong frame PPP là gói dữ liệu IP datagram) ứng với giá trị 21h, giao thức AppleTalk là 29h, DFCnet là 27h, giao thức điều khiển đường truyền PPP là C021h và giao thức điều khiển IP là 8021. Giao thức IP Control được PPP sử dụng khi kích hoạt kênh truyền lần đầu tiên để cấu hình IP giữa các thiết bị trên hai đầu kênh truyền.
- ☛ **Thông tin:** Trường này chứa gói tin được giao thức tầng mạng gửi đi trên đường truyền PPP (là IP datagram với giao thức IP). Độ dài lớn nhất của trường thông tin này là 1500 byte, mặc dù giá trị này có thể thay đổi lúc đặt cấu hình cho đường truyền.
- ☛ **Tổng kiểm tra (Checksum):** Trường checksum được sử dụng để phát hiện các bit bị lỗi trong frame nhận được. Nó là mã CRC 2 hoặc 4 byte giống như trong giao thức HDLC.

### **Chèn byte (Byte stuffing):**

Trước khi kết thúc việc trình bày về PPP frame, xét vấn đề phát sinh khi trong gói dữ liệu của giao thức tầng mạng lại có một byte giống byte cờ đánh dấu điểm bắt đầu và kết thúc của frame. Điều gì xảy ra nếu phát hiện thấy có byte cờ với giá trị 01111110 ở giữa gói tin trong trường thông tin. Bên nhận sẽ cho rằng, đây là điểm kết thúc của frame PPP – mặc dù trên thực tế không phải như vậy.

Một cách để giải quyết vấn đề này là cấm các giao thức tầng trên gửi dữ liệu chứa byte cờ. Yêu cầu về tính trong suốt của PPP đã nêu ở trên không chấp nhận giải pháp này. Giải pháp thay thế được PPP cũng như nhiều giao thức khác áp dụng là kỹ thuật *chèn byte* (byte stuff).

PPP định nghĩa byte điều khiển đặc biệt có giá trị 01111101 làm nhiệm vụ đánh dấu. Nếu byte cờ – 01111110 xuất hiện trong frame (trừ vị trí mở đầu và kết thúc của frame), PPP đặt byte đánh dấu trước byte cờ. Như vậy, nó đã "chèn" thêm một byte điều khiển để đánh dấu rằng byte 01111110 không phải là cờ mà là dữ liệu thực. Bên nhận thấy 01111110 đứng trước 01111101 nên biết được 01111101 không phải là cờ mà là dữ liệu, nên nó sẽ tự động loại bỏ byte đánh dấu 01111110 được phía nhận chèn vào bên cạnh dòng dữ liệu thực.



**Hình 5.44. Chèn byte trong byte**

Hình 5.44 minh họa chèn byte trong PPP. Tương tự như thế, nếu chính byte đánh dấu cũng xuất hiện trong dòng dữ liệu thực sự, thì nó cũng cần được đánh dấu.

## 5.8.2. Giao thức điều khiển đường truyền PPP (LCP) và kiểm soát mạng

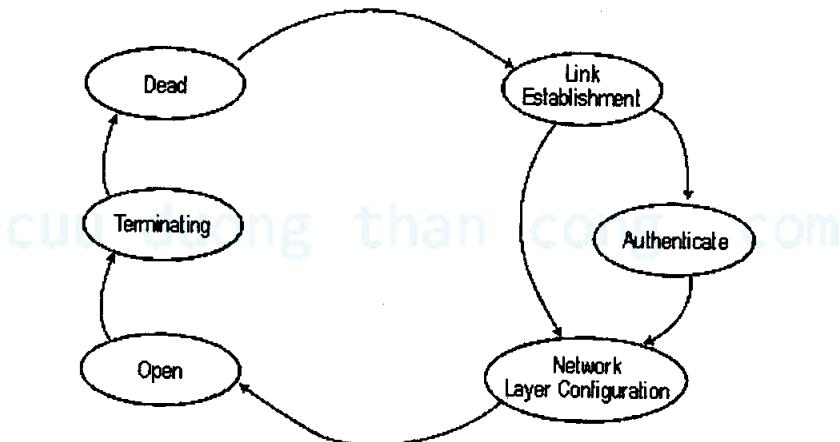
Trong phần trước, ta đã thấy PPP đóng khung dữ liệu được gửi đi trên đường truyền. Nhưng đường truyền được khởi tạo như thế nào khi máy tính hoặc router ở một phía của đường truyền bật trước? Quá trình khởi tạo, duy trì, báo lỗi và đóng đường truyền PPP được thực hiện nhờ giao thức điều khiển đường truyền (LCP –Link Control Protocol) và các giao thức điều khiển mạng của PPP.

Trước khi trao đổi bất kỳ dữ liệu nào trên đường truyền PPP, hai phía (mỗi phía ở một đầu đường truyền) phải thực hiện nhiều công việc để thiết lập cấu hình cho đường truyền, điều này cũng tương tự như thực thể TCP bên gửi và bên nhận thực hiện bắt tay ba bước (xem mục 3.5) để đặt các tham số của kết nối TCP trước khi trao đổi TCP segment. Hình 5.45 minh họa biểu đồ chuyển trạng thái của giao thức LCP để đặt cấu hình, duy trì và kết thúc đường truyền PPP.

Đường truyền PPP bắt đầu và kết thúc trong trạng thái đóng (dead). Khi phát sinh sự kiện như phát hiện sóng mang, hay người quản trị mạng tác động để chỉ tầng vật lý sẵn sàng sử dụng, PPP bước sang trạng thái thiết lập đường truyền (link establishment). Trong trạng thái này, một phía của đường truyền gửi tùy chọn cấu hình mong muốn qua frame yêu cầu cấu hình LCP (là frame PPP có giá trị trường protocol ứng với giao thức LCP và trường information chứa nội dung cấu hình yêu cầu). Sau đó, phía bên kia trả lời với frame *configure – ack* (chấp nhận tất cả các lựa chọn), frame *configure – nak* (hiểu nhưng không chấp nhận các lựa chọn) hoặc frame *configure – reject* (không thể ghi nhận hoặc chấp nhận các lựa chọn để đàm phán). Tuỳ chọn cấu hình LCP bao gồm kích thước tối đa của frame trên đường truyền, giao thức kiểm chứng được sử dụng (nếu cần) và một tuỳ chọn xác định có bỏ qua việc sử dụng trường địa chỉ và trường điều khiển trong frame PPP hay không.

Sau khi đường truyền được thiết lập, thoả thuận xong các tuỳ chọn của đường truyền và kiểm chứng thành công, hai phía của đường truyền PPP sẽ

trao đổi các gói tin kiểm soát của tầng mạng. Nếu IP chạy phía trên PPP, giao thức điều khiển IP [RFC 1332] được sử dụng để thiết lập cấu hình cho module giao thức IP tại mỗi đầu của đường truyền PPP. Gói tin IPCP được đặt trong frame PPP. IPCP cho phép hai module IP thay đổi hoặc đặt cấu hình địa chỉ IP hay thoả thuận có nén dữ liệu IP hay không. Những giao thức kiểm soát mạng tương tự được đưa ra cho những giao thức tầng mạng khác như DECnet [RFC 1762] và AppleTalk [RFC 1378]. Sau khi cấu hình xong tầng mạng, PPP có thể bắt đầu gửi gói tin của tầng mạng – đường truyền ở trạng thái mở và dữ liệu bắt đầu chuyển trên đường truyền PPP. Các frame yêu cầu phản hồi và frame trả lời phản hồi LCP có thể được hai phía của đường truyền trao đổi để kiểm tra trạng thái đường truyền.



**Hình 5.45. Sơ đồ chuyển trạng thái của LCP**

Đường truyền PPP được duy trì cho đến khi gói tin LCP yêu cầu kết thúc được gửi đi. Nếu frame LCP yêu cầu kết thúc (terminate – request) từ một phía kết nối được trả lời bởi frame, LCP chấp nhận kết thúc (terminate – ack) từ phía bên kia thì đường truyền bước vào trạng thái đóng.

Tóm lại, PPP là giao thức tầng liên kết dữ liệu cho hai thiết bị ở hai đầu của một đường truyền kiểu điểm nối điểm, trao đổi các frame chứa gói dữ liệu của tầng mạng. Những chức năng chủ yếu của PPP là:

- ☛ **Đóng gói dữ liệu:** Phương thức đặt gói dữ liệu trong frame PPP, xác định vị trí bắt đầu và kết thúc của frame và phát hiện lỗi trong frame.

- ☞ **Giao thức điều khiển đường truyền:** Khởi tạo, duy trì và kết thúc đường truyền PPP.
- ☞ **Giao thức điều khiển mạng:** Một nhóm giao thức, mỗi giao thức ứng với một giao thức mạng ở tầng trên, cho phép module tầng mạng tự đặt cấu hình trước khi gói dữ liệu tầng mạng bắt đầu chuyên qua đường truyền PPP.

cuu duong than cong. com

cuu duong than cong. com

## Tài liệu tham khảo

---

- [1]. J. Kurose and K. Ross, *Computer Networking: A Top – Down Approach Featuring the Internet*", Addison – Wesley, 3rd edition, May, 2004.
- [2]. Andrew S. Tanenbaum, Computer Networks, *Prentice Hall PTR*, 2004.
- [3]. S.Keshav, *An Engineering Approach to Computer Networking*, Addison – Wesley, 2003.
- [4]. Jean Walrand, *Communication Networks: A first cost*, second edition, McGraw Hill, 2003.
- [5]. Douglas E. Comer, *Computer Networks and Internet with Internet Application*, 3rd edi, Prentice Hall, 2003.
- [6]. Michael A. Gallo and William M. Hancock, *Computer Communications and Network Technologies*, Brooks/Cole Thomson Learning, 2003.
- [7]. William Stalling, *Data and Computers Communications*, 6<sup>th</sup> edition, Prentice Hall, 2003.
- [8]. Behrouz A. Forouzan, *Data Communications and Networking*, McGraw Hill, 2003
- [9]. Ramesh Subramanian and Brian D. Goodman, *Peer To Peer Computing The Evolution Of A Disruptive Technology*, IDEA Group Publishing, 2006.
- [10]. Wei Luo, *Layer 2 VPN Architectures*, Cisco Press, 2006.

*Chịu trách nhiệm xuất bản :*

Chủ tịch HĐQT kiêm Tổng Giám đốc NGÔ TRẦN ÁI  
Phó Tổng Giám đốc kiêm Tổng biên tập NGUYỄN QUÝ THAO

*Biên tập nội dung và sửa bản in :*

ĐỖ HỮU PHÚ

*Thiết kế mỹ thuật và trình bày bìa :*

BÍCH LA

*Thiết kế sách và chế bản :*

ĐỖ PHÚ

cuu duong than cong. com

cuu duong than cong. com

---

## **Giáo trình NHẬP MÔN MẠNG MÁY TÍNH**

**Mã số: 7B752Y9 – DAI**

In 1.000 bản (QĐ : 37), khổ 16 x 24 cm. In tại Nhà in Đại học Quốc Gia Hà Nội.  
Địa chỉ : 16 Hàng Chuối, Hà Nội.

Số ĐKKH xuất bản : 375 – 2009/CXB/7 – 726/GD.

In xong và nộp lưu chiểu tháng 6 năm 2009.