

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

NGUYỄN NGỌC ĐIệp

BÀI GIẢNG
CÁC GIAO THỨC CỦA INTERNET

HÀ NỘI 2014

GIỚI THIỆU

Hiện nay, các công nghệ liên quan đến mạng và liên mạng đang phát triển vô cùng mạnh mẽ và có ảnh hưởng rất lớn đến các hoạt động văn hóa, xã hội. Ngoài những kiến thức cơ bản về mạng máy tính và Internet, để có thể hiểu biết sâu sắc hơn về lĩnh vực này, không thể không tìm hiểu những kiến thức chuyên sâu về các giao thức trên mạng, cụ thể là mạng Internet mà chúng ta sử dụng hàng ngày.

Bài giảng "Các giao thức của Internet" được biên soạn nhằm hỗ trợ sinh viên Đại học Công nghệ thông tin học, đặc biệt là chuyên ngành An toàn thông tin, tìm hiểu kiến thức chuyên sâu về các giao thức trên mạng Internet, hoặc những người quan tâm đến lĩnh vực này có thể tham khảo. Nội dung trong bài giảng được tham khảo từ một số tài liệu chuyên ngành về mạng và các giao thức mạng, đặc biệt là cuốn sách *"TCP/IP Protocol Suite"* của tác giả Behrouz A. Forouzan. Đây là một tài liệu được rất nhiều giáo sư, sinh viên đã sử dụng, đọc phục vụ cho mục đích nghiên cứu và học tập.

Bài giảng được cấu trúc với năm nội dung chính như sau:

Chương 1 giới thiệu về Internet và kiến trúc của mạng Internet, kiến trúc của chồng giao thức TCP/IP và giới thiệu một số các công cụ cần thiết cho việc thực hành và nghiên cứu mạng Internet.

Chương 2 trình bày các giao thức tầng ứng dụng, gồm các giao thức phổ biến như HTTP, FTP, SMTP, DHCP, DNS, TELNET, SNMP, v.v. Đây cũng chính là những ứng dụng phổ biến trên mạng Internet hiện nay, giúp cho người dùng tiếp xúc và tham gia được vào môi trường mạng.

Chương 3 thảo luận về các giao thức tầng giao vận, như TCP và UDP. Các giao thức này giúp cho việc truyền tải dữ liệu trên mạng một cách thông suốt, trong đó TCP thực hiện các cơ chế điều khiển luồng dữ liệu, điều khiển tắc nghẽn, quản lý các kết nối, kiểm soát lỗi, ghép và phân kênh dữ liệu.

Chương 4 trình bày về các giao thức tầng mạng, gồm các giao thức giúp cho việc định địa chỉ mạng (IPv4, IPv6), các giao thức thông báo lỗi mạng (ICMPv4, ICMPv6), giao thức phân giải địa chỉ ARP, và các giao thức định tuyến như RIP và OSPF.

Cuối cùng, chương 5 sẽ thảo luận về các giao thức an toàn cho mạng Internet. Cụ thể trong chương này sẽ xem xét một số vấn đề an ninh cho các tầng ứng dụng, giao vận và mạng, với các giao thức như PGP, S/MIME, SSL, IPSec, v.v. Chương này cũng trình bày một số khái niệm về tường lửa.

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT VÀ THUẬT NGỮ.....	iii
CHƯƠNG 1 TỔNG QUAN VỀ INTERNET VÀ CHỖNG GIAO THỨC TCP/IP	1
1.1 GIỚI THIỆU VỀ INTERNET VÀ KIẾN TRÚC CỦA MẠNG INTERNET.....	1
1.1.1 Giới thiệu về Internet.....	1
1.1.2 Kiến trúc của mạng Internet	3
1.2 KIẾN TRÚC CHỖNG GIAO THỨC INTERNET	5
1.2.1 So sánh giữa OSI và chỖng giao thức TCP/IP.....	5
1.2.2 Các tầng trong chỖng giao thức TCP/IP	6
1.3 MỘT SỐ CÔNG CỤ CẦN CHO NGHIÊN CỨU MẠNG INTERNET.....	11
1.3.1 Ipconfig.....	11
1.3.2 Netstat.....	12
1.3.3 Wireshark.....	13
1.3.4 Telnet.....	14
1.3.5 Ping.....	14
1.3.6 Tracert.....	15
CHƯƠNG 2 CÁC GIAO THỨC TẦNG ỨNG DỤNG	16
2.1 GIỚI THIỆU VỀ TẦNG ỨNG DỤNG.....	16
2.1.1 Mô hình client-server.....	16
2.1.2 Mô hình peer-to-peer	19
2.2 MỘT SỐ GIAO THỨC PHỖ BIẾN.....	19
2.2.1 HTTP (<i>Hypertext Transfer Protocol</i>).....	19
2.2.2 FTP (<i>File Transfer Protocol</i>)	24
2.2.3 SMTP, POP, IMAP và MIME (<i>Các giao thức thư điện tử</i>)	27
2.3 GIAO THỨC DHCP (<i>Dynamic Host Configuration Protocol</i>).....	32
2.3.1 Hoạt động của DHCP	32
2.3.2 Điều khiển lỗi	34
2.3.3 Định dạng gói tin DHCP	35
2.4 HỖ THỖNG TÊN MIỀN DNS (<i>Domain Name System</i>).....	37
2.4.1 Không gian tên, không gian tên miền và phân bố tên miền	37
2.4.2 DNS trên Internet.....	40
2.4.3 Phân giải tên-địa chỉ	41
2.4.4 Các thông điệp DNS	43
2.4.5 Đóng gói dữ liệu	47
2.4.6 Bảo mật DNS.....	47
2.5 ĐĂNG NHẬP TỪ XA: TELNET VÀ SSH.....	48
2.5.1 TELNET	48
2.5.2 SSH (<i>Secure Shell</i>)	56
2.6 GIAO THỨC QUẢN LÝ MẠNG ĐƠN GIẢN SNMP	58
2.6.1 Khái niệm	59
2.6.2 Các thành phần quản lý	59
2.6.3 Định dạng của SNMP PDUs	63
2.6.4 Thông điệp SNMP.....	64
2.6.5 Bảo mật.....	64
CHƯƠNG 3 CÁC GIAO THỨC TẦNG GIAO VẬN.....	66
3.1 GIỚI THIỆU VỀ TẦNG GIAO VẬN.....	66

3.2	UDP (User Datagram Protocol).....	66
3.2.1	Gói tin người dùng (User datagram)	67
3.2.2	Các dịch vụ của UDP.....	68
3.2.3	Các ứng dụng UDP.....	70
3.3	TCP (Transmission Control Protocol).....	71
3.3.1	Các dịch vụ của TCP	71
3.3.2	Các đặc điểm của TCP.....	73
3.3.3	Định dạng đoạn (Segment).....	74
3.3.4	Kết nối TCP	76
3.3.5	Thiết lập lại kết nối.....	81
3.3.6	Cửa sổ trong TCP	81
3.3.7	Điều khiển luồng	82
3.3.8	Kiểm soát lỗi trong TCP.....	84
3.3.9	Điều khiển tắc nghẽn trong TCP	86
	CHƯƠNG 4 CÁC GIAO THỨC TẦNG MẠNG	91
4.1	GIỚI THIỆU VỀ TẦNG MẠNG	91
4.2	IPv4 VÀ IPv6	92
4.2.1	Giao thức IPv4.....	92
4.2.2	Giao thức IPv6.....	100
4.3	ICMPv4 VÀ ICMPv6.....	109
4.3.1	Giao thức ICMPv4.....	109
4.3.2	Giao thức ICMPv6.....	115
4.4	GIAO THỨC ARP (Address Resolution Protocol).....	118
4.4.1	Ánh xạ địa chỉ.....	118
4.4.2	Giao thức ARP.....	119
4.5	CÁC GIAO THỨC ĐỊNH TUYẾN: RIP VÀ OSPF.....	124
4.5.1	RIP (Routing Information Protocol).....	124
4.5.2	OSPF (Open Shortest Path First).....	128
	CHƯƠNG 5 CÁC GIAO THỨC AN TOÀN MẠNG INTERNET	135
5.1	AN NINH TẦNG MẠNG	135
5.1.1	Hai chế độ hoạt động của IPSec	135
5.1.2	Hai giao thức an ninh của IPSec.....	137
5.1.3	Dịch vụ cung cấp bởi IPSec.....	140
5.1.4	Security Association (SA)	140
5.1.5	Internet Key Exchange (IKE).....	143
5.1.6	Mạng riêng ảo (VPN)	144
5.2	AN NINH TẦNG GIAO VẬN.....	145
5.2.1	Kiến trúc SSL	145
5.2.2	Bốn giao thức SSL.....	147
5.3	AN NINH TẦNG ỨNG DỤNG.....	149
5.3.1	An ninh cho e-mail	149
5.3.2	Pretty Good Privacy (PGP).....	150
5.3.3	Secure/Multipurpose Internet Mail Extension (S/MIME).....	153
5.4	TƯỜNG LỬA.....	156
5.4.1	Tường lửa lọc gói tin (<i>Packet-Filter Firewall</i>)	157
5.4.2	Tường lửa proxy	158
	TÀI LIỆU THAM KHẢO.....	159

DANH MỤC CÁC TỪ VIẾT TẮT VÀ THUẬT NGỮ

AH	Authentication Header
ARP	Address Resolution Protocol
ARPA	Advanced Research Projects Agency
ARPANET	Advanced Research Projects Agency Network
AS	Autonomous System
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation 1
BGP	Border Gateway Protocol
BOOTP	Bootstrap Protocol
CA	Certification Authority
DARPA	Defense Advanced Research Projects Agency
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol (Giao thức cấu hình địa chỉ động)
DNS	Domain Name System (Hệ thống tên miền)
ESP	Encapsulating Security Payload
FTP	File Transfer Protocol (Giao thức truyền tệp)
HMAC	Hashed Message Authentication Code
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IKE	Internet Key Exchange
IP	Internet Protocol
IPSec	IP Security
ISAKMP	Internet Security Association and Key Management Protocol
ISO	International Organization for Standardization
ISOC	Internet Society (Hiệp hội Internet)
ISP	Internet Service Provider (nhà cung cấp dịch vụ Internet)
IV	Initial Vector (véc-tơ khởi tạo)
LAN	Local Area Network (mạng cục bộ)
LSA	Link State Advertisement
MAC	Media Access Control/Message Authentication Code
MD	Message Digest (Thông điệp tóm tắt)
MIB	Management Information Base
MILNET	Military Network
MIME	Multipurpose Internet Mail Extension
MSS	Maximum Segment Size
MTA	Message Transfer Agent
MTU	Maximum Transfer Unit
NAP	Network Access Point (Điểm truy cập mạng)
NAT	Network Address Translation
NIC	Network Information Center (Trung tâm thông tin mạng)
NIC	Network Interface Card
NSA	National Security Agency (Cơ quan an ninh quốc gia)

NSFNET	National Science Foundation Network
NVT	Network Virtual Terminal
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
PGP	Pretty Good Privacy
POP	Post Office Protocol
PPP	Point-to-Point Protocol
RARP	Reverse Address Resolution Protocol
RFC	Request for Comment
RIP	Routing Information Protocol
RTP	Real-time Transport Protocol
RTT	Round Trip Time
S/MIME	Secure/Multipurpose Internet Mail Extension
SA	Security Association
SAD	Security Association Database
SCTP	Stream Control Transmission Protocol
SHA	Secure Hash Algorithm
SKEME	Secure Key Exchange Mechanism
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol (Giao thức quản lý mạng đơn giản)
SP	Security Policy
SPD	Security Policy Database
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TELNET	Terminal Network
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
TOS	Type of Service (loại dịch vụ)
TTL	Time to Live (thời gian sống)
UA	User Agent
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VPN	Virtual Private Network (Mạng riêng ảo)
WAN	Wide Area Network (Mạng diện rộng)
WWW	World Wide Web
message	thông điệp
segment	đoạn
datagram	gói tin
frame	khung (dữ liệu)
header	tiêu đề (gói tin)
trailer	phần đuôi (gói tin)
link	liên kết
digest	thông điệp tóm tắt

CHƯƠNG 1

TỔNG QUAN VỀ INTERNET VÀ CHỒNG GIAO THỨC TCP/IP

Chương này trình bày các vấn đề cơ bản về mạng Internet và chồng giao thức TCP/IP, bao gồm một số nội dung sau:

- Giới thiệu về mạng Internet và kiến trúc tổng quát của mạng Internet.
- Chồng giao thức TCP/IP và mô tả quá trình truyền thông tại các tầng.
- Giới thiệu một số công cụ cần cho nghiên cứu và thực hành mạng.

1.1 GIỚI THIỆU VỀ INTERNET VÀ KIẾN TRÚC CỦA MẠNG INTERNET

1.1.1 Giới thiệu về Internet

Mạng Internet là một hệ thống thông tin toàn cầu có thể được truy nhập công cộng bao gồm hàng triệu các thiết bị tính toán kết nối lại với nhau. Người dùng truy nhập vào Internet thông qua các chương trình ứng dụng mạng chạy trên thiết bị đầu cuối (*hosts*). Dữ liệu được truyền theo phương thức chuyển mạch gói (*packet switching*) dựa trên một giao thức liên mạng đã được chuẩn hóa (*giao thức IP*), qua các phương tiện truyền thông có dây (cáp quang, cáp đồng) hoặc không dây (sóng radio, vệ tinh,...).

Mạng Internet mang lại rất nhiều tiện ích cho người dùng, như tìm kiếm thông tin bằng cách sử dụng các công cụ tìm kiếm (*search engine*), trò chuyện, trao đổi thông tin với những người khác qua các hệ thống trò chuyện trực tuyến (*chat*), hội thảo truyền hình (*video conference*) hoặc hệ thống thư điện tử (*email*), sử dụng các dịch vụ thương mại, mua bán trên mạng (*e-commerce*), các dịch vụ chăm sóc sức khỏe, giáo dục từ xa,... Đặc biệt, một ứng dụng phổ thông nhất, cung cấp nguồn thông tin khổng lồ trên Internet là WWW (*World Wide Web*), là hệ thống các trang web liên kết với nhau bằng các siêu liên kết (*hyperlink*) và các địa chỉ URL (*Uniform Resource Locator*). Ứng dụng này được coi là một cuộc cách mạng trên Internet vì nhờ đó người dùng có thể dễ dàng truy cập và trao đổi thông tin.

1.1.1.1 Sơ lược lịch sử phát triển của Internet

Tiền thân của mạng Internet ngày nay là ARPANET, được kết nối thành công từ 4 trạm đầu tiên (*Viện nghiên cứu Stanford, Đại học California (Los Angeles), Đại học Utah và Đại học California (Santa Barbara)*) vào năm 1969, bởi Cục các dự án nghiên cứu tiên tiến ARPA (*Advanced Research Projects Agency*) của Bộ Quốc phòng Mỹ. Giao thức truyền thông được sử dụng trong mạng ARPANET là NCP (*Network Control Protocol*).

Khoảng năm 1973-1974, thuật ngữ “Internet” lần đầu tiên xuất hiện, cùng với đề xuất một số vấn đề cơ bản của Internet bởi các tác giả Vint Cerf và B Kahn, là những nét chính của chồng giao thức TCP/IP. Trong thời kỳ đầu những năm 1980, ARPANET được coi là trụ cột của Internet. Đến năm 1983, chồng giao thức TCP/IP (*TCP/IP protocol suite*) chính thức được coi là chuẩn và được dùng hoàn toàn thay thế cho NCP, do ưu điểm mạnh và rất quan trọng của nó là khả năng liên kết các mạng với nhau một cách dễ dàng. Tại thời điểm này,

ARPANET được tách thành ARPANET và MILNET, trong đó MILNET tích hợp với mạng dữ liệu quốc phòng, còn ARPANET trở thành một mạng dân sự.

Một mốc lịch sử quan trọng của Internet là năm 1986, mạng NSFNet chính thức được thiết lập bởi Ủy ban khoa học quốc gia của Mỹ (*NSF – National Science Foundation*), kết nối 5 trung tâm máy tính lại với nhau phục vụ cho nghiên cứu khoa học, có tốc độ truyền là 1,5Mbps, nhanh hơn so với tốc độ truyền của ARPANET là 56Kbps. Như vậy, thời gian này, ARPANET và NSFNet cùng đồng thời tồn tại, sử dụng chung bộ giao thức chuẩn và có kết nối với nhau. Tuy nhiên, sau đó, nhiều doanh nghiệp có xu hướng chuyển sang sử dụng NSFNet, và đến năm 1990, ARPANET chính thức dừng hoạt động. Cũng từ đây, mạng do ARPANET và NSFNet tạo ra dần được đưa vào sử dụng dân dụng, chính là tiền thân của mạng Internet. Các doanh nghiệp bắt đầu tổ chức kinh doanh trên mạng.

Đến năm 1995, NSFNet thu lại thành một mạng nghiên cứu, còn Internet thì vẫn tiếp tục phát triển và dần trở thành một phương tiện đại chúng với các ứng dụng phổ biến như thư điện tử (*email*), truyền tệp tin FTP, dịch vụ thông tin toàn cầu WWW, đồng thời những hình ảnh video cũng bắt đầu được truyền đi trên mạng Internet.

1.1.1.2 Quản lý Internet

Việc điều phối, quản lý hoạt động cấp cao nhất của Internet được thực hiện bởi Hiệp hội Internet (*ISOC – Internet Society*). Khi người dùng muốn kết nối vào Internet, Hiệp hội Internet sẽ thực hiện việc phân phối địa chỉ cho các máy tính của họ. Tuy nhiên, do tốc độ phát triển nhanh chóng của Internet, từ năm 1992, việc phân phối địa chỉ được phân cấp cho các Trung tâm thông tin mạng của các khu vực (*NIC – Network Information Center*). Việt Nam thuộc sự quản lý của Trung tâm thông tin mạng Châu Á-Thái Bình Dương (gọi là *APNIC – Asia-Pacific Network Information Centre*) có trụ sở đặt tại Nhật Bản.

1.1.1.3 Giao thức và các chuẩn

Trong mạng máy tính, việc truyền thông liên lạc giữa các thực thể được diễn ra trong các hệ thống khác nhau. Thực thể là một đối tượng bất kỳ nào đó có khả năng gửi và nhận thông tin. Tuy nhiên, hai thực thể không chỉ đơn thuần là gửi dòng bit cho nhau mà còn phải hiểu được nhau. Các thực thể sẽ phải cùng thỏa thuận về các quy tắc, quy ước khi tham gia truyền thông và tập hợp các quy tắc, quy ước này gọi là *giao thức (protocol)*. Như vậy, giao thức sẽ định nghĩa những thông tin gì được trao đổi qua hệ thống truyền thông, cách thức thực hiện việc trao đổi và thời gian sẽ thực hiện việc trao đổi liên lạc đó. Hay nói một cách khác, các yếu tố chính của một giao thức là *cú pháp, ngữ nghĩa và thời gian*.

- **Cú pháp:** Cú pháp là cấu trúc hoặc định dạng của dữ liệu, có nghĩa là thứ tự mà chúng được trình bày. Ví dụ, một giao thức đơn giản có thể quy định 8 bit đầu tiên của dữ liệu là địa chỉ của người gửi, 8 bit tiếp theo là địa chỉ của người nhận, và phần còn lại là thông điệp. Thứ tự dữ liệu cũng chính là thứ tự của các bit khi chúng được lưu trữ hoặc truyền đi. Các hệ thống máy tính khác nhau có thể lưu trữ dữ liệu theo các thứ tự khác nhau, và các khác biệt này cần phải được xử lý khi các hệ thống máy tính này giao tiếp, trao đổi thông tin với nhau.

- **Ngữ nghĩa:** Ngữ nghĩa là ý nghĩa của từng phần (nhóm) bit. Cần phải giải thích và diễn dịch được các phần trong chuỗi bit, và dựa vào đó để đưa ra những hành động. Ví dụ, cần phải xác định được địa chỉ để định tuyến thông điệp đi đến đích.
- **Thời gian:** Thời gian đề cập đến hai đặc điểm: thời điểm dữ liệu phải được gửi đi và khoảng thời gian mà nó được phép truyền đi trên mạng. Ví dụ, nếu bên gửi gửi dữ liệu với tốc độ 100 megabits mỗi giây (100 Mbps), nhưng bên nhận chỉ có thể xử lý dữ liệu với tốc độ 1 Mbps, thì việc truyền tải sẽ là quá tải với bên nhận và dữ liệu phần lớn sẽ bị mất.

Nếu giao thức được đồng nghĩa với các “quy tắc” (*rules*) thì các *chuẩn* (*standards*) được coi là sự đồng thuận trên mức quy tắc. Việc đưa ra các chuẩn là rất cần thiết trong việc tạo và duy trì một thị trường mở và cạnh tranh cho các nhà sản xuất thiết bị, cũng như đảm bảo khả năng tương thích về dữ liệu, công nghệ truyền thông và các quy trình trong nước và quốc tế. Các chuẩn cung cấp những hướng dẫn cho các nhà sản xuất, nhà cung cấp, cơ quan chính phủ và các nhà cung cấp dịch vụ khác nhằm đảm bảo các liên kết truyền thông có thể thực hiện được trong thị trường hiện nay và trong truyền thông quốc tế. Chuẩn truyền thông dữ liệu bao gồm hai loại: *theo thực tế* (*de facto*) và *theo tính hợp pháp* (*de jure*).

- **Theo thực tế:** Chuẩn không được chấp thuận bởi cơ quan tổ chức nhưng lại được thông qua chấp thuận sử dụng rộng rãi trên thực tế. Chuẩn này thường được thiết lập bởi các nhà sản xuất khi họ lần đầu muốn tìm cách xác định chức năng của một sản phẩm hoặc công nghệ mới. Ví dụ, một số các chuẩn DVD.
- **Theo tính hợp pháp:** Chuẩn được hợp pháp hóa và công nhận bởi một cơ quan chính thống.

1.1.1.4 Sự phát triển của Internet

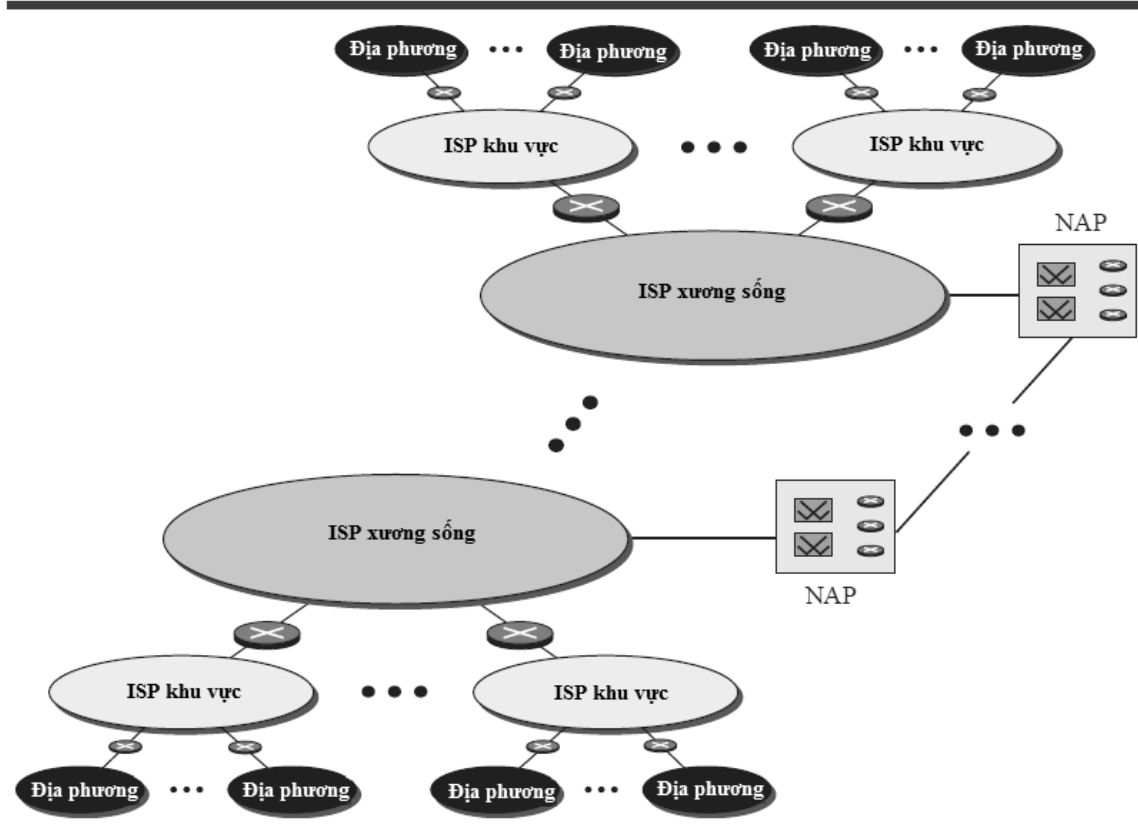
Internet đã phát triển một cách vô cùng mạnh mẽ. Chỉ trong vài thập kỷ qua, số lượng các mạng đã tăng từ hàng chục đến hàng trăm ngàn. Đồng thời, số lượng máy tính kết nối với mạng lưới đã phát triển từ hàng trăm đến hàng trăm triệu. Internet vẫn đang tiếp tục phát triển. Các yếu tố ảnh hưởng đến sự tăng trưởng này:

- **Các giao thức mới:** Các giao thức mới cần phải được bổ sung và đồng thời cần phải loại bỏ những giao thức không còn phù hợp. Ví dụ, một giao thức (IPv6) tốt hơn IPv4 đã được chấp thuận như một tiêu chuẩn nhưng vẫn chưa được chuyển đổi hoàn toàn.
- **Công nghệ mới:** Các công nghệ mới đang được phát triển sẽ tăng cường năng lực mạng lưới và cung cấp băng thông tốt hơn cho người dùng Internet.
- **Tăng cường sử dụng dữ liệu đa phương tiện:** Internet sẽ ngày càng được sử dụng nhiều hơn như là một phương tiện chia sẻ dữ liệu, và đặc biệt là dữ liệu đa phương tiện (âm thanh, hình ảnh, video).

1.1.2 Kiến trúc của mạng Internet

Internet ngày nay không chỉ là một mạng phân cấp đơn giản mà nó được tạo ra từ nhiều mạng lưới địa phương bằng cách kết nối các thiết bị và các trạm chuyển mạch. Internet có tính mở (còn gọi Internet là “*mạng của các mạng*”), thường xuyên thay đổi, ví dụ như thêm vào

các mạng mới, các mạng đang tồn tại cần có thêm địa chỉ, hoặc cần loại bỏ mạng của các công ty không còn tồn tại,... Do vậy, rất khó có thể có được một biểu diễn chính xác về Internet. Ngày nay, hầu hết những người dùng cuối muốn kết nối Internet đều thông qua dịch vụ của các Nhà cung cấp dịch vụ Internet (ISPs – *Internet Service Providers*). Việc quản lý ISPs được phân thành nhiều cấp: cấp quốc tế, cấp quốc gia, cấp khu vực và cấp địa phương. Hiện nay, Internet được điều hành bởi các công ty tư nhân, không phải là chính phủ. Hình 1.1 trình bày một mô hình kiến trúc của mạng Internet.



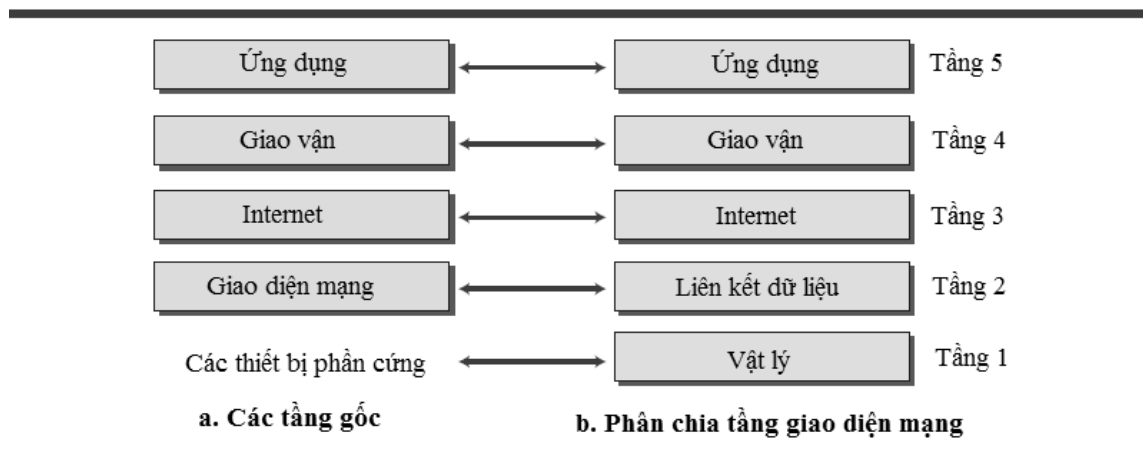
Hình 1.1 Mô hình kiến trúc của mạng Internet

- **ISP xương sống (Backbone ISPs):** ISP xương sống được tạo ra và duy trì bởi các công ty chuyên biệt. Đây được coi là các ISPs cấp một trong hệ thống phân cấp ISP. Có khá nhiều ISP xương sống hoạt động tại Bắc Mỹ, nổi tiếng nhất là Sprint-Link, PSINet, UUNet Technology, AGIS, và Internet MCI. Để cung cấp kết nối giữa những người dùng cuối với nhau, các mạng xương sống được kết nối bởi các trạm chuyển mạch phức tạp (thường do một bên thứ ba) được gọi là các điểm truy cập mạng (*NAPs* - *network access points*). Một số ISP khu vực cũng được kết nối với nhau bằng các trạm chuyển mạch riêng (gọi là *peering points*). ISP xương sống thường hoạt động ở tốc độ cao (khoảng 10 Gbps).
- **ISP khu vực (Regional ISPs):** ISP khu vực là các ISP nhỏ được kết nối với một hoặc nhiều ISP xương sống. Các ISP này thuộc cấp độ thứ hai của hệ thống phân cấp ISP với tốc độ dữ liệu thấp hơn.

- **ISP địa phương (Local ISPs):** Các ISP địa phương cung cấp dịch vụ trực tiếp đến người dùng cuối. Các ISP này có thể được kết nối với các ISP khu vực hoặc kết nối trực tiếp đến ISP xương sống. Hầu hết người dùng cuối được kết nối với các ISP địa phương. Một ISP địa phương có thể là một công ty chuyên cung cấp dịch vụ Internet, hoặc một công ty có một mạng lưới cung cấp dịch vụ cho người lao động riêng của họ, hoặc một tổ chức phi lợi nhuận, chẳng hạn như một trường cao đẳng hoặc đại học, chạy một hệ thống mạng riêng.

1.2 KIẾN TRÚC CHỒNG GIAO THỨC INTERNET

Chồng giao thức TCP/IP được phát triển trước mô hình OSI, vì vậy, các tầng trong chồng giao thức TCP/IP không hoàn toàn chính xác với các tầng trong mô hình OSI. Bản gốc chồng giao thức TCP/IP được xác định bao gồm bốn tầng phần mềm xây dựng dựa trên phần cứng. Tuy nhiên, hiện nay, chồng giao thức TCP/IP được coi như là một mô hình năm tầng với các tầng có tên tương tự như các tầng trong mô hình OSI. Hình 1.2 thể hiện các tầng trong mô hình TCP/IP.



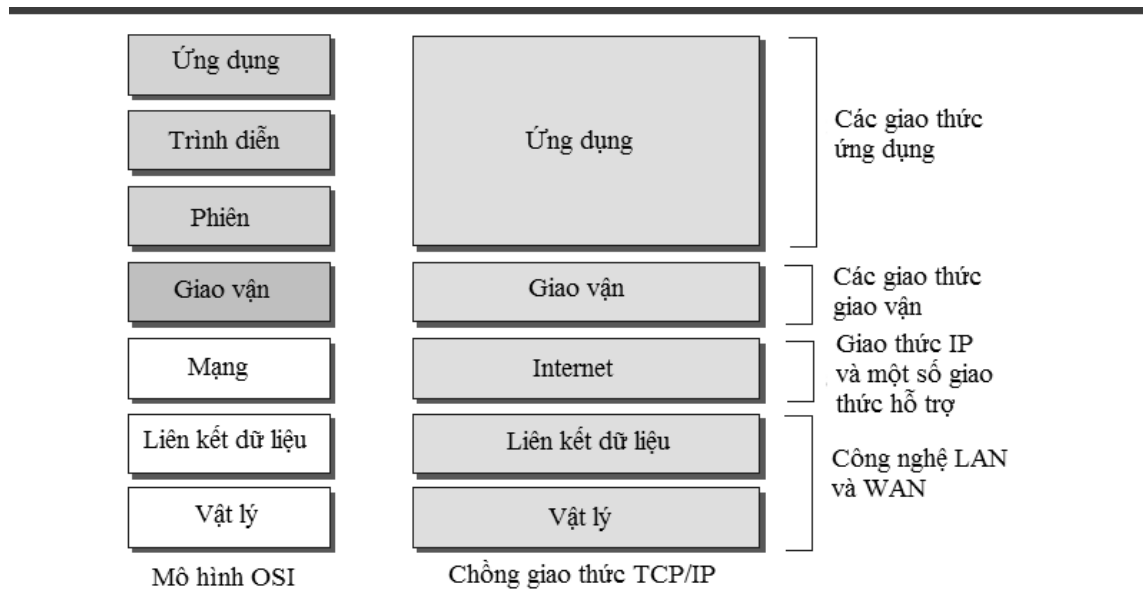
Hình 1.2 Các tầng trong chồng giao thức TCP/IP

1.2.1 So sánh giữa OSI và chồng giao thức TCP/IP

Khi so sánh hai mô hình OSI và TCP/IP, thấy rằng, hai tầng phiên và tầng trình diễn, không có trong TCP/IP. Hai tầng này không được bổ sung vào TCP/IP sau khi mô hình OSI được công bố. Tầng ứng dụng trong TCP/IP thường được coi là sự kết hợp của ba tầng trong mô hình OSI, như thể hiện trong hình 1.3. Có hai lý do như sau: thứ nhất, chồng giao thức TCP/IP có nhiều hơn một giao thức tầng giao vận, và một số các chức năng của tầng phiên có sẵn trong một số giao thức tầng giao vận; thứ hai, tầng ứng dụng không chỉ là một phần của phần mềm mà có rất nhiều ứng dụng có thể được phát triển ở tầng này. Nếu một số các chức năng của tầng phiên và tầng trình diễn là cần thiết cho một ứng dụng cụ thể, thì nó có thể được thêm vào trong quá trình phát triển phần mềm và được coi như là một phần của phần mềm đó.

TCP/IP là một chồng giao thức phân cấp bao gồm các mô-đun tương tác với nhau, trong đó, mỗi mô-đun cung cấp một chức năng cụ thể, nhưng các mô-đun không nhất thiết phải phụ thuộc lẫn nhau. Trong khi mô hình OSI chỉ rõ các chức năng thuộc về từng tầng cụ thể, thì các tầng trong chồng giao thức TCP/IP bao gồm các giao thức tương đối độc lập, có thể được dùng pha trộn và phù hợp, tùy thuộc vào nhu cầu của hệ thống. Thuật ngữ “*phân cấp*” có

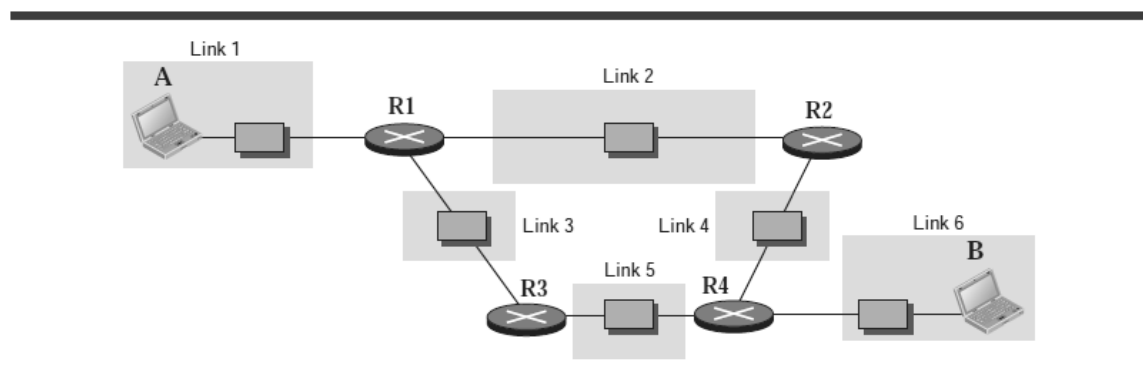
nghĩa là mỗi giao thức ở tầng phía trên được hỗ trợ bởi một hoặc nhiều giao thức ở các tầng thấp hơn.



Hình 1.3 Mô hình OSI và chồng giao thức TCP/IP

1.2.2 Các tầng trong chồng giao thức TCP/IP

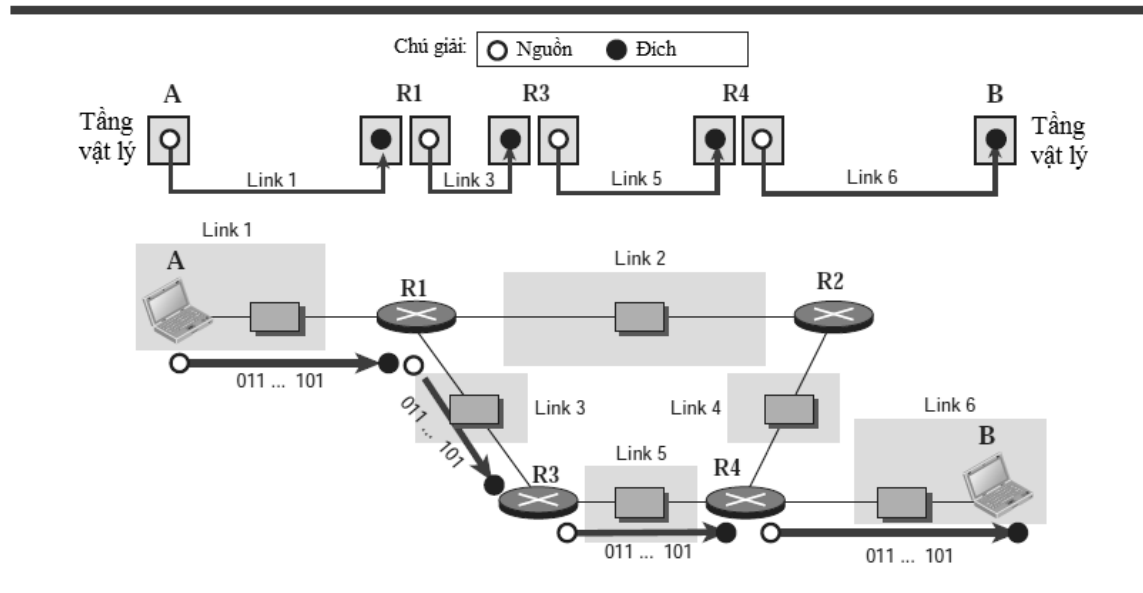
Phần này trình bày tóm tắt mục đích của mỗi tầng trong chồng giao thức TCP/IP. Internet được tạo thành từ nhiều các mạng nhỏ, gọi là các *liên kết* (*links*). Mỗi liên kết là một mạng trong đó cho phép các máy tính có thể truyền thông liên lạc với nhau. Ví dụ, nếu tất cả các máy tính trong một văn phòng được nối lại với nhau bằng dây, thì việc kết nối đó sẽ tạo thành một liên kết. Nếu có một số máy tính thuộc về một công ty tư nhân được kết nối với nhau thông qua kênh vệ tinh, thì việc kết nối này cũng là một liên kết. Như vậy, một liên kết có thể là một mạng cục bộ (*LAN – Local Area Network*) phục vụ cho một khu vực nhỏ, hoặc cũng có thể là một mạng diện rộng (*WAN – Wide Area Network*) phục vụ cho một khu vực lớn hơn. Giả thiết rằng các liên kết khác nhau được kết nối với nhau bằng các thiết bị như là các bộ định tuyến hoặc các bộ chuyển mạch để định hướng cho dữ liệu đi đến đích của nó. Hình 1.4 là một ví dụ thể hiện mục đích của mỗi tầng trong Internet, gồm có sáu liên kết, bốn bộ định tuyến (R1 đến R4) và hai máy tính A và B.



Hình 1.4 Ví dụ một phần của mạng Internet

1.2.2.1 Tầng vật lý

TCP/IP không định nghĩa bất kỳ giao thức cụ thể nào cho tầng vật lý. Tầng vật lý hỗ trợ tất cả các giao thức chuẩn và độc quyền. Ở mức này, việc thông tin liên lạc được thực hiện giữa hai *hop* hoặc hai nút, hoặc máy tính hoặc bộ định tuyến. *Đơn vị truyền thông tại tầng vật lý là một bit đơn*. Khi kết nối được thiết lập giữa hai nút, một dòng bit sẽ được truyền giữa các nút đó. Tuy nhiên, tầng vật lý sẽ xử lý từng bit riêng một. Hình 1.5 biểu diễn việc truyền tin giữa các nút. Giả thiết trong trường hợp này là hai máy tính đã tìm ra được đường truyền hiệu quả nhất là thông qua các bộ định tuyến R1, R3 và R4.



Hình 1.5 Truyền thông tại tầng vật lý

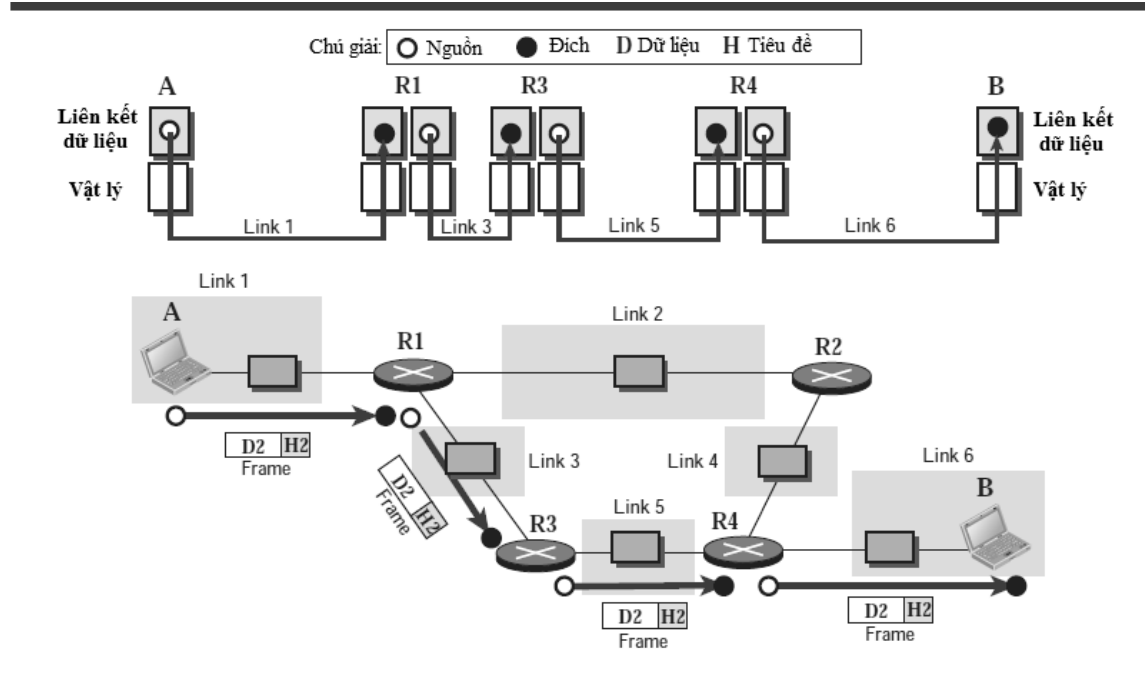
Nếu một nút được kết nối với n liên kết, nó cần n giao thức tầng vật lý, mỗi giao thức cho một liên kết, do các liên kết khác nhau có thể sử dụng các giao thức tầng vật lý khác nhau. Tuy nhiên, hình 1.5 ở đây chỉ biểu diễn các tầng vật lý có tham gia vào quá trình truyền thông. Mỗi máy tính chỉ liên quan đến một liên kết và mỗi bộ định tuyến liên quan đến hai liên kết. Đường đi của các bit giữa máy tính A và máy tính B qua bốn chặng độc lập: máy tính A gửi từng bit tới bộ định tuyến R1 theo định dạng của giao thức liên kết 1; bộ định tuyến R1 gửi từng bit đến bộ định tuyến R3 theo định dạng của giao thức liên kết 3; và cứ tiếp tục thế. Bộ định tuyến R1 có ba liên kết vật lý, trong đó có hai liên kết nằm trên đường đi của máy tính A và máy tính B: lớp kết nối với liên kết 1 nhận bit theo định dạng của giao thức liên kết 1; lớp kết nối với liên kết 3 gửi bit theo định dạng của giao thức liên kết 3. Tương tự đối với các bộ định tuyến R3 và R4.

Chức năng của tầng vật lý, ngoài việc truyền các bit, theo chức năng đã được đề cập trong mô hình OSI, còn chủ yếu phụ thuộc vào các công nghệ được thực hiện trên các liên kết. Trên thực tế có nhiều giao thức cho tầng vật lý của các mạng LAN hoặc WAN.

1.2.2.2 Tầng liên kết dữ liệu

TCP/IP không xác định bất kỳ giao thức cụ thể nào cho các tầng liên kết dữ liệu, mà nó hỗ trợ tất cả các giao thức chuẩn và giao thức độc quyền. Ở lớp này, việc truyền thông cũng

được thực hiện giữa các *hop* hoặc các nút. Đơn vị truyền thông tại tầng liên kết dữ liệu là một *khung* (*frame*). Một khung là một gói tin được đóng gói từ dữ liệu ở tầng mạng chuyển xuống và được thêm vào phần tiêu đề (*header*) và đôi khi cả phần đuôi (*trailer*). Phần đầu trong dữ liệu truyền thông sẽ chứa thông tin về nguồn và đích của khung. Địa chỉ đích được sử dụng để xác định nút nhận khung phía bên phải vì có thể có nhiều nút được kết nối với liên kết. Địa chỉ nguồn được sử dụng để trả lời hoặc xác nhận theo yêu cầu bởi các giao thức. Hình 1.6 thể hiện việc truyền thông tại tầng liên kết dữ liệu.



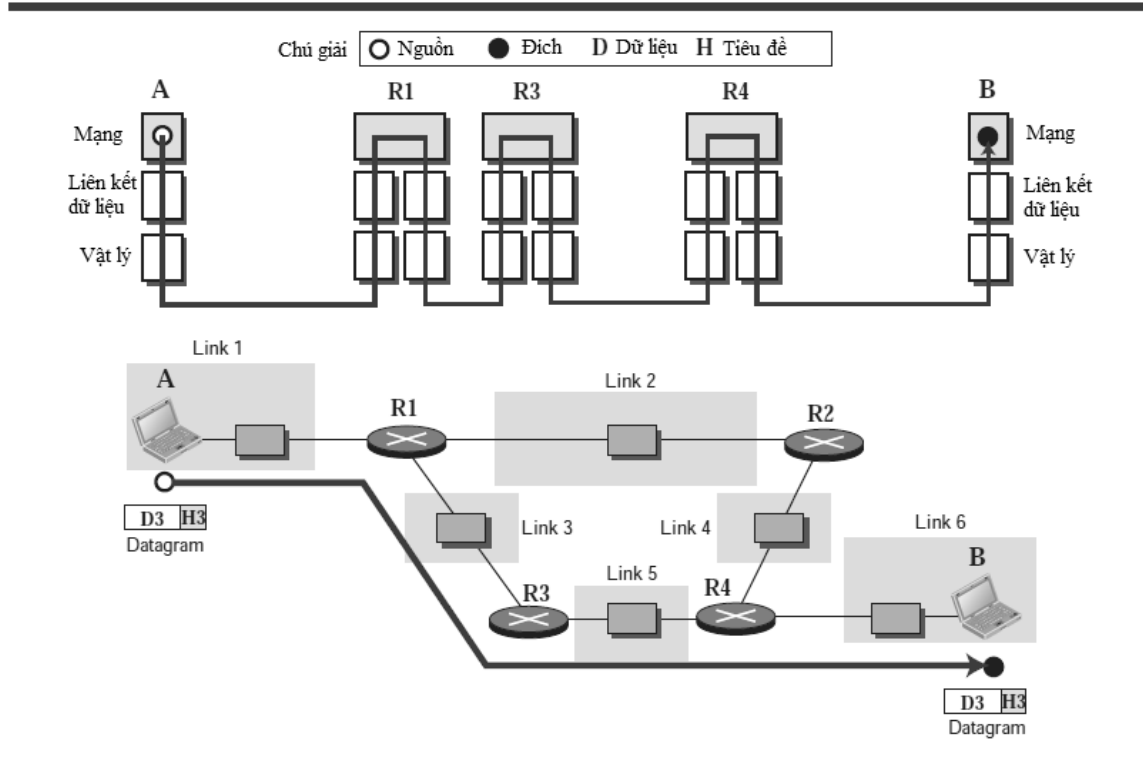
Hình 1.6 Truyền thông tại tầng liên kết dữ liệu

Các khung dữ liệu được di chuyển giữa máy tính A và bộ định tuyến R1 có thể khác các khung di chuyển giữa bộ định tuyến R1 và R3. Khi một khung được chuyển đến bộ định tuyến R1, bộ định tuyến này sẽ mở khung và tách phần dữ liệu theo định dạng của giao thức tầng liên kết dữ liệu phía bên trái của nó. Sau đó dữ liệu sẽ được tạo khung mới theo định dạng của giao thức tầng liên kết dữ liệu phía bên phải và được chuyển tới R3. Lý do là vì hai liên kết, liên kết 1 và liên kết 3, có thể sử dụng các giao thức khác nhau và như vậy các khung dữ liệu sẽ có những định dạng khác nhau. Hình 1.6 không biểu diễn việc di chuyển dữ liệu vật lý của khung, vì việc này chỉ được thực hiện tại tầng vật lý. Tại tầng liên kết dữ liệu, hai nút chỉ truyền thông logic với nhau. Mặt khác, tầng liên kết dữ liệu tại bộ định tuyến R1 chỉ biết được là khung được gửi trực tiếp từ tầng liên kết dữ liệu của máy tính A. Tuy nhiên, thực tế dữ liệu được gửi từ A đến R1 là một dòng bit từ tầng vật lý của A di chuyển sang tầng vật lý của R1. Như vậy, một khung dữ liệu tại A được chuyển thành dòng bit, và các bit tại R1 lại được chuyển thành một khung dữ liệu, đây chính là điểm quan trọng với hai tầng liên kết dữ liệu mà khung dữ liệu được trao đổi.

1.2.2.3 Tầng mạng

Chồng giao thức TCP/IP hỗ trợ giao thức IP (*Internet Protocol*) cho tầng mạng (*network layer*), hay chính xác hơn là tầng liên mạng (*Internetwork layer*). Giao thức IP thực hiện kỹ

thuật truyền tải. Đơn vị truyền thông của tầng mạng là một gói tin (*datagram*). Mỗi gói tin được vận chuyển riêng rẽ và chúng có thể di chuyển theo các tuyến đường khác nhau từ nguồn đến đích. Các gói tin có thể đến đích không theo thứ tự hoặc bị trùng lặp. Giao thức IP không theo dõi các tuyến đường và không có cơ chế sắp xếp lại các gói tin tại điểm đích. Hình 1.7 biểu diễn việc truyền thông tại tầng mạng.



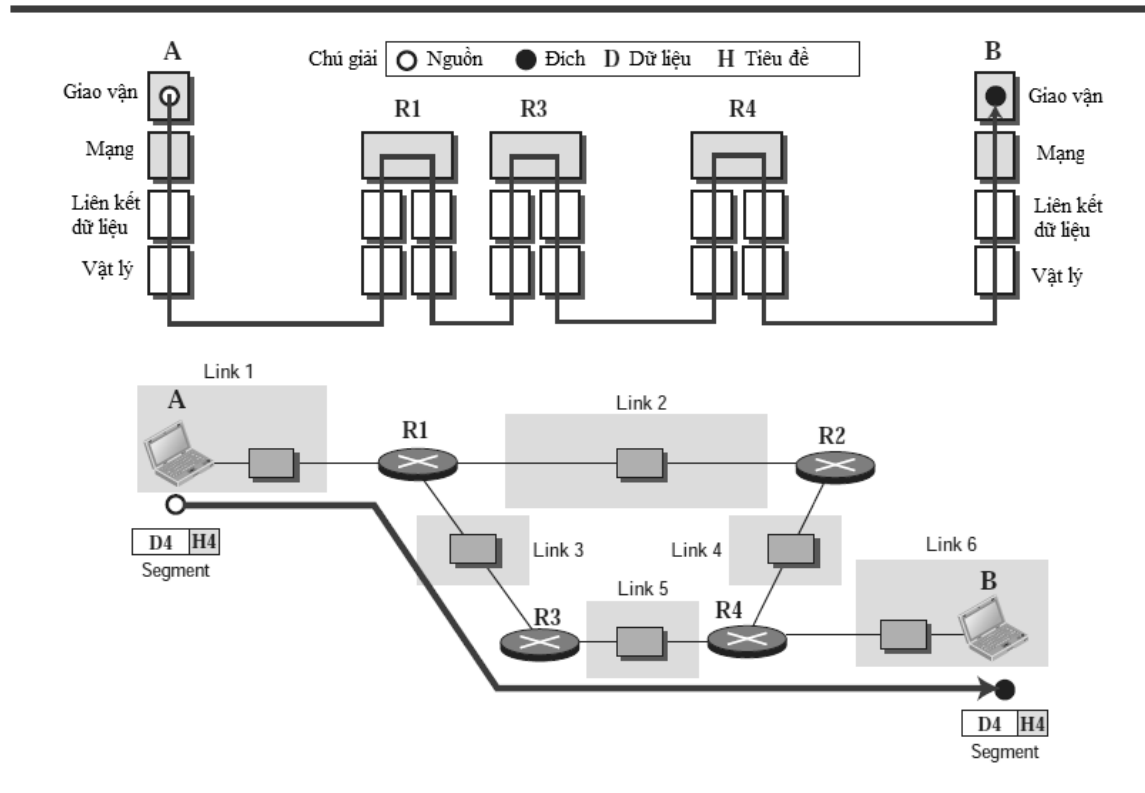
Hình 1.7 Truyền thông tại tầng mạng

Có một sự khác biệt chính giữa việc truyền thông tại tầng mạng và truyền thông tại tầng liên kết dữ liệu hoặc tầng vật lý. Truyền thông tại tầng mạng là đầu cuối đến đầu cuối (*end to end*) trong khi truyền thông tại hai tầng kia là nút đến nút (*node to node*). Giả sử, gói tin bắt đầu từ máy tính A cần đi đến đích là máy tính B. Các tầng mạng của các bộ định tuyến có thể kiểm tra địa chỉ nguồn và đích của gói tin để tìm ra con đường tốt nhất, nhưng không được phép thay đổi nội dung gói tin. Tất nhiên, việc truyền thông tại tầng mạng mang tính logic chứ không phải vật lý. Tầng mạng của các máy tính A và B chỉ biết được là đang gửi và nhận các gói tin, nhưng thực tế, việc truyền thông được thực hiện ở tầng vật lý.

1.2.2.4 Tầng giao vận

Có một sự khác biệt chính giữa tầng giao vận và tầng mạng. Mặc dù tất cả các nút trong mạng đều cần phải có tầng mạng, nhưng chỉ có hai máy tính đầu cuối là cần phải có tầng giao vận. Tầng mạng chịu trách nhiệm gửi các gói tin riêng lẻ (*datagram*) từ máy tính A đến máy tính B, còn tầng giao vận chịu trách nhiệm chuyển toàn bộ thông điệp, được gọi là một *đoạn* (*segment*), từ A đến B. Một đoạn có thể bao gồm một vài cho đến hàng chục gói tin. Các đoạn cần được chia thành các gói tin và mỗi gói sẽ được chuyển xuống tầng mạng để truyền đi. Vì Internet xác định các tuyến đường đi khác nhau cho mỗi gói tin, nên các gói tin có thể đến đích không theo thứ tự và có thể bị mất mát. Tầng giao vận tại máy tính đích B cần phải đợi

cho đến khi tất cả các gói tin đến, tập hợp chúng lại và tạo thành các đoạn dữ liệu. Các tầng giao vận chỉ biết được việc truyền thông giữa chúng được thực hiện bằng việc trao đổi các đoạn dữ liệu, nhưng trên thực tế thì việc truyền thông được thực hiện tại tầng vật lý bằng cách trao đổi các bit dữ liệu. Hình 1.8 biểu diễn việc truyền thông tại tầng giao vận.



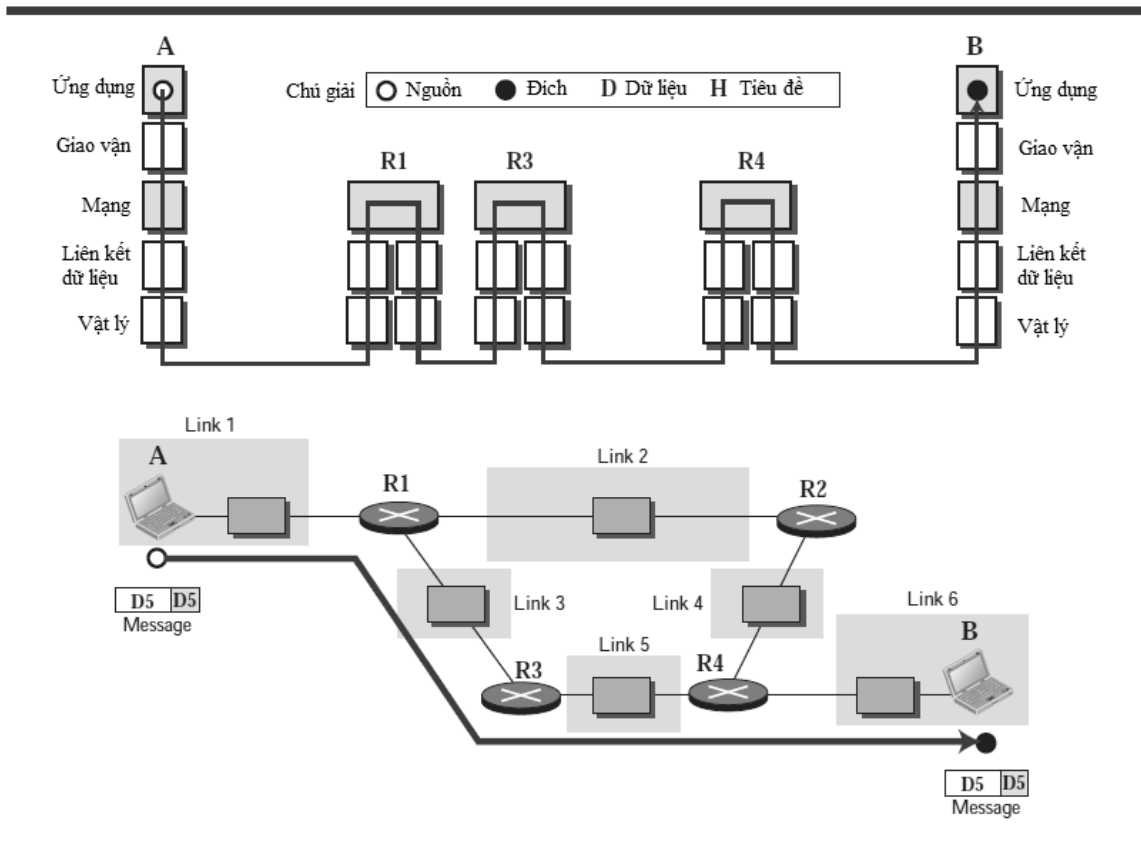
Hình 1.8 Truyền thông tại tầng giao vận

Trong chồng giao thức TCP/IP, tầng giao vận bao gồm hai giao thức, là giao thức TCP (*Transmission Control Protocol*) và giao thức UDP (*User Datagram Protocol*). Trong vài năm gần đây xuất hiện một giao thức mới là SCTP (*Stream Control Transmission Protocol*).

1.2.2.5 Tầng ứng dụng

Tầng ứng dụng trong chồng giao thức TCP/IP tương đương với sự kết hợp ba tầng trong mô hình OSI là tầng phiên, tầng trình diễn và tầng ứng dụng. Tầng ứng dụng cho phép người dùng truy nhập vào hệ thống mạng riêng hoặc hệ thống Internet toàn cầu. Nhiều giao thức được định nghĩa ở tầng này để cung cấp các dịch vụ cho người dùng, như thư điện tử (*e-mail*), truyền tệp tin, truy nhập Web, v.v. Hình 1.9 biểu diễn việc truyền thông ở tầng ứng dụng.

Việc truyền thông tại tầng ứng dụng cũng giống như tầng giao vận, là đầu cuối đến đầu cuối (*end to end*). Đơn vị truyền thông tại tầng ứng dụng là một thông điệp (*message*). Một thông điệp được tạo ra tại máy tính A được gửi đến máy tính B sẽ không bị thay đổi trong suốt quá trình truyền.



Hình 1.9 Truyền thông tại tầng ứng dụng

1.3 MỘT SỐ CÔNG CỤ CẦN CHO NGHIÊN CỨU MẠNG INTERNET

Phần này giới thiệu một số công cụ giúp cho người dùng có thể nghiên cứu và phân tích mạng, bao gồm: ipconfig, netstat, wireshark, telnet, ping và tracert.

1.3.1 Ipconfig

Lệnh *ipconfig* (*internet protocol configuration*) được sử dụng khi cần biết trạng thái cấu hình TCP/IP hiện tại của máy tính (giao diện cục bộ) và có thể thay đổi thiết lập DHCP và DNS. Trong hầu hết các trường hợp, lệnh ipconfig được dùng với tham số */all*. Cú pháp:

<i>ipconfig /all</i>	(hiển thị báo cáo chi tiết về cấu hình của tất cả các card mạng)
<i>/flushdns</i>	(xóa bỏ cache chứa tên trong DNS)
<i>/registerdns</i>	(đăng ký lại tên DNS)
<i>/displaydns</i>	(hiển thị cache của bộ phân giải DNS)
<i>/release</i>	(hủy bỏ địa chỉ IP cho một card)
<i>/renew</i>	(phát hành địa chỉ IP mới cho một card)
<i>/?</i>	(để biết thêm các tham số ipconfig khác)

```

C:\Windows\system32\cmd.exe

C:\Users\Diep>ipconfig/all

Windows IP Configuration

    Host Name . . . . . : diep
    Primary Dns Suffix . . . . . :
    Node Type . . . . . : Mixed
    IP Routing Enabled. . . . . : No
    WINS Proxy Enabled. . . . . : No

Wireless LAN adapter Local Area Connection* 12:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . :
    Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter
    Physical Address. . . . . : 84-A6-C8-23-D3-AA
    DHCP Enabled. . . . . : Yes
    Autoconfiguration Enabled . . . . : Yes

Ethernet adapter Bluetooth Network Connection:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . :
    Description . . . . . : Bluetooth Device (Personal Area Network)
  
```

Hình 1.10 *ipconfig /all* hiển thị giao diện cục bộ

Chú ý thận trọng khi sử dụng *ipconfig* trên máy client vì khi dùng lệnh với một số tham số mà không có sự trợ giúp của quản trị mạng thì có thể dẫn đến tình trạng mất kết nối với server.

1.3.2 Netstat

Active Connections

Proto	Local Address	Foreign Address	State	PID
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING	868
TCP	0.0.0.0:443	0.0.0.0:0	LISTENING	2720
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING	4
TCP	0.0.0.0:554	0.0.0.0:0	LISTENING	4976
TCP	0.0.0.0:902	0.0.0.0:0	LISTENING	2020
TCP	0.0.0.0:912			
TCP	0.0.0.0:2869			
TCP	0.0.0.0:5357			
TCP	0.0.0.0:10243			
TCP	0.0.0.0:42176			
TCP	0.0.0.0:49152			
TCP	0.0.0.0:49153			
TCP	0.0.0.0:49154			
TCP	0.0.0.0:49156			
TCP	0.0.0.0:49157			
TCP	127.0.0.1:8307			
TCP	127.0.0.1:10000			
TCP	192.168.1.103:139			
TCP	192.168.1.103:6206			

CurrPorts

Process Name	Process ID	Protocol	Local Port	Local Por...	Local Address	Rem
BitComet.exe	2844	UDP	1720		0.0.0.0	
ieexplore.exe	644	UDP	2446		127.0.0.1	
lsass.exe	1520	UDP	500		0.0.0.0	
lsass.exe	1520	UDP	4500		0.0.0.0	
mcnascv.exe	1740	TCP	6646		0.0.0.0	
mcnascv.exe	1740	UDP	6646		99.240.214.105	
mDNSRespond...	404	TCP	5354		127.0.0.1	
mDNSRespond...	404	UDP	5353		99.240.214.105	
mDNSRespond...	404	UDP	1025		0.0.0.0	
MwlSvc.exe	1008	TCP	6636		0.0.0.0	
MwlSvc.exe	1008	UDP	6636		99.240.214.105	
svchost.exe	1776	TCP	135		0.0.0.0	
svchost.exe	532	TCP	2869		0.0.0.0	
svchost.exe	1972	UDP	1114		127.0.0.1	
svchost.exe	1972	UDP	1113		0.0.0.0	
svchost.exe	348	UDP	1028		0.0.0.0	

Hình 1.11 *netstat* hiển thị các kết nối cục bộ

Netstat (network statistics) là một công cụ dòng lệnh hiển thị các kết nối mạng (cả đi lẫn đến), các bảng định tuyến và một số giao diện mạng (bộ điều khiển giao diện mạng hoặc giao diện mạng xác định phần mềm), và thống kê giao thức mạng. Netstat có sẵn trên nhiều hệ điều

hành như Linux, Windows NT (bao gồm Windows XP, Windows Vista, Windows 7, 8), v.v. Netstat cũng được dùng để tìm kiếm các vấn đề trong mạng và xác định lưu lượng mạng như một thước đo hiệu suất.

Cú pháp: `netstat [/a][/e][/n][...]`

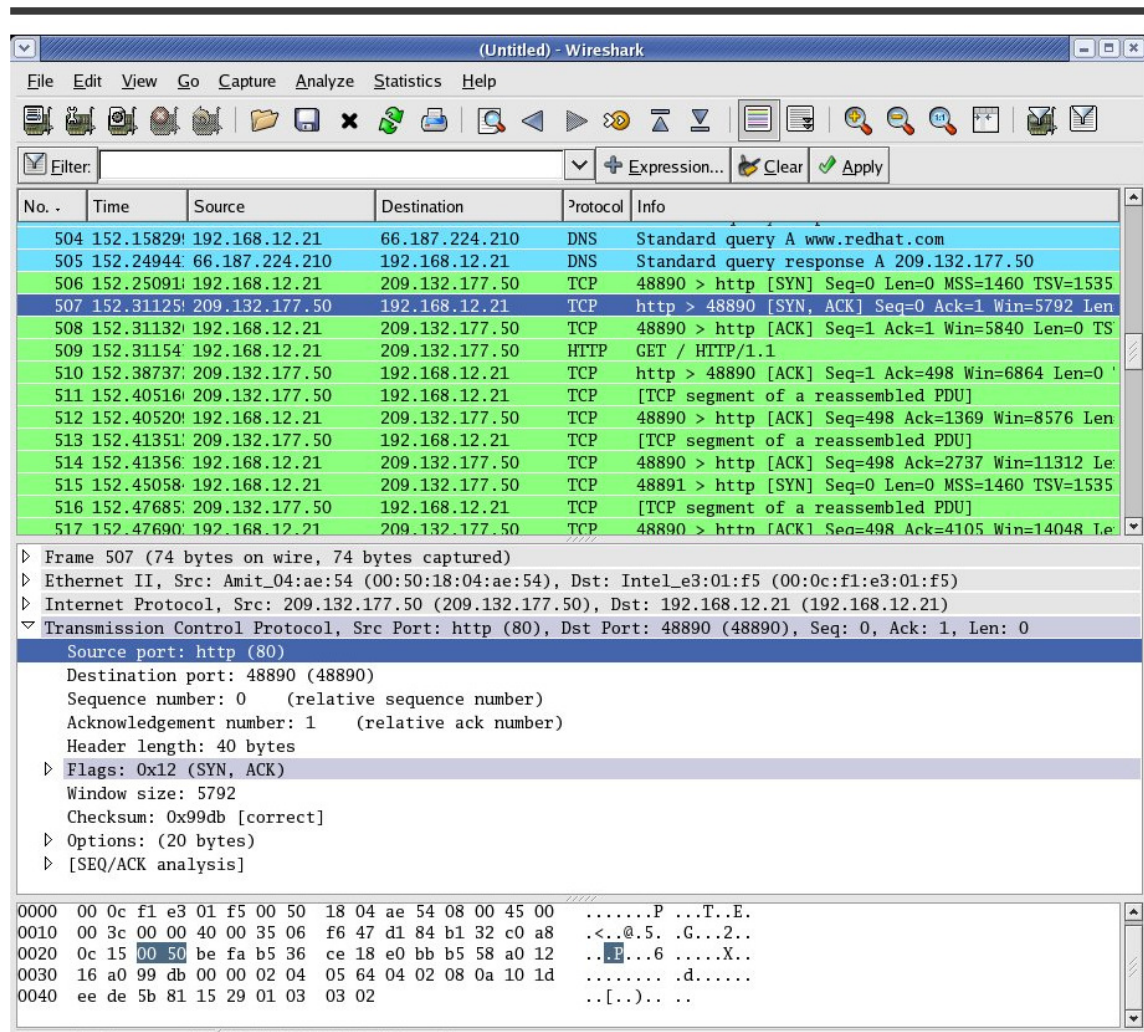
`/a` (hiển thị tất cả các kết nối và các cổng TCP và UDP đang lắng nghe)

`/e` (hiển thị các thông tin thống kê Ethernet)

`/n` (hiển thị địa chỉ và số cổng kết nối)

`/?` (để biết thêm các tham số khác)

1.3.3 Wireshark



Hình 1.12 Wireshark hiển thị thông tin của gói tin

Wireshark là một phần mềm mã nguồn mở được sử dụng để phân tích gói tin nhằm giải quyết một vấn đề cụ thể của mạng. Ví dụ, muốn xác định nguyên nhân bị mất kết nối mạng, băng thông bị chiếm, hay một số tình huống an ninh mạng cơ bản, v.v. Wireshark có thể hỗ trợ gần như tất cả các giao thức, từ những loại phổ biến như IP, TCP,... đến những loại đặc biệt như Bittorrent,... Với giao diện ứng dụng đồ họa với các menu được bố trí rõ ràng, dễ

hiệu, wireshark rất thân thiện với người dùng. Mặt khác, wireshark có thể được cài đặt trên hầu hết các hệ điều hành hiện nay. Vì vậy, wireshark là một công cụ rất tốt cho những người muốn nghiên cứu, phân tích giao thức, kể cả chuyên nghiệp hay nghiệp dư.

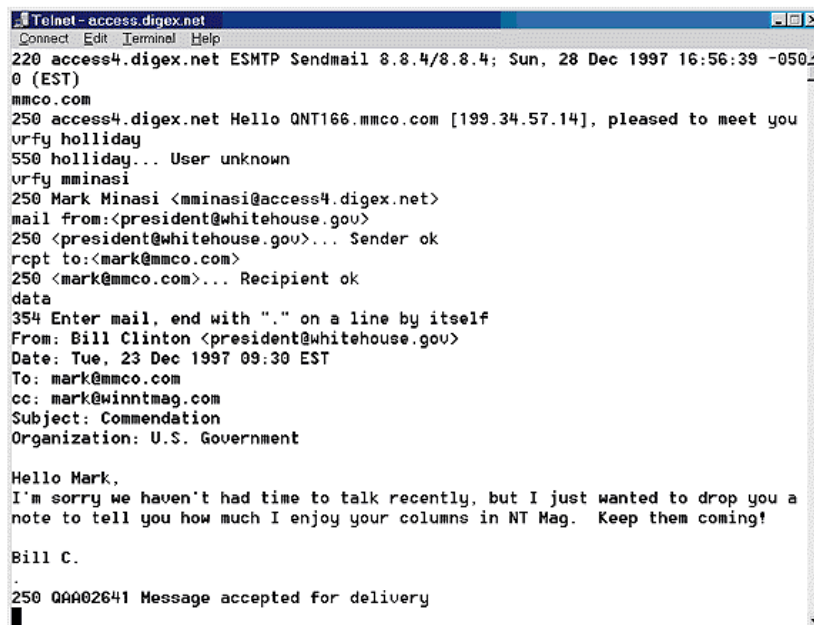
1.3.4 Telnet

Telnet là một ứng dụng cho phép người dùng tại một thiết bị (nguồn) có thể thông qua kết nối mạng đến một thiết bị (đầu cuối) ở xa để điều khiển nó bằng câu lệnh như là đang làm việc tại chính máy tính ở xa đó. Telnet cho phép tạo kết nối với thiết bị ở xa, thu thập thông tin và chạy chương trình.

Cú pháp: *telnet IP/host port*

trong đó, IP là địa chỉ IP của thiết bị đầu cuối, host là tên của thiết bị đầu cuối, port là cổng để giao tiếp với thiết bị đầu cuối.

Có thể sử dụng *telnet /?* để tìm hiểu thêm về các tham số khác.



```

Telnet - access.digex.net
Connect Edit Terminal Help
220 access4.digex.net ESMTp Sendmail 8.8.4/8.8.4; Sun, 28 Dec 1997 16:56:39 -0500
0 (EST)
mmco.com
250 access4.digex.net Hello QNT166.mmco.com [199.34.57.14], pleased to meet you
vrfy holliday
550 holliday... User unknown
vrfy mminasi
250 Mark Minasi <mminasi@access4.digex.net>
mail from:<president@whitehouse.gov>
250 <president@whitehouse.gov>... Sender ok
rcpt to:<mark@mmco.com>
250 <mark@mmco.com>... Recipient ok
data
354 Enter mail, end with "." on a line by itself
From: Bill Clinton <president@whitehouse.gov>
Date: Tue, 23 Dec 1997 09:30 EST
To: mark@mmco.com
cc: mark@winntmag.com
Subject: Commendation
Organization: U.S. Government

Hello Mark,
I'm sorry we haven't had time to talk recently, but I just wanted to drop you a
note to tell you how much I enjoy your columns in NT Mag. Keep them coming!

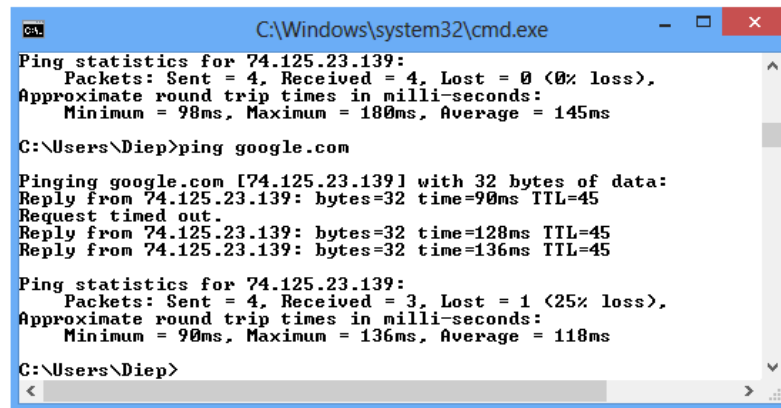
Bill C.
.
250 QAA02641 Message accepted for delivery
  
```

Hình 1.13 Telnet đến một mail server

1.3.5 Ping

Ping là một tiện ích quản trị mạng máy tính được sử dụng để kiểm tra khả năng đi đến đích của một host trên mạng và đo thời gian đi một vòng (RTT) của thông điệp từ nguồn đến đích. *Ping* hoạt động bằng cách gửi gói tin ICMP *echo request* đến host đích và chờ đợi một đáp ứng ICMP. Trong quá trình này nó sẽ đo thời gian truyền đến tiếp nhận (RTT) và ghi lại bất kỳ gói tin nào bị mất. Kết quả được hiển thị dưới dạng một bản tóm tắt thống kê các gói tin đáp ứng nhận được, bao gồm thời gian tối thiểu, thời gian tối đa và trung bình các RTT, và đôi khi có cả thông tin về độ lệch chuẩn của giá trị trung bình. Tùy thuộc vào cài đặt thực tế, tiện ích *ping* có thể được thực hiện với một số các tham số dòng lệnh khác nhau theo các chế độ hoạt động cụ thể.

Ping có thể bị lạm dụng như một hình thức đơn giản của tấn công từ chối dịch vụ dưới hình thức gây ra lụt *ping*, trong đó kẻ tấn công lần ất nặn nhân với các gói tin ICMP *echo request*.



```

C:\Windows\system32\cmd.exe

Ping statistics for 74.125.23.139:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 98ms, Maximum = 180ms, Average = 145ms

C:\Users\Diep>ping google.com

Pinging google.com [74.125.23.139] with 32 bytes of data:
Reply from 74.125.23.139: bytes=32 time=90ms TTL=45
Request timed out.
Reply from 74.125.23.139: bytes=32 time=128ms TTL=45
Reply from 74.125.23.139: bytes=32 time=136ms TTL=45

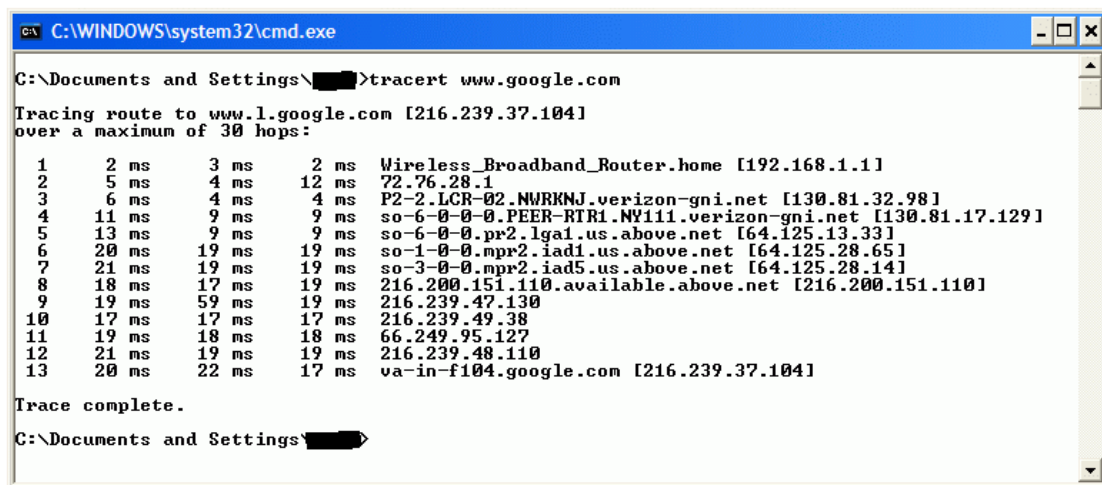
Ping statistics for 74.125.23.139:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 90ms, Maximum = 136ms, Average = 118ms

C:\Users\Diep>
    
```

Hìn 1.14 . Ping tới google.com

1.3.6 Tracert

Tracert là một công cụ dùng để hiển thị các tuyến đường (đường dẫn) và đo độ trễ vận chuyển các gói tin qua mạng Internet trên Microsoft Windows (trong các hệ điều hành khác như Linux hay Apple Mac OS nó có tên là *traceroute*). Thông tin lịch sử của tuyến đường được ghi lại là RTT của các gói tin nhận được từ mỗi host liên tiếp (nút ở xa) trong tuyến đường; tổng thời gian trung bình trong mỗi *hop* xác định tổng thời gian dùng để thiết lập kết nối. Trong quá trình *tracert*, nếu 3 gói dữ liệu gửi đi bị mất hơn hai lần thì kết nối bị mất và không cho kết quả đánh giá.



```

C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\>tracert www.google.com

Tracing route to www.l.google.com [216.239.37.104]
over a maximum of 30 hops:
  0  0 ms  0 ms  0 ms  Wireless_Broadband_Router.home [192.168.1.1]
  1  2 ms  3 ms  2 ms  72.76.28.1
  2  5 ms  4 ms  12 ms  P2-2.LCR-02.NWRKNJ.verizon-gni.net [130.81.32.98]
  3  6 ms  4 ms  4 ms  so-6-0-0-0.PEER-RTT1.NY111.verizon-gni.net [130.81.17.129]
  4  11 ms  9 ms  9 ms  so-6-0-0.pr2.lga1.us.above.net [64.125.13.33]
  5  13 ms  9 ms  9 ms  so-1-0-0.mpr2.iad1.us.above.net [64.125.28.65]
  6  20 ms  19 ms  19 ms  so-3-0-0.mpr2.iad5.us.above.net [64.125.28.14]
  7  21 ms  19 ms  19 ms  216.200.151.110.available.above.net [216.200.151.110]
  8  18 ms  17 ms  19 ms  216.239.47.130
  9  19 ms  59 ms  19 ms  216.239.49.38
 10  17 ms  17 ms  17 ms  66.249.95.127
 11  19 ms  18 ms  18 ms  216.239.48.110
 12  21 ms  19 ms  19 ms  va-in-f104.google.com [216.239.37.104]
 13  20 ms  22 ms  17 ms

Trace complete.

C:\Documents and Settings>
    
```

Hình 1.15 Tracert tới www.google.com

CHƯƠNG 2

CÁC GIAO THỨC TẦNG ỨNG DỤNG

Chương này giới thiệu về tầng ứng dụng và tập trung trình bày về các giao thức phổ biến được sử dụng tại tầng ứng dụng trong mạng Internet. Nội dung chính bao gồm:

- Các mô hình ứng dụng: client/server (khách/chủ), peer-to-peer (điểm-điểm).
- Các giao thức phổ biến: HTTP (web), FTP (truyền tệp tin), SMTP, POP, IMAP và MIME (thư điện tử).
- Giao thức cấu hình địa chỉ động DHCP
- Hệ thống tên miền DNS
- Đăng nhập từ xa: TELNET và SSH
- Giao thức quản lý mạng đơn giản SNMP

2.1 GIỚI THIỆU VỀ TẦNG ỨNG DỤNG

Tầng ứng dụng là nơi giúp cho người dùng tham gia vào môi trường mạng thông qua các chương trình ứng dụng. Việc truyền thông tại tầng này phụ thuộc vào các ứng dụng cụ thể, dữ liệu được truyền từ chương trình ứng dụng theo định dạng được quy định bởi ứng dụng, sau đó được đóng gói để chuyển xuống tầng giao vận. Phần sau trình bày hai mô hình ứng dụng mạng là mô hình client/server (khách/chủ) và mô hình peer-to-peer (điểm-điểm hay ngang hàng).

2.1.1 Mô hình client-server

Mục đích của mạng hay liên mạng là cung cấp các dịch vụ cho người dùng: một người dùng tại trạm cục bộ muốn nhận một dịch vụ từ một máy tính ở xa. Để thực hiện được việc này cần có hai chương trình chạy trên hai máy tính được kết nối Internet, một chương trình tại máy tính cục bộ yêu cầu dịch vụ và một chương trình tại máy tính ở xa cung cấp dịch vụ cho chương trình yêu cầu. Có một số vấn đề cần xem xét như sau:

1. Nên để cả hai chương trình ứng dụng đều có thể yêu cầu dịch vụ và cung cấp dịch vụ, hay mỗi chương trình chỉ thực hiện một việc và chương trình kia thực hiện việc còn lại? Giải pháp là phải có một chương trình ứng dụng, được gọi là *client*, chạy trên máy cục bộ, yêu cầu dịch vụ từ một chương trình ứng dụng khác, được gọi là *server*, chạy trên máy ở xa. Nói cách khác, nhiệm vụ yêu cầu dịch vụ và cung cấp dịch vụ là tách biệt nhau ở mỗi máy. Một chương trình ứng dụng, hoặc là *bên yêu cầu (client)*, hoặc là *bên đáp ứng (server)*. Nghĩa là, chương trình ứng dụng được xây dựng theo cặp, client và server, cả hai có cùng tên.
2. Nên để server chỉ cung cấp dịch vụ cho một client cụ thể hay server có thể cung cấp dịch vụ cho bất kỳ client nào yêu cầu loại dịch vụ mà nó cung cấp? Giải pháp chung nhất là server cung cấp dịch vụ cho bất kỳ client nào cần loại dịch vụ nó có. Nghĩa là server-client là mối quan hệ một-nhiều.

3. Có nên để một máy tính chỉ chạy một chương trình (client hoặc server)? Giải pháp là bất kỳ máy tính nào được kết nối với Internet đều có thể chạy bất kỳ chương trình client nào nếu phần mềm phù hợp có sẵn. Các chương trình server nên chạy trên một máy tính mà có thể chạy liên tục trong thời gian dài.
4. Khi nào một chương trình ứng dụng nên chạy? Tất cả mọi thời điểm hay chỉ khi có một yêu cầu dịch vụ? Nói chung, chương trình client, là chương trình yêu cầu dịch vụ, chỉ nên chạy khi có yêu cầu. Còn chương trình server, là chương trình cung cấp dịch vụ, nên chạy mọi thời điểm vì nó không thể biết được khi nào dịch vụ của nó được yêu cầu.
5. Nên có một chương trình ứng dụng toàn cầu có thể cung cấp bất kỳ loại dịch vụ nào mà người dùng muốn? Hay nên có một chương trình ứng dụng cho mỗi loại dịch vụ? Trong TCP/IP, các dịch vụ được yêu cầu thường xuyên và bởi nhiều người dùng có các chương trình ứng dụng client-server cụ thể. Ví dụ, có thể tách các chương trình ứng dụng client-server cho phép người dùng truy nhập tệp tin, gửi thư điện tử, v.v. Với các dịch vụ có nhiều tùy chọn hơn, cần có một chương trình ứng dụng chung cho phép người dùng truy nhập các dịch vụ có sẵn trên máy tính ở xa. Ví dụ, một chương trình ứng dụng client-server cho phép người dùng đăng nhập vào máy tính ở xa và sau đó sử dụng các dịch vụ máy tính đó cung cấp.

2.1.1.1 Server

Server là một chương trình chạy trên máy ở xa cung cấp dịch vụ cho các client. Khi server khởi động, nó mở cổng cho các yêu cầu từ nhiều client đến. Khi một yêu cầu đến, server sẽ đáp ứng lại yêu cầu, hoặc lặp đi lặp lại, hoặc đồng thời.

Server lặp đi lặp lại chỉ có thể xử lý một yêu cầu tại một thời điểm: nhận một yêu cầu, xử lý và gửi đáp ứng tới bộ đáp ứng trước khi xử lý yêu cầu khác. *Server đồng thời* có thể xử lý nhiều yêu cầu tại cùng một thời điểm và do vậy có thể chia sẻ thời gian của nó cho nhiều yêu cầu.

Server sử dụng UDP, giao thức tầng giao vận không hướng kết nối, hoặc TCP/SCTP, giao thức tầng giao vận hướng kết nối. Vì thế, server hoạt động phụ thuộc vào hai yếu tố: giao thức tầng giao vận và phương thức dịch vụ. Về mặt lý thuyết, có bốn loại server: lặp không kết nối, đồng thời không kết nối, lặp hướng kết nối và đồng thời hướng kết nối.

2.1.1.2 Client

Client là một chương trình chạy trên máy cục bộ yêu cầu dịch vụ từ một server. Một chương trình client là *hữu hạn*, nghĩa là nó được bắt đầu bởi người dùng (hoặc chương trình ứng dụng) và kết thúc khi dịch vụ hoàn thành. Thông thường, client mở kênh truyền thông bằng việc sử dụng địa chỉ IP của trạm ở xa và địa chỉ cổng đã được biết trước của chương trình server cụ thể hiện đang chạy trên máy đó. Sau khi kênh truyền thông được mở, client gửi yêu cầu và nhận đáp ứng. Mặc dù phần yêu cầu-đáp ứng có thể lặp đi lặp lại vài lần, nhưng toàn bộ quá trình là hữu hạn và cuối cùng sẽ phải đi đến kết thúc.

Các client có thể chạy trên một máy hoặc lặp đi lặp lại hoặc đồng thời. Việc chạy các client *lặp đi lặp lại* nghĩa là chạy chúng lần lượt: một client khởi tạo, chạy và kết thúc trước

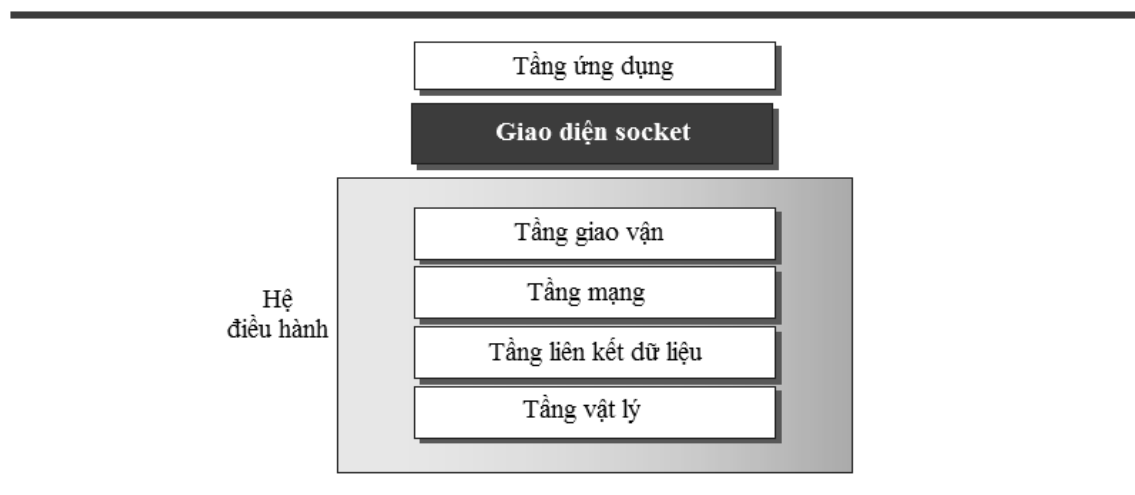
khi máy tính có thể khởi tạo một client khác. Tuy nhiên, hầu hết các máy tính ngày nay đều cho phép các client *đồng thời*, nghĩa là hai hoặc nhiều client có thể chạy tại cùng một thời điểm.

2.1.1.3 Giao diện Socket

Làm thế nào một tiến trình client có thể truyền thông với một tiến trình server? Chương trình máy tính là một tập chỉ dẫn được định nghĩa trước cho máy tính biết việc nó cần làm. Chương trình máy tính có một tập chỉ dẫn cho các tính toán toán học, tập chỉ dẫn khác cho việc xử lý chuỗi, tập chỉ dẫn khác cho truy nhập vào/ra. Khi cần một chương trình có thể truyền thông với một chương trình khác đang chạy trên một máy tính khác, cần một tập chỉ dẫn mới cho tầng giao vận biết cách mở kết nối, gửi dữ liệu và nhận dữ liệu từ đầu cuối khác và đóng kết nối. Một tập các chỉ dẫn loại này thường được coi như là một *giao diện (interface)*.

Có ba giao diện chung nhất được thiết kế cho việc truyền thông là: *giao diện socket*, *giao diện tầng giao vận (TLI)* và *luồng (stream)*. Chương trình mạng cần phải thân thiện với tất cả các giao diện này. Ở đây chỉ tóm tắt giao diện socket nhằm cung cấp một hiểu biết chung nhất về truyền thông mạng tại tầng ứng dụng.

Khái niệm giao diện socket được bắt đầu vào đầu những năm 1980 tại trường đại học Berkeley, như là một phần của môi trường UNIX. Để hiểu rõ hơn về khái niệm giao diện socket, cần phải hiểu được mối quan hệ giữa nền tảng hệ điều hành, như UNIX hoặc Windows, và chồng giao thức TCP/IP. Hình 2.1 biểu diễn quan hệ khái niệm giữa hệ điều hành và chồng giao thức TCP/IP.



Hình 2.1 Mối quan hệ giữa hệ điều hành và chồng giao thức TCP/IP

Giao diện socket, như là một tập chỉ dẫn, được đặt giữa hệ điều hành và các chương trình ứng dụng. Để truy nhập các dịch vụ được cung cấp bởi chồng giao thức TCP/IP, chương trình ứng dụng cần sử dụng các chỉ dẫn được định nghĩa trong giao diện socket.

Socket là một phần mềm khái quát mô phỏng một socket phần cứng như thường thấy trong cuộc sống. Để sử dụng kênh truyền thông, chương trình ứng dụng (client hoặc server) cần yêu cầu hệ điều hành tạo một socket. Sau đó chương trình ứng dụng có thể *cắm (plug)* vào

socket để gửi và nhận dữ liệu. Để việc truyền thông dữ liệu được diễn ra, cần có một cặp socket, mỗi cái ở tại một đầu cuối truyền thông.

Trong Internet, socket là một cấu trúc dữ liệu phần mềm. Định dạng cấu trúc dữ liệu để định nghĩa socket phụ thuộc vào nền tảng ngôn ngữ được sử dụng bởi các tiến trình. Người lập trình không được phép định nghĩa lại cấu trúc này mà chỉ cần yêu cầu sử dụng tệp tiêu đề (*header file*) có định nghĩa cấu trúc. Ví dụ, trong ngôn ngữ C, một socket được định nghĩa bởi cấu trúc gồm 5 trường: *family* (xác định nhóm giao thức: IPv4, IPv6,...), *type* (xác định loại socket), *protocol* (xác định giao thức sử dụng giao diện, được thiết lập là 0 cho TCP/IP), *local socket address* (xác định địa chỉ socket cục bộ, gồm địa chỉ IP và số hiệu cổng), và *remote socket address* (xác định địa chỉ socket ở xa).

2.1.2 Mô hình peer-to-peer

Mặc dù hầu hết các ứng dụng có sẵn trên Internet ngày nay sử dụng mô hình client-server, nhưng ý tưởng sử dụng mô hình peer-to-peer (P2P) hiện tại cũng đang được quan tâm. Trong mô hình này, hai máy tính ngang hàng (máy xách tay, máy để bàn, hay máy tính lớn mainframe) có thể truyền thông với nhau để trao đổi dịch vụ. Mô hình này khá được quan tâm trong một số lĩnh vực như truyền tệp tin, trong đó việc sử dụng mô hình client/server có thể đặt tải nặng lên server nếu client muốn truyền tệp tin lớn như audio hoặc video; hoặc trong trường hợp hai máy ngang hàng muốn trao đổi tệp hoặc thông tin với nhau mà không cần thông qua server.

Tuy nhiên, cần phải đề cập đến là mô hình P2P không bỏ qua mô hình client-server. Trong một số trường hợp, nó vẫn duy trì server để hỗ trợ cho quá trình chia sẻ bởi một số người dùng cần. Ví dụ, thay vì cho phép một số client tạo kết nối và tải về một tệp tin lớn, một server có thể để cho mỗi client tải về một phần của một tệp tin và sau đó chia sẻ với nhau. Tuy nhiên, trong quá trình tải một phần của tệp tin hoặc chia sẻ các tệp tin tải về, một máy tính cần phải đóng vai trò là client và một máy tính khác đóng vai trò là server. Nói cách khác, một máy tính có thể là client cho một ứng dụng cụ thể tại một thời điểm và là server vào thời điểm khác.

2.2 MỘT SỐ GIAO THỨC PHỔ BIẾN

Phần này trình bày một số giao thức phổ biến hoạt động tại tầng ứng dụng như HTTP, FTP, SMTP, POP, IMAP và MIME. Đây là các giao thức được sử dụng cho các ứng dụng duyệt web, truyền tệp và trao đổi thư điện tử, là những dịch vụ mà người dùng mạng thường xuyên sử dụng nhất.

2.2.1 HTTP (*Hypertext Transfer Protocol*)

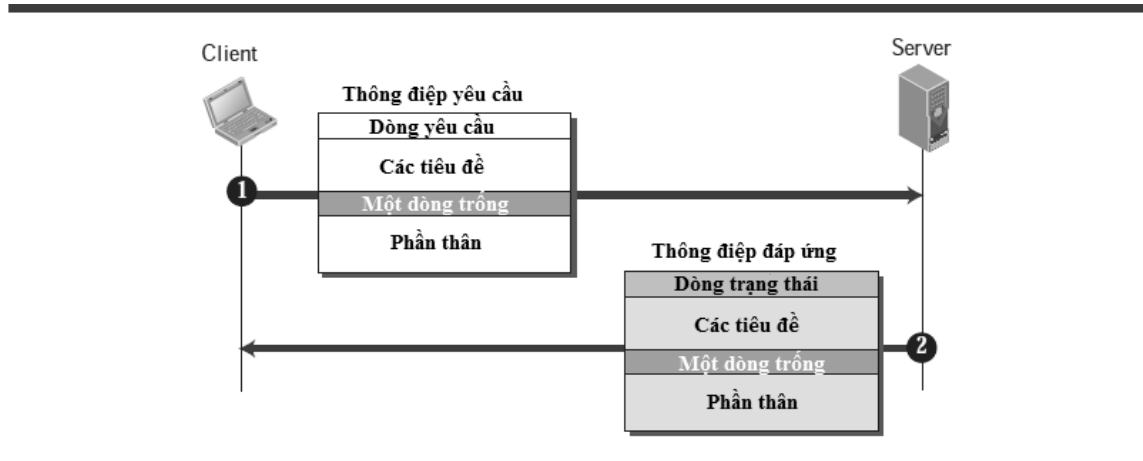
WWW (World Wide Web) là một dịch vụ client/server, trong đó client sử dụng trình duyệt có thể truy cập đến một dịch vụ ở server ở xa. Mỗi trạm có thể chứa một hoặc nhiều tài liệu, được gọi là các trang web, và mỗi trang web có thể chứa một số liên kết đến các trang web khác. Mỗi trang web là một tệp tin với tên và địa chỉ.

Hypertext Transfer Protocol (HTTP) là một giao thức được sử dụng chủ yếu để truy cập dữ liệu trên World Wide Web. HTTP đơn giản chỉ sử dụng một kết nối TCP trên cổng số 80 và dữ liệu được chuyển giao giữa client và server. Các thông điệp HTTP được truyền trực

tiếp, được đọc và hiểu bởi HTTP server và HTTP client (trình duyệt). Các lệnh từ client đến server được nhúng vào trong một thông điệp yêu cầu; nội dung trả lời cho yêu cầu và các thông tin khác được nhúng trong một thông điệp đáp ứng.

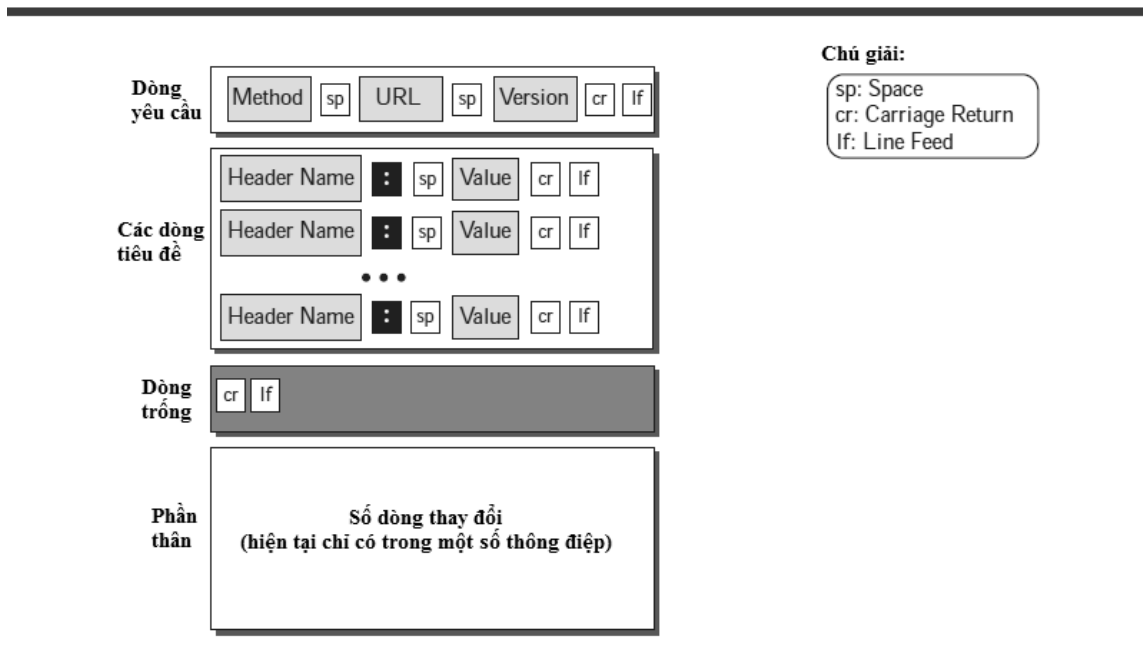
a. Giao dịch HTTP

Hình 2.2 minh họa giao dịch HTTP giữa client và server. Mặc dù HTTP sử dụng các dịch vụ của TCP, nhưng HTTP lại là một giao thức không trạng thái, nghĩa là server không lưu giữ thông tin về client. Các client khởi tạo giao dịch bằng cách gửi một yêu cầu, server trả lời bằng cách gửi một đáp ứng.



Hình 2.2 Giao dịch HTTP

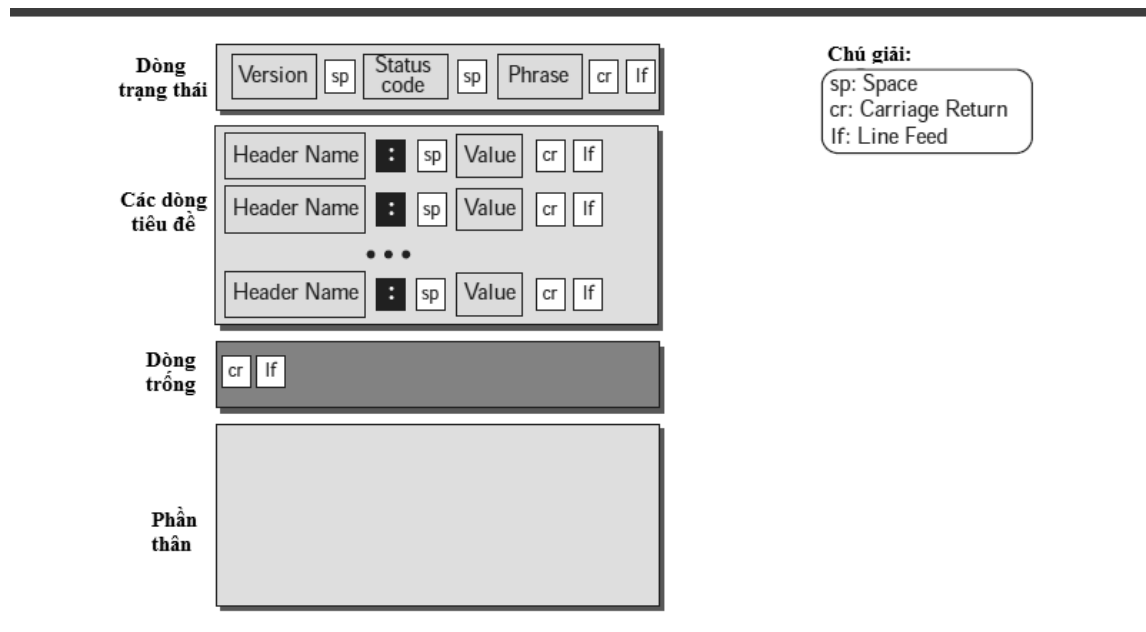
Thông điệp yêu cầu bao gồm một dòng yêu cầu, phần tiêu đề, một dòng trống, và (đôi khi) có một phần thân. Định dạng thông điệp yêu cầu được trình bày trong hình 2.3



Hình 2.3 Định dạng thông điệp yêu cầu

- **Dòng yêu cầu:** gồm 3 trường là *methods* (các phương thức), *URL* và *version* (phiên bản). Trong đó, các phương thức xác định loại yêu cầu, bao gồm: GET, HEAD, POST, PUT, TRACE, CONNECT, DELETE và OPTIONS (trong phiên bản HTTP v1.1, là phiên bản hiện đang được sử dụng).
- **Các dòng tiêu đề:** Trong thông điệp yêu cầu, có thể có hoặc không các dòng tiêu đề. Mỗi dòng gửi một thông tin bổ sung từ client tới server. Ví dụ, client có thể yêu cầu dữ liệu phải được gửi theo một định dạng đặc biệt nào đó. Mỗi dòng có tên tiêu đề, một dấu hai chấm, một ký tự trống và một giá trị tiêu đề.
- **Phần thân** trong thông điệp yêu cầu: có thể có hoặc không. Thông thường, nó chứa các bình luận được gửi tới.

Thông điệp đáp ứng bao gồm một dòng trạng thái, phần tiêu đề, một dòng trống, và (đôi khi) có một phần thân. Định dạng thông điệp đáp ứng được trình bày trong hình 2.4.



Hình 2.4 Định dạng của thông điệp đáp ứng

- **Dòng trạng thái:** gồm 3 trường là *version* (phiên bản của HTTP, hiện tại là 1.1), *status code* (xác định trạng thái của yêu cầu, biểu diễn dưới dạng mã số với 3 digit), và *phrase* (giải thích trạng thái, biểu diễn dưới dạng văn bản theo mã số trạng thái tương ứng). Ví dụ: 200 OK (yêu cầu thành công), 304 Not modified (tài liệu không được thay đổi), 400 Bad request (có lỗi cú pháp trong yêu cầu), v.v. Mã trạng thái nằm trong 5 dải số 100, 200, 300, 400 và 500.
- **Dòng tiêu đề:** có thể có hoặc không có dòng này. Mỗi dòng gửi một số thông tin bổ sung từ server cho client. Ví dụ, phía gửi có thể gửi thông tin thêm về tài liệu. Mỗi dòng tiêu đề bao gồm tên tiêu đề, dấu hai chấm, ký tự trống và một giá trị tiêu đề.
- **Phần thân:** Phần thân chứa tài liệu được gửi từ server cho client. Phần thân sẽ có trừ khi đáp ứng là một thông điệp báo lỗi.

b. Cookie

WWW được thiết kế như một thực thể không trạng thái: khi client gửi yêu cầu, server đáp ứng và kết thúc mối liên hệ. Trước đây, web chỉ thường được sử dụng để lấy các tài liệu có sẵn công cộng. Tuy nhiên, hiện nay web có thêm nhiều chức năng khác như:

- *Thương mại điện tử*: Người dùng có thể vào các trang web, là những cửa hàng điện tử, xem hàng, đặt hàng muốn mua và thanh toán tiền.
- *Bảo mật*: Một số trang web có thể yêu cầu đăng ký tham gia và đăng nhập khi muốn truy cập thông tin.
- *Cổng thông tin (portal)*: Người dùng chọn các trang Web mà họ muốn xem.
- *Quảng cáo*: Một số trang web chỉ có mục đích quảng cáo.

Vì các mục đích này, cơ chế cookie đã được đặt ra. Việc tạo ra và lưu trữ các tệp tin cookie phụ thuộc vào việc thực hiện, tuy nhiên, nguyên tắc là giống nhau:

- Khi server nhận được yêu cầu từ một khách hàng, nó sẽ lưu trữ thông tin về khách hàng trong một tệp tin hoặc một chuỗi. Thông tin có thể bao gồm các tên miền của khách hàng, nội dung của các tệp tin cookie (thông tin máy chủ đã thu thập được về khách hàng như tên, số đăng ký, v.v.), một dấu thời gian, và các thông tin khác tùy thuộc vào việc thực hiện.
- Server chứa các tệp tin cookie trong đáp ứng mà nó sẽ gửi cho khách hàng.
- Khi khách hàng nhận được phản hồi, trình duyệt lưu trữ cookie trong thư mục cookie, được sắp xếp theo tên máy chủ miền.

Khi khách hàng gửi một yêu cầu tới server, trình duyệt sẽ tìm trong thư mục cookie để xem có thể tìm thấy một cookie được gửi bởi server đó hay không. Nếu tìm thấy, các tệp tin cookie sẽ được gắn vào trong yêu cầu. Khi server nhận được yêu cầu, nó sẽ biết rằng đây là một khách hàng cũ chứ không phải là người mới. Chú ý là nội dung của các tệp tin cookie không bao giờ được đọc bởi trình duyệt hoặc tiết lộ cho người sử dụng. Đó là một cookie được tạo ra và được sử dụng bởi server. Phần sau là các phương thức mà một cookie được sử dụng cho bốn mục đích đã đề cập ở trên:

- *Thương mại điện tử*: Cookie được dùng cho người mua sắm (khách hàng) của cửa hàng điện tử. Khi một khách hàng chọn một mục và chèn nó vào một giỏ hàng, một cookie có chứa thông tin về các sản phẩm, chẳng hạn như số lượng và đơn giá của nó, được gửi đến trình duyệt. Nếu khách hàng chọn một mục thứ hai, cookie sẽ được cập nhật với những thông tin lựa chọn mới. Và cứ tiếp tục như vậy. Khi khách hàng hoàn thành mua sắm và muốn kiểm tra, mục cuối cùng được lấy và tổng số phí được tính toán.
- *Bảo mật*: Các trang web, ví dụ như các trang web thương mại điện tử, giới hạn việc truy cập cho các khách hàng đã đăng ký bằng cách chỉ gửi một cookie cho khách hàng khi họ đăng ký lần đầu tiên. Đối với những truy cập lặp đi lặp lại, chỉ có những khách hàng gửi cookie thích hợp mới được cho phép.

- *Cổng thông tin:* Một cổng thông tin web sử dụng cookie theo cách trên. Khi người dùng chọn các trang ưa thích của mình, một cookie sẽ được thực hiện và gửi đi. Nếu trang web được truy cập một lần nữa, các cookie được gửi đến máy chủ để hiển thị những gì khách hàng đang tìm kiếm.
- *Quảng cáo:* Một công ty quảng cáo có thể đặt banner (biểu ngữ) quảng cáo trên một số trang web chính mà thường được người dùng xem. Các công ty quảng cáo chỉ cung cấp một URL cho các địa chỉ banner thay vì chính banner riêng của họ. Khi người dùng truy cập các trang web chính và nhấp chuột vào biểu tượng của một công ty quảng cáo, một yêu cầu sẽ được gửi đến công ty quảng cáo. Công ty quảng cáo sẽ gửi banner, ví dụ là một tệp ảnh GIF, nhưng nó cũng đính kèm một cookie với ID của người sử dụng. Sau đó, bất kỳ việc sử dụng banner nào trong tương lai cũng sẽ được thêm vào cơ sở dữ liệu về hồ sơ hành vi sử dụng web của người dùng. Các công ty quảng cáo sẽ biên soạn những thông tin người dùng quan tâm và có thể bán những thông tin này cho các công ty khác. Điều này khiến cho việc sử dụng cookie rất gây tranh cãi. Hy vọng là trong tương lai sẽ có một số quy định mới được đưa ra để bảo vệ sự riêng tư của người dùng.

c. Web caching: Proxy server

HTTP hỗ trợ *proxy server*. Một proxy server là một máy tính lưu giữ bản sao của những đáp ứng cho các yêu cầu gần đây nhất. Khi HTTP client gửi một yêu cầu đến proxy server, nó sẽ kiểm tra bộ nhớ cache (tạm) của nó. Nếu câu trả lời không được lưu trữ trong bộ nhớ cache, proxy server sẽ gửi yêu cầu đến server tương ứng. Khi câu trả lời đến nó sẽ được gửi đến proxy server và được lưu trữ lại cho các yêu cầu trong tương lai.

Việc sử dụng proxy server sẽ giúp giảm tải trên các server gốc, làm giảm lưu lượng truy cập, và cải thiện độ trễ. Tuy nhiên, để sử dụng proxy server, client phải được cấu hình để truy cập proxy thay vì máy chủ đích.

Các proxy server hoạt động với cả hai vai trò là client và server. Khi nhận được yêu cầu từ client mà nó có câu trả lời, nó sẽ hoạt động như là server và gửi phản hồi cho client. Ngược lại, khi nhận được yêu cầu từ client mà nó không có câu trả lời, thì đầu tiên nó sẽ hoạt động như một client và gửi yêu cầu đến máy chủ đích. Sau đó, khi nhận được câu trả lời, nó sẽ hoạt động trở lại như là server và gửi phản hồi cho client.

Proxy server thường được đặt tại phía client. Nghĩa là có thể có một hệ thống phân cấp proxy server như sau:

1. Máy tính của khách cũng có thể được sử dụng như một proxy server với một công suất nhỏ, có khả năng lưu trữ những đáp ứng cho các yêu cầu thường xuyên của khách hàng.
2. Trong một công ty, một proxy server có thể được cài đặt trên máy tính trong mạng LAN để giảm tải lưu lượng đi ra ngoài và đi vào trong mạng LAN.
3. ISP với nhiều khách hàng có thể cài đặt một proxy server để giảm tải lưu lượng đi ra ngoài và đi vào trong mạng của ISP.

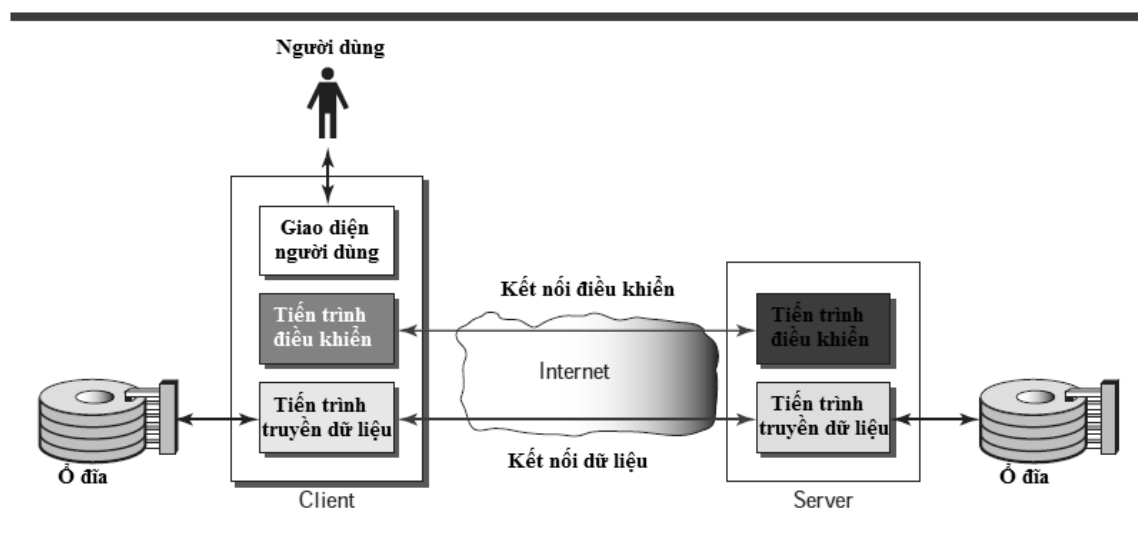
Một câu hỏi rất quan trọng là bao lâu thì cần phải cập nhật (thay đổi/xóa) một câu trả lời trong proxy server? Có một số chiến lược khác nhau, ví dụ như sau:

- Lưu trữ danh sách các trang web có thông tin trong một khoảng thời gian. Ví dụ, một hãng thông tấn có thể thay đổi trang tin tức của mình vào mỗi buổi sáng. Vì vậy, một proxy server có thể nhận được những tin tức vào buổi sáng sớm và giữ nó cho đến ngày hôm sau.
- Thêm một số tiêu đề để hiển thị thời gian sửa đổi cuối cùng của thông tin. Sau đó các proxy server có thể sử dụng các thông tin trong tiêu đề này để dự đoán về thời gian thông tin còn có giá trị.

2.2.2 FTP (File Transfer Protocol)

FTP là cơ chế chuẩn được cung cấp bởi TCP/IP cho việc sao chép một tệp tin từ một máy sang máy khác. Mặc dù việc chuyển các tệp tin từ một hệ thống này sang một hệ thống khác có vẻ là đơn giản và dễ hiểu, song có một số vấn đề phải được xử lý. Ví dụ, hai hệ thống có thể sử dụng quy ước tên tệp tin khác nhau, hai hệ thống có thể có nhiều cách khác nhau để trình bày văn bản và dữ liệu, hai hệ thống có thể có cấu trúc thư mục khác nhau, v.v. Tất cả những vấn đề này đều được giải quyết bởi FTP.

FTP khác với các ứng dụng client-server khác trong việc hình thành hai kết nối giữa các host. Một kết nối được sử dụng để truyền dữ liệu, còn một kết nối cho các thông tin điều khiển (lệnh và các đáp ứng). Việc phân chia các lệnh và truyền dữ liệu làm cho FTP hiệu quả hơn. Kết nối điều khiển sử dụng các quy tắc rất đơn giản của truyền thông, chỉ cần phải truyền một dòng lệnh hoặc một dòng đáp ứng tại một thời điểm. Mặt khác, kết nối dữ liệu cần các quy tắc phức tạp hơn do sự đa dạng của các kiểu dữ liệu được truyền. FTP sử dụng hai cổng TCP nổi tiếng: cổng 21 được dùng cho kết nối điều khiển và cổng 20 được dùng cho kết nối dữ liệu.



Hình 2.5 FTP (File Transfer Protocol)

Hình 2.5 cho thấy mô hình cơ bản của FTP. Client có ba thành phần: giao diện người dùng, tiến trình điều khiển client, và tiến trình truyền dữ liệu client. Server có hai thành phần: tiến trình điều khiển server và tiến trình truyền dữ liệu server. Kết nối điều khiển được tạo ra

giữa giữa hai tiến trình điều khiển. Kết nối dữ liệu được tạo ra giữa hai tiến trình truyền dữ liệu.

Kết nối điều khiển sẽ được duy trì trong suốt toàn bộ phiên hoạt động của FTP. Kết nối dữ liệu sẽ được mở ra (khi có lệnh liên quan đến việc truyền tệp) và sau đó sẽ được đóng lại khi mỗi tệp tin được truyền xong. Nói cách khác, khi người dùng bắt đầu một phiên FTP, kết nối điều khiển sẽ được mở ra. Trong khi kết nối điều khiển mở, kết nối dữ liệu có thể được mở và đóng nhiều lần nếu có một số tệp tin được truyền.

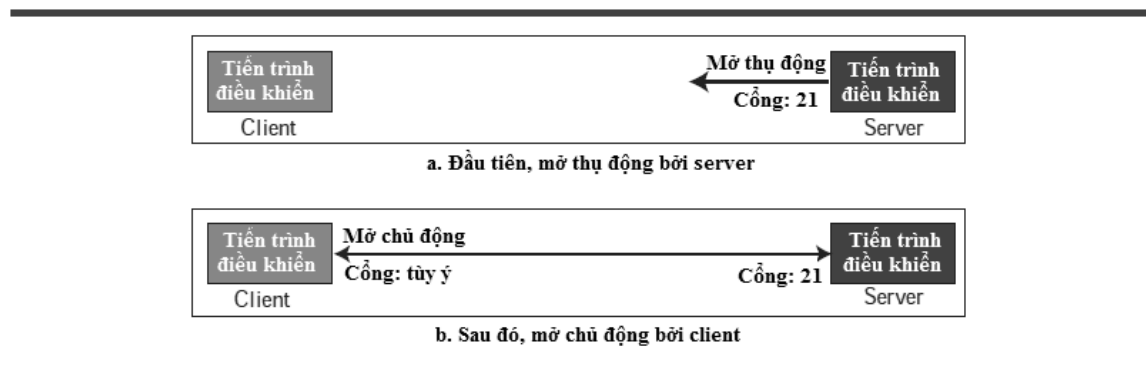
a. Các kết nối

Có hai kết nối FTP, điều khiển và dữ liệu, sử dụng các chiến lược khác nhau và các số hiệu cổng khác nhau.

Các kết nối điều khiển được tạo ra theo cách tương tự như các chương trình ứng dụng, gồm hai bước:

1. Server đưa ra một mở thụ động trên cổng (đã biết) số 21 và chờ đợi client.
2. Client dùng một cổng tùy ý và đưa ra một mở chủ động.

Kết nối sẽ được duy trì trong suốt toàn bộ quá trình. Dịch vụ được sử dụng ở đây, là giao thức IP, sẽ giảm tối thiểu sự chậm trễ vì đây là một kết nối tương tác giữa người dùng (là con người) và một server (máy tính). Người dùng gõ các lệnh và mong muốn nhận được đáp ứng ngay lập tức (gần như không có trễ). Hình 2.6 biểu diễn kết nối ban đầu giữa server và client.

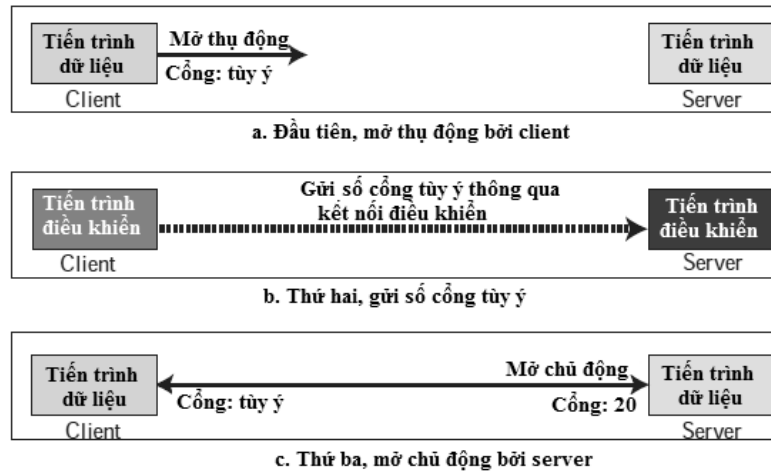


Hình 2.6 Mở kết nối điều khiển

Kết nối dữ liệu được sử dụng bởi cổng (đã biết) số 20 tại phía server. Tuy nhiên, việc tạo ra một kết nối dữ liệu khác với những gì chúng ta đã biết. Các bước tạo ra một kết nối dữ liệu FTP như sau:

1. Client (chứ không phải là server) đưa ra một mở thụ động với một cổng tùy ý, là do client là phía đưa ra các lệnh yêu cầu truyền tệp tin.
2. Client gửi số cổng này đến server bằng cách sử dụng lệnh PORT.
3. Server nhận số cổng và đưa ra một mở chủ động bằng cách sử dụng cổng (đã biết) số 20 và số cổng tùy ý đã nhận được.

Hình 2.7 trình bày các bước tạo ra các kết nối dữ liệu ban đầu.



Hình 2.7 Tạo kết nối dữ liệu

b. Việc truyền thông

FTP client và FTP server chạy trên các máy tính khác nhau và phải giao tiếp với nhau. Hai máy tính có thể sử dụng các hệ điều hành khác nhau, các bộ ký tự khác nhau, cấu trúc các tệp tin khác nhau, và định dạng các tệp tin khác nhau. FTP sẽ phải làm tương thích tất cả những tính chất không đồng nhất này.

FTP có hai cách tiếp cận khác nhau, một cho kết nối điều khiển và một cho kết nối dữ liệu.

Truyền thông qua kết nối điều khiển: FTP sử dụng cách tiếp cận tương tự như TELNET hoặc SMTP để giao tiếp qua kết nối điều khiển bằng bộ ký tự ASCII NVT. Truyền được thực hiện thông qua các lệnh và đáp ứng. Việc truyền thông rất đơn giản vì tại mỗi thời điểm chỉ cần gửi một lệnh (hoặc một đáp ứng). Mỗi lệnh hoặc đáp ứng chỉ là một dòng ngắn nên sẽ không cần lo lắng về định dạng hoặc cấu trúc tệp tin. Cuối mỗi dòng sẽ có ký tự báo kết thúc dòng.

Truyền thông qua kết nối dữ liệu: Mục đích là muốn chuyển các tệp tin. Client phải xác định các loại tệp tin được truyền, cấu trúc của dữ liệu, và các chế độ truyền dẫn. Trước khi gửi các tệp tin thông qua kết nối dữ liệu, cần phải chuẩn bị truyền thông qua kết nối điều khiển. Vấn đề không đồng nhất được giải quyết bằng cách xác định ba tính chất truyền thông: loại tệp tin, cấu trúc dữ liệu, và chế độ truyền dẫn.

- FTP có thể truyền một trong ba loại tệp tin qua kết nối dữ liệu, là: tệp ASCII (mặc định cho dữ liệu văn bản), tệp EBCDIC và tệp ảnh.
- FTP có thể truyền một tệp qua kết nối dữ liệu theo một trong các cách biên dịch sau về dạng cấu trúc dữ liệu: cấu trúc tệp (mặc định), cấu trúc bản ghi và cấu trúc phân trang.
- FTP có thể truyền tệp qua kết nối dữ liệu theo một trong ba chế độ truyền dẫn: chế độ dòng (mặc định), chế độ khối và chế độ nén.

c. Xử lý các lệnh

FTP sử dụng kết nối điều khiển để thiết lập việc truyền thông giữa tiến trình điều khiển client và tiến trình điều khiển server. Trong suốt quá trình truyền thông, các lệnh được gửi từ client tới server và các đáp ứng được gửi từ server tới client.

Các lệnh, được gửi từ tiến trình điều khiển client FTP, là ở dạng ASCII chữ hoa, và có thể có hoặc không có theo sau đối số. Có thể phân chia các lệnh thành sáu nhóm: lệnh truy cập, lệnh quản lý tệp tin, lệnh định dạng dữ liệu, lệnh xác định cổng, lệnh truyền tệp tin và các lệnh khác.

Mỗi lệnh FTP sẽ tạo ra ít nhất một đáp ứng. Mỗi đáp ứng gồm hai phần: một số gồm 3 digit (xác định mã) và tiếp theo là phần văn bản (xác định các thông số cần thiết hoặc giải thích thêm).

d. Truyền tệp tin

Việc truyền tệp tin xảy ra qua kết nối dữ liệu dưới sự điều khiển của các lệnh được gửi qua kết nối điều khiển. Truyền tin FTP được thực hiện theo một trong ba ngữ cảnh sau:

- Một tệp tin được sao chép từ server cho client (tải về), nghĩa là *lấy một tệp tin*. Việc này được thực hiện dưới sự giám sát của lệnh RETR.
- Một tệp tin được sao chép từ client đến server (tải lên), nghĩa là *lưu trữ một tệp tin*. Việc này được thực hiện dưới sự giám sát của lệnh STOR.
- Một danh sách các tên thư mục hoặc tên tệp tin được gửi từ server cho client. Việc này được thực hiện dưới sự giám sát của lệnh LIST. Chú ý là FTP xử lý một danh sách các tên thư mục hoặc tên tệp tin như là một tệp tin. Nó được gửi qua kết nối dữ liệu.

e. FTP vô danh (Anonymous FTP)

Để sử dụng FTP, người dùng cần một tài khoản (user name) và mật khẩu trên server ở xa. Một số trang web có một bộ các tệp tin có sẵn để truy cập công cộng. Để truy cập các tệp tin này, người dùng không cần phải có tài khoản và mật khẩu. Thay vào đó, người dùng có thể sử dụng *anonymous* (vô danh) như tên người dùng và mật khẩu là *guest*.

Người sử dụng truy cập vào hệ thống này sẽ bị giới hạn quyền. Một số trang web cho phép người dùng vô danh chỉ được sử dụng một tập con của các lệnh. Ví dụ, hầu hết các trang web cho phép người dùng sao chép một số tệp tin, nhưng sẽ không cho phép điều hướng thông qua các thư mục.

2.2.3 SMTP, POP, IMAP và MIME (Các giao thức thư điện tử)

Thư điện tử là một trong những dịch vụ Internet phổ biến nhất. Trước đây, vào lúc bắt đầu của thời đại Internet, các tin nhắn được gửi bằng thư điện tử ngắn và chỉ bao gồm văn bản, tuy nhiên, hiện nay, thư điện tử phức tạp hơn rất nhiều. Nội dung tin nhắn bao gồm văn bản, hình ảnh, âm thanh và video. Các hệ thống thư điện tử cũng cho phép tin nhắn được gửi đến một hoặc nhiều người nhận. Kiến trúc chung của một hệ thống thư điện tử bao gồm ba thành phần chính: thành phần người dùng (*user agent*), thành phần truyền tin nhắn (*message transfer agent*) và thành phần truy nhập thông điệp (*message access agent*).

2.2.3.1 Thành phần người dùng (User agent)

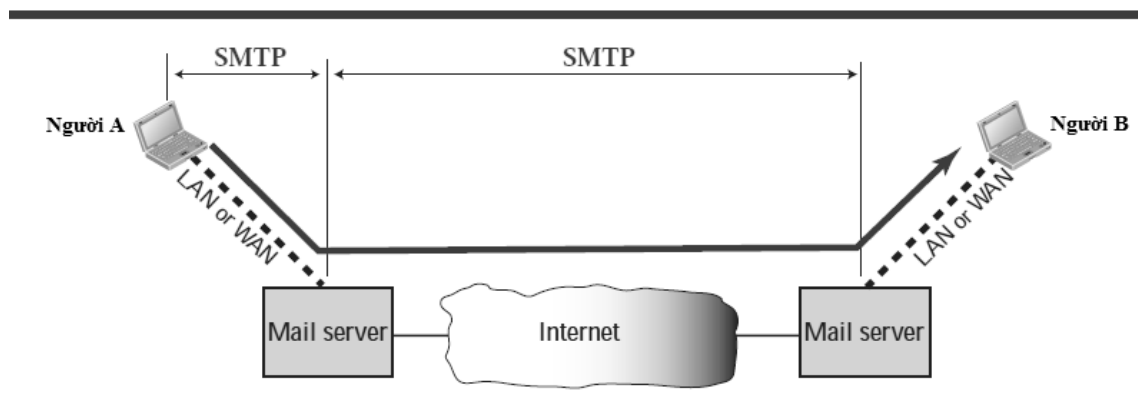
User agent (UA) cung cấp dịch vụ cho người dùng để thực hiện quá trình gửi và nhận tin nhắn dễ dàng hơn. UA là một gói phần mềm (chương trình) mà có thể giúp người dùng soạn thảo, đọc, trả lời và chuyển tiếp thư. Nó cũng có thể xử lý hộp thư cục bộ trên máy tính người dùng.

Có 2 loại user agent là hướng câu lệnh (*command-driven*) và hướng giao diện (*GUI-based*). Loại user agent hướng câu lệnh được sử dụng từ những ngày đầu của thư điện tử. Hiện loại này vẫn được sử dụng như là các lệnh cơ bản trong các server. Một user agent hướng câu lệnh thường chấp nhận một lệnh ký tự từ bàn phím để thực hiện nhiệm vụ. Ví dụ, người dùng gõ ký tự *r* tại dấu nhắc lệnh để trả lời cho người gửi, hoặc gõ ký tự *R* để trả lời cho người gửi và tất cả người nhận.

User agent hướng giao diện sử dụng giao diện đồ họa người dùng (GUI) để giúp người dùng tương tác với các phần mềm bằng cách sử dụng cả bàn phím và chuột. Loại này có các thành phần đồ họa như biểu tượng, các thanh menu, và cửa sổ làm cho việc truy nhập dịch vụ được dễ dàng.

2.2.3.2 Thành phần truyền tin nhắn (Message transfer agent - MTA): SMTP

Việc truyền thư thực tế được thực hiện thông qua thành phần truyền tin nhắn (MTA). Để gửi thư, hệ thống phải có MTA client, và để nhận thư, hệ thống phải có MTA server. Giao thức chuẩn định nghĩa MTA client và MTA server trong Internet được gọi là *giao thức truyền thư đơn giản* (Simple Mail Transfer Protocol - SMTP). Hình 2.8 biểu diễn việc sử dụng giao thức SMTP trong quá trình gửi thư điện tử.



Hình 2.8 Giao thức SMTP trong quá trình gửi thư điện tử

SMTP được sử dụng hai lần, giữa người gửi và mail server của người gửi và giữa hai mail server với nhau. Một giao thức khác sẽ được sử dụng giữa mail server và người nhận. SMTP chỉ đơn giản định nghĩa cách các lệnh và đáp ứng được gửi qua lại. Mỗi mạng sẽ tùy ý lựa chọn gói phần mềm thực hiện. Phần sau sẽ trình bày cơ chế truyền thư bằng SMTP.

a. Các lệnh và đáp ứng

SMTP sử dụng các lệnh và đáp ứng để truyền tin nhắn giữa MTA client và MTA server.

Các lệnh được gửi từ client tới server, gồm một từ khóa và theo sau là một số đối số (hoặc có thể không có). SMTP định nghĩa 14 lệnh, bao gồm: HELO, MAIL FROM, RCPT TO, DATA, QUIT, RSET, VRFY, NOOP, TURN, EXPN, HELP, SEND FROM, SMOL FROM, SMAL FROM. Ví dụ: *MAIL FROM: forouzan@challenger.atc.fhda.edu* (xác định người gửi tin nhắn, đối số là địa chỉ email của người gửi).

Các đáp ứng được gửi từ server tới client. Mỗi đáp ứng là một mã gồm 3 digit và có thể kèm theo sau là thông tin văn bản bổ sung. Ví dụ, 220 Service ready (dịch vụ sẵn sàng), 450 Mailbox not available (hộp thư không sẵn sàng), 500 Syntax error; unrecognized command (lỗi cú pháp; không hiểu lệnh), v.v.

b. Các pha thực hiện chuyển thư

Quá trình chuyển thư điện tử diễn ra theo ba giai đoạn: thiết lập kết nối, chuyển thư, và chấm dứt kết nối.

Thiết lập kết nối: Sau khi client tạo một kết nối TCP tới cổng số 25, SMTP server bắt đầu pha kết nối. Các bước như sau:

1. Server gửi mã 220 (dịch vụ sẵn sàng) để client biết nó đã sẵn sàng nhận thư. Nếu server chưa sẵn sàng, nó sẽ gửi mã số 421 (dịch vụ không có sẵn).
2. Client gửi tin nhắn HELO để xác định bản thân bằng cách sử dụng địa chỉ tên miền của nó. Mục đích là để thông báo cho các server tên miền của client. (Cần nhớ rằng trong quá trình thiết lập kết nối TCP, người gửi và người nhận biết nhau thông qua địa chỉ IP của họ).
3. Server trả lời với mã 250 (yêu cầu lệnh hoàn thành) hoặc một số mã khác tùy thuộc vào tình trạng thực tế.

Chuyển thư: Sau khi kết nối được thiết lập giữa SMTP client và SMTP server, một thông điệp đơn giữa một người gửi và một hoặc nhiều người nhận có thể được trao đổi. Giai đoạn này bao gồm 8 bước, trong đó bước 3 và bước 4 được lặp lại nếu có nhiều hơn một người nhận:

1. Client gửi thông điệp MAIL FROM để giới thiệu người gửi. Thông điệp này bao gồm địa chỉ email của người gửi (hộp thư và tên miền). Bước này cung cấp địa chỉ mail cho mail server để có thể thông báo trong trường hợp có lỗi và gửi thông điệp báo cáo.
2. Server đáp ứng với mã 250 hoặc một số mã thích hợp khác.
3. Client sẽ gửi RCPT TO (người nhận) thông báo, trong đó bao gồm địa chỉ email của người nhận.
4. Server đáp ứng với mã 250 hoặc một số mã thích hợp khác.
5. Client gửi thông điệp DATA để khởi tạo việc chuyển tin nhắn.
6. Server đáp ứng với mã 354 (bắt đầu đầu vào email) hoặc một số thông điệp thích hợp khác.
7. Client gửi nội dung của tin nhắn trong các dòng liên tiếp. Mỗi dòng có ký tự kết thúc dòng. Tin nhắn được kết thúc bằng một dòng chỉ chứa một ký tự cách.

8. Server đáp ứng với mã 250 (OK) hoặc một số mã thích hợp khác.

Chấm dứt kết nối: Sau khi tin nhắn được chuyển thành công, client sẽ chấm dứt kết nối. Giai đoạn này bao gồm hai bước:

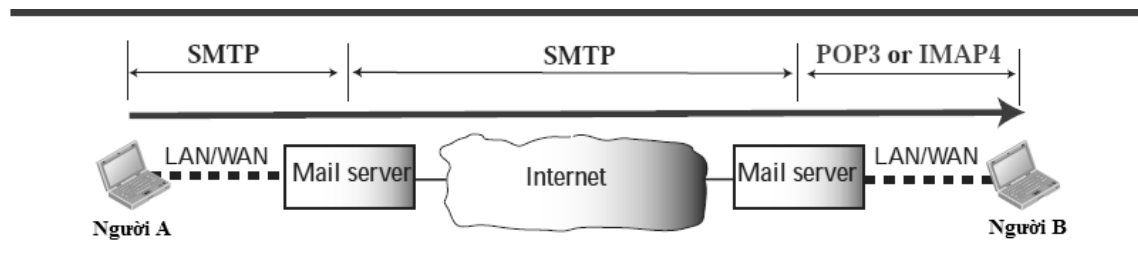
1. Client gửi lệnh QUIT.
2. Client đáp ứng với mã 221 (dịch vụ đóng kênh truyền) hoặc một số mã thích hợp khác.

Sau pha chấm dứt kết nối, kết nối TCP sẽ phải được đóng lại.

2.2.3.3 Thành phần truy nhập thông điệp (Message access agent): POP và IMAP

Giai đoạn đầu tiên và giai đoạn thứ hai của quá trình chuyển thư điện tử sử dụng giao thức SMTP. Tuy nhiên, SMTP không tham gia vào giai đoạn thứ ba vì SMTP là một giao thức đẩy (*push*), nó đẩy thông điệp từ client đến server. Nói cách khác, hướng của phần lớn dữ liệu (thông điệp) là từ client đến server. Mặt khác, giai đoạn thứ ba cần một giao thức kéo (*pull*), client phải kéo thông điệp từ server về. Hướng của phần lớn dữ liệu là từ server đến client. Giai đoạn thứ ba cần sử dụng một thành phần truy nhập thông điệp.

Hiện tại có hai giao thức truy nhập thông điệp có sẵn là POP (*Post Office Protocol*), phiên bản 3 (POP3) và IMAP (*Internet Mail Access Protocol*), phiên bản 4 (IMAP4). Vị trí của hai giao thức này trong quá trình chuyển thư điện tử được trình bày trong hình 2.9.



Hình 2.9 POP3 và IMAP4

a. POP3

POP3 là giao thức đơn giản và hạn chế về tính năng. Các phần mềm POP3 client được cài đặt trên máy tính của người nhận, còn phần mềm POP3 server được cài đặt trên mail server.

Việc truy cập thư bắt đầu với client khi người dùng cần tải thư về từ hộp thư (mail box) trên mail server. Client mở một kết nối đến server trên cổng TCP số 110. Sau đó, nó sẽ gửi tên người dùng và mật khẩu để truy nhập vào hộp thư. Người dùng sau đó có thể liệt kê và lấy thư về từng cái một.

POP3 có hai chế độ: *chế độ xóa* và *chế độ giữ*. Trong chế độ xóa, thư sẽ bị xóa khỏi hộp thư sau mỗi lần truy xuất. Trong chế độ giữ, thư vẫn còn trong hộp thư sau khi truy xuất. Chế độ xóa thường được sử dụng khi người dùng đang làm việc tại máy tính thường xuyên sử dụng của mình (vĩnh viễn) và có thể lưu và sắp xếp các thư nhận được sau khi đọc hoặc trả lời. Chế độ giữ thường được sử dụng khi người dùng truy nhập thư ở một máy tính ở xa máy tính vẫn thường sử dụng (ví dụ, máy tính xách tay). Thư đã được đọc nhưng vẫn sẽ được giữ trong hệ thống để có thể truy xuất và tổ chức sau.

b. IMAP4

IMAP4 là một giao thức truy nhập mail tương tự POP3 nhưng có nhiều tính năng hơn, IMAP4 mạnh hơn và phức tạp hơn POP3. POP3 không cho phép người dùng tổ chức mail trên server; người dùng không thể có các thư mục khác nhau trên server. (Mặc dù, người dùng có thể tạo các thư mục trên máy tính của mình.) Ngoài ra, POP3 không cho phép người dùng kiểm tra một phần nội dung của các email trước khi tải về. IMAP4 cung cấp các chức năng bổ sung sau:

- Người dùng có thể kiểm tra các tiêu đề thư trước khi tải về.
- Người dùng có thể tìm kiếm các nội dung của thư với một chuỗi cụ thể các ký tự trước khi tải về.
- Người dùng có thể tải về một phần của thư. Điều này đặc biệt hữu ích nếu băng thông bị hạn chế và thư có chứa dữ liệu đa phương tiện với yêu cầu băng thông cao.
- Người dùng có thể tạo, xóa, hoặc đổi tên các hộp thư trên mail server.
- Người dùng có thể tạo ra một cây phân cấp các hộp thư trong một thư mục để lưu trữ thư.

2.2.3.4 MIME (Multipurpose Internet Mail Extensions)

MIME là một giao thức bổ sung cho phép dữ liệu ASCII được gửi qua thư điện tử. MIME biến đổi dữ liệu không phải ASCII tại phía người gửi thành dữ liệu NVT ASCII và phân phối cho MTA client để gửi qua Internet. Ở phía nhận, các thông điệp được chuyển trở lại theo dạng dữ liệu gốc. Có thể coi MIME là một tập chức năng phần mềm giúp chuyển đổi dữ liệu không phải ASCII thành dữ liệu ASCII và ngược lại.

MIME định nghĩa năm tiêu đề có thể được thêm vào phần tiêu đề thư điện tử gốc để xác định các tham số chuyển đổi: *phiên bản MIME (MIME-Version)*, *loại nội dung (Content-Type)*, *mã hóa nội dung truyền (Content-Transfer-Encoding)*, *mã nội dung (Content-Id)*, *mô tả nội dung (Content-Description)*.

- *Phiên bản MIME*: Phiên bản hiện tại là 1.1
- *Loại nội dung*: xác định loại dữ liệu được sử dụng trong phần thân của tin nhắn. MIME sử dụng 7 loại dữ liệu khác nhau, gồm: văn bản, multipart (nhiều phần, độc lập), thông điệp, ảnh, video, âm thanh, ứng dụng (thông điệp gốc là loại dữ liệu không được định nghĩa trước).
- *Mã hóa nội dung truyền*: xác định các phương pháp được dùng để mã hóa tin nhắn thành dạng 0 và 1 để truyền đi, gồm 5 loại phương pháp mã hóa: 7bit (các ký tự NVT ASCII và dòng ngắn), 8bit (các ký tự không phải ASCII và dòng ngắn), Binary (các ký tự không phải ASCII và dòng không giới hạn), Base64 (các khối dữ liệu 6 bit được mã hóa thành ký tự 8 bit ASCII), Quoted-printable (các ký tự không phải ASCII được mã hóa thành một dấu bằng cộng với một mã ASCII).
- *Mã nội dung*: xác định duy nhất toàn bộ tin nhắn trong một môi trường nhiều tin nhắn.
- *Mô tả nội dung*: xác định xem phần thân là hình ảnh, âm thanh hay video.

2.3 GIAO THỨC DHCP (*Dynamic Host Configuration Protocol*)

Mỗi máy tính sử dụng chồng giao thức TCP/IP cần phải biết địa chỉ IP của nó. Nếu máy tính sử dụng địa chỉ không phân lớp hoặc là thành viên của một mạng con (*subnet*) thì nó cũng phải biết mặt nạ mạng (*subnet mark*). Thông thường cần bốn thông tin về địa chỉ như sau:

- Địa chỉ IP của máy tính
- Địa chỉ mặt nạ mạng của máy tính
- Địa chỉ IP của bộ định tuyến
- Địa chỉ của máy chủ đã được đặt tên.

Bốn thông tin này có thể được lưu lại thành một tệp cấu hình và được truy nhập bởi máy tính trong suốt quá trình bootstrap (mỗi khởi động). Các thông tin này phụ thuộc vào cấu hình máy cá nhân và hệ thống mạng mà máy tính được kết nối.

Trước khi DHCP trở thành giao thức chuẩn cho việc cấu hình địa chỉ cho các hệ thống đầu cuối (*hosts*), có một số giao thức khác đã được sử dụng cho mục đích này, ví dụ như giao thức RARP (*Reverse Address Resolution Protocol*) hay giao thức BOOTP (*Bootstrap Protocol*). DHCP là một giao thức hoạt động theo kiến trúc client/server, được thiết kế để cung cấp bốn thông tin về địa chỉ cho một máy tính không có ổ đĩa hoặc một máy tính được khởi động lần đầu tiên.

2.3.1 Hoạt động của DHCP

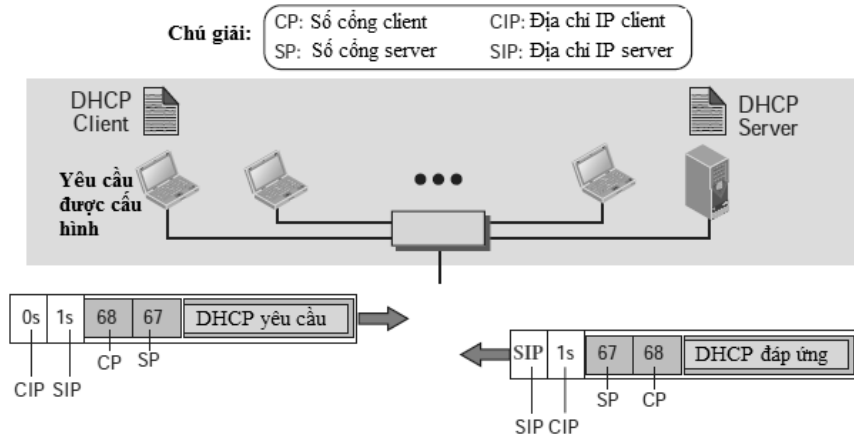
Máy chủ DHCP (*DHCP server*) và máy khách DHCP (*DHCP client*) hoặc có thể ở cùng trên một mạng hoặc có thể ở trên các mạng khác nhau.

2.3.1.1 DHCP client và DHCP server ở trên cùng một mạng

Trên thực tế tuy không phổ biến nhiều trường hợp DHCP client và DHCP server ở cùng trên một mạng nhưng các nhà quản trị hoàn toàn có thể làm được điều này. Hình 2.10 là một ví dụ.

Trong trường hợp này, hoạt động của DHCP được mô tả như sau:

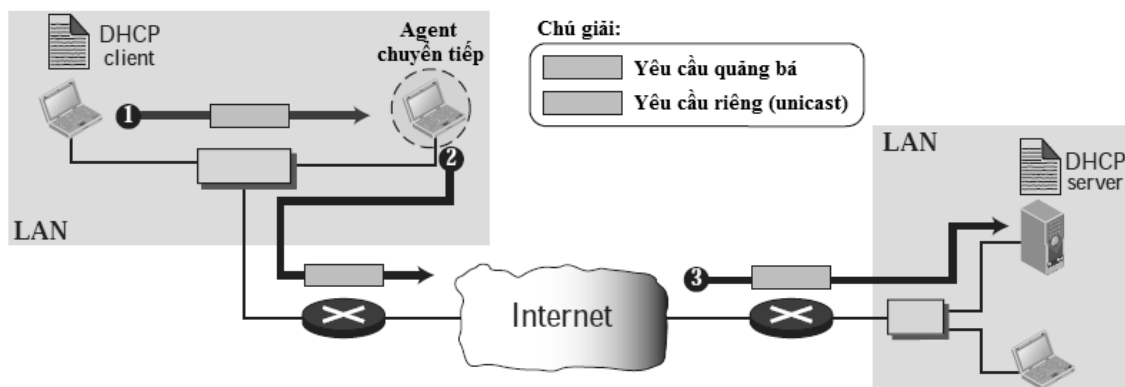
1. DHCP server đặt một lệnh mở thụ động trên cổng UDP 67 và chờ đợi client.
2. Một client đã khởi động đặt một lệnh mở chủ động trên cổng số 68. Thông điệp được đóng gói trong một gói tin UDP, sử dụng cổng số 67 là đích và cổng số 68 là nguồn. Các gói tin UDP lần lượt được đóng gói trong một gói tin IP. Vậy để client có thể gửi một gói tin IP khi nó hoặc chỉ biết địa chỉ IP của nó (địa chỉ nguồn), hoặc chỉ biết địa chỉ IP của server (địa chỉ đích), nó sử dụng tất cả các số 0 như là địa chỉ nguồn và tất cả các số 1 như là địa chỉ đích.
3. Server sẽ đáp ứng với hoặc là thông điệp quảng bá (*broadcast*) hoặc là thông điệp riêng (*unicast*) dùng cổng nguồn số 67 và cổng đích số 68. Đáp ứng có thể được gửi riêng (*unicast*) vì server biết được địa chỉ IP của client. Server cũng biết địa chỉ vật lý của client, có nghĩa là nó không cần đến dịch vụ phân giải địa chỉ ARP để chuyển từ địa chỉ logic sang địa chỉ vật lý. Tuy nhiên, một số hệ thống không cho phép bỏ qua ARP nên kết quả là vẫn cần phải sử dụng các địa chỉ quảng bá (*broadcast*).



Hình 2.10 DHCP client và DHCP server trên cùng một mạng

2.3.1.2 DHCP client và DHCP server ở trên hai mạng khác nhau

Như trong các tiến trình của tầng ứng dụng, một client có thể thuộc một mạng và server có thể thuộc một mạng khác. Hình 2.11 mô tả tình huống này.



Hình 2.11 DHCP client và DHCP server trên hai hệ thống mạng khác nhau

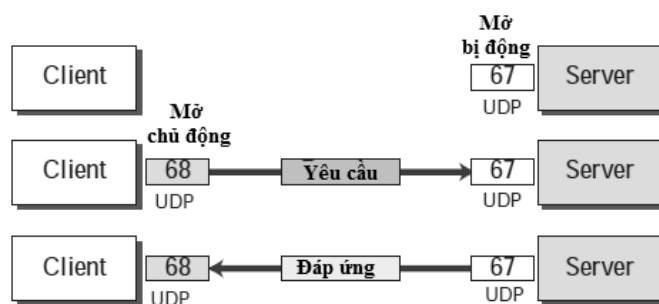
Tuy nhiên, có một vấn đề cần phải được giải quyết ở đây là DHCP yêu cầu (*DHCP request*) cần được quảng bá vì client không biết địa chỉ của server. Một gói tin IP quảng bá không thể đi qua bất kỳ bộ định tuyến nào, nghĩa là khi bộ định tuyến nhận được thì gói tin sẽ bị hủy bỏ. Lưu ý là một địa chỉ IP gồm toàn số 1 là một địa chỉ quảng bá giới hạn.

Để giải quyết vấn đề này, cần có một bộ phận trung gian, được gọi là *điểm chuyển tiếp* (*relay agent*), có thể là một trong các hệ thống đầu cuối (hoặc một bộ định tuyến có thể được cấu hình để hoạt động tại tầng ứng dụng). Các điểm chuyển tiếp biết địa chỉ riêng (*unicast*) của DHCP server và lắng nghe các thông điệp quảng bá trên cổng 67. Khi nhận được các gói tin loại trên, điểm chuyển tiếp sẽ thực hiện đóng gói các thông điệp trong một gói tin unicast và gửi yêu cầu tới DHCP server. Gói tin, mang địa chỉ đích unicast, được định tuyến bởi tất cả các bộ định tuyến đến được DHCP server. DHCP server biết điểm chuyển tiếp của thông điệp đến vì có một trường trong thông điệp yêu cầu có chứa địa chỉ IP của điểm chuyển tiếp. Sau khi nhận trả lời, điểm chuyển tiếp sẽ gửi thông điệp đến DHCP client.

2.3.1.3 Các cổng UDP

Hình 2.12 biểu diễn tương tác giữa một client và một DHCP server. Ở đây, server sử dụng cổng đã biết (số 67) là bình thường, tuy nhiên, client sử dụng cổng số 68 là không được thông dụng lắm. Lý do chọn cổng số 68 thay vì một cổng bất kỳ nào khác là vì muốn ngăn chặn vấn đề phát sinh khi server trả lời client theo kiểu quảng bá.

Để hiểu được vấn đề này, chúng ta xem xét cụ thể tình huống sử dụng một cổng bất kỳ. Giả sử máy A trên mạng sử dụng DHCP client trên cổng 2017 (chọn ngẫu nhiên). Máy B, trên cùng mạng, sử dụng DAYTIME client trên cổng 2017 (cũng là một sự chọn lựa ngẫu nhiên). Bây giờ, DHCP server gửi một thông điệp trả lời quảng bá với đích là cổng 2017 và địa chỉ IP quảng bá là $FFFFFFFF_{16}$. Tất cả các máy sẽ mở gói tin mang địa chỉ IP đích này. Máy A thấy thông điệp từ một chương trình ứng dụng trên cổng 2017, nó sẽ nhận được đúng chính xác thông điệp được chuyển tới nó (DHCP client). Máy B (DAYTIME client) sẽ nhận được một thông điệp không chính xác. Sự nhầm lẫn là do việc phân kênh (*demultiplexing*) của các gói tin dựa trên địa chỉ của socket (bao gồm địa chỉ IP và số cổng), mà trong trường hợp này, cả hai socket là giống nhau.



Hình 2.12 Sử dụng các cổng UDP

Việc sử dụng một cổng đã biết rõ (dưới 1024) ngăn chặn được tình huống dùng hai số cổng đích giống nhau. Máy B không thể chọn ngẫu nhiên cổng 68 vì các cổng chọn ngẫu nhiên có số hiệu lớn hơn 1023.

Câu hỏi đặt ra là điều gì sẽ xảy ra khi máy B cũng đang chạy DHCP client? Trong trường hợp này, địa chỉ của socket là giống nhau và cả hai client sẽ nhận được cùng một thông điệp. Với tình huống này, một số định danh thứ ba sẽ được dùng để phân biệt các client. DHCP dùng một số khác, được gọi là *mã giao dịch* (*transaction ID*), được lựa chọn ngẫu nhiên, cho mỗi kết nối liên quan đến DHCP. Như vậy sẽ rất khó có thể xảy ra trường hợp hai máy chọn cùng một mã tại cùng một thời điểm.

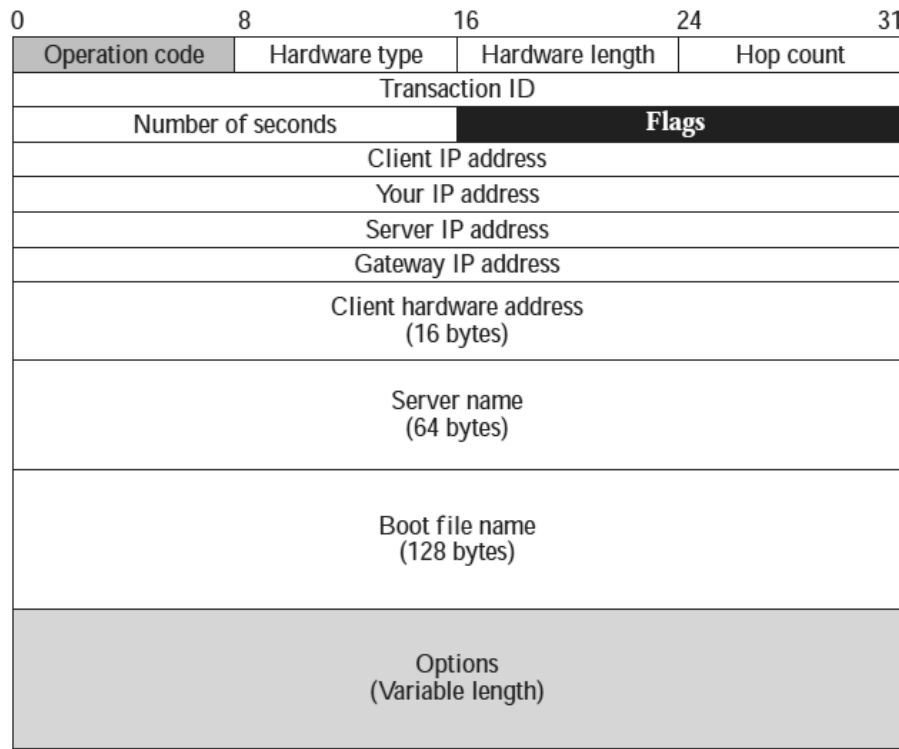
2.3.2 Điều khiển lỗi

Cần có một cơ chế điều khiển lỗi khi sử dụng DHCP, vì rất có thể các yêu cầu bị mất hoặc bị lỗi, hoặc các đáp ứng bị lỗi. Do DHCP sử dụng giao thức UDP, là giao thức không cung cấp cơ chế điều khiển lỗi, vậy nên bản thân DHCP sẽ phải cung cấp cơ chế này. Điều khiển lỗi được thực hiện qua hai chiến lược sau:

1. DHCP yêu cầu UDP sử dụng checksum, do việc sử dụng checksum trong UDP là tùy chọn.
2. DHCP client sử dụng các bộ định thời (timer) và chính sách truyền lại nếu không nhận được trả lời của DHCP khi nó yêu cầu. Tuy nhiên, để tránh trường hợp tắc nghẽn tại một số điểm cần truyền lại yêu cầu (ví dụ, khi mất nguồn), DHCP buộc client sử dụng một số ngẫu nhiên để thiết lập bộ định thời của nó.

2.3.3 Định dạng gói tin DHCP

Hình 2.13 trình bày định dạng của một gói tin DHCP.



Hình 2.13 Định dạng gói tin DHCP

Phần sau sẽ mô tả các trường dữ liệu:

- *Operation Code*: Gồm 8 bit xác định loại gói tin DHCP: yêu cầu (1) hoặc đáp ứng (2)
- *Hardware type*: Gồm 8 bit xác định loại mạng vật lý. Mỗi loại mạng được gán bởi một số nguyên, ví dụ 1 cho mạng Ethernet.
- *Hardware length*: Gồm 8 bit xác định độ dài của địa chỉ vật lý theo byte, ví dụ, 6 cho mạng Ethernet.
- *Hop count*: Gồm 8 bit xác định số bước (*hop*) lớn nhất mà gói tin có thể đi qua.
- *Transaction ID*: Gồm 4 byte, là một số nguyên. Đây là định danh giao dịch được xác định bởi client khi thực hiện một kết nối một đáp ứng với một yêu cầu. Server sẽ trả lại cùng một giá trị trong đáp ứng của nó.

- *Number of seconds*: Gồm 16 bit, xác định khoảng thời gian (tính bằng giây) từ thời điểm client bắt đầu khởi động.
- *Flag*: Gồm 16 bit, trong đó chỉ các bit phía tận cùng bên trái được sử dụng, còn phần còn lại được thiết lập là các bit 0. Bit phía ngoài cùng bên trái xác định đáp ứng từ server là quảng bá (*broadcast*) (thay vì *unicast*). Nếu đáp ứng là unicast tới client, thì địa chỉ IP đích của gói tin IP là địa chỉ được gán cho client. Vì client không biết địa chỉ IP của nó, nên gói tin có thể sẽ bị loại bỏ. Tuy nhiên, nếu gói tin IP là quảng bá, thì mọi máy đều sẽ nhận được và xử lý thông điệp quảng bá.
- *Client IP address*: Gồm 4 byte chứa địa chỉ IP của client. Nếu client không có thông tin này, thì giá trị của trường này là 0.
- *Your IP address*: Gồm 4 byte chứa địa chỉ IP của client. Trường này sẽ được điền bởi server (trong thông điệp đáp ứng) khi nhận được yêu cầu từ client.
- *Server IP address*: Gồm 4 byte chứa địa chỉ IP của server, được điền bởi server trong thông điệp đáp ứng.
- *Gateway IP address*: Gồm 4 byte chứa địa chỉ IP của bộ định tuyến, được điền bởi server trong thông điệp đáp ứng.
- *Client hardware address*: Là địa chỉ vật lý của client. Mặc dù server có thể lấy địa chỉ này từ khung dữ liệu được gửi từ client, nhưng sẽ hiệu quả hơn nếu địa chỉ được cung cấp một cách chính xác bởi client trong thông điệp yêu cầu.
- *Server name*: Gồm 64 byte, được điền tùy ý bởi server trong gói tin đáp ứng. Nó là một chuỗi không có kết cuối, bao gồm tên miền của server. Nếu server không muốn điền thông tin vào trường dữ liệu này, thì nó phải điền đầy tất cả là các bit 0.
- *Boot filename*: Gồm 128 byte có thể được điền tùy ý bởi server trong gói tin đáp ứng. Nó là một chuỗi không có kết cuối, bao gồm đầy đủ tên đường dẫn của tệp khởi động. Client có thể sử dụng đường dẫn này để lấy các thông tin khởi động khác. Nếu server không muốn điền thông tin vào trường dữ liệu này, thì nó phải điền đầy tất cả là các bit 0.
- *Options*: Gồm 64 byte, với hai mục đích. Nó có thể chứa thông tin bổ sung (như mật nạ mạng hoặc địa chỉ bộ định tuyến mặc định), hoặc một số thông tin nhà sản xuất cụ thể. Trường này chỉ được sử dụng trong thông điệp đáp ứng. Server sử dụng một số, gọi là *magic cookie* (*cookie ảo thuật*), theo định dạng của địa chỉ IP với giá trị là 99.130.83.99. Khi client đọc xong thông điệp, nó sẽ tìm kiếm magic cookie. Nếu có, 60 byte tiếp theo là tùy chọn. Một tùy chọn bao gồm ba trường dữ liệu: một trường 1 byte, một trường độ dài 1 byte, và một trường giá trị độ dài thay đổi. Trường độ dài xác định độ dài của trường giá trị chứ không phải toàn bộ phần tùy chọn.

Độ dài của các trường chứa địa chỉ IP là bội số của 4 byte. Phần *padding option*, chỉ có độ dài là 1 byte, được sử dụng cho các điều chỉnh. Phía cuối cùng của trường option, chỉ gồm 1 byte, xác định điểm kết thúc của trường này. Các nhà cung cấp có thể sử dụng thẻ tùy chọn từ 128 đến 254 để cung cấp thông tin mở rộng trong thông điệp đáp ứng.

2.4 HỆ THỐNG TÊN MIỀN DNS (Domain Name System)

Hệ thống tên miền (DNS), là một ứng dụng client/server, dùng để ánh xạ tên máy chủ (host) trong tầng ứng dụng thành một địa chỉ IP trong tầng mạng. Với sự phát triển của Internet ngày nay, các giải pháp sử dụng một *tệp tin host* hoặc lưu trữ toàn bộ các tệp tin host vào một hệ thống máy tính duy nhất là không còn phù hợp, bởi hệ thống liên tục phải được cập nhật tại mọi thời điểm có những thay đổi và lưu lượng truy cập trên Internet là vô cùng lớn. Một giải pháp khác, được sử dụng cho đến ngày nay, là phân chia lượng thông tin khổng lồ thành các phần nhỏ hơn và lưu trữ mỗi phần trên một hệ thống máy tính khác nhau, trong đó, các host cần ánh xạ có thể liên hệ với hệ thống máy tính gần nhất lưu giữ các thông tin cần thiết. Đây chính là giải pháp mà hệ thống tên miền (DNS) sử dụng.

Giả sử người dùng muốn sử dụng một client truyền tệp tin truy nhập vào một server truyền tệp tin chạy trên một máy tính ở xa. Người dùng chỉ biết tên gọi của server truyền tệp tin, ví dụ như *forouzan.com*. Tuy nhiên, TCP/IP cần địa chỉ IP của server truyền tệp tin để tạo kết nối. Sau đây là 6 bước ánh xạ tên host thành một địa chỉ IP:

1. Người dùng chuyển tên gọi của host đến client truyền tệp tin.
2. Client truyền tệp tin chuyển tên gọi máy chủ đến DNS client.
3. Mỗi máy tính, sau khi khởi động, sẽ biết được địa chỉ của một DNS server. Do vậy, DNS client sẽ gửi một thông điệp, là một câu truy vấn cho tên gọi của server truyền tệp tin, đến DNS server với địa chỉ IP của DNS server đã được biết.
4. DNS server trả lời địa chỉ IP của server truyền tệp tin mong muốn.
5. DNS client chuyển địa chỉ IP tới server truyền tệp tin.
6. Client truyền tệp tin lúc này sẽ sử dụng địa chỉ IP nhận được để truy nhập tới server truyền tệp tin.

Ở đây, mục đích của việc truy nhập Internet là để tạo ra một kết nối giữa client và server truyền tệp tin, nhưng trước khi có thể thực hiện được điều này thì cần phải thực hiện được kết nối giữa DNS client và DNS server. Nói cách khác, cần phải thực hiện hai kết nối: một kết nối đầu tiên cho việc ánh xạ tên gọi và địa chỉ IP; và kết nối thứ hai cho việc truyền các tệp tin (như ví dụ).

2.4.1 Không gian tên, không gian tên miền và phân bố tên miền

2.4.1.1 Không gian tên

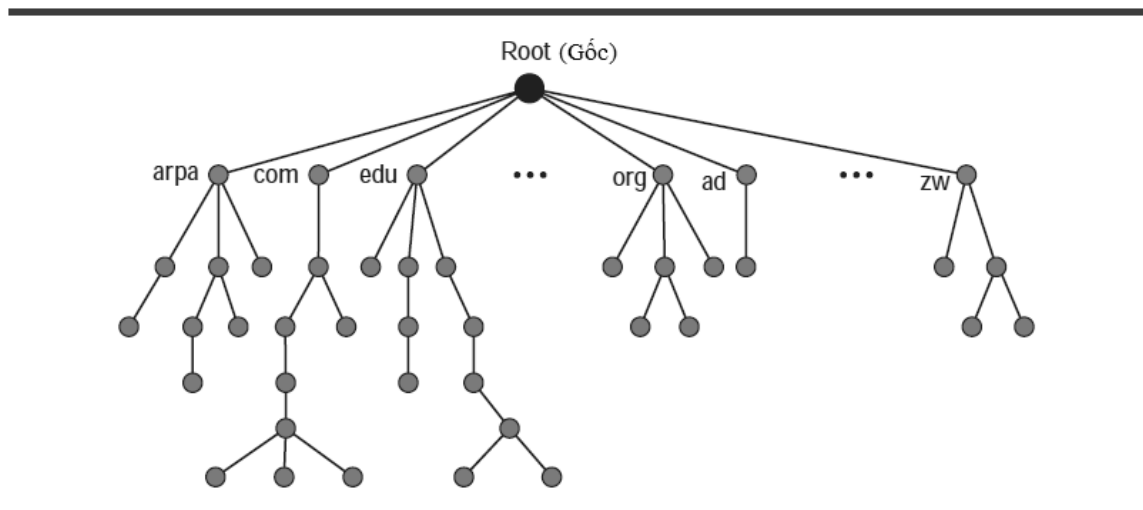
Để cho rõ ràng, tên gọi để gán cho máy cần phải được lựa chọn cẩn thận từ một không gian tên với tên gọi phải là duy nhất vì các địa chỉ cũng là duy nhất. Không gian tên cần phải được tổ chức theo hai cách: ngang bằng hoặc phân cấp.

- **Không gian tên ngang bằng:** Mỗi tên gọi trong không gian này là một chuỗi các ký tự không có cấu trúc. Tên gọi có thể có hoặc không có phần chung; nếu có nó cũng có thể không có ngữ nghĩa cụ thể. Nhược điểm chính của không gian tên ngang bằng là nó không thể được sử dụng trong các hệ thống lớn như mạng Internet vì cần phải được điều khiển tập trung để tránh không rõ nghĩa và trùng lặp.

- **Không gian tên phân cấp:** Mỗi tên gọi được tạo ra từ một số phần: phần đầu có thể xác định bản chất của tổ chức, phần thứ hai có thể là tên của tổ chức, phần thứ ba có thể là các phòng ban của tổ chức,... Trong trường hợp này, cơ quan có thẩm quyền thực hiện gán và điều khiển các không gian tên theo hướng không tập trung. Cấp có thẩm quyền trung tâm có thể gán phần tên xác định bản chất của tổ chức và tên của tổ chức. Trách nhiệm gán phần còn lại của tên có thể được giao cho chính tổ chức đó. Tổ chức có thể thêm các hậu tố (hoặc tiền tố) để tên xác định host hoặc các nguồn tài nguyên. Việc quản lý của tổ chức không cần quan tâm đến các tiền tố được chọn cho host bởi một tổ chức khác, vì thậm chí nếu một phần của địa chỉ là giống nhau thì toàn bộ địa chỉ sẽ vẫn khác nhau. Tên gọi là phần duy nhất không cần phải được gán bởi cơ quan có thẩm quyền trung tâm. Cơ quan này chỉ cần kiểm soát một phần của tên gọi chứ không cần kiểm soát toàn bộ.

2.4.1.2 Không gian tên miền

Để có được không gian tên phân cấp, cần phải thiết kế *không gian tên miền*: các tên gọi được định nghĩa theo một cấu trúc hình cây ngược với gốc là đỉnh trên cùng. Một cây có thể có 128 mức, từ mức 0 (là gốc), đến mức 127 (hình 2.14).



Hình 2.14 Không gian tên miền

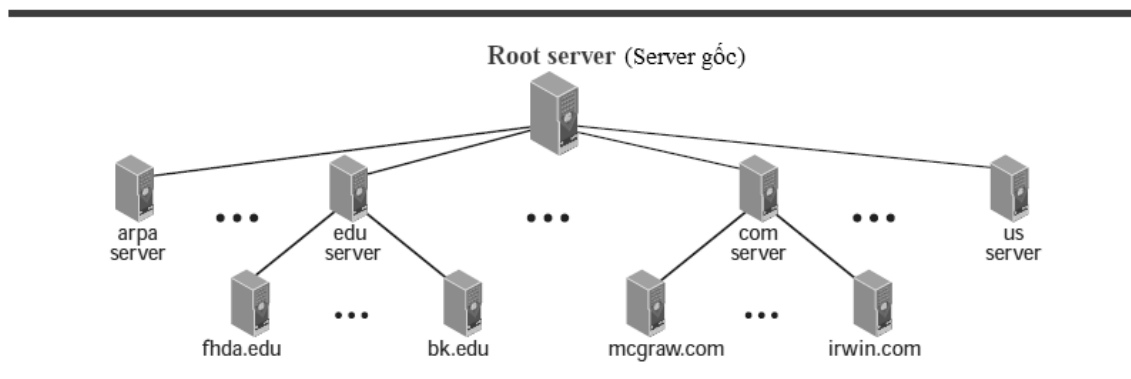
- **Nhãn:** Mỗi nút trên cây có một nhãn (*label*), là một chuỗi với tối đa là 63 ký tự. Nhãn của gốc là một chuỗi có giá trị là *null* (xâu rỗng). DNS yêu cầu tất cả các con của một nút (là các nút thuộc nhánh của nút đó) có các nhãn khác nhau, để đảm bảo tính duy nhất cho các tên miền.
- **Tên miền:** Mỗi nút trong cây có một tên miền. Một tên miền đầy đủ là một chuỗi gồm các nhãn được phân cách bởi dấu chấm (.). Các tên miền thường được đọc từ nút lên tới gốc. Nhãn cuối cùng là nhãn của gốc (*null*). Điều này có nghĩa là một tên miền đầy đủ luôn kết thúc bởi một nhãn có giá trị là *null*, nghĩa là ký tự cuối cùng là một dấu chấm vì chuỗi *null* không có giá trị gì cả. Ví dụ một số tên miền: edu., fhda.edu., atc.fhda.edu., v.v.

- **Miền:** là một cây con của không gian tên miền. Tên của miền là tên gọi của nút tại đỉnh của cây con. Một miền tự nó có thể phân chia thành các miền nhỏ hơn (hoặc miền phụ).

2.4.1.3 Phân bố tên miền

Thông tin chứa trong các không gian tên miền cần phải được lưu trữ. Tuy nhiên, sẽ là không hiệu quả và không tin cậy nếu chỉ lưu trữ lượng thông tin vô cùng lớn như này trên một máy tính. Không hiệu quả bởi vì việc đáp ứng lại tất cả những yêu cầu từ khắp mọi nơi trên thế giới sẽ làm hệ thống bị quá tải. Không tin cậy là do với bất kỳ một lỗi nào thì dữ liệu đều sẽ không thể được truy nhập.

Giải pháp cho các vấn đề này là phân tán thông tin trên nhiều máy tính, gọi là các *DNS server*. Việc này được thực hiện bằng cách phân chia toàn bộ không gian thành nhiều miền dựa trên mức đầu tiên. Mỗi miền được tạo ra theo cách này có thể rất lớn, nên DNS cho phép các miền có thể tiếp tục được chia ra thành các miền nhỏ hơn (gọi là *miền phụ*, *subdomain*). Mỗi server có thể chịu trách nhiệm (có thẩm quyền) với một miền lớn hoặc một miền nhỏ. Nghĩa là, có một cây phân cấp các server theo cùng cách phân cấp tên miền (xem hình 2.15).



Hình 2.15 Phân cấp các server tên miền

Vì phân cấp toàn bộ tên miền không thể lưu trữ được trên một server duy nhất, nên nó được phân chia giữa nhiều server. Tất cả những gì server phải chịu trách nhiệm hoặc có thẩm quyền được gọi là một *vùng* (*zone*). Có thể định nghĩa một vùng như là một phần tiếp giáp của toàn bộ cây. Nếu một server nhận trách nhiệm cho một miền và không thực hiện phân chia miền thành các miền nhỏ hơn nữa, thì khi đó “miền” và “vùng” là như nhau. Server tạo một cơ sở dữ liệu gọi là *tệp vùng* (*zone file*) và lưu tất cả các thông tin cho từng nút trong miền đó. Tuy nhiên, nếu server phân chia miền thành các miền phụ và ủy nhiệm một phần thẩm quyền của nó đến các server khác, thì khi đó “miền” và “vùng” là khác nhau. Thông tin về các nút trong miền phụ được lưu trữ trong các server tại các mức thấp hơn, trong khi server gốc giữ những thông tin tham chiếu đến các server mức thấp hơn đó. Dĩ nhiên, server gốc không giải phóng trách nhiệm hoàn toàn, mà nó vẫn có một vùng, nhưng thông tin chi tiết được lưu trữ tạo các server cấp thấp hơn.

Một server cũng vẫn có thể vừa phân chia miền của nó, vừa ủy quyền trách nhiệm, nhưng vẫn lưu giữ một phần thông tin miền cho chính nó. Trong trường hợp này, vùng được

tạo thành từ thông tin chi tiết của phần miền mà nó không ủy quyền và tham chiếu tới phần được ủy quyền.

- **Server gốc:** là server bao gồm toàn bộ cây. Server gốc thường không lưu trữ tất cả các thông tin về các miền mà nó sẽ ủy quyền cho các server khác, và giữ tham chiếu tới các server này. Có một số server gốc, mỗi server bao phủ toàn bộ không gian tên miền. Các server gốc được phân tán trên khắp thế giới.
- **Server sơ cấp:** là một server lưu trữ tập tin trong vùng mà nó có thẩm quyền. Nó có trách nhiệm tạo, bảo trì và cập nhật các tập vùng (được lưu trên ổ đĩa cục bộ).
- **Server thứ cấp:** là server chuyển toàn bộ thông tin vùng từ một server khác (sơ cấp hoặc thứ cấp) và lưu trữ tập trên ổ đĩa cục bộ của nó. Server thứ cấp không tạo mới và cũng không cập nhật các tập vùng. Nếu có yêu cầu cập nhật, thì việc này sẽ phải được thực hiện bởi server sơ cấp, và nó sẽ gửi phiên bản đã được cập nhật cho server thứ cấp.

Các server sơ cấp và thứ cấp đều có thẩm quyền trên các vùng mà nó quản lý. Ý tưởng ở đây là không đặt server thứ cấp tại mức thẩm quyền thấp hơn, mà dùng nó để tạo ra dự phòng dữ liệu cho tình huống server lỗi, và vẫn có thể tiếp tục phục vụ cho các client. Một server có thể là một server sơ cấp cho một vùng cụ thể và cũng có thể là server thứ cấp cho một vùng khác. Vì vậy, khi tham chiếu tới một server như là một server sơ cấp hoặc thứ cấp, thì cần phải cẩn thận xác định vùng nào mà mình cần tham chiếu tới.

2.4.2 DNS trên Internet

DNS là một giao thức có thể được sử dụng trên nhiều nền tảng khác nhau. Trên Internet, không gian tên miền (cây) được chia thành ba phần khác nhau: miền chung, miền quốc gia, và miền ngược.

- **Miền chung:** Miền chung xác định các host được đăng ký theo hoạt động chung của họ. Mức đầu tiên trong phần các miền chung bao gồm 14 nhãn, mô tả các loại tổ chức như được liệt kê như sau: *aero* (các công ty hàng không, vũ trụ), *com* (tổ chức thương mại), *edu* (tổ chức giáo dục), *gov* (tổ chức chính phủ), *mil* (quân đội), *coop*, *info*, *int*, *museum*, *name*, *net*, *org*, *pro*, *biz*.
- **Miền quốc gia:** Phần miền quốc gia sử dụng hai ký tự viết tắt cho tên quốc gia (ví dụ: *us* cho United States (Mỹ), *vn* cho Việt Nam). Các nhãn thứ hai có thể là tổ chức, hoặc chỉ một vùng cụ thể hơn trong quốc gia.
- **Miền ngược (Inverse Domain):** Miền ngược được sử dụng để ánh xạ một địa chỉ thành một tên gọi. Điều này có thể cần thiết trong một số trường hợp, ví dụ như client có yêu cầu gửi đến server để làm một việc gì đó. Mặc dù server có lưu tập tin chứa danh sách các client được ủy quyền, nhưng chỉ địa chỉ IP của client (được trích xuất từ các gói tin IP) được liệt kê ra. Server sẽ yêu cầu bộ phân giải của nó gửi truy vấn đến DNS server để ánh xạ địa chỉ thành một tên gọi nhằm xác định xem client nào thuộc danh sách có thẩm quyền. Loại truy vấn này được gọi là truy vấn ngược hoặc truy vấn con trỏ (PTR). Để xử lý một truy vấn con trỏ, miền ngược được thêm vào không gian tên miền với nút ở mức đầu tiên gọi là *arpa* (vì lý do lịch sử). Mức thứ hai cũng là một

nút đơn được gọi tên là *in-addr* (cho địa chỉ ngược). Phần còn lại của miền xác định địa chỉ IP.

2.4.3 Phân giải tên-địa chỉ

Việc ánh xạ tên thành địa chỉ hoặc địa chỉ thành tên được gọi là phân giải tên-địa chỉ (*name-address resolution*).

2.4.3.1 Bộ phân giải

DNS được thiết kế như một ứng dụng client/server. Một host cần ánh xạ một địa chỉ thành một tên gọi hoặc một tên gọi thành một địa chỉ được gọi là *phân giải*. Bộ phân giải truy nhập vào DNS server gần nhất để yêu cầu ánh xạ. Nếu server có thông tin, sẽ trả lời thỏa mãn yêu cầu của bộ phân giải, ngược lại, hoặc là nó sẽ tham chiếu bộ phân giải đến các server khác, hoặc là nó sẽ yêu cầu các server khác cung cấp thông tin.

Sau khi bộ phân giải nhận được ánh xạ, nó sẽ diễn giải đáp ứng để xem đây có phải là một phân giải đúng hay có lỗi gì không, và cuối cùng cung cấp kết quả cho tiến trình yêu cầu.

2.4.3.2 Ánh xạ tên thành địa chỉ

Trong hầu hết thời gian, các bộ phân giải gửi tên miền cho server và yêu cầu địa chỉ tương ứng. Với trường hợp này, server kiểm tra miền chung hoặc miền quốc gia để tìm ánh xạ.

Nếu tên miền thuộc phần miền chung, thì bộ phân giải sẽ nhận một tên miền có dạng như "*chal.atc.fhda.edu*". Truy vấn được gửi bởi bộ phân giải tới DNS địa phương để phân giải. Nếu server địa phương không thể phân giải truy vấn, nó sẽ hoặc là tham chiếu bộ phân giải tới các server khác hoặc yêu cầu các server khác trực tiếp thực hiện.

Nếu tên miền thuộc phần miền quốc gia, thì bộ phân giải sẽ nhận một tên miền có dạng "*ch.fhda.cu.ca.us*". Thủ tục thực hiện phân giải địa chỉ tương tự trên.

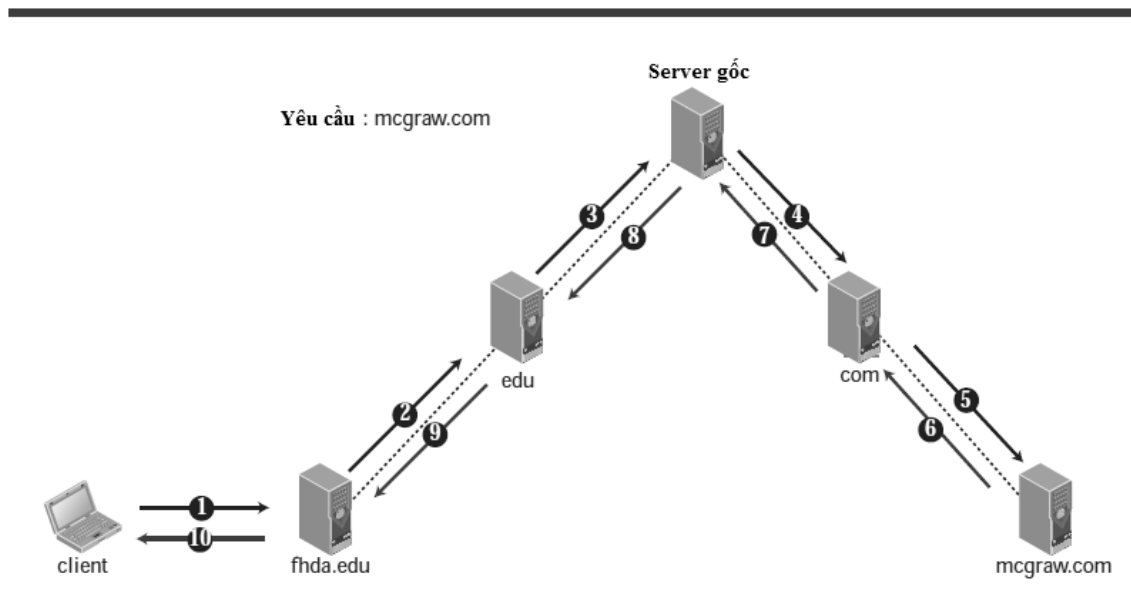
2.4.3.3 Ánh xạ địa chỉ thành tên

Client có thể gửi một địa chỉ IP tới server để yêu cầu ánh xạ thành một tên miền. Như đã đề cập ở trên, việc này được gọi là truy vấn con trỏ PTR. Để trả lời các truy vấn dạng này, DNS sử dụng miền ngược. Tuy nhiên, trong yêu cầu, địa chỉ IP cần được chuyển ngược và hai nhãn, *in-addr* và *arpa*, được nối với nhau để tạo thành một tên miền được chấp nhận bởi phần miền ngược. Ví dụ, nếu bộ phân giải nhận được địa chỉ IP là 132.34.45.121, bộ phân giải đầu tiên sẽ đảo ngược địa chỉ và sau đó thêm hai nhãn trước khi gửi. Tên miền được gửi sẽ là "*121.45.34.132.in-addr.arpa*", là tên sẽ được nhận bởi DNS địa phương và cần được phân giải.

2.4.3.4 Phân giải đệ quy

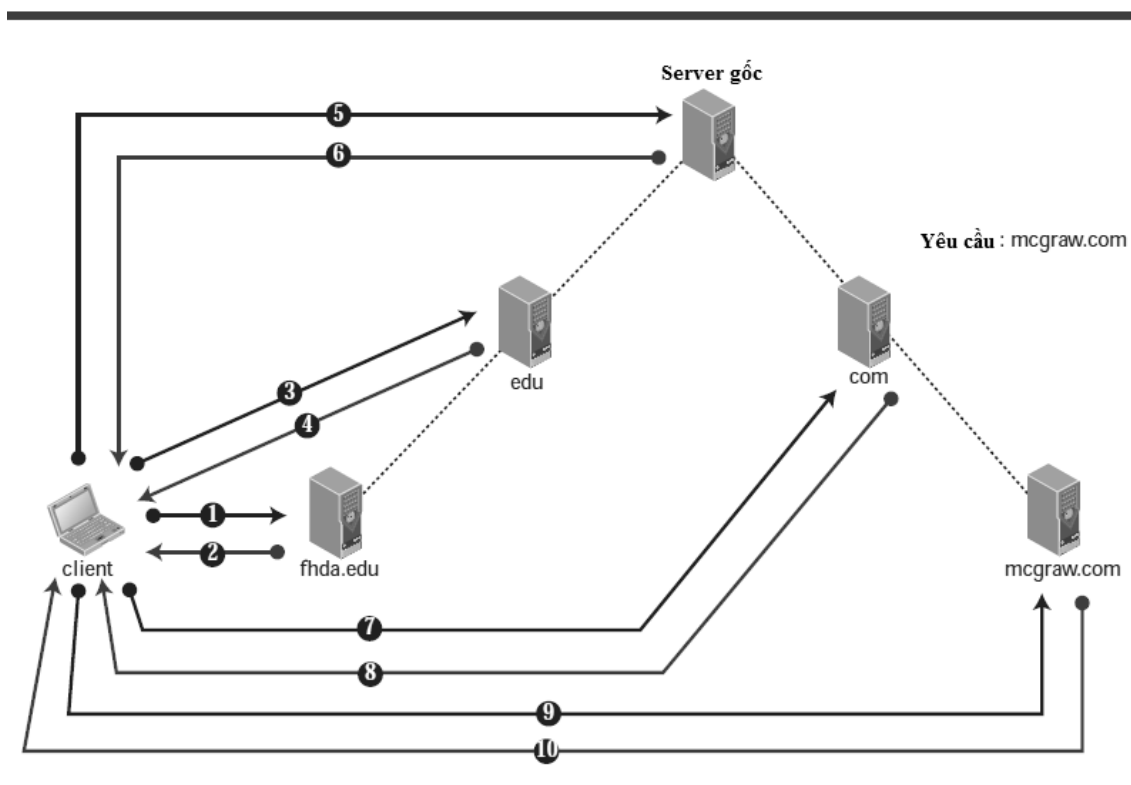
Client (bộ phân giải) có thể yêu cầu một câu trả lời đệ quy từ một server tên miền. Điều này có nghĩa là bộ phân giải mong muốn server đưa ra câu trả lời cuối cùng. Nếu server có thẩm quyền với tên miền, nó sẽ tìm kiếm trong cơ sở dữ liệu của nó và trả lời. Nếu server không có thẩm quyền, nó sẽ gửi yêu cầu tới một server khác (thường là server mức cao hơn (bố, mẹ)) và đợi câu trả lời. Nếu server mức cao hơn có thẩm quyền, nó sẽ trả lời, ngược lại,

nó lại tiếp tục gửi truy vấn tới một server khác. Khi cuối cùng truy vấn được phân giải, câu trả lời sẽ được chuyển ngược trở lại tới client yêu cầu (hình 2.16).



Hình 2.16 Phân giải đệ quy

2.4.3.5 Phân giải lặp



Hình 2.17 Phân giải lặp

Nếu client không yêu cầu một câu trả lời đệ quy, ánh xạ có thể được thực hiện theo phương pháp lặp. Nếu server có thẩm quyền với tên miền, nó sẽ gửi câu trả lời. Nếu server

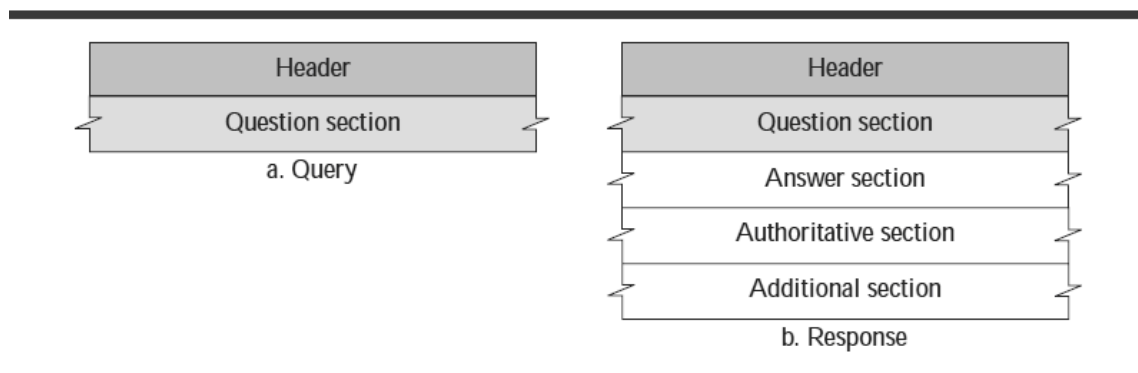
không có thẩm quyền, nó sẽ trả lại client địa chỉ IP của server mà nó “nghĩ” là server đó có thể phân giải được truy vấn. Client có trách nhiệm lặp lại truy vấn tới server thứ hai này. Nếu server mới có thể giải quyết vấn đề, nó sẽ đáp trả với một địa chỉ IP, ngược lại, nó sẽ gửi địa chỉ IP của server mới cho client. Khi này, client lại tiếp tục lặp lại truy vấn đến server thứ ba. Quá trình này được gọi là *lấp* vì client lấp đi lấp lại cùng một truy vấn tới nhiều server. Hình 2.17 biểu diễn client truy vấn tới 5 server trước khi nhận được câu trả lời từ server *mcgraw.com*.

2.4.3.6 Sử dụng bộ nhớ đệm (Caching)

Mỗi lần server nhận được một truy vấn tên miền mà không thuộc phạm vi thì nó cần phải tìm trong cơ sở dữ liệu của nó địa chỉ IP của các server khác. Việc giảm thời gian tìm kiếm sẽ làm tăng hiệu quả thực hiện. DNS xử lý việc này bằng cách sử dụng một kỹ thuật gọi là dùng *bộ nhớ đệm* (*caching*). Khi server yêu cầu ánh xạ từ server khác và nhận về đáp ứng, các thông tin này sẽ được lưu trữ trong bộ nhớ đệm trước khi gửi đi cho client. Nếu client đó hoặc các client khác tiếp tục yêu cầu ánh xạ này, server có thể kiểm tra bộ nhớ đệm và giải quyết vấn đề. Tuy nhiên, cần phải báo cho client biết rằng đáp ứng được lấy từ bộ nhớ đệm chứ không phải từ nguồn server có thẩm quyền, server sẽ đánh dấu đáp ứng là *unauthoritative* (không có thẩm quyền).

Việc sử dụng bộ nhớ đệm làm tăng tốc độ phân giải, tuy nhiên nó cũng có thể gây ra một số vấn đề. Nếu server lưu một ánh xạ trong bộ đệm trong một khoảng thời gian dài, nó có thể gửi một ánh xạ cũ chưa được cập nhật đến client. Để ngăn chặn vấn đề này, có hai kỹ thuật được sử dụng. Thứ nhất là, server có thẩm quyền thường thêm thông tin *thời gian sống* (*time-to-live, TTL*) vào trong ánh xạ. Thông số này là thời gian, tính bằng giây, mà server được phép lưu thông tin trong bộ nhớ đệm. Sau thời gian này, ánh xạ sẽ được coi là không hợp lệ và tất cả các truy vấn sẽ phải được gửi lại cho server có thẩm quyền. Thứ hai là, DNS yêu cầu mỗi server giữ một bộ đếm TTL cho mỗi ánh xạ trong bộ nhớ đệm. Bộ nhớ đệm sẽ được kiểm tra định kỳ và những ánh xạ nào có TTL hết hạn sẽ bị loại bỏ.

2.4.4 Các thông điệp DNS



Hình 2.18 Thông điệp yêu cầu và đáp ứng

DNS có hai loại thông điệp: yêu cầu (truy vấn) và đáp ứng (trả lời). Cả hai loại này đều có cùng định dạng. Thông điệp yêu cầu bao gồm phần tiêu đề (*header*) và các bản ghi yêu cầu (*question section*); thông điệp đáp ứng bao gồm phần tiêu đề (*header*), các bản ghi yêu

câu (*question section*), các bản ghi đáp ứng (*answer section*), các bản ghi có thẩm quyền (*authoritative section*) và các bản ghi bổ sung (*additional section*) (hình 2.18).

2.4.4.1 Định dạng của thông điệp

a. Phần tiêu đề (Header)

Cả hai thông điệp yêu cầu và đáp ứng có cùng định dạng phần tiêu đề với một số trường được thiết lập là 0 cho các thông điệp yêu cầu. Phần tiêu đề gồm 12 byte và có khuôn dạng như trong hình 2.19.

Identification	Flags
Number of question records	Number of answer records (Giá trị này là 0 trong thông điệp yêu cầu)
Number of authoritative records (Giá trị này là 0 trong thông điệp yêu cầu)	Number of additional records (Giá trị này là 0 trong thông điệp yêu cầu)

Hình 2.19 Định dạng tiêu đề

Các trường trong phần tiêu đề như sau:

- *Identification (Định danh)*: Gồm 16 bit được client sử dụng để kết nối đáp ứng với yêu cầu. Client sử dụng các số định danh khác nhau cho mỗi lần gửi truy vấn. Server lấy số này điền vào câu trả lời tương ứng.
- *Flag (Cờ)*: Gồm 16 bit chứa thông tin về các trường con như hình 2.20.

QR	OpCode	AA	TC	RD	RA	Three 0s	rCode
----	--------	----	----	----	----	----------	-------

Hình 2.20 Các trường trong cờ

Các trường trong cờ được mô tả như sau:

- ✓ *QR (query/response, yêu cầu/đáp ứng)*: Gồm 1 bit, xác định loại thông điệp: 0 là thông điệp yêu cầu, 1 là thông điệp đáp ứng.
- ✓ *OpCode*: Gồm 4 bit, xác định loại yêu hoặc loại đáp ứng: 0 là chuẩn, 1 là ngược, 2 là yêu cầu trạng thái server).
- ✓ *AA (authoritative answer, trả lời có thẩm quyền)*: Gồm 1 bit. Khi bit này được thiết lập là 1 thì server tên miền là server có thẩm quyền. Nó được dùng trong thông điệp đáp ứng.
- ✓ *TC (truncated, cắt bỏ)*: Gồm 1 bit. Khi bit này được thiết lập là 1 thì thông điệp đáp ứng có độ dài lớn hơn 512 byte và cần phải được cắt bỏ về còn 512 byte. Nó được sử dụng khi DNS dùng dịch vụ của UDP (trong quá trình đóng gói dữ liệu).
- ✓ *RD (recursion desired, yêu cầu đệ quy)*: Gồm 1 bit. Khi bit này được thiết lập là 1 thì client có mong muốn là nhận được một đáp ứng đệ quy. Bit này được thiết lập trong thông điệp yêu cầu và được lặp lại trong thông điệp đáp ứng.

- ✓ *RA (recursion available, đệ quy sẵn sàng)*: Gồm 1 bit. Khi bit này được thiết lập trong đáp ứng, có nghĩa là đáp ứng đệ quy đã sẵn sàng. Bit này chỉ được thiết lập trong thông điệp đáp ứng.
- ✓ *Reserved (dự phòng)*: Gồm 3 bit được thiết lập là 000.
- ✓ *rCode*: Gồm 4 bit biểu diễn trạng thái lỗi trong đáp ứng, chỉ có server có thẩm quyền mới có thể chỉ ra.
- *Number of question records (Số bản ghi yêu cầu)*: Gồm 16 bit chứa số lượng yêu cầu trong phần truy vấn của thông điệp.
- *Number of answer records (Số bản ghi trả lời)*: Gồm 16 bit chứa số lượng bản ghi trả lời trong phần trả lời của thông điệp đáp ứng. Giá trị này là 0 trong thông điệp yêu cầu.
- *Number of authoritative records (Số bản ghi có thẩm quyền)*: Gồm 16 bit chứa số bản ghi có thẩm quyền trong phần thẩm quyền của thông điệp đáp ứng. Giá trị này là 0 trong thông điệp yêu cầu.
- *Number of additional records (Số bản ghi bổ sung)*: Gồm 16 bit chứa số bản ghi bổ sung trong phần bổ sung của thông điệp đáp ứng. Giá trị này là 0 trong thông điệp yêu cầu.

b. Phần truy vấn

Phần này bao gồm một hoặc nhiều bản ghi yêu cầu, có trong cả thông điệp yêu cầu và thông điệp đáp ứng.

c. Phần trả lời

Phần này bao gồm một hoặc nhiều bản ghi tài nguyên, chỉ có trong thông điệp đáp ứng. Đây là câu trả lời từ server cho client (bộ phân giải).

d. Phần thẩm quyền

Phần này bao gồm một hoặc nhiều bản ghi tài nguyên, chỉ có trong thông điệp đáp ứng, cho biết thông tin (tên miền) của một hoặc nhiều server có thẩm quyền với câu truy vấn.

e. Phần thông tin bổ sung

Phần này bao gồm một hoặc nhiều bản ghi tài nguyên, chỉ có trong thông điệp đáp ứng, cung cấp các thông tin bổ sung có thể cần cho bộ phân giải. Ví dụ, một server có thể đưa thông tin về tên miền của server có thẩm quyền cho bộ phân giải trong phần thẩm quyền, và gồm cả địa chỉ IP của server thẩm quyền trong phần thông tin bổ sung.

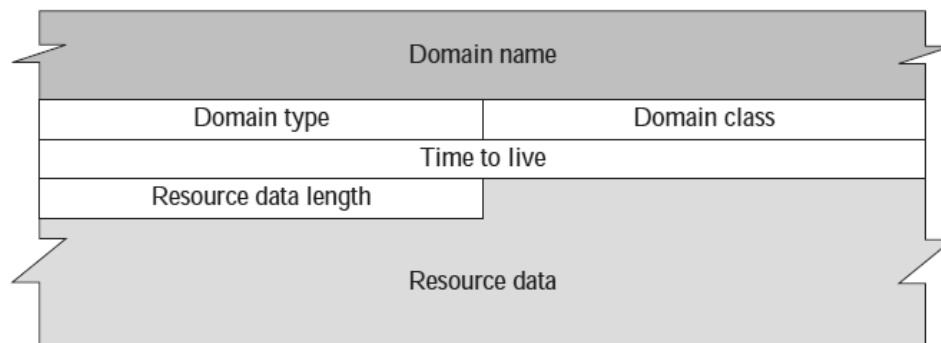
2.4.4.2 Các loại bản ghi

Như đã thấy trong phần trước, có hai loại bản ghi được sử dụng trong DNS. Các bản ghi yêu cầu được dùng trong phần yêu cầu của các thông điệp yêu cầu và đáp ứng. Các bản ghi tài nguyên được dùng trong các phần trả lời, thẩm quyền, và thông tin bổ sung của thông điệp đáp ứng.

a. Bản ghi yêu cầu: Một bản ghi yêu cầu được client dùng để lấy thông tin từ server, chứa tên miền, bao gồm các trường được mô tả như sau:

- *Query name (tên yêu cầu)*: Trường này có độ dài thay đổi, chứa tên miền.
- *Query type (loại yêu cầu)*: Gồm 16 bit xác định loại yêu cầu.
- *Query class (lớp yêu cầu)*: Gồm 16 bit xác định giao thức cụ thể dùng DNS. Ở đây, chỉ quan tâm đến lớp 1 (Internet).

b. Bản ghi tài nguyên: Mỗi tên miền (mỗi nút trên cây) được kết hợp với một bản ghi được gọi là bản ghi tài nguyên. Cơ sở dữ liệu của server chứa các bản ghi tài nguyên này. Các bản ghi tài nguyên cũng là những gì mà server trả lời lại cho client. Hình 2.21 trình bày định dạng của một bản ghi tài nguyên.



Hình 2.21 Định dạng bản ghi tài nguyên

- *Domain name (tên miền)*: Trường này có độ dài thay đổi, chứa tên miền. Nó được lặp lại từ tên miền trong bản ghi yêu cầu. Vì DNS yêu cầu nén ở mọi nơi nên nếu tên bị lặp lại thì trường này sẽ là một con trỏ chỉ ra trường tên miền tương ứng trong bản ghi yêu cầu.
- *Domain type (loại miền)*: Trường này giống như trường *query type* (loại yêu cầu) trong bản ghi yêu cầu.
- *Domain class (lớp miền)*: Trường này giống như trường *query class* trong bản ghi yêu cầu.
- *Time-to-live (thời gian sống)*: Gồm 32 bit xác định thời gian (tính bằng giây) mà câu trả lời còn hợp lệ. Đầu nhận có thể lưu câu trả lời trong bộ đệm trong khoảng thời gian này. Nếu nó có giá trị là 0 thì bản ghi tài nguyên chỉ được sử dụng trong một giao dịch đơn và không được lưu trong bộ đệm.
- *Resource data length (độ dài dữ liệu tài nguyên)*: Gồm 16 bit xác định độ dài của dữ liệu tài nguyên.
- *Resource data (dữ liệu tài nguyên)*: Trường này có độ dài thay đổi, chứa câu trả lời của truy vấn (trong phần đáp ứng) hoặc tên miền của server có thẩm quyền (trong phần thẩm quyền) hoặc thông tin bổ sung (trong phần thông tin bổ sung). Định dạng và nội dung của trường này phụ thuộc vào giá trị của trường loại miền, là một trong các trường hợp sau:

- ✓ *A number (Số)*: Được viết trong các octet. Ví dụ, một địa chỉ IPv4 gồm 4 octet số nguyên và địa chỉ IPv6 gồm 16 octet số nguyên.
- ✓ *A domain name (Tên miền)*: Tên miền được biểu diễn như là một chuỗi các nhãn. Trước mỗi nhãn là một trường 1 byte xác định số các ký tự trong nhãn. Vì mỗi tên miền kết thúc với một nhãn *null*, nên byte cuối cùng của mỗi tên miền là một trường độ dài với giá trị là bằng 0. Để phân biệt giữa một trường độ dài và một con trỏ lệch (*offset pointer*), hai bit thứ tự cao của trường độ dài luôn luôn là 00. Điều này không tạo ra rắc rối vì độ dài của nhãn không thể nhiều hơn 63, là số lớn nhất có 6 bit (111111).
- ✓ *An offset pointer (Con trỏ lệch)*: Các tên miền có thể được thay thế bởi một con trỏ lệch. Một con trỏ lệch là một trường 2 byte và mỗi trường có 2 bit thứ tự cao được thiết lập bằng 1 (11).
- ✓ *A character string (Chuỗi ký tự)*: Một chuỗi ký tự được biểu diễn bởi một trường có độ dài 1 byte tiếp theo bởi số ký tự xác định trường độ dài. Trường độ dài không bị giới hạn như trường độ dài của tên miền. Chuỗi ký tự có thể dài tới 255 ký tự (bao gồm cả trường độ dài).

2.4.5 Đóng gói dữ liệu

DNS có thể sử dụng một trong hai giao thức TCP và UDP. Trong cả hai trường hợp đều thường sử dụng cổng có số hiệu là 53. UDP được sử dụng trong trường hợp thông điệp đáp ứng ngắn hơn 512 byte, vì hầu hết các gói tin UDP có kích thước giới hạn là 512 byte. Nếu kích thước của thông điệp đáp ứng dài hơn 512 byte, thì sẽ phải sử dụng kết nối TCP. Trong trường hợp này, một trong hai kịch bản sau sẽ xảy ra:

Nếu bộ phân giải biết được trước kích thước của thông điệp đáp ứng, thì nó sẽ sử dụng kết nối TCP. Ví dụ, nếu một server tên miền thứ cấp (hoạt động như là một client) yêu cầu chuyển vùng từ một server sơ cấp, thì nó sẽ sử dụng kết nối TCP vì kích thước của thông tin cần chuyển thường vượt quá 512 byte.

Nếu bộ phân giải không biết trước được kích thước của gói tin đáp ứng thì nó có thể sử dụng cổng UDP. Tuy nhiên, nếu kích thước của thông điệp đáp ứng dài hơn 512 byte, thì server sẽ cắt thông điệp và thiết lập bit TC (xem hình 2.20). Bộ phân giải sẽ mở kết nối TCP và lặp lại yêu cầu cho đến khi nhận được đáp ứng đầy đủ từ server.

2.4.6 Bảo mật DNS

DNS là một trong những hệ thống quan trọng nhất trong cơ sở hạ tầng Internet, cung cấp các dịch vụ quan trọng cho người dùng Internet. Các ứng dụng, như truy cập web hoặc thư điện tử, phụ thuộc nhiều vào hoạt động đúng đắn của DNS. Tuy nhiên, DNS có thể bị tấn công theo nhiều cách như sau:

1. Kẻ tấn công có thể đọc các đáp ứng của DNS server để tìm thấy tên của các trang mà người dùng thường hay truy cập. Loại thông tin này có thể được sử dụng để tìm hồ sơ (profile) của người dùng. Để ngăn chặn các cuộc tấn công này, thông điệp DNS cần phải được giữ bí mật.

2. Kẻ tấn công có thể chặn các đáp ứng của DNS server và thay đổi nó hoặc tạo ra một đáp ứng không có thật hoàn toàn mới và chuyển hướng người dùng đến trang web hoặc miền kẻ tấn công muốn người dùng truy cập. Kiểu tấn công này có thể được bảo vệ bằng cách sử dụng xác thực nguồn gốc thông điệp và toàn vẹn thông điệp.
3. Kẻ tấn công có thể làm ngập lụt DNS server để áp đảo hay thậm chí làm sụp đổ nó. Kiểu tấn công này có thể được bảo vệ bằng cách sử dụng phương pháp ngăn chặn tấn công từ chối dịch vụ (DOS attack).

Để bảo vệ DNS, IETF (*Internet Engineering Task Force, tổ chức chuyên trách về kỹ thuật mạng Internet*) đã phát minh ra một công nghệ có tên DNS Security (DNSSEC). Công nghệ này cung cấp xác thực nguồn gốc và toàn vẹn thông điệp sử dụng dịch vụ bảo mật là *chữ ký số (digital signature)*. Tuy nhiên, DNSSEC không đảm bảo bí mật cho các thông điệp DNS. DNSSEC không có đặc tả cụ thể nào cho bảo vệ tấn công từ chối dịch vụ. Tuy nhiên, hệ thống bộ nhớ đệm bảo vệ các server ở cấp độ cao hơn, có thể chống lại các tấn công này trong một số trường hợp.

2.5 ĐĂNG NHẬP TỪ XA: TELNET VÀ SSH

2.5.1 TELNET

Telnet được viết tắt từ *Terminal Network*, là một giao thức TCP/IP chuẩn cho dịch vụ đầu cuối ảo được đề xuất bởi ISO. Telnet có thể thiết lập một kết nối tới một hệ thống ở xa và thiết bị đầu cuối hiện tại được hiển thị như là một thiết bị đầu cuối tại hệ thống ở xa đó. Telnet cũng là tên của chương trình ứng dụng truy nhập từ xa dựa theo kiến trúc client/server.

2.5.1.1 Một số khái niệm liên quan

a. Môi trường chia sẻ thời gian (*Time-Sharing Environment*)

Telnet được thiết kế tại thời điểm mà hầu hết các hệ điều hành, như UNIX, hoạt động trong môi trường chia sẻ thời gian. Trong môi trường này, một máy tính lớn sẽ hỗ trợ nhiều người dùng. Tương tác giữa người dùng và máy tính được thực hiện thông qua một thiết bị đầu cuối, thường có kết nối bàn phím, màn hình và chuột. Thậm chí một máy tính siêu nhỏ (*microcomputer*) cũng có thể mô phỏng một thiết bị đầu cuối với một giả lập thiết bị đầu cuối.

Trong môi trường chia sẻ thời gian, tất cả quá trình xử lý phải được thực hiện bởi máy tính trung tâm. Khi người dùng gõ ký tự từ bàn phím, ký tự thường được gửi tới máy tính và hiển thị ra màn hình. Chia sẻ thời gian tạo ra một môi trường trong đó mỗi người dùng có được ảo ảnh của một máy tính chuyên dụng. Người dùng có thể chạy chương trình, truy nhập vào tài nguyên hệ thống, chuyển đổi từ chương trình này sang chương trình khác, v.v.

b. Đăng nhập (*Login*)

Trong môi trường chia sẻ thời gian, người dùng là một phần của hệ thống và có một số quyền truy nhập vào nguồn tài nguyên. Mỗi người dùng có thẩm quyền có một định danh và có thể có một mật khẩu. Định danh của người dùng xác định người dùng như là một phần của hệ thống. Để truy nhập vào hệ thống, người dùng đăng nhập vào hệ thống với một mã người dùng (*user id*) hoặc tên đăng nhập (*login name*). Hệ thống cũng phải có mật khẩu kiểm tra nhằm ngăn chặn người dùng trái phép truy nhập vào tài nguyên.

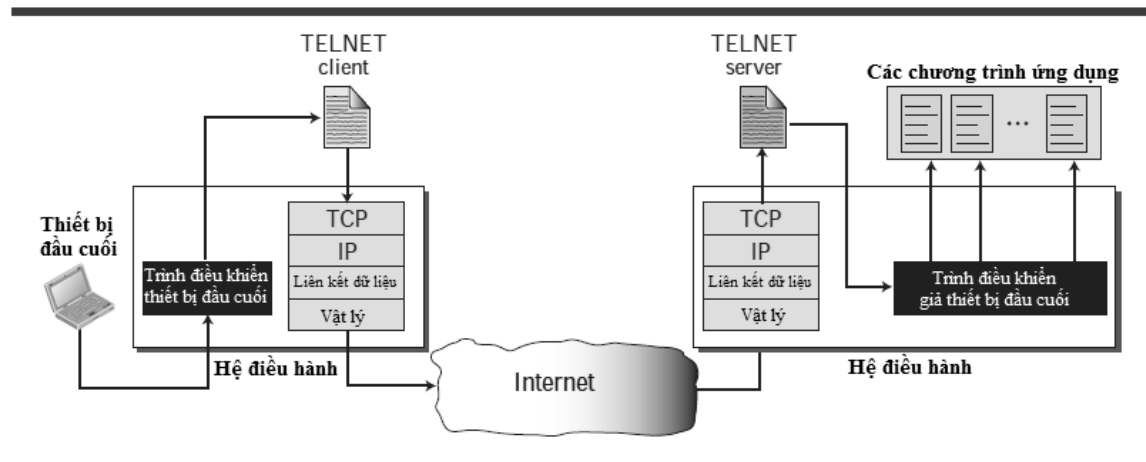
c. Đăng nhập cục bộ (Local login)

Việc người dùng đăng nhập vào hệ thống chia sẻ thời gian cục bộ được gọi là *đăng nhập cục bộ*. Khi người dùng nhập vào một thiết bị đầu cuối hoặc một máy trạm chạy mô phỏng thiết bị đầu cuối, các tổ hợp phím được chấp nhận bởi các trình điều khiển (*driver*) thiết bị đầu cuối. Trình điều khiển thiết bị đầu cuối chuyển các ký tự cho hệ điều hành. Hệ điều hành, lần lượt, dịch tổ hợp các ký tự và gọi chương trình ứng dụng hoặc tiện ích mong muốn.

Tuy nhiên, cơ chế không đơn giản như đã thấy vì hệ điều hành có thể gán các ngữ nghĩa đặc biệt cho các ký tự đặc biệt. Ví dụ, trong UNIX, một số các ký tự kết hợp với nhau có ngữ nghĩa đặc biệt, như *control* kết hợp với *z* nghĩa là đình chỉ (*suspend*), *control* kết hợp với *c* là hủy bỏ (*abort*),... Trong khi các tình huống đặc biệt này không tạo ra bất kỳ vấn đề gì trong đăng nhập cục bộ vì mô phỏng thiết bị đầu cuối và trình điều khiển đầu cuối biết chính xác ngữ nghĩa của mỗi ký tự đặc biệt kết hợp với các ký tự khác, nhưng lại có thể phát sinh một số vấn đề khi thực hiện đăng nhập từ xa. Vậy tiến trình nào có thể dịch các ký tự đặc biệt: client hay là server? Vấn đề sẽ được giải thích rõ trong các phần sau.

d. Đăng nhập từ xa (Remote login)

Khi người dùng muốn truy nhập một chương trình ứng dụng hoặc tiện ích tại một máy tính ở xa, người ta thực hiện *đăng nhập từ xa*. Ở đây, các chương trình Telnet client và Telnet server được đưa vào sử dụng. Người dùng gửi các tổ hợp phím tới trình điều khiển thiết bị đầu cuối mà hệ điều hành cục bộ chấp nhận các ký tự nhưng không dịch chúng. Các ký tự được gửi cho Telnet client và sau đó được chuyển tới tập ký tự tổng quát, được gọi là các ký tự *thiết bị đầu cuối ảo mạng* (Network Virtual Terminal, NVT), và chuyển tới chồng giao thức TCP/IP (hình 2.22).



Hình 2.22 Đăng nhập từ xa

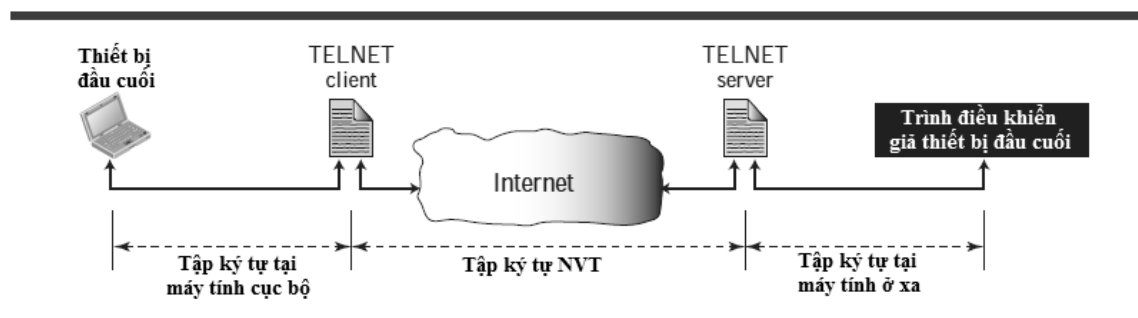
Các lệnh hoặc văn bản, theo dạng NVT, đi vào Internet và đến chồng giao thức TCP/IP tại máy tính ở xa. Khi đó, các ký tự được chuyển tới hệ điều hành và chuyển cho Telnet server để đổi thành các ký tự tương ứng mà máy tính ở xa có thể hiểu được. Tuy nhiên, các ký tự không được chuyển trực tiếp cho hệ điều hành vì hệ điều hành ở xa không được thiết kế để nhận các ký tự từ Telnet server, mà nó được thiết kế để nhận các ký tự từ trình điều khiển thiết bị đầu cuối. Giải pháp là cần phải thêm vào một phần phần mềm, trình điều khiển giả thiết bị

đầu cuối (*pseudoterminal driver*), nó sẽ giả như là các ký tự được gửi đến từ thiết bị đầu cuối. Hệ điều hành sau đó sẽ chuyển các ký tự tới chương trình ứng dụng phù hợp.

e. Thiết bị đầu cuối ảo mạng (Network Virtual Terminal, NVT)

Kỹ thuật truy nhập vào máy tính ở xa là phức tạp, bởi vì mọi máy tính và hệ điều hành của nó chấp nhận kết nối các ký tự và các tổ hợp phím đặc biệt. Ví dụ, tổ hợp phím kết thúc tệp trong một máy tính đang chạy hệ điều hành DOS là *Ctrl + z*, trong khi với hệ điều hành UNIX lại là *Ctrl + d*.

Khi muốn truy nhập đến một máy tính bất kỳ ở xa nào trên thế giới, đầu tiên cần phải biết về loại máy tính sẽ được kết nối đến, sau đó cần phải cài đặt giả lập thiết bị đầu cuối cụ thể cho máy tính. Telnet xử lý vấn đề này bằng cách định nghĩa giao diện chung, được gọi là *tập ký tự thiết bị đầu cuối ảo mạng*, NVT. Qua giao diện này, Telnet client dịch các ký tự (dữ liệu hoặc lệnh) đến từ thiết bị đầu cuối cục bộ thành dạng NVT và gửi chúng đi vào mạng. Mặt khác, telnet server dịch dữ liệu và lệnh từ dạng NVT thành dạng chấp nhận được bởi máy tính ở xa. Hình 2.23 minh họa khái niệm này.



Hình 2.23 Khái niệm NVT

NVT sử dụng hai tập ký tự, một cho dữ liệu và một cho điều khiển. Cả hai đều bao gồm 1 byte (8 bits).

- **Các ký tự dữ liệu:** Với dữ liệu, NVT thường sử dụng NVT ASCII. Đây là một bộ ký tự 8 bit, trong đó 7 bit sau là cùng US ASCII và bit đầu là 0. Mặc dù nó có thể gửi 8 bit ASCII (với bit đầu được thiết lập là 0 hoặc 1), thì đầu tiên cũng phải được sự đồng ý của cả client và server sử dụng tùy chọn đàm phán.
- **Các ký tự điều khiển:** Để gửi các ký tự điều khiển giữa các máy tính (từ client đến server hoặc ngược lại), NVT sử dụng tập ký tự 8 bit trong đó bit đầu tiên được thiết lập là 1. Ví dụ, ký tự EOF (hệ cơ số 10: 236, hệ cơ số 2: 11101100, ý nghĩa: kết thúc tệp); ký tự IP (hệ cơ số 10: 244, hệ cơ số 2: 11110100, ý nghĩa: ngắt tiến trình), v.v.

f. Nhúng (Embedding)

Telnet chỉ sử dụng một kết nối TCP. Server sử dụng một cổng đã biết là 23 và client sử dụng một cổng không bền vững. Kết nối này được sử dụng cho việc gửi cả ký tự dữ liệu và ký tự điều khiển.

Telnet thực hiện việc này bằng cách nhúng các ký tự điều khiển trong luồng dữ liệu. Tuy nhiên, để phân biệt các ký tự dữ liệu với các ký tự điều khiển, mỗi chuỗi ký tự điều khiển được

bắt đầu bởi một ký tự điều khiển đặc biệt, được gọi là IAC (*interpret as control*). Ví dụ, giả sử người dùng muốn server hiển thị một tệp tin (*file1*) trên một máy tính từ xa, lệnh được gõ như sau: *cat file1*, trong đó *cat* là một lệnh UNIX được sử dụng để hiển thị nội dung của tệp tin lên màn hình. Tuy nhiên, tên của tệp bị gõ sai (*filea* thay vì *file1*). Người dùng sử dụng phím *backspace* để sửa lại cho đúng tình huống này: *cat filea<backspace>1*

Tuy nhiên, trong cài đặt mặc định của Telnet, người dùng không thể sửa cục bộ, việc sửa đổi phải được thực hiện tại server ở xa. Ký tự *backspace* sẽ được dịch thành hai ký tự ở xa (IAC EC), các ký tự này sẽ được nhúng vào trong dữ liệu và gửi tới server ở xa.

2.5.1.2 Tùy chọn và đàm phán tùy chọn

Telnet cho phép client và server đàm phán tùy chọn trước và trong khi sử dụng dịch vụ. Các tùy chọn là các đặc tính mở rộng có sẵn cho người dùng với một thiết bị đầu cuối phức tạp hơn. Người dùng với thiết bị đầu cuối đơn giản có thể sử dụng các đặc tính mặc định. Một số ký tự điều khiển đã được thảo luận ở phần trước được dùng để định nghĩa tùy chọn. Các tùy chọn chung bao gồm:

- *Binary*: cho phép phía nhận dịch mỗi 8 bit ký tự nhận được, ngoại trừ IAC, thành dữ liệu nhị phân. Khi nhận được IAC, ký tự tiếp theo hoặc các ký tự sẽ được dịch thành các lệnh. Tuy nhiên, nếu nhận được 2 ký tự IAC liên tiếp, ký tự đầu tiên sẽ bị loại bỏ, còn ký tự thứ hai sẽ được dịch như là dữ liệu.
- *Echo*: cho phép server dội lại dữ liệu nhận được từ client. Điều này có nghĩa là mọi ký tự được gửi từ client tới đầu gửi sẽ được server dội lại trên màn hình của thiết bị đầu cuối client. Trong trường hợp này, người dùng tại thiết bị đầu cuối thường không dội lại các ký tự khi chúng được gõ, nhưng sẽ đợi cho đến khi nhận lại được chúng từ server.
- *Suppress go-ahead*: ngăn chặn ký tự đi trước (GA).
- *Status*: cho phép người dùng hoặc tiến trình chạy trên máy client nhận được trạng thái tùy chọn được kích hoạt bởi phía server.
- *Timing mark*: cho phép một bên ấn định dấu thời gian xác định tất cả các dữ liệu nhận được từ trước đã được xử lý.
- *Terminal type*: cho phép client gửi thông tin về loại thiết bị đầu cuối của nó.
- *Terminal speed*: cho phép client gửi thông tin về tốc độ của thiết bị đầu cuối của nó.
- *Line mode*: cho phép client chuyển chế độ dòng.

a. Đàm phán tùy chọn

Để sử dụng bất kỳ tùy chọn nào đã được đề cập ở phần trước, đầu tiên phải yêu cầu đàm phán tùy chọn giữa client và server. Bốn ký tự điều khiển được sử dụng cho mục đích này được trình bày trong bảng 2.1.

Bảng 2.1 Tập các ký tự NVT cho đàm phán tùy chọn

Ký tự	Mã	Ý nghĩa 1	Ý nghĩa 2	Ý nghĩa 3
WILL	251	Cấp phép kích hoạt	Chấp nhận kích hoạt	
WONT	252	Từ chối kích hoạt	Cấp phép không kích hoạt	Chấp nhận không kích hoạt
DO	253	Phê duyệt kích hoạt	Yêu cầu kích hoạt	
DONT	254	Không phê duyệt kích hoạt	Phê duyệt không kích hoạt	Yêu cầu không kích hoạt

b. Kích hoạt một tùy chọn

Một số tùy chọn chỉ có thể được kích hoạt bởi server, một số chỉ được kích hoạt bởi client, và một số có thể được kích hoạt bởi cả hai. Một tùy chọn được kích hoạt hoặc thông qua một *cấp phép* (*offer*) hoặc một *yêu cầu* (*request*).

c. Cấp phép kích hoạt

Một bên có thể cho phép kích hoạt tùy chọn nếu nó có quyền thực hiện việc đó. Việc cấp phép có thể được chấp nhận hoặc không chấp nhận từ bên khác. Bên cho phép gửi lệnh *WILL*, nghĩa là "tôi sẽ kích hoạt tùy chọn?". Bên kia gửi hoặc lệnh *DO*, nghĩa là "hãy làm", hoặc lệnh *DONT*, nghĩa là "không làm".

d. Yêu cầu kích hoạt

Một bên có thể yêu cầu phía khác kích hoạt tùy chọn. Yêu cầu có thể được chấp nhận hoặc bị từ chối. Bên yêu cầu gửi lệnh *DO*, nghĩa là "đề nghị kích hoạt tùy chọn". Bên kia gửi hoặc là lệnh *WILL*, nghĩa là "Tôi sẽ làm", hoặc lệnh *WONT*, nghĩa là "Tôi sẽ không làm".

e. Vô hiệu hóa một tùy chọn

Một tùy chọn đã được kích hoạt có thể bị vô hiệu hóa bởi một trong các bên, thông qua một *cấp phép* (*offer*) hoặc một *yêu cầu* (*request*).

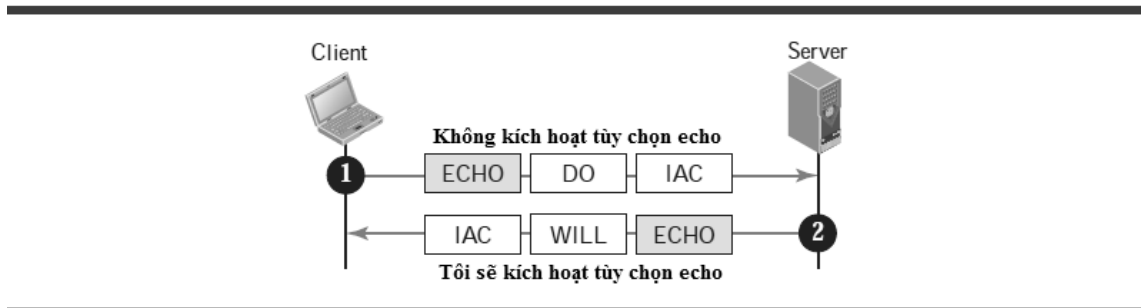
f. Cấp phép vô hiệu hóa

Một bên có thể cho phép vô hiệu hóa một tùy chọn. Bên khác sẽ buộc phải phê duyệt việc cấp phép. Bên cấp phép sẽ gửi lệnh *WONT*, nghĩa là "Sẽ không sử dụng tùy chọn nữa". Câu trả lời phải là lệnh *DONT*, nghĩa là "Không sử dụng nữa".

g. Yêu cầu vô hiệu hóa

Một bên có thể yêu cầu bên kia vô hiệu hóa một tùy chọn. Bên kia phải chấp nhận yêu cầu mà không thể từ chối. Bên yêu cầu gửi lệnh *DONT*, nghĩa là "Không sử dụng tùy chọn nữa". Câu trả lời là lệnh *WONT*, nghĩa là "Sẽ không sử dụng nữa".

Ví dụ: Hình 2.24 biểu diễn một ví dụ đàm phán tùy chọn. Trong ví dụ này, client muốn server dội lại mỗi ký tự được gửi đến server. Mặt khác, khi một ký tự được gõ tại bàn phím thiết bị đầu cuối của người dùng, nó sẽ đi đến server và được gửi lại màn hình của người dùng trước khi được xử lý. Tùy chọn dội lại (*echo*) được kích hoạt bởi server vì nó là server gửi các ký tự trở lại thiết bị đầu cuối của người dùng. Vì thế client nên *yêu cầu* server kích hoạt tùy chọn sử dụng *DO*. Yêu cầu bao gồm 3 ký tự: *IAC*, *DO*, và *ECHO*. Server chấp nhận yêu cầu và kích hoạt tùy chọn. Nó báo cho client bằng cách gửi 3 ký tự: *IAC*, *WILL* và *ECHO*.



Hình 2.24 Tùy chọn Echo

h. Tính đối xứng

Một đặc điểm thú vị của Telnet là đàm phán tùy chọn đối xứng, trong đó client và server có vai trò bình đẳng nhau. Điều này nghĩa là, khi bắt đầu kết nối, giả sử cả hai bên cùng sử dụng cài đặt mặc định Telnet và không có kích hoạt tùy chọn. Nếu một bên muốn kích hoạt tùy chọn, nó có thể cấp phép hoặc yêu cầu. Bên kia có quyền phê duyệt cấp phép hoặc từ chối yêu cầu nếu bên này không có khả năng sử dụng tùy chọn hoặc không muốn sử dụng tùy chọn. Điều này cho phép mở rộng Telnet. Một client hoặc server có thể cài một phiên bản phức tạp hơn của Telnet với nhiều tùy chọn hơn. Khi được kết nối với một bên, nó có thể cấp phép hoặc yêu cầu các tùy chọn mới này. Nếu bên kia cũng hỗ trợ các tùy chọn mới này, các tùy chọn có thể được kích hoạt; ngược lại, chúng sẽ bị từ chối.

i. Đàm phán tùy chọn con

Một số tùy chọn có thể yêu cầu thêm vào một số thông tin. Ví dụ, để xác định loại hoặc tốc độ của thiết bị đầu cuối, thì đàm phán cần có một xâu hoặc một số xác định loại hoặc tốc độ. Trong cả hai trường hợp, hai ký tự tùy chọn con cần phải được chỉ ra. Tập ký tự tùy chọn con bao gồm: SE (kết thúc tùy chọn con), SB (bắt đầu tùy chọn con).

2.5.1.3 Việc điều khiển server

Một số các ký tự điều khiển có thể được sử dụng để điều khiển server ở xa. Khi một chương trình ứng dụng đang chạy trên máy tính cục bộ, các ký tự đặc biệt được dùng để ngắt (hủy bỏ) chương trình (ví dụ, *Ctrl+c*), hoặc xóa ký tự cuối cùng đã được gõ (ví dụ, phím *delete* hoặc phím *backspace*), v.v. Tuy nhiên, khi một chương trình ứng dụng đang chạy trên server ở xa, các ký tự điều khiển này sẽ được gửi đến máy ở xa. Người dùng vẫn gõ chuỗi ký tự đó, nhưng chúng sẽ được chuyển thành các ký tự đặc biệt và gửi tới server. Một số ký tự được gửi tới server để điều khiển chương trình ứng dụng đang chạy như sau:

- **IP (Interrupt process):** Khi chương trình đang chạy cục bộ, người dùng có thể ngắt (hủy bỏ) chương trình khi, ví dụ, chương trình đi vào vòng lặp không giới hạn. Người dùng có thể gõ tổ hợp phím *Ctrl+c*, hệ điều hành gọi một hàm, và hàm sẽ thực hiện hủy bỏ chương trình. Tuy nhiên, nếu chương trình đang chạy trên máy tính ở xa, hàm phù hợp sẽ được gọi bởi hệ điều hành của máy ở xa. Telnet định nghĩa ký tự IP để đọc hoặc dịch lệnh phù hợp để gọi hàm ngắt tại máy ở xa.
- **AO (Abort output):** Hàm này cũng tương tự như IP, nhưng nó cho phép tiến trình tiếp tục mà không cần tạo đầu ra. Điều này rất hữu ích nếu tiến trình có thêm tác dụng khác

(mà người dùng muốn) ngoài việc tạo đầu ra (có thể người dùng không cần). Ví dụ, hầu hết các lệnh trong UNIX đều tạo đầu ra và có một trạng thái tồn tại. Người dùng có thể sử dụng muốn dùng trạng thái này trong tương lai, nhưng không quan tâm đến đầu ra dữ liệu.

- *AYT (are you there?)*: Ký tự điều khiển này được dùng để xác định xem máy ở xa đã được khởi động và chạy hay chưa, đặc biệt là sau một thời gian dài phía server im lặng. Khi nhận được ký tự này server thường gửi một tín hiệu âm thanh hoặc hình ảnh để xác nhận là nó đang chạy.
- *EC (erase character)*: Khi người dùng gửi dữ liệu từ bàn phím tới máy địa phương, ký tự *delete* hoặc *backspace* có thể xóa ký tự được gõ cuối cùng. Để làm việc này với máy ở xa, Telnet định nghĩa ký tự điều khiển EC.
- *EL (erase line)*: Ký tự này được dùng để xóa dòng hiện tại trong máy ở xa.

a. Báo hiệu ngoài (out-of-band signaling)

Để tạo các ký tự điều khiển hiệu quả trong các tình huống đặc biệt, Telnet sử dụng *báo hiệu ngoài*. Trong báo hiệu ngoài, các ký tự điều khiển được đặt trước bởi IAC và gửi tới tiến trình ở xa.

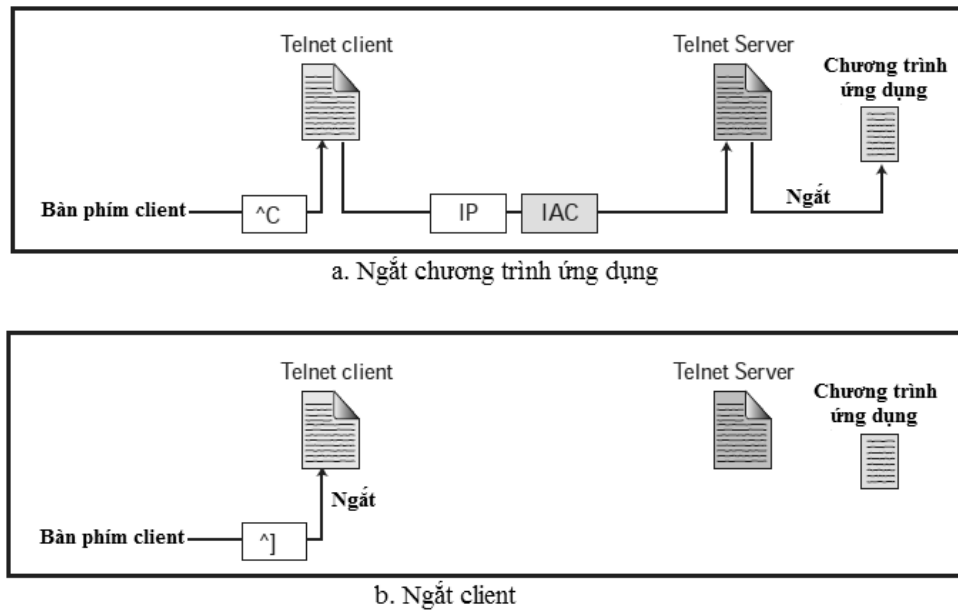
Tưởng tượng một tình huống trong đó chương trình ứng dụng đang chạy tại phía server đi vào vòng lặp không giới hạn và không chấp nhận thêm bất kỳ dữ liệu vào nào. Người dùng muốn ngắt chương trình ứng dụng, nhưng chương trình không đọc dữ liệu từ vùng đệm nữa. TCP tại phía server phải thấy là vùng đệm bị đầy và cần phải gửi một gói tin (TCP segment) của nó để báo là kích thước cửa sổ của client hiện đang là 0. Mặt khác, TCP tại phía server cần thông báo là không có lưu lượng thông thường nào được chấp nhận thêm nữa. Để khắc phục tình trạng này, một gói tin TCP khẩn cần được gửi từ client đến server. Gói tin khẩn sẽ bỏ qua kỹ thuật điều khiển luồng thông thường. Mặc dù TCP không chấp nhận gói tin bình thường, nhưng nó sẽ chấp nhận gói tin khẩn.

Khi một tiến trình Telnet (client hoặc server) muốn gửi một chuỗi ký tự ngoài tới một tiến trình khác (client hoặc server), nó nhúng chuỗi trong luồng dữ liệu và chèn một ký tự đặc biệt được gọi là DM (*data mark*). Tuy nhiên, để báo cho phía kia, nó sẽ tạo ra một gói tin TCP với tập bit khẩn và con trỏ khẩn trỏ tới ký tự DM. Khi tiến trình nhận nhận được dữ liệu, nó đọc dữ liệu và bỏ qua bất kỳ dữ liệu nào phía trước của các ký tự điều khiển (ví dụ, IAC và IP). Khi nó đọc đến ký tự DM, phần dữ liệu còn lại sẽ được xử lý như bình thường. Mặt khác, ký tự DM được dùng như là ký tự đồng bộ để chuyển tiến trình nhận từ chế độ khẩn sang chế độ bình thường và đồng bộ lại hai đầu cuối. Theo cách này, ký tự điều khiển (IP) được phân phối bên ngoài hệ điều hành, trong đó sử dụng các hàm thích hợp để ngắt chương trình ứng dụng đang chạy.

b. Ký tự thoát

Một ký tự được gõ bởi người dùng thường được gửi tới server. Tuy nhiên, thi thoảng người dùng muốn các ký tự được dịch bởi client thay vì server. Trong trường hợp này, người dùng có thể sử dụng ký tự thoát, thường là *Ctrl+J* (được biểu diễn là *^J*). Hình 2.25 so sánh việc ngắt một chương trình ứng dụng tại phía ở xa với việc ngắt tiến trình client tại phía địa

phương sử dụng ký tự thoát. Phím nhắc Telnet (*Telnet prompt*) được hiển thị sau ký tự thoát này.



Hình 2.25 So sánh hai tiến trình ngắt khác nhau

2.5.1.4 Các chế độ hoạt động

Hầu hết các cài đặt Telnet hoạt động ở một trong ba chế độ sau: chế độ mặc định, chế độ ký tự và chế độ dòng.

- **Chế độ mặc định:** Chế độ mặc định được sử dụng khi không có chế độ khác được gọi qua đàm phán tùy chọn. Trong chế độ này, việc dội lại được thực hiện bởi client. Người dùng gõ một ký tự và client dội ký tự đó lại màn hình (hoặc máy in) nhưng không gửi nó đi cho đến khi toàn bộ dòng được hoàn thành. Sau khi gửi toàn bộ dòng đến server, client đợi lệnh GA (*go ahead*) từ server trước khi chấp nhận một dòng mới từ người dùng. Hoạt động này là đơn công (*half-duplex*). Hoạt động đơn công không phù hợp khi kết nối TCP là song công (*full-duplex*), và do vậy chế độ này trở nên lỗi thời.
- **Chế độ ký tự:** Trong chế độ ký tự, mỗi ký tự được gõ sẽ được gửi từ client đến server. Server thường dội lại các ký tự để hiển thị trên màn hình của client. Trong chế độ này, việc dội lại các ký tự có thể bị trễ nếu thời gian truyền lâu (như trong kết nối vệ tinh). Nó cũng tạo ra quá tải lưu lượng cho mạng vì cần đến 3 gói tin TCP được gửi đi cho mỗi ký tự dữ liệu: người dùng nhập một ký tự và gửi đến server; server báo nhận được ký tự và dội ký tự lại (trong một gói tin); và client báo nhận đã nhận được ký tự dội lại.
- **Chế độ dòng:** Một chế độ mới được đề xuất để khắc phục những thiếu sót của chế độ mặc định và chế độ ký tự. Trong chế độ này, được gọi là chế độ dòng, việc sửa đổi dòng (dội lại, xóa ký tự, xóa dòng, v.v) được thực hiện bởi client. Sau đó client sẽ gửi toàn bộ dòng đến server. Chế độ mặc định hoạt động trong chế độ đơn công; chế độ

dòng là song công, với việc client gửi một dòng sau dòng khác, mà không cần một ký tự can thiệp GA (go ahead) từ server.

2.5.1.5 Vấn đề bảo mật

Telnet bị một số vấn đề về bảo mật. Mặc dù Telnet có yêu cầu tên đăng nhập và mật khẩu (khi thay đổi văn bản), tuy nhiên điều này thường là chưa đủ. Một máy tính nhỏ (microcomputer) kết nối tới một mạng LAN quảng bá có thể dễ dàng nghe lén bằng cách sử dụng phần mềm snoop (ăn trộm) và lấy được tên đăng nhập và mật khẩu tương ứng (thậm chí đã bị mã hóa).

2.5.2 SSH (Secure Shell)

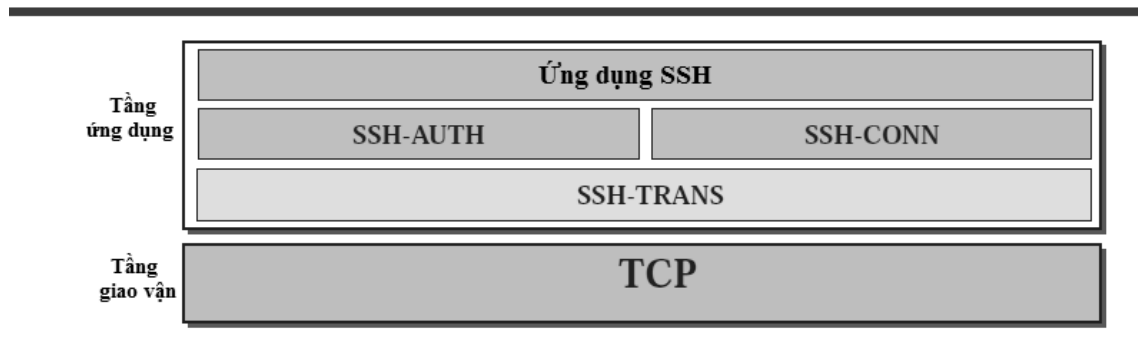
Một chương trình ứng dụng đăng nhập từ xa thông dụng khác là Secure Shell (SSH). SSH, giống như Telnet, sử dụng TCP như là giao thức vận chuyển tầng dưới, nhưng SSH bảo mật tốt hơn và cung cấp nhiều dịch vụ hơn Telnet.

2.5.2.1 Các phiên bản

Có hai phiên bản SSH là SSH-1 và SSH-2, và chúng không hoàn toàn tương thích. Phiên bản thứ nhất, SSH-1 hiện nay đang bị phản đối vì lỗ hổng bảo mật trong nó. Phần sau chỉ trình bày về SSH-2.

2.5.2.2 Các thành phần

SSH là một giao thức tầng ứng dụng đã được đề xuất với 4 thành phần, như trình bày trong hình 2.26.



Hình 2.26 Các thành phần của SSH

a. Giao thức tầng giao vận SSH (SSH-TRANS)

Vì TCP không phải là một giao thức tầng giao vận bảo mật, nên đầu tiên SSH phải sử dụng một giao thức mà có thể tạo được kênh bảo mật phía trên của TCP. Tầng mới này là một giao thức độc lập, được quy vào SSH-TRANS. Khi phần mềm cài đặt giao thức này được gọi, đầu tiên client và server sử dụng giao thức TCP để thiết lập một kết nối trước không an toàn. Sau đó, họ trao đổi một vài tham số bảo mật để thiết lập kênh bảo mật phía trên TCP. Phần sau sẽ tóm tắt một số các dịch vụ giao thức này cung cấp:

1. Sự riêng tư hoặc bí mật của thông điệp trao đổi.

2. Sự toàn vẹn dữ liệu, nghĩa là nó phải đảm bảo là các thông điệp trao đổi giữa client và server không bị thay đổi bởi kẻ xâm nhập.
3. Chứng thực server, nghĩa là bây giờ client sẽ đảm bảo server là một trong những phần nó yêu cầu (*claim*).
4. Việc nén các thông điệp sẽ giúp nâng cao hiệu quả của hệ thống và làm cho việc tấn công khó khăn hơn.

b. Giao thức chứng thực SSH (SSH-AUTH)

Sau khi kênh bảo mật được thiết lập giữa client và server, và server được xác thực cho client, SSH có thể gọi một phần mềm khác để xác thực client cho server.

c. Giao thức kết nối SSH (SSH-CONN)

Sau khi kênh bảo mật được thiết lập, và cả client và server được chứng thực lẫn nhau, SSH có thể gọi một phần của phần mềm cài đặt giao thức thứ ba, SSH-CONN. Một trong các dịch vụ được cung cấp bởi giao thức SSH-CONN là thực hiện ghép kênh. SSH-CONN lấy kênh bảo mật được thiết lập bởi hai giao thức trước, và cho phép client tạo nhiều kênh logic trên nó.

d. Ứng dụng SSH

Sau khi pha kết nối hoàn thành, SSH cho phép một vài chương trình ứng dụng sử dụng kết nối này. Mỗi ứng dụng có thể tạo một kênh logic như đã mô tả ở phần trên, và sau đó có quyền lợi từ kết nối bảo mật. Mặt khác, đăng nhập từ xa là một dịch vụ có thể sử dụng các giao thức SSH-CONN; các ứng dụng khác, như là ứng dụng truyền tệp có thể sử dụng các kênh logic cho mục đích này.

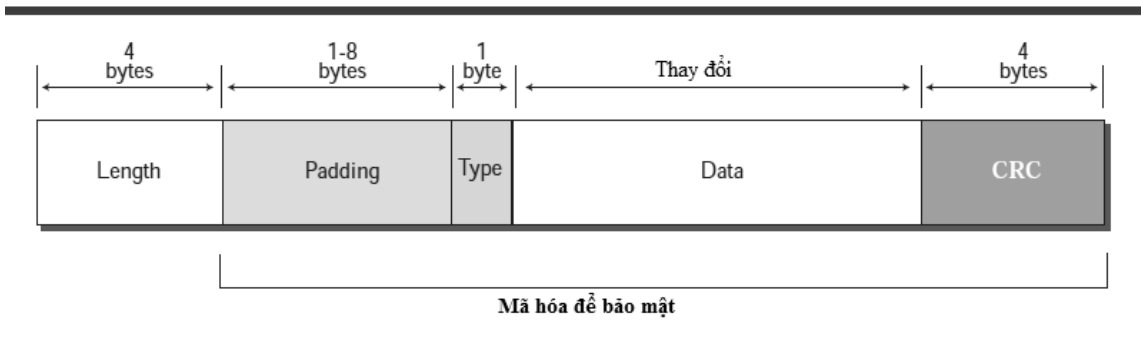
2.5.2.3 Chuyển tiếp cổng

SSH cung cấp một dịch vụ được quan tâm là *chuyển tiếp cổng*. Ở đây, có thể sử dụng các kênh được bảo mật có sẵn trong SSH để truy nhập tới chương trình ứng dụng mà không cần cung cấp các dịch vụ bảo mật. Chương trình ứng dụng như Telnet hoặc SMTP có thể sử dụng các dịch vụ của SSH dùng kỹ thuật chuyển tiếp cổng. Kỹ thuật chuyển tiếp cổng SSH tạo ra một đường hầm thông qua đó các thông điệp thuộc giao thức khác có thể di chuyển. Vì lý do này, kỹ thuật này đôi khi coi như là *SSH đường hầm (tunneling)*.

Người ta có thể thay đổi kết nối trực tiếp, nhưng không an toàn, giữa Telnet client và Telnet server bằng chuyển tiếp cổng. Telnet client có thể sử dụng SSH client tại trạm cục bộ để tạo ra một kết nối bảo mật với SSH server tại phía ở xa. Bất kỳ yêu cầu nào từ Telnet client tới Telnet server đều được chuyển qua đường hầm được cung cấp bởi SSH client và SSH server. Bất kỳ đáp ứng nào từ Telnet server tới Telnet client cũng đều được chuyển qua đường hầm được cung cấp bởi SSH client và SSH server.

2.5.2.4 Khuôn dạng của các gói tin SSH

Hình 2.27 biểu diễn khuôn dạng của các gói tin được sử dụng với các giao thức SSH.



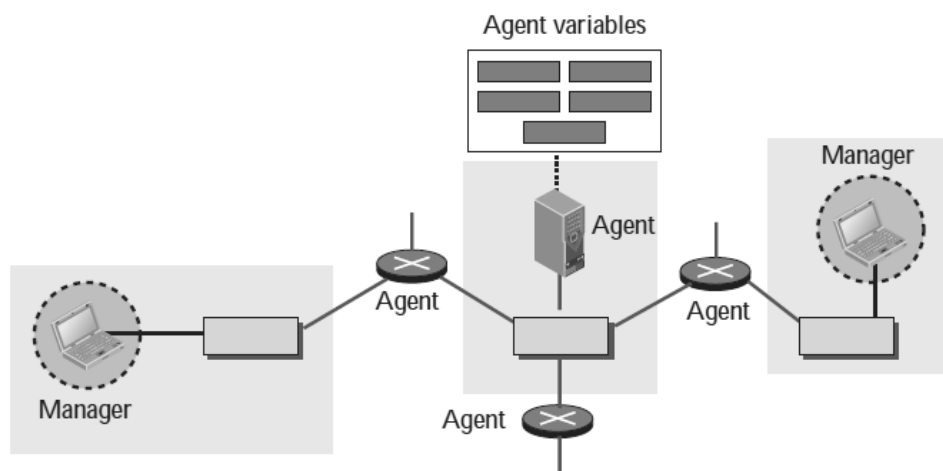
Hình 2.27 Định dạng gói tin SSH

Mô tả tóm tắt các trường dữ liệu như sau:

- **Length (độ dài):** Gồm 4 byte xác định độ dài của gói tin bao gồm các trường: loại (*type*), dữ liệu (*data*) và trường CRC; không bao gồm trường Padding và trường độ dài (*length*).
- **Padding:** Có thể bao gồm từ 1 đến 8 byte, được thêm vào gói tin để làm cho việc tấn công vào tính năng cung cấp bảo mật khó khăn hơn.
- **Type (loại):** Gồm 1 byte, xác định loại gói tin được sử dụng bởi các giao thức SSH.
- **Data (dữ liệu):** Trường này có độ dài thay đổi.
- **CRC (cycle redundancy check):** được sử dụng để kiểm tra lỗi.

2.6 GIAO THỨC QUẢN LÝ MẠNG ĐƠN GIẢN SNMP

Giao thức quản lý mạng đơn giản SNMP (*Simple Network Management Protocol*) là một khung cho việc quản lý các thiết bị trong mạng Internet dùng chồng giao thức TCP/IP. Nó cung cấp một tập các thao tác nền tảng cho việc giám sát và bảo trì mạng Internet.



Hình 2.28 Khái niệm SNMP

2.6.1 Khái niệm

SNMP sử dụng các khái niệm trạm quản lý (*manager*) và đại lý (*agent*). Manager thường là một host, điều khiển và giám sát một tập các agents, thường là các bộ định tuyến (*router*) hoặc các máy chủ (*servers*) (hình 2.28).

SNMP là một giao thức mức ứng dụng trong đó, một số trạm quản lý sẽ điều khiển một tập các agents. Giao thức được thiết kế tại mức ứng dụng để có thể giám sát các thiết bị được cung cấp bởi nhiều nhà sản xuất khác nhau và được cài đặt trên các mạng vật lý khác nhau. Nói cách khác, SNMP giải phóng các nhiệm vụ quản lý từ cả đặc tính vật lý và kỹ thuật mạng bên dưới. Nó có thể được sử dụng trong một mạng hỗn hợp được tạo bởi các mạng LAN và WAN kết nối với nhau bằng các bộ định tuyến được cung cấp bởi nhiều nhà sản xuất khác nhau.

Trạm quản lý (*manager*) là một host chạy chương trình SNMP client. Trạm bị quản lý (*agent*) là một bộ định tuyến (hoặc một host), chạy chương trình SNMP server. Việc quản lý được thực hiện thông qua tương tác đơn giản giữa manager và agent.

Các agent lưu trữ thông tin về hiệu năng trong cơ sở dữ liệu. Server phải truy nhập vào các giá trị trong cơ sở dữ liệu. Ví dụ, một bộ định tuyến có thể lưu trữ số lượng gói tin nhận được và chuyển đi trong các biến thích hợp. Manager có thể lấy và so sánh các biến này để xem bộ định tuyến có bị tắc nghẽn hay không.

Manager cũng có thể làm cho bộ định tuyến thực hiện một số hành động nhất định. Ví dụ, một bộ định tuyến kiểm tra định kỳ giá trị của một bộ đếm khởi động lại để xem khi nào nó cần tự khởi động lại. Ví dụ, nó sẽ tự khởi động lại khi giá trị của bộ đếm bằng 0. Manager có thể sử dụng đặc điểm này để khởi động lại agent ở xa bất kỳ thời điểm nào. Nó chỉ đơn giản gửi một gói tin để buộc bộ đếm có giá trị là 0.

Agent cũng có thể đóng góp vào tiến trình quản lý. Các chương trình server chạy trên agent có thể kiểm tra môi trường và, nếu nó nhận thấy điều gì đó không bình thường, nó có thể gửi thông điệp cảnh báo (được gọi là *bẫy (trap)*) tới server.

Nói cách khác, việc quản lý với SNMP dựa trên ba ý tưởng cơ bản sau:

- Manager kiểm tra một agent bằng cách yêu cầu gửi thông tin về hành vi của agent.
- Manager buộc agent thực hiện một nhiệm vụ bằng cách đặt lại các giá trị trong cơ sở dữ liệu của agent.
- Agent đóng góp vào quá trình quản lý bằng cách cảnh báo với manager về tình huống bất thường.

2.6.2 Các thành phần quản lý

Để thực hiện nhiệm vụ quản lý, SNMP sử dụng hai thành phần khác: *SMI (Structure of Management Information)* và *MIB (Management Information Base)*. Nói cách khác, việc quản lý trên Internet được thực hiện thông qua sự phối hợp của ba thành phần: SNMP, SMI và MIB.

SNMP có một số vai trò cụ thể trong quản lý mạng. Nó xác định định dạng của gói tin được gửi từ manager tới agent và ngược lại, đồng thời cũng dịch kết quả và thực hiện các thông kê (thường với sự trợ giúp của các phần mềm quản lý khác). Các gói tin trao đổi chứa tên đối tượng (biến) và trạng thái của nó (các giá trị). SNMP chịu trách nhiệm đọc và thay đổi các giá trị này.

Để sử dụng SNMP, cần phải có các luật để đặt tên cho các đối tượng hoặc xác định loại đối tượng. Cần phải có một luật chung toàn cầu vì chúng ta không biết được kiến trúc của máy tính gửi, nhận và lưu trữ các giá trị này. Máy gửi có thể là một máy tính mạnh trong đó số nguyên được lưu trữ dạng 8 byte dữ liệu; máy nhận có thể là một máy tính nhỏ mà chỉ lưu trữ số nguyên dạng 4 byte dữ liệu. SMI là một giao thức xác định các luật này. Tuy nhiên, chúng ta phải hiểu là SMI chỉ xác định các luật; nó không xác định bao nhiêu đối tượng được quản lý trong một thực thể hoặc đối tượng nào sử dụng loại nào. SMI là một tập các luật chung để đặt tên đối tượng và liệt kê loại của chúng. Việc kết hợp một đối tượng với một loại không được thực hiện bởi SMI.

Với mỗi một đối tượng bị quản lý, cần một giao thức để xác định số lượng các đối tượng, tên của chúng theo các luật được định nghĩa bởi SMI, và kết hợp một loại với mỗi đối tượng đã có tên. Giao thức này là MIB. MIB tạo một tập các đối tượng xác định cho mỗi thực thể tương tự như một cơ sở dữ liệu (hầu hết là siêu dữ liệu trong cơ sở dữ liệu, tên và các loại không kèm giá trị). *MIB tạo ra một tập các đối tượng có tên, loại của nó, và mối quan hệ của chúng với nhau trong một thực thể được quản lý.*

2.6.2.1 SMI (The Structure of Management Information)

SMI phiên bản 2 (SMIv2) là một thành phần của quản lý mạng. Các chức năng:

1. Đặt tên các đối tượng.
2. Định nghĩa loại dữ liệu có thể được lưu trữ trong một đối tượng.
3. Hiện thị cách mã hóa dữ liệu để truyền qua mạng.

SMI là một chỉ dẫn cho SNMP. Nó nhấn mạnh ba thuộc tính để xử lý một đối tượng: tên, loại dữ liệu, và phương pháp mã hóa.

- **Tên:** SMI yêu cầu mỗi đối tượng được quản lý (như một bộ định tuyến, một biến trong một bộ định tuyến, một giá trị, v.v) có một tên duy nhất. Để đặt tên các đối tượng toàn cục, SMI sử dụng một *định danh đối tượng*, là một định danh phân cấp dựa trên cấu trúc cây. Cấu trúc cây bắt đầu với một gốc không có tên. Mỗi đối tượng có thể được định nghĩa bằng cách sử dụng một chuỗi số nguyên được phân cách bởi dấu chấm (.). Cấu trúc cây cũng có thể định nghĩa một đối tượng bằng cách sử dụng một chuỗi tên dạng văn bản cách nhau bởi dấu chấm. Biểu diễn theo dạng số nguyên-dấu chấm được sử dụng trong SNMP. Ký pháp tên-dấu chấm được sử dụng bởi người dùng thông thường. Ví dụ, biểu diễn sau là về cùng một đối tượng với hai ký pháp khác nhau:

iso.org.dod.Internet.mgmt.mib-2 ↔ *1.3.6.1.2.1*

- **Loại:** Thuộc tính thứ hai của một đối tượng là loại dữ liệu. Để xác định loại dữ liệu, SMI sử dụng các định nghĩa ký pháp cú pháp trừu tượng (*Abstract Syntax Notation 1*,

ASN.1) cơ bản và thêm vào một số định nghĩa mới. Nói cách khác, SMI gồm cả tập con (*subset*) và tập siêu (*superset*) của ASN.1. SMI có hai loại kiểu dữ liệu: *đơn giản* và *có cấu trúc*.

- ✓ *Loại đơn giản*: Loại dữ liệu đơn giản là các loại dữ liệu nguyên tử. Một số trong chúng được lấy trực tiếp từ ASN.1 và một số được thêm vào bởi SMI.
- ✓ *Loại có cấu trúc*: Bằng cách kết hợp các loại dữ liệu đơn giản và có cấu trúc có thể tạo ra các loại dữ liệu có cấu trúc mới. SMI định nghĩa hai loại dữ liệu có cấu trúc: *sequence* và *sequence of*. Sequence là kết hợp của các loại dữ liệu đơn giản, không nhất thiết cùng loại (tương tự với khái niệm *cấu trúc* hoặc *bản ghi* được sử dụng trong các ngôn ngữ lập trình). Sequence of là kết hợp của các loại dữ liệu đơn giản cùng loại hoặc là một kết hợp các loại dữ liệu sequence cùng loại (tương tự với khái niệm *mảng* trong các ngôn ngữ lập trình).
- **Phương pháp mã hóa**: SMI sử dụng một chuẩn khác, *Basic Encoding Rules (BER)*, để mã hóa dữ liệu truyền đi trên mạng. BER chỉ ra rằng mỗi phần của dữ liệu được mã hóa theo định dạng 3 phần tử: tag (thẻ), length (độ dài) và value (giá trị).
 - ✓ *Tag*: gồm 1 byte, xác định loại dữ liệu.
 - ✓ *Length*: gồm một hoặc nhiều byte. Nếu là 1 byte, bit quan trọng nhất phải là 0, 7 bit còn lại xác định độ dài của dữ liệu. Nếu có nhiều hơn 1 byte, bit quan trọng nhất của byte đầu tiên phải là 1, 7 bit còn lại của byte đầu tiên xác định số lượng byte cần để định nghĩa chiều dài dữ liệu.
 - ✓ *Value*: mã hóa giá trị của dữ liệu theo các luật được xác định trong BER.

2.6.2.2 MIB (The Management Information Base)

MIB phiên bản 2 (MIB2) là thành phần thứ hai được dùng trong quản lý mạng. Mỗi agent có một MIB2 của nó, là một tập tất cả các đối tượng mà manager có thể quản lý. Các đối tượng trong MIB2 được phân thành 10 loại khác nhau: *system* (hệ thống), *interface* (giao diện), *address translation* (dịch địa chỉ), *ip*, *icmp*, *tcp*, *udp*, *egp*, *transmission* (truyền), và *snmp*. Các nhóm này thuộc đối tượng mib-2 trong cây định danh đối tượng. Mỗi nhóm xác định các biến và/hoặc các bảng.

- *sys*. Đối tượng này (system) xác định thông tin chung về nút (hệ thống), như là tên, vị trí và thời gian tồn tại (lifetime).
- *if*. Đối tượng này (interface) xác định thông tin về tất cả các loại giao diện của nút bao gồm mã số giao diện, địa chỉ vật lý và địa chỉ IP.
- *at*. Đối tượng này (address translation) xác định thông tin về bảng ARP.
- *ip*. Đối tượng này xác định thông tin liên quan đến địa chỉ IP, như là bảng định tuyến và địa chỉ IP.
- *icmp*. Đối tượng này xác định thông tin liên quan đến ICMP, như số lượng gói tin đã gửi và nhận và tổng số lỗi.

- *tcp*. Đối tượng này xác định thông tin chung liên quan đến TCP, như là bảng kết nối, giá trị time-out, số lượng cổng, và số lượng gói tin đã gửi và nhận.
- *udp*. Đối tượng này xác định thông tin chung liên quan đến UDP, như là lượng cổng và số lượng gói tin đã gửi và nhận.
- *snmp*. Đối tượng này xác định thông tin chung liên quan đến SNMP.

2.6.2.3 SNMP

SNMP dùng cả SMI và MIB trong quản lý mạng Internet. Nó là một chương trình ứng dụng cho phép:

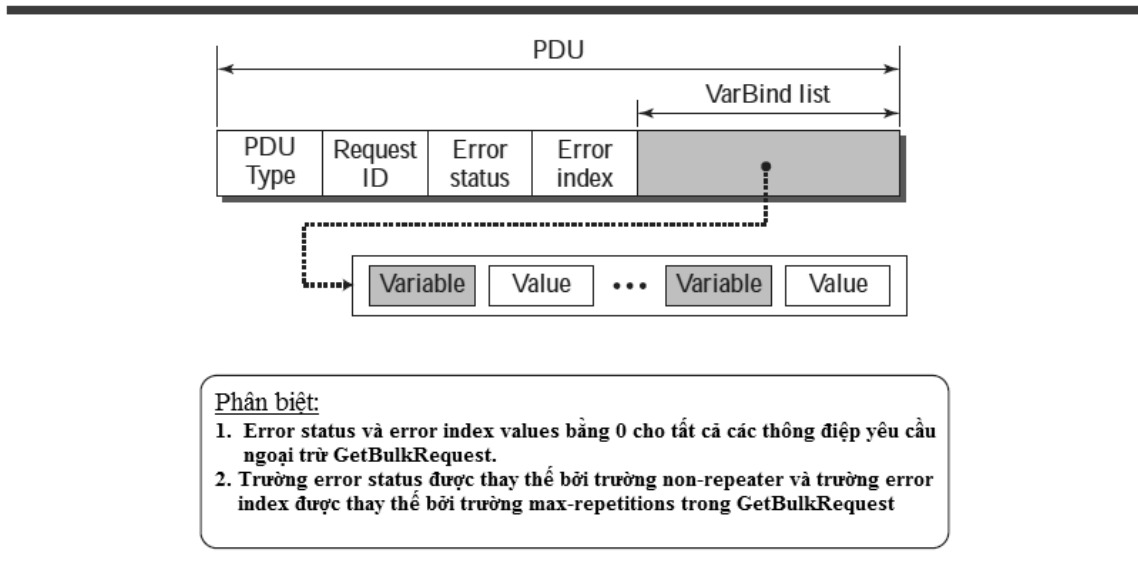
1. Manager lấy giá trị của một đối tượng đã được định nghĩa trong agent.
2. Manager lưu trữ giá trị trong một đối tượng được định nghĩa trong agent.
3. Agent gửi một thông điệp cảnh báo về tình huống bất thường cho manager.

SNMPv3 định nghĩa tám loại đơn vị dữ liệu giao thức (PDUs): *GetRequest*, *GetNextRequest*, *GetBulkRequest*, *SetRequest*, *Response*, *Trap*, *InformRequest*, và *Report*.

- *GetRequest*: *GetRequest* PDU được gửi từ manager (client) đến agent (server) để lấy giá trị của một biến hoặc một tập biến.
- *GetNextRequest*: *GetNextRequest* PDU được gửi từ manager tới agent để lấy giá trị của biến. Giá trị thu được là giá trị của đối tượng theo *ObjectId* đã được định nghĩa trong PDU. Nó chủ yếu được sử dụng để lấy giá trị của các mục trong một bảng. Nếu manager không biết chỉ mục của các mục, nó không thể lấy được giá trị. Tuy nhiên, nó có thể sử dụng *GetNextRequest* và định nghĩa *ObjectId* của bảng. Vì mục đầu tiên có *ObjectId* ngay sau *ObjectId* của bảng, giá trị của mục đầu tiên này sẽ được trả về. Manager có thể sử dụng *ObjectId* này để lấy giá trị của các mục tiếp theo, v.v.
- *GetBulkRequest*: *GetBulkRequest* PDU được gửi từ manager tới agent để lấy số lượng lớn dữ liệu. Nó có thể được sử dụng thay vì dùng nhiều *GetRequest* và *GetNextRequest* PDUs.
- *SetRequest*: *SetRequest* PDU được gửi từ manager tới agent để thiết lập (lưu) một giá trị trong một biến.
- *Response*: *Response* PDU được gửi từ agent tới manager để đáp ứng với *GetRequest* or *GetNextRequest*. Nó chứa giá trị của biến được yêu cầu bởi manager.
- *Trap*: *Trap* (cũng được gọi là *SNMPv2 Trap* để phân biệt nó với *SNMPv1 Trap*) PDU được gửi từ agent tới manager để báo cáo một sự kiện. Ví dụ, nếu agent bị khởi động lại, nó phải báo cho manager và báo cáo thời gian khởi động lại.
- *InformRequest*: *InformRequest* PDU được gửi từ một manager tới một manager khác ở xa để lấy giá trị của một số biến từ các agent chịu sự quản lý bởi manager ở xa. Manager ở xa đáp ứng lại với một *Response* PDU.
- *Report*: *Report* PDU được thiết kế để báo cáo về một số loại lỗi giữa các manager. Nó vẫn chưa được sử dụng.

2.6.3 Định dạng của SNMP PDUs

Định dạng của 8 SNMP PDUs được trình bày trong hình 2.29 GetBulkRequest PDU khác với các PDU khác ở 2 điểm như trong hình vẽ.



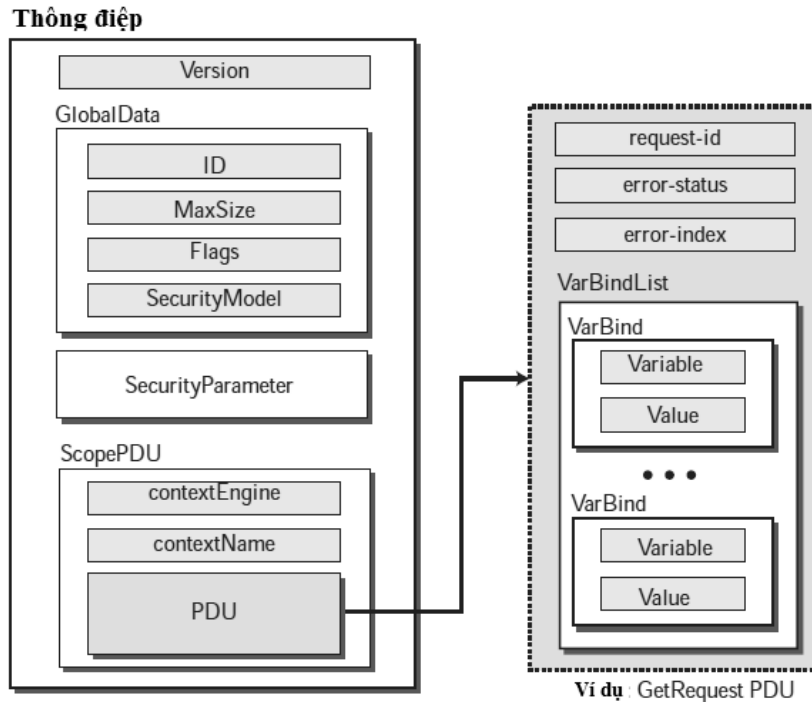
Hình 2.29 Định dạng SNMP PDU

Các trường được liệt kê như sau:

- *PDU type*: Trường này xác định loại PDU, gồm có 8 loại như trên.
- *Request ID*: Trường này là một chuỗi số được sử dụng bởi manager trong một PDU yêu cầu và lặp lại bởi agent trong đáp ứng. Nó được sử dụng để so khớp yêu cầu và đáp ứng.
- *Error status*: Đây là một số nguyên chỉ được sử dụng trong PDU đáp ứng để biểu diễn loại lỗi được báo cáo bởi agent. Giá trị của nó là 0 trong các PDU yêu cầu. Một số loại lỗi có thể xảy ra: noError (không có lỗi), tooBig (quá lớn cho một thông điệp), noSuchName (biến không tồn tại), badValue (giá trị lưu trữ không hợp lệ), readOnly (giá trị không được thay đổi), và genErr (các lỗi khác).
- *Non-repeaters*: Trường này chỉ sử dụng trong GetBulkRequest và thay thế các trường trạng thái lỗi (error status), là rỗng trong PDU yêu cầu.
- *Error index*: Chỉ mục lỗi là một giá trị tương đối cho manager biết các biến nào là nguyên nhân của lỗi.
- *Max-repetition*: Trường này cũng chỉ được sử dụng trong GetBulkRequest và thay thế các trường chỉ mục lỗi, là rỗng trong PDU yêu cầu.
- *VarBind list*: Đây là một tập các biến với các giá trị tương ứng mà manager muốn lấy hoặc thiết lập. Các giá trị là rỗng trong GetRequest và GetNextRequest. Trong một Trap PDU, nó biểu diễn các biến và giá trị liên quan đến một PDU cụ thể.

2.6.4 Thông điệp SNMP

SNMP không chỉ gửi một PDU, mà nó còn nhúng PDU trong một thông điệp. Một thông điệp trong SNMPv3 là một chuỗi bao gồm 4 phần tử: Version, GlobalData, SecurityParameters, và ScopePDU (bao gồm PDU đã mã hóa) như hình 2.30. Phần tử thứ nhất và thứ ba là loại dữ liệu đơn giản; phần tử thứ hai và thứ tư là sequence.



Hình 2.30 Thông điệp SNMP

- **Version:** Trường Version là loại dữ liệu số nguyên xác định trường version (phiên bản). Phiên bản hiện tại là 3.
- **GlobalData:** Trường GlobalData là một sequence với 4 phần tử của loại dữ liệu đơn giản: ID, Max-Size, Flags, và SecurityModel.
- **SecurityParameter:** Phần tử này là một sequence có thể rất phức tạp, phụ thuộc vào loại bảo mật được sử dụng trong phiên bản 3.
- **ScopePDU:** Phần tử cuối cùng gồm loại dữ liệu đơn giản và PDU thực tế. Chúng tôi chỉ trình bày một ví dụ về GetRequest PDU. Chú ý là VarBindList là một sequence được tạo thành từ một hoặc nhiều sequence, được gọi là VarBind. Mỗi VarBind tạo từ hai phần tử dữ liệu đơn giản là biến và giá trị.

2.6.5 Bảo mật

SNMPv3 có thêm hai đặc điểm mới so với các phiên bản trước là bảo mật và quản trị từ xa. SNMPv3 cho phép manager chọn một hoặc nhiều mức bảo mật khi truy nhập vào một agent. Các chức năng khác nhau của bảo mật có thể được cấu hình bởi manager cho phép xác thực thông điệp, tính bí mật và tính toàn vẹn.

SNMPv3 cũng cho phép cấu hình bảo mật từ xa mà không yêu cầu nhà quản trị phải có mặt thực tế tại nơi đặt thiết bị.

CHƯƠNG 3

CÁC GIAO THỨC TẦNG GIAO VẬN

Chương này trình bày về tầng giao vận với hai giao thức chính được sử dụng là TCP và UDP. Nội dung chương bao gồm ba phần chính như sau:

- Giới thiệu về các dịch vụ tầng giao vận cung cấp
- Giao thức UDP
- Giao thức TCP

3.1 GIỚI THIỆU VỀ TẦNG GIAO VẬN

Tầng giao vận nằm giữa các tầng mạng và tầng ứng dụng. Tầng giao vận có trách nhiệm cung cấp các dịch vụ cho tầng ứng dụng; và nó nhận được dịch vụ từ tầng mạng. Các dịch vụ tầng giao vận cung cấp bao gồm:

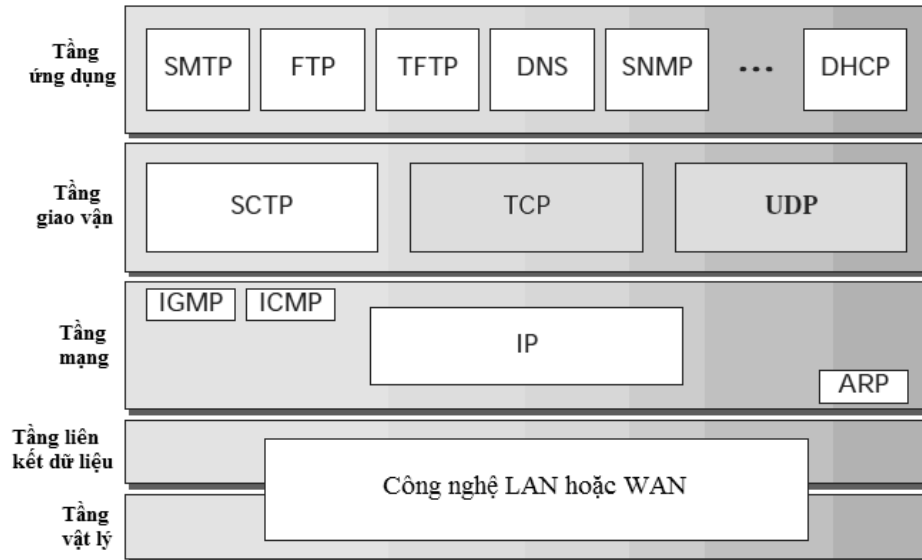
- Truyền thông tiến trình tới tiến trình
- Địa chỉ hóa thông qua các số hiệu cổng
- Đóng gói (tại phía gửi) và mở gói dữ liệu (tại phía nhận)
- Ghép kênh và phân kênh dữ liệu
- Điều khiển luồng
- Kiểm soát lỗi
- Điều khiển tắc nghẽn
- Dịch vụ truyền thông hướng kết nối và không kết nối

Hình 3.1 biểu diễn mối liên hệ của các giao thức tầng giao vận (TCP, UDP) với các giao thức khác các tầng trong chồng giao thức TCP/IP: TCP, UDP được đặt giữa tầng ứng dụng và lớp IP, được coi là trung gian giữa chương trình ứng dụng và các hoạt động mạng.

Giao thức tầng giao vận thực hiện các chức năng nhằm cung cấp các dịch vụ cho tầng ứng dụng (như đã nói ở trên). Có hai giao thức trong tầng giao vận là TCP và UDP (hiện tại có một giao thức mới được phát triển là giao thức SCTP), trong đó giao thức TCP cung cấp đầy đủ các chức năng của tầng giao vận, tuy nhiên giao thức UDP thực hiện các chức năng này ở mức hạn chế. Phần sau sẽ trình bày chi tiết về hai giao thức TCP và UDP.

3.2 UDP (*User Datagram Protocol*)

UDP là một giao thức tầng giao vận *không hướng kết nối, không tin cậy*. Nó không thêm bất kỳ điều gì vào dịch vụ IP ngoại trừ cung cấp truyền thông giữa các tiến trình thay vì truyền thông giữa các điểm đầu cuối (host-to-host). Tuy nhiên, UDP cung cấp kiểm soát lỗi trong một số trường hợp. Nếu UDP phát hiện ra một lỗi trong gói tin nhận được, nó sẽ âm thầm bỏ qua gói tin đó.

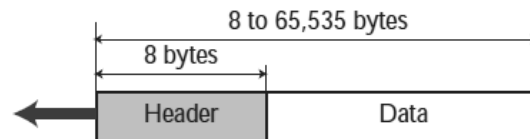


Hình 3.1 Vị trí của TCP và UDP trong chồng giao thức TCP/IP

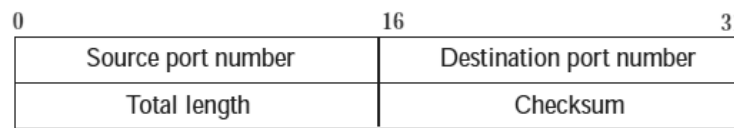
Khả năng của UDP là hạn chế, vậy tại sao các tiến trình vẫn sử dụng nó? Với một số nhược điểm có thể dẫn tới những ưu điểm. UDP là giao thức rất đơn giản sử dụng tối thiểu chi phí (phần tiêu đề gói tin). Nếu một tiến trình muốn gửi một thông điệp nhỏ và không quan tâm nhiều đến độ tin cậy, nó có thể sử dụng UDP. Việc gửi một gói tin nhỏ sử dụng UDP mất ít tương tác giữa người gửi và người nhận hơn là sử dụng giao thức TCP.

3.2.1 Gói tin người dùng (User datagram)

Gói tin UDP, được gọi là *user datagram* (gói tin người dùng), có phần tiêu đề gồm 8 byte. Hình 3.2 biểu diễn định dạng của gói tin UDP.



a. Gói tin người dùng UDP



b. Định dạng tiêu đề

Hình 3.2 Định dạng gói tin UDP

Các trường như sau:

- *Source port number* (Số hiệu cổng nguồn): Đây là số hiệu cổng được sử dụng bởi tiến trình chạy trên host nguồn. Nó có độ dài 16 bit, nghĩa là số hiệu cổng nguồn có thể từ 0 đến 65.535. Nếu host nguồn là client (gửi yêu cầu), thì trong hầu hết các trường hợp số hiệu cổng là số tùy ý được yêu cầu bởi tiến trình và được chọn bởi phần mềm UDP

chạy trên host nguồn. Nếu host nguồn là server (gửi đáp ứng), thì trong hầu hết các trường hợp số hiệu cổng là một số đã được biết trước.

- *Destination port number (Số hiệu cổng đích):* Đây là số hiệu cổng được sử dụng bởi tiến trình chạy trên host đích. Nó cũng có độ dài 16 bit. Nếu host đích là server (client đang gửi yêu cầu đến), thì trong hầu hết các trường hợp số hiệu cổng là một số đã được biết trước. Nếu host đích là client (server đang gửi đáp ứng đến), thì trong hầu hết các trường hợp số hiệu cổng là một số tùy ý. Trong trường hợp này, server sẽ sao chép số hiệu cổng tùy ý mà nó nhận được trong gói tin yêu cầu.
- *Length (Độ dài):* Trường này gồm 16 bit, xác định tổng chiều dài của gói tin người dùng, cả phần tiêu đề và dữ liệu. 16 bit có thể xác định chiều dài từ 0 đến 65.535 byte. Tuy nhiên, tổng chiều dài cần nhỏ hơn, vì một gói tin người dùng UDP được lưu trong một gói tin IP có độ dài 65.535 byte. Trường độ dài trong gói tin người dùng UDP thực sự là cần thiết. Một gói tin người dùng được đóng gói trong một gói tin IP. Có một trường trong gói tin IP xác định độ dài. Có một trường khác trong gói tin IP xác định độ dài phần tiêu đề. Nếu chúng ta lấy giá trị trường thứ nhất trừ đi trường thứ hai thì có thể biết được độ dài của gói tin UDP được đóng gói trong gói tin IP.

$$\text{Độ dài UDP} = \text{Độ dài IP} - \text{Độ dài tiêu đề IP}$$

Tuy nhiên, các nhà thiết kế UDP thấy rằng sẽ là hiệu quả hơn cho UDP đích khi tính toán chiều dài của dữ liệu từ những thông tin đã được cung cấp trong gói tin UDP, hơn là sử dụng phần mềm IP cung cấp thông tin này. Vì khi phần mềm IP cung cấp các gói tin UDP tới tầng giao vận, nó đã bỏ đi phần tiêu đề của gói tin IP.

- *Checksum (Kiểm tra lỗi):* Trường này được dùng để kiểm tra lỗi trên toàn bộ gói tin UDP (cả phần tiêu đề và phần dữ liệu).

3.2.2 Các dịch vụ của UDP

Truyền thông tiến trình tới tiến trình: UDP cung cấp truyền thông tiến trình tới tiến trình sử dụng các socket, là tổ hợp của địa chỉ IP và số hiệu cổng. Ví dụ một số cổng được sử dụng bởi UDP: Cổng 9 (giao thức Discard, từ chối bất kỳ gói tin nào nhận được), cổng 161 (giao thức SNMP, giao thức quản lý mạng đơn giản), cổng 111 (giao thức RPC, gọi thủ tục ở xa), v.v.

Các dịch vụ không hướng kết nối: Như đã đề cập ở trước, UDP cung cấp một dịch vụ không hướng kết nối. Nghĩa là mỗi gói tin người dùng được gửi bởi UDP là một gói tin độc lập. Không có mối liên hệ giữa các gói tin người dùng khác nhau ngay cả khi chúng đến từ cùng một tiến trình nguồn và đi đến cùng một chương trình đích. Các gói tin người dùng không được đánh số. Cũng vậy, không có việc thiết lập kết nối và hủy bỏ kết nối như trong trường hợp với TCP. Nghĩa là gói tin người dùng có thể đi bằng nhiều con đường khác nhau từ nguồn đến đích.

Một trong những ảnh hưởng của không hướng kết nối đó là tiến trình sử dụng UDP không thể gửi luồng dữ liệu tới UDP và mong UDP cắt chúng thành các gói tin người dùng khác nhau có liên quan. Thay vào đó, mỗi yêu cầu phải đủ nhỏ để vừa trong một gói tin người

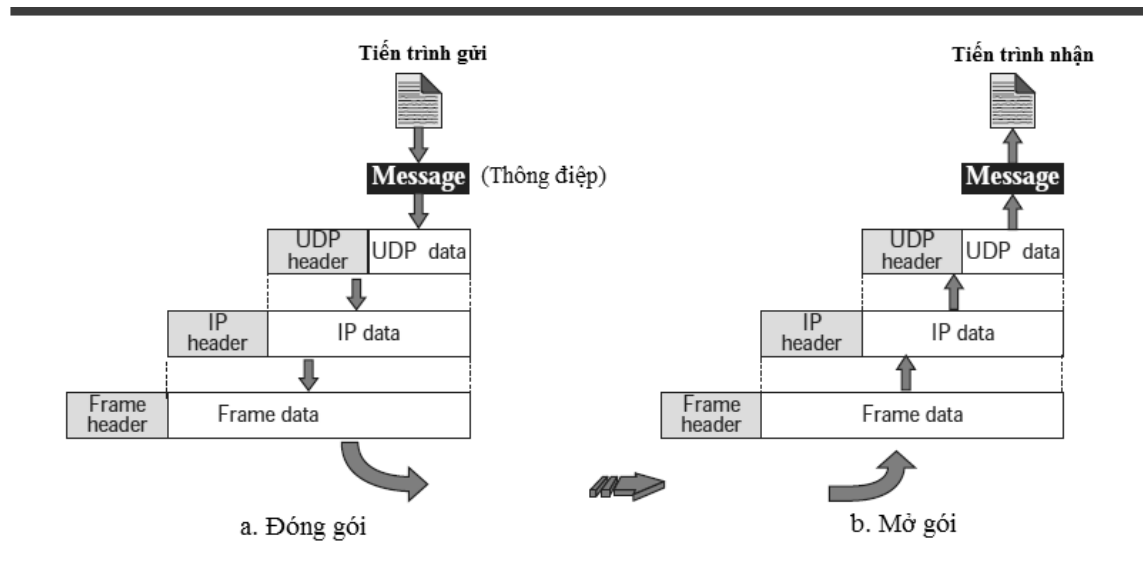
dùng. Chỉ tiến trình gửi các thông điệp ngắn, các thông điệp nhỏ hơn 65.507 byte (65.535 trừ đi 8 byte cho tiêu đề UDP và trừ đi 20 byte cho tiêu đề IP), có thể sử dụng UDP.

Điều khiển luồng: UDP là một giao thức rất đơn giản. Không có cơ chế điều khiển luồng, và do vậy không có cơ chế cửa sổ. Phía nhận có thể bị tràn các thông điệp đến. Không có điều khiển luồng nghĩa là tiến trình sử dụng UDP cần phải cung cấp dịch vụ này khi cần.

Kiểm soát lỗi: Không có cơ chế kiểm soát lỗi trong UDP ngoại trừ checksum. Nghĩa là phía bên gửi không biết được thông điệp có bị mất hay bị lặp hay không. Khi phía nhận dò được lỗi bằng checksum, gói tin người dùng sẽ âm thầm bị bỏ qua. UDP không có cơ chế kiểm soát lỗi, nghĩa là tiến trình sử dụng UDP cần phải cung cấp dịch vụ này khi cần.

Điều khiển tắc nghẽn: Vì UDP là giao thức không hướng kết nối, nên nó không cung cấp cơ chế điều khiển tắc nghẽn. UDP giả thiết là các gói tin được gửi đi là nhỏ và không thường xuyên, và không thể tạo ra tắc nghẽn trong mạng. Giả sử này có thể là đúng hoặc không đúng ngày nay khi mà UDP được sử dụng cho truyền dữ liệu video và audio thời gian thực.

Đóng gói và mở gói dữ liệu: Để gửi thông điệp từ một tiến trình tới một tiến trình khác, giao thức UDP phải thực hiện đóng gói và mở gói dữ liệu (hình 3.3).



Hình 3.3 Đóng gói và mở gói dữ liệu

- **Đóng gói:** Khi tiến trình có một thông điệp cần gửi qua UDP, nó sẽ gửi thông điệp tới UDP theo cặp địa chỉ socket và độ dài của dữ liệu. UDP nhận dữ liệu và thêm vào phần tiêu đề UDP. Sau đó UDP chuyển gói tin người dùng tới IP với các địa chỉ socket. IP sẽ thêm vào phần thông tin điều khiển của nó, sử dụng giá trị 17 trong trường giao thức, xác định rằng dữ liệu đến từ giao thức UDP. Tiếp theo, gói tin IP được chuyển cho tầng liên kết dữ liệu. Tầng liên kết dữ liệu nhận gói tin IP, thêm vào phần tiêu đề của nó (và có thể cả phần đuôi) và chuyển nó cho tầng vật lý. Tầng vật lý mã hóa các bit thành các tín hiệu điện hoặc tín hiệu ánh sáng và gửi nó đi đến máy ở xa.

- **Mở gói:** Khi thông điệp đến tại một host đầu cuối, tầng vật lý giải mã tín hiệu thành bit và chuyển nó tới tầng liên kết dữ liệu. Tầng liên kết dữ liệu sử dụng phần tiêu đề (và phần đuôi) để kiểm tra dữ liệu. Nếu không có lỗi, thì phần tiêu đề và phần đuôi sẽ được loại bỏ và gói tin được chuyển tới IP. Phần mềm IP tự kiểm tra. Nếu không có lỗi, phần tiêu đề sẽ được loại bỏ và gói tin người dùng sẽ được chuyển tới UDP với địa chỉ IP bên gửi và bên nhận. UDP sử dụng checksum để kiểm tra toàn bộ gói tin người dùng. Nếu không có lỗi, thì phần tiêu đề sẽ được loại bỏ và dữ liệu ứng dụng cùng với địa chỉ socket gửi sẽ được chuyển cho tiến trình. Địa chỉ socket gửi được chuyển tới tiến trình trong trường hợp cần để đáp ứng với thông điệp nhận được.

Ghép kênh và phân kênh: Trong một host đang chạy chồng giao thức TCP/IP, chỉ có một UDP nhưng có thể có một vài tiến trình muốn dùng dịch vụ của UDP. Để xử lý tình huống này, UDP thực hiện ghép kênh và phân kênh dữ liệu.

- **Ghép kênh:** Tại phía gửi, có thể có một vài tiến trình cần gửi gói dữ liệu người dùng. Tuy nhiên, chỉ có một UDP. Đây là mối quan hệ nhiều-một và yêu cầu ghép kênh. UDP chấp nhận các thông điệp từ các tiến trình khác nhau, được phân biệt bởi số cổng đã được gán cho nó. Sau khi thêm phần tiêu đề, UDP chuyển gói tin người dùng tới IP.
- **Phân kênh:** Tại phía nhận, chỉ có một UDP. Tuy nhiên, có thể có nhiều tiến trình có thể nhận gói tin người dùng. Đây là mối quan hệ một-nhiều và yêu cầu phân kênh. UDP nhận các gói tin người dùng từ IP. Sau khi kiểm tra lỗi và loại bỏ phần tiêu đề, UDP phân phối mỗi thông điệp cho một tiến trình phù hợp dựa trên số hiệu cổng.

3.2.3 Các ứng dụng UDP

Phần sau là một số ứng dụng có nhiều lợi ích hơn khi sử dụng các dịch vụ của UDP so với khi sử dụng TCP.

- UDP phù hợp cho một tiến trình yêu cầu truyền thông yêu cầu-đáp ứng đơn giản với rất ít liên quan đến kiểm soát lỗi và điều khiển luồng. Nó không thường được sử dụng cho một tiến trình như FTP cần gửi số lượng dữ liệu lớn.
- UDP thích hợp cho một tiến trình với cơ chế kiểm soát lỗi và điều khiển luồng cục bộ. Ví dụ, tiến trình TFTP bao gồm kiểm soát lỗi và điều khiển luồng. Nó có thể dễ dàng sử dụng UDP.
- UDP là một giao thức giao vận phù hợp cho quảng bá. Khả năng quảng bá được nhúng trong phần mềm UDP chứ không được nhúng trong phần mềm TCP.
- UDP được sử dụng cho các tiến trình quản lý như SNMP.
- UDP được sử dụng cho một số giao thức cập nhật định tuyến như RIP (*Routing Information Protocol*).
- UDP thường được sử dụng cho các ứng dụng thời gian thực mà không chấp nhận sự chậm trễ giữa các phần của thông điệp nhận được.

3.3 TCP (Transmission Control Protocol)

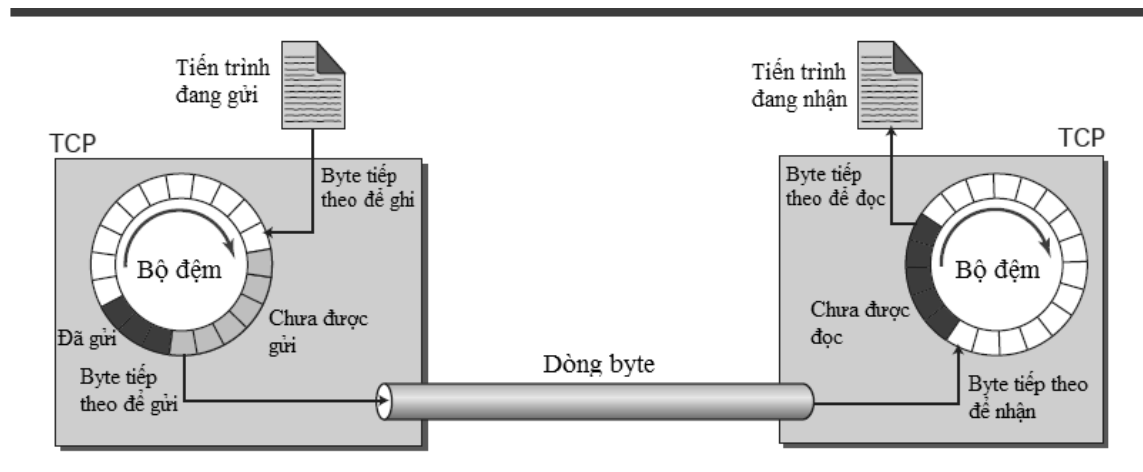
3.3.1 Các dịch vụ của TCP

Truyền thông tiến trình tới tiến trình: Giống như UDP, TCP cung cấp truyền thông tiến trình tới tiến trình sử dụng cổng. Ví dụ, cổng 20 và 21 (giao thức FTP, giao thức truyền tệp tin, dữ liệu và điều khiển), cổng 23 (giao thức TELNET), cổng 25 (giao thức SMTP, giao thức truyền mail đơn giản), cổng 53 (giao thức DNS, server tên miền), cổng 80 (giao thức HTTP, giao thức truyền siêu văn bản), v.v.

Dịch vụ phân phối dòng: TCP, không giống như UDP, là một giao thức hướng dòng. Trong UDP, tiến trình gửi thông điệp với giới hạn được xác định trước để UDP phân phối. UDP thêm phần tiêu đề của nó vào mỗi thông điệp và phân phối nó tới IP để truyền đi. Mỗi thông điệp trong tiến trình này gọi là một gói tin người dùng (*user datagram*), và cuối cùng trở thành một gói tin IP. IP hay UDP đều không nhận ra được bất kỳ mối liên hệ nào giữa các gói dữ liệu.

Tuy nhiên, TCP cho phép tiến trình gửi phân phối dữ liệu theo dòng bytes và tiến trình nhận thu dữ liệu theo dòng bytes. TCP tạo một môi trường trong đó hai tiến trình cùng được kết nối bởi một "đường ống" (*tube*) truyền các bytes đi trên mạng Internet.

Các bộ đệm gửi và nhận: Vì các tiến trình gửi và nhận có thể không cần thiết ghi và đọc dữ liệu cùng tốc độ, nên TCP cần các bộ đệm cho việc lưu trữ. Có 2 bộ đệm, bộ đệm gửi và bộ đệm nhận, mỗi bộ đệm cho một hướng. Các bộ đệm này cũng cần thiết cho việc điều khiển luồng và kiểm soát lỗi trong TCP. Phương pháp cài đặt bộ đệm là sử dụng một mảng vòng các đoạn 1 byte như trong hình 3.4. Để đơn giản, chúng tôi trình bày 2 bộ đệm, mỗi bộ 20 bytes; thông thường các bộ đệm gồm hàng trăm đến hàng nghìn bytes, tùy thuộc cài đặt.



Hình 3.4 Các bộ đệm gửi và nhận dữ liệu

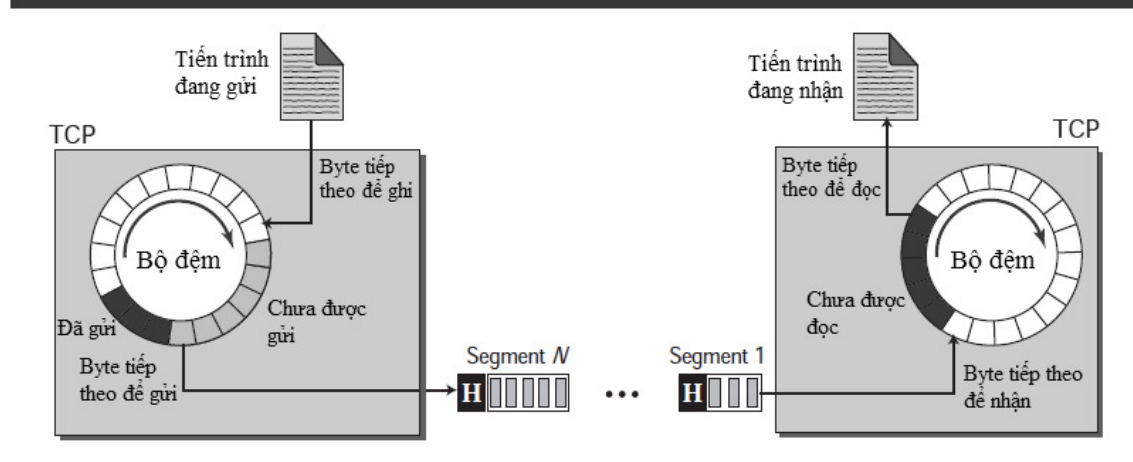
Hình vẽ biểu diễn chuyển động của dữ liệu theo một hướng. Tại phía gửi, bộ đệm chia thành 3 loại vùng. Phần màu trắng là khoảng trống có thể được điền đầy bởi tiến trình gửi. Các vùng tô màu (đen) đậm là các bytes đã được gửi nhưng chưa được báo nhận. TCP gửi sẽ lưu các bytes này trong vùng đệm cho đến khi nhận được báo nhận. Vùng tô màu (ghi) nhạt chứa các bytes được gửi bởi TCP gửi. Tuy nhiên, như sẽ thấy trong phần sau, TCP có thể chỉ gửi một phần trong phần vùng màu nhạt này. Điều này có thể có nguyên nhân là do sự chậm

trễ của tiến trình nhận, hoặc do tình trạng tắc nghẽn của mạng. Cũng cần chú ý rằng sau khi các byte trong vùng tô màu đậm được báo nhận, thì các vùng được xoay vòng và sẵn sàng để tiến trình gửi sử dụng. Đây là lý do sử dụng bộ đệm xoay vòng.

Hoạt động của bộ đệm tại phía nhận khá đơn giản. Bộ đệm vòng tròn được chia thành 2 vùng (trắng và được tô màu (đen) đậm). Vùng màu trắng là khoảng trống sẽ được điền đầy bởi các byte nhận được từ mạng. Phần tô màu đậm chứa các byte nhận được và có thể được đọc bởi tiến trình nhận, vùng được xoay vòng và được thêm vào vùng trống.

Phân đoạn (Segment): Mặc dù việc đệm xử lý sự chênh lệch về tốc độ giữa các tiến trình gửi và nhận, nhưng vẫn cần thêm một bước nữa trước khi có thể gửi dữ liệu đi. Lớp IP, như một nhà cung cấp dịch vụ cho TCP, cần phải gửi dữ liệu trong các gói tin, chứ không phải là các dòng bytes. Tại tầng giao vận, TCP nhóm một số các bytes lại với nhau thành một gói tin được gọi là *đoạn* (segment). TCP thêm vào phần tiêu đề cho mỗi đoạn (mục đích là để điều khiển) và phân phối đoạn tới lớp IP để truyền đi. Các đoạn được đóng gói trong một gói tin IP và truyền đi. Toàn bộ hoạt động này là trong suốt đối với tiến trình nhận. Sau đó các đoạn sẽ được nhận ra trình tự, bị mất, bị lỗi và gửi lại. Tất cả những việc này được xử lý bởi TCP gửi với tiến trình ứng dụng nhận không biết gì về các hoạt động của TCP. Hình 3.5 biểu diễn các đoạn đã được tạo ra bởi các byte trong vùng đệm.

Lưu ý là các đoạn không nhất thiết có cùng kích thước. Trong hình vẽ, để đơn giản, ở đây biểu diễn một đoạn gồm 3 byte và một đoạn khác gồm 5 byte. Trong thực tế, các đoạn có kích thước hàng trăm đến hàng nghìn bytes.



Hình 3.5 Các đoạn TCP

Truyền thông song công (Full-Duplex Communication): TCP cho phép dịch vụ truyền song công, nghĩa là dữ liệu có thể đi theo cả hai hướng tại cùng một thời điểm. Tại mỗi điểm cuối TCP có vùng đệm gửi và nhận của riêng nó, các đoạn được di chuyển theo cả hai hướng.

Ghép kênh và phân kênh: Như UDP, TCP thực hiện ghép kênh tại phía gửi và phân kênh tại phía nhận. Tuy nhiên, vì TCP là giao thức hướng kết nối, nên một kết nối cần phải được thiết lập cho mỗi cặp tiến trình.

Dịch vụ hướng kết nối: TCP, không giống UDP, là giao thức hướng kết nối. Khi một tiến trình tại trạm A muốn gửi và nhận dữ liệu từ tiến trình khác tại trạm B, thì diễn ra ba pha như sau:

1. Hai TCP thiết lập một kết nối ảo giữa chúng.
2. Dữ liệu trao đổi theo cả hai hướng.
3. Chấm dứt kết nối.

Đây là một kết nối ảo chứ không phải là một kết nối vật lý. TCP segment được đóng gói trong một IP datagram và có thể được gửi đi theo trình tự, hoặc bị mất, bị lỗi và sau đó truyền lại. Mỗi gói tin có thể được truyền đi đến đích bằng những con đường khác nhau. Không có kết nối vật lý nào ở đây. TCP tạo môi trường hướng dòng chịu trách nhiệm phân phối các byte theo trình tự đến trạm khác.

Dịch vụ truyền tin cậy: TCP là giao thức truyền tin cậy. Nó sử dụng kỹ thuật báo nhận để kiểm tra tính an toàn của dữ liệu đến (sẽ thảo luận đặc trưng này ở phần điều khiển lỗi).

3.3.2 Các đặc điểm của TCP

Để cung cấp được các dịch vụ đã đề cập ở phần trên, TCP có một số các đặc điểm được tóm tắt như sau.

Hệ thống đánh số: Mặc dù phần mềm TCP theo dõi các segment được truyền đi hoặc nhận về, nhưng không có trường nào chứa giá trị số hiệu của segment trong phần tiêu đề cả. Thay vào đó, có hai trường được gọi là *số thứ tự (sequence number)* và *số báo nhận (acknowledgment number)*. Hai trường này chỉ ra số byte chứ không phải là số hiệu segment.

- **Số byte:** TCP đánh số tất cả các byte dữ liệu (octet) được truyền đi trong kết nối. Số là độc lập trong mỗi hướng. Khi TCP nhận các byte dữ liệu từ một tiến trình, TCP lưu trữ chúng trong bộ đệm gửi và đánh số chúng. Số không nhất thiết phải bắt đầu từ 0. Thay vào đó, TCP chọn một số bất kỳ trong khoảng 0 đến $2^{32}-1$ là số cho byte đầu tiên. Ví dụ, nếu số được chọn là 1057 và tổng số dữ liệu được gửi là 6000 byte, thì các byte được đánh số từ 1057 đến 7056. Số byte cũng được sử dụng trong điều khiển luồng và kiểm soát lỗi.
- **Số thứ tự:** Sau khi các byte được đánh số, TCP sẽ gán một số thứ tự cho mỗi segment mà nó sẽ được gửi đi. Số thứ tự cho mỗi segment là số byte đầu tiên của dữ liệu trong segment đó. Khi một segment mang một kết nối dữ liệu và thông tin điều khiển, nó sử dụng một số thứ tự. Nếu một segment không mang dữ liệu người dùng, nó sẽ không xác định hợp lệ một số thứ tự. Vẫn có các trường, nhưng giá trị là không hợp lệ. Tuy nhiên, một số segment, khi chỉ mang thông tin điều khiển, cần một số thứ tự để cho phép có được một báo nhận từ phía nhận. Các segment này được sử dụng để thiết lập, kết thúc hoặc hủy bỏ kết nối. Mỗi segment này sử dụng một số thứ tự qua một byte nó mang, mà không có dữ liệu thực tế nào.
- **Số báo nhận:** Như đã nói ở trước, truyền thông trong TCP là song công; khi một kết nối được thiết lập, cả hai phía có thể gửi và nhận dữ liệu đồng thời. Tại mỗi phía, số byte thường được bắt đầu bằng các số khác nhau. Số thứ tự trong mỗi hướng cho biết

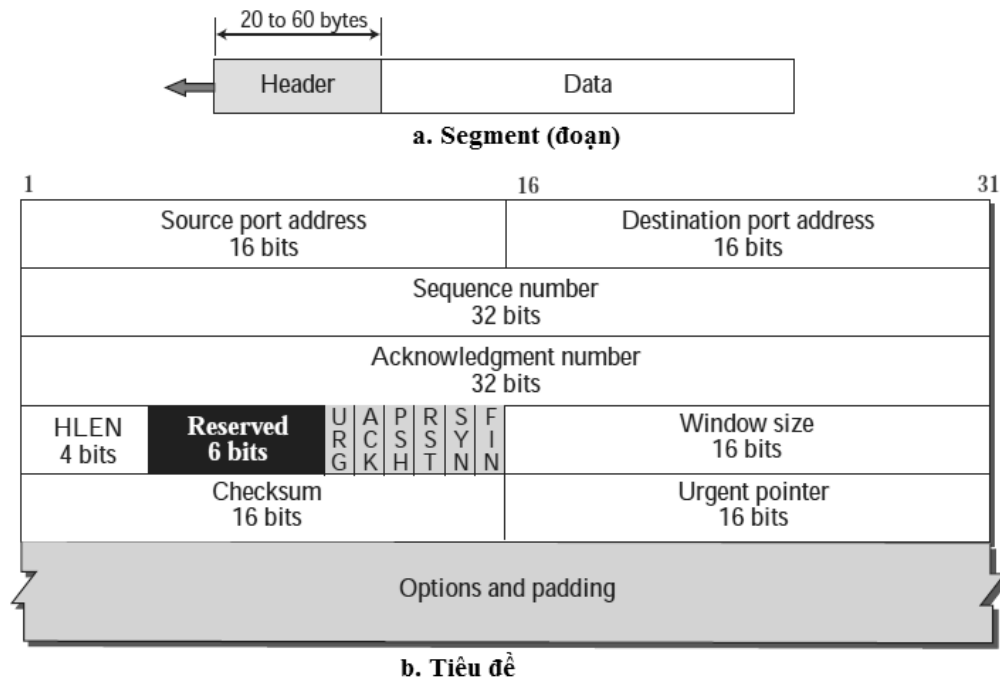
số byte đầu tiên mà segment mang. Mỗi phía cũng sử dụng một số báo nhận để xác nhận số byte mà nó nhận được. Tuy nhiên, số báo nhận xác định số của byte kế tiếp mà phía đó mong muốn nhận được. Ngoài ra, số báo nhận được tích lũy, nghĩa là các bên có số byte cuối cùng mà nó nhận được, đã được chuyển đến an toàn, thêm 1 vào và thông báo rằng tổng này là số báo nhận. Thuật ngữ *tích lũy* ở đây nghĩa là nếu một bên dùng 5643 là một số báo nhận thì nghĩa là nó đã nhận được tất cả các byte từ đầu đến 5642. Điều này không có nghĩa là bên đó đã nhận được 5.642 byte vì số byte đầu tiên không phải bắt đầu từ 0.

Điều khiển luồng: TCP, không giống như UDP, cung cấp cơ chế điều khiển luồng. Các TCP gửi cho biết bao nhiêu dữ liệu có thể được chấp nhận từ tiến trình gửi; TCP nhận cho biết bao nhiêu dữ liệu có thể được gửi đi bởi tiến trình gửi. Điều này giúp ngăn cản phía nhận bị quá tải với dữ liệu. Hệ thống đánh số cho phép TCP sử dụng điều khiển luồng hướng byte.

Kiểm soát lỗi: Để cung cấp dịch vụ tin cậy, TCP cài đặt cơ chế kiểm soát lỗi. Mặc dù cơ chế này coi segment như là một đơn vị dữ liệu cho kiểm soát lỗi (bị lỗi hoặc bị mất), thực tế kiểm soát lỗi được thực hiện theo hướng byte.

Điều khiển tắc nghẽn: TCP, không giống như UDP, có cơ chế điều khiển tắc nghẽn mạng. Số lượng dữ liệu được gửi bởi phía gửi không chỉ được điều khiển bởi phía nhận (điều khiển luồng), mà còn được xác định bởi mức độ tắc nghẽn (nếu có) trong mạng.

3.3.3 Định dạng đoạn (Segment)



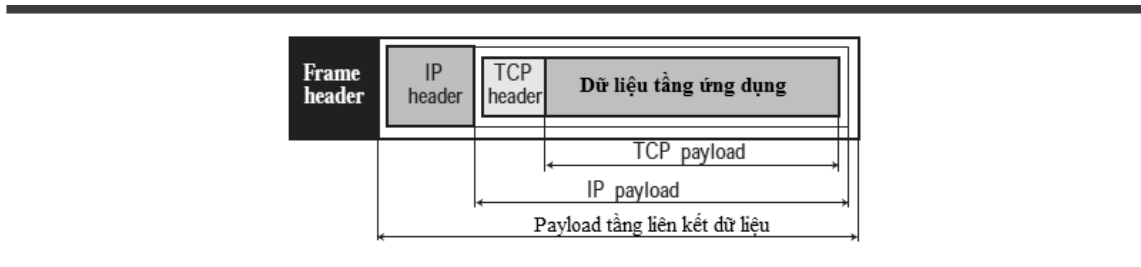
Hình 3.6 Định dạng đoạn

Định dạng của một đoạn được thể hiện trên hình 3.6. Đoạn bao gồm một tiêu đề từ 20 đến 60 byte, tiếp theo là dữ liệu từ chương trình ứng dụng. Tiêu đề là 20 byte nếu không có

phần tùy chọn và lên đến 60 byte nếu nó chứa các tùy chọn. Chúng ta sẽ thảo luận về một số trường tiêu đề trong phần này để hiểu rõ ý nghĩa và mục đích của chúng.

- *Source port address (Địa chỉ cổng nguồn)*: Trường này gồm 16 bit xác định số cổng của chương trình ứng dụng trong host đang gửi đoạn.
- *Destination port address (Địa chỉ cổng đích)*: Trường này gồm 16 bit xác định số cổng của các chương trình ứng dụng trong các host được nhận đoạn.
- *Sequence number (Số thứ tự)*: Trường 32 bit này xác định số được gán cho byte đầu tiên của dữ liệu chứa trong đoạn. Trong quá trình thiết lập kết nối, mỗi bên sẽ sử dụng một số ngẫu nhiên (thường là khác nhau tùy mỗi bên) để tạo ra một *số thứ tự khởi tạo (initial sequence number - ISN)*.
- *Acknowledgment number (Số báo nhận)*: Trường 32 bit này xác định số byte mà đoạn mong muốn sẽ nhận được từ phía bên kia. Nếu bên nhận đã nhận được thành công x byte từ phía bên kia, nó sẽ trả về $x + 1$ là số báo nhận. Báo nhận và dữ liệu có thể được đóng gói cùng nhau.
- *Header length (Độ dài tiêu đề)*: Trường này gồm 4 bit xác định chiều dài theo đơn vị từ 4 byte của phần tiêu đề gói tin TCP. Chiều dài của tiêu đề có thể từ 20 đến 60 byte. Vì vậy, giá trị của trường này luôn luôn nằm trong khoảng từ 5 ($5 \times 4 = 20$) đến 15 ($15 \times 4 = 60$).
- *Reserved (Dự phòng)*: Đây là một trường 6 bit dành cho sử dụng dự phòng trong tương lai.
- *Control (Điều khiển)*: Trường này xác định 6 bit điều khiển hoặc cờ khác nhau, bao gồm URG (cờ trả báo khẩn), ACK (báo nhận), PSH (yêu cầu đẩy dữ liệu), RST (thiết lập lại kết nối), SYN (đồng bộ số thứ tự), FIN (chấm dứt kết nối). Một hoặc một số bit có thể được thiết lập tại một thời điểm. Các bit này cho phép điều khiển luồng, thiết lập và hủy bỏ kết nối, ngắt kết nối, và chế độ truyền dữ liệu trong TCP.
- *Window size (Kích thước cửa sổ)*: Trường này xác định kích thước cửa sổ gửi của TCP theo byte. Lưu ý rằng độ dài của trường này là 16 bit, có nghĩa là kích thước tối đa của cửa sổ là 65.535 byte. Giá trị này thường được gọi là *cửa sổ nhận (receiving window - rwnd)* và được xác định bởi phía nhận. Phía gửi phải tuân theo phía nhận trong trường hợp này.
- *Checksum*: Trường 16 bit này chứa checksum (tổng kiểm tra). Sử dụng checksum trong TCP là bắt buộc.
- *Urgent Pointer (Con trỏ khẩn)*: Trường này 16 bit, trong đó chỉ có hiệu lực nếu cờ khẩn được thiết lập, được sử dụng khi đoạn có chứa dữ liệu khẩn. Nó xác định một giá trị cần phải được thêm vào số thứ tự để có được số lượng các byte khẩn cuối cùng trong phần dữ liệu của đoạn.
- *Options (Tùy chọn)*: Có thể có đến 40 byte thông tin tùy chọn trong tiêu đề TCP.

Một đoạn TCP đóng gói dữ liệu nhận được từ tầng ứng dụng. Đoạn TCP sẽ được đóng gói trong một gói tin IP, sau đó được đóng gói trong một khung (frame) ở tầng liên kết dữ liệu như trong hình 3.7.



Hình 3.7 Đóng gói dữ liệu

3.3.4 Kết nối TCP

TCP là giao thức vận chuyển hướng kết nối. Như đã biết, giao thức vận chuyển hướng kết nối cần phải thiết lập một đường ảo giữa nguồn và đích. Tất cả đoạn thuộc thông điệp sau đó được gửi qua đường dẫn ảo này. Việc sử dụng một con đường ảo duy nhất cho toàn bộ thông điệp tạo điều kiện cho quá trình báo nhận cũng như truyền lại các khung bị hư hỏng hoặc bị mất. Ý tưởng là: TCP nào sử dụng dịch vụ của IP, một giao thức không hướng kết nối, có thể được định hướng kết nối. Vấn đề ở đây kết nối TCP là ảo, không phải là vật lý. TCP hoạt động ở mức cao hơn. TCP sử dụng các dịch vụ của IP để phân phối các đoạn riêng lẻ đến bên nhận, nhưng nó tự điều khiển kết nối của nó. Nếu một đoạn bị mất hoặc bị hỏng, nó sẽ được truyền lại. Không giống như TCP, IP không biết việc truyền lại này. Nếu một đoạn đến không theo trình tự, TCP giữ nó cho đến khi đoạn bị thiếu đến; IP không biết đến việc sắp xếp lại này.

Trong TCP, vận chuyển hướng kết nối cần ba giai đoạn: thiết lập kết nối, truyền dữ liệu, và chấm dứt (giải phóng) kết nối.

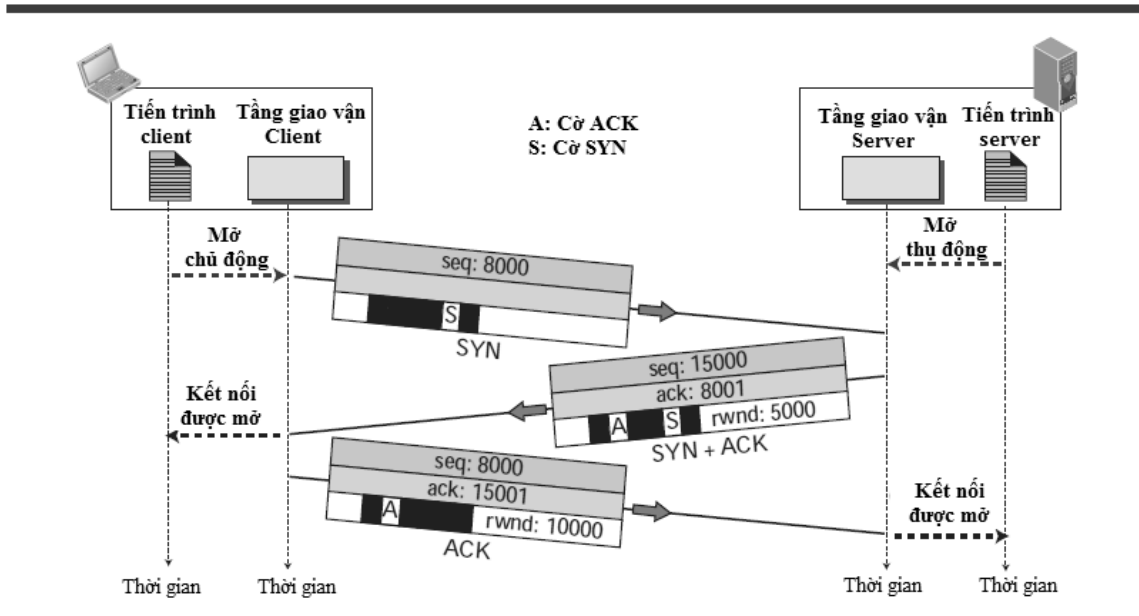
3.3.4.1 Thiết lập kết nối

TCP truyền dữ liệu ở chế độ song công. Khi hai TCP trong hai máy tính kết nối với nhau, chúng có thể gửi các đoạn cho nhau tại cùng một thời điểm. Điều này có nghĩa là mỗi bên phải khởi tạo việc truyền thông và nhận được sự chấp thuận của phía bên kia trước khi dữ liệu được truyền đi.

Thiết lập kết nối trong TCP được gọi là bắt tay ba bước. Trong ví dụ, một chương trình ứng dụng, được gọi là client, muốn thực hiện một kết nối với một chương trình ứng dụng khác, được gọi là server, sử dụng TCP là giao thức tầng giao vận.

Tiến trình bắt đầu với server. Chương trình server báo cho TCP của nó biết là nó đã sẵn sàng chấp nhận một kết nối. Yêu cầu này được gọi là *mở thụ động (passive open)*. Mặc dù TCP server sẵn sàng chấp nhận một kết nối nào từ bất kỳ các máy tính trên thế giới, nhưng nó không thể tự tạo kết nối.

Chương trình client đưa ra một yêu cầu *mở chủ động (active open)*. Client muốn kết nối tới một server mở sẽ báo cho TCP của nó kết nối tới server cụ thể. Lúc này, TCP có thể bắt đầu quá trình bắt tay ba bước như trong hình 3.8.



Hình 3.8 Thiết lập kết nối sử dụng bắt tay ba bước

Để hiển thị quá trình chúng ta sử dụng đường thời gian. Mỗi đoạn có giá trị cho tất cả các trường tiêu đề của nó và cũng có thể có một số trường tùy chọn. Tuy nhiên, ở đây chỉ hiển thị một số các trường cần thiết để hiểu từng bước: số thứ tự, số báo nhận, các cờ điều khiển, và kích thước cửa sổ nếu có liên quan. Ba bước trong giai đoạn này như sau.

1. Client gửi đoạn đầu tiên, là đoạn SYN, trong đó chỉ có cờ SYN được thiết lập. Đoạn này là để đồng bộ hóa các số thứ tự. Client trong ví dụ chọn một số ngẫu nhiên là số thứ tự đầu tiên gửi số này đến server. Số thứ tự này được gọi là số thứ tự ban đầu (ISN). Lưu ý rằng đoạn này không chứa số báo nhận. Nó cũng không xác định kích thước cửa sổ; định nghĩa kích thước cửa sổ chỉ có ý nghĩa chỉ khi đoạn có chứa số báo nhận. Đoạn cũng có thể bao gồm một số tùy chọn (sẽ thảo luận sau). Chú ý là đoạn SYN là một đoạn điều khiển và không chứa dữ liệu. Tuy nhiên, nó có sử dụng một số thứ tự. Khi việc truyền dữ liệu bắt đầu, ISN được tăng lên 1. Chúng ta có thể nói rằng đoạn SYN không mang dữ liệu thực, nhưng có thể coi nó như là có chứa một byte tưởng tượng.
2. Server gửi đoạn thứ hai, đoạn SYN + ACK với hai bit cờ thiết lập: SYN và ACK. Đoạn này có hai mục đích. Đầu tiên, nó là một đoạn SYN để truyền thông theo một hướng khác. Server sử dụng đoạn này để khởi tạo một số thứ tự cho việc đánh số các byte được gửi từ server cho client. Server cũng báo đã nhận được đoạn SYN từ client bằng cách thiết lập cờ ACK và hiển thị số thứ tự tiếp theo dự kiến sẽ nhận được từ client. Bởi vì nó có chứa báo nhận, nó cũng cần phải xác định kích thước cửa sổ nhận, *rwnd* (được sử dụng bởi client), như sẽ thấy trong phần điều khiển luồng.
3. Client sẽ gửi đoạn thứ ba. Đây chỉ là một đoạn ACK. Nó báo đã nhận được đoạn thứ hai với cờ ACK và trường số báo nhận. Lưu ý rằng các số thứ tự trong đoạn giống như trong đoạn SYN; đoạn ACK không sử dụng bất kỳ số thứ tự nào. Client cũng phải xác định kích thước cửa sổ server. Một số cài đặt cho phép đoạn thứ ba trong giai đoạn kết

nối này có thể mang phần dữ liệu đầu tiên từ client. Trong trường hợp này, đoạn thứ ba phải có một số thứ tự mới cho biết số byte của byte đầu tiên của dữ liệu. Nói chung, đoạn thứ ba thường không mang dữ liệu và không sử dụng số thứ tự.

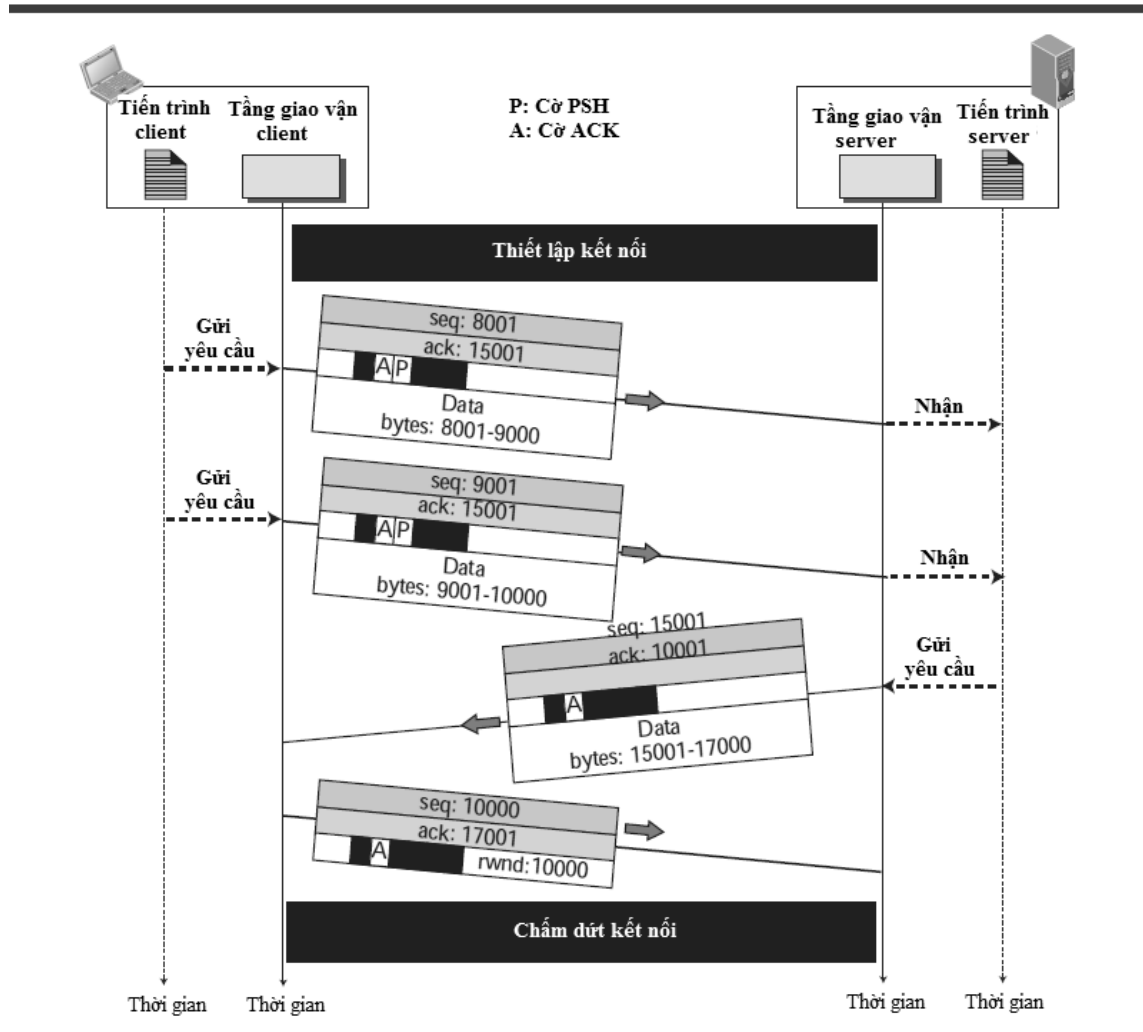
Thủ tục thiết lập kết nối trong giao thức TCP có dẫn đến một vấn đề an ninh nghiêm trọng được gọi là *tấn công ngập lụt SYN (SYN Flooding Attack)*. Điều này xảy ra khi một hoặc nhiều kẻ tấn công độc hại gửi một số lượng lớn các đoạn SYN đến một server, giả vờ rằng mỗi đoạn trong đó đến từ một client khác nhau bằng cách giả mạo địa chỉ IP nguồn trong gói tin. Server, giả định rằng các client đang phát hành mở chủ động, phân bổ các nguồn tài nguyên cần thiết, chẳng hạn như tạo các bảng khối điều khiển truyền (*transfer control block - TCB*) và thiết lập các bộ định thời. TCP server sau đó gửi các đoạn SYN + ACK cho client giả, bị mất. Server chờ đợi bước thứ ba của quá trình bắt tay, tuy nhiên, các nguồn lực đã được phân bổ lại không được sử dụng. Nếu, trong thời gian ngắn này, số lượng các đoạn SYN quá lớn, thì thậm chí server sẽ bị cạn kiệt tài nguyên và có thể không còn khả năng chấp nhận yêu cầu kết nối từ các client hợp lệ nữa. Tấn công ngập lụt SYN này thuộc về một nhóm các cuộc tấn công bảo mật được gọi là *tấn công từ chối dịch vụ (denial of service attack - DOS)*, trong đó kẻ tấn công chiếm độc quyền một hệ thống với rất nhiều yêu cầu dịch vụ làm cho hệ thống bị quá tải và từ chối cung cấp dịch vụ cho những yêu cầu hợp lệ.

Một số cài đặt của TCP có chiến lược để giảm bớt tác động của tấn công SYN. Một số phương pháp thực hiện áp đặt giới hạn yêu cầu kết nối trong một khoảng thời gian xác định. Một số phương pháp khác cố gắng lọc ra các gói tin đến từ những địa chỉ nguồn không mong muốn. Gần đây có một chiến lược là trì hoãn phân bổ nguồn lực cho đến khi server có thể xác minh được các yêu cầu kết nối đến từ một địa chỉ IP hợp lệ, bằng cách sử dụng cookie.

3.3.4.2 Truyền dữ liệu

Sau khi kết nối được thiết lập, truyền dữ liệu hai chiều có thể được diễn ra. Client và server có thể gửi dữ liệu và báo nhận theo cả hai chiều. Báo nhận được mang cùng với dữ liệu. Hình 3.9 là một ví dụ.

Trong ví dụ này, sau khi kết nối được thiết lập, client gửi 2.000 byte dữ liệu trong hai đoạn. Sau đó server gửi 2.000 byte trong một đoạn. Client gửi hơn một đoạn. Ba đoạn đầu tiên mang cả dữ liệu và báo nhận, nhưng đoạn cuối cùng chỉ mang một báo nhận vì không có nhiều dữ liệu được gửi đi. Chú ý các giá trị của các số thứ tự và số báo nhận. Các đoạn dữ liệu được gửi bởi client phải có cờ PSH (push) được thiết lập để TCP server cố gắng phân phối dữ liệu cho tiến trình server sớm nhất ngay khi chúng nhận được. Mặt khác, đoạn từ server không thiết lập cờ PSH. Hầu hết các cài đặt TCP có tùy chọn thiết lập hoặc không thiết lập cờ này.



Hình 3.9 Truyền dữ liệu

a. Việc đẩy dữ liệu

Việc gửi trong TCP cần sử dụng một bộ đệm để lưu trữ các dòng dữ liệu đến từ chương trình ứng dụng gửi và có thể lựa chọn kích thước đoạn. Việc tiếp nhận TCP cũng cần bộ đệm dữ liệu khi dữ liệu đến và phân phối dữ liệu đến các chương trình ứng dụng khi chương trình ứng dụng đã sẵn sàng hoặc khi có điều kiện thuận tiện cho việc tiếp nhận TCP. Sự linh hoạt này làm tăng hiệu quả của TCP.

Tuy nhiên, có trường hợp chương trình ứng dụng không cần đến sự linh hoạt này. Ví dụ, xem xét trường hợp một chương trình ứng dụng giao tiếp tương tác với một chương trình ứng dụng ở một đầu cuối khác. Chương trình ứng dụng ở một phía muốn gửi một phím tắt tới ứng dụng ở phía kia và nhận phản hồi ngay lập tức. Việc truyền dẫn bị trễ và việc phân phối dữ liệu bị trễ có thể sẽ không được chấp nhận bởi chương trình ứng dụng.

TCP có thể xử lý các tình huống như vậy. Chương trình ứng dụng ở phía gửi có thể yêu cầu thao tác push (đẩy). Điều này có nghĩa là TCP gửi không phải chờ đợi cửa sổ được lấp đầy. Nó phải tạo ra một đoạn và gửi đi ngay lập tức. TCP gửi cũng phải thiết lập bit đẩy

(PSH) để TCP nhận biết rằng đoạn bao gồm dữ liệu phải được phân phối tới chương trình ứng dụng nhận càng sớm càng tốt và không phải chờ đợi thêm dữ liệu.

Mặc dù thao tác đẩy có thể được yêu cầu bởi chương trình ứng dụng, nhưng hầu hết các cài đặt TCP hiện tại thường bỏ qua các yêu cầu như vậy. TCP có thể chọn hoặc sử dụng hoặc không sử dụng tính năng này.

b. Dữ liệu khẩn

TCP là một giao thức hướng dòng. Điều này có nghĩa là dữ liệu được biểu diễn từ chương trình ứng dụng đến TCP như một dòng các byte. Mỗi byte dữ liệu có một vị trí trong dòng. Tuy nhiên, có trường hợp trong đó chương trình ứng dụng cần gửi byte *khẩn cấp*, một số byte cần được xử lý một cách đặc biệt bởi ứng dụng ở đầu kia. Giải pháp là gửi một đoạn với bit URG được thiết lập. Chương trình ứng dụng gửi báo cho TCP gửi đây là phần dữ liệu là khẩn cấp. TCP gửi sẽ tạo ra một đoạn và chèn dữ liệu khẩn tại đầu của đoạn. Phần còn lại của đoạn có thể chứa dữ liệu thường từ bộ đệm. Trường con trỏ khẩn (urgent pointer) trong tiêu đề xác định điểm cuối của dữ liệu khẩn (byte cuối cùng của dữ liệu khẩn).

Khi TCP nhận nhận được đoạn với bit URG được thiết lập, nó sẽ thông báo cho ứng dụng nhận biết. Việc này được thực hiện phụ thuộc vào hệ điều hành. Sau đó, phụ thuộc vào chương trình nhận để có hành động tiếp.

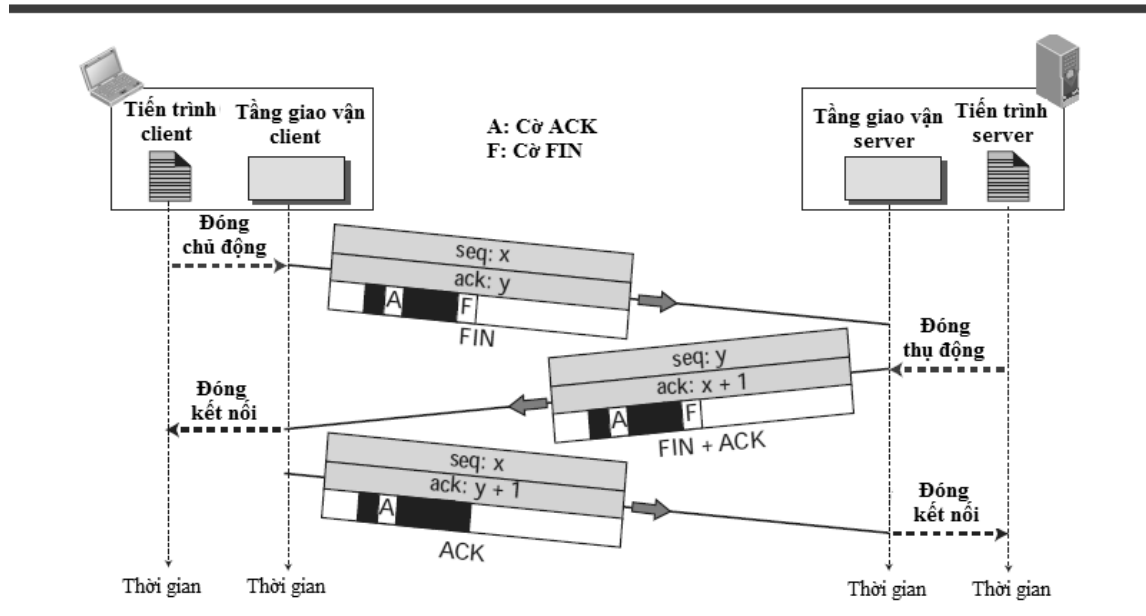
Điều quan trọng là phải đề cập là dữ liệu khẩn của TCP không phải là một dịch vụ ưu tiên, cũng không phải là dịch vụ dữ liệu nhanh. Thay vào đó, chế độ khẩn trong TCP là một dịch vụ từ chương trình ứng dụng ở phía gửi để đánh dấu một số phần của dòng byte cần xử lý đặc biệt bởi chương trình ứng dụng ở phía nhận.

Vì vậy, báo hiệu sự hiện diện của dữ liệu khẩn và đánh dấu vị trí của nó trong dòng dữ liệu chỉ dùng để phân biệt việc phân phối dữ liệu khẩn với việc phân phối dữ liệu TCP khác. Đối với tất cả các mục đích khác, dữ liệu khẩn được xử lý như phần còn lại của dòng byte TCP. Chương trình ứng dụng tại phía nhận phải đọc từng byte dữ liệu một cách chính xác theo thứ tự nó đã được gửi bất kể có sử dụng hay không chế độ khẩn. TCP chuẩn, như đã cài đặt, không bao giờ phân phối bất kỳ dữ liệu nào không đúng thứ tự.

3.3.4.3 Chấm dứt kết nối

Một trong hai bên (client hoặc server), tùy theo dữ liệu trao đổi, có thể đóng kết nối, mặc dù việc này thường được xuất phát từ phía client. Hầu hết các cài đặt hiện nay cho phép hai tùy chọn để chấm dứt kết nối: bắt tay ba bước và bắt tay bốn bước với một tùy chọn half-close. Phần sau chỉ trình bày tùy chọn chấm dứt kết nối bắt tay ba bước (vì hiện nay hầu hết các cài đặt đều cho phép dùng tùy chọn này), xem hình 3.10.

1. Trong một tình huống chung, TCP client, sau khi nhận được lệnh đóng từ tiến trình, sẽ gửi đoạn đầu tiên là đoạn FIN với cờ FIN được thiết lập. Lưu ý rằng đoạn FIN có thể bao gồm đoạn cuối cùng của dữ liệu được gửi bởi client hoặc có thể chỉ là đoạn điều khiển như thể hiện trong hình vẽ. Nếu nó chỉ là đoạn điều khiển, nó sẽ chỉ sử dụng một số thứ tự.



Hình 3.10 Chấm dứt kết nối sử dụng bắt tay ba bước

- TCP server, sau khi nhận được đoạn FIN, thông báo cho tiến trình của nó về tình trạng và gửi đoạn thứ hai, đoạn FIN + ACK, để xác nhận việc tiếp nhận đoạn FIN từ client và đồng thời thông báo đóng kết nối theo một hướng khác. Đoạn này cũng có thể chứa phần cuối cùng của dữ liệu từ server. Nếu nó không mang dữ liệu, nó chỉ sử dụng một số thứ tự.
- TCP client gửi đoạn cuối, đoạn ACK, để xác nhận việc đã nhận được đoạn FIN từ TCP server. Đoạn này có chứa số báo nhận, là 1 cộng với số thứ tự nhận được trong đoạn FIN từ server. Đoạn này không mang dữ liệu và không dùng số thứ tự.

3.3.5 Thiết lập lại kết nối

TCP ở một đầu có thể từ chối một yêu cầu kết nối, có thể hủy bỏ kết nối hiện có, hoặc có thể chấm dứt một kết nối. Tất cả được thực hiện với cờ RST (reset).

- Từ chối một kết nối:** Giả sử TCP ở một phía yêu cầu kết nối đến một cổng không tồn tại. TCP ở phía bên kia có thể gửi một đoạn với bit RST được thiết lập để từ chối yêu cầu.
- Hủy bỏ một kết nối:** TCP có thể muốn hủy bỏ một kết nối hiện tại do tình trạng bất thường. Nó có thể gửi một đoạn RST để đóng kết nối.
- Chấm dứt một kết nối treo (Idle):** TCP ở một bên có thể phát hiện ra rằng TCP ở phía bên kia đã bị treo trong một khoảng thời gian dài. Nó có thể gửi một đoạn RST để chấm dứt kết nối. Quá trình này tương tự như hủy bỏ kết nối.

3.3.6 Cửa sổ trong TCP

Trước khi thảo luận việc truyền dữ liệu trong TCP và các vấn đề như điều khiển luồng, kiểm soát lỗi, và điều khiển tắc nghẽn, phần này mô tả cửa sổ được sử dụng trong TCP. TCP sử dụng hai cửa sổ (cửa sổ gửi và cửa sổ nhận) cho mỗi hướng truyền dữ liệu, có nghĩa là bốn

cửa sổ cho truyền thông liên lạc hai chiều. Tuy nhiên, để đơn giản, ở đây giả định không thực tế là việc giao tiếp chỉ thực hiện một chiều (từ client đến server); truyền thông hai chiều có thể được suy diễn bằng cách sử dụng hai truyền thông một chiều.

3.3.6.1 Cửa sổ gửi

Kích thước cửa sổ gửi được quyết định bởi người nhận (điều khiển luồng) và tình trạng tắc nghẽn trong mạng (điều khiển tắc nghẽn): cửa sổ gửi mở ra (opens), đóng lại (closes), hoặc co lại (shrinks).

Các đặc trưng của cửa sổ gửi trong TCP:

1. Cửa sổ TCP được tính theo byte. Mặc dù thực tế truyền thông trong TCP là đoạn theo đoạn, nhưng các biến điều khiển cửa sổ được thể hiện trong byte.
2. Trong một số cài đặt, TCP có thể lưu trữ dữ liệu nhận được từ tiến trình và sau đó mới gửi đi, nhưng ở đây giả thiết là TCP gửi có khả năng gửi các đoạn dữ liệu đi ngay khi nó nhận được từ tiến trình.
3. Giao thức TCP chỉ sử dụng một bộ định thời cho mỗi gói tin được gửi đi.

3.3.6.2 Cửa sổ nhận

Cửa sổ nhận có thể mở ra và đóng lại; trong thực tế, các cửa sổ sẽ không bao giờ thu nhỏ. Các đặc trưng của cửa sổ nhận trong giao thức TCP:

1. TCP cho phép tiến trình nhận lấy dữ liệu với tốc độ của riêng của nó. Điều này có nghĩa rằng một phần của bộ đệm tại phía nhận có thể bị chiếm bởi byte đã được nhận và được báo nhận, nhưng đang chờ đợi để được tiến trình nhận tiếp nhận. Kích thước cửa sổ nhận là sau đó luôn luôn nhỏ hơn hoặc bằng kích thước bộ đệm. Kích thước cửa sổ nhận xác định số byte mà cửa sổ nhận có thể chấp nhận từ phía gửi trước khi bị quá tải (điều khiển luồng). Nói cách khác, kích thước cửa sổ, thường được gọi là *rwnd*, có thể được xác định như sau:

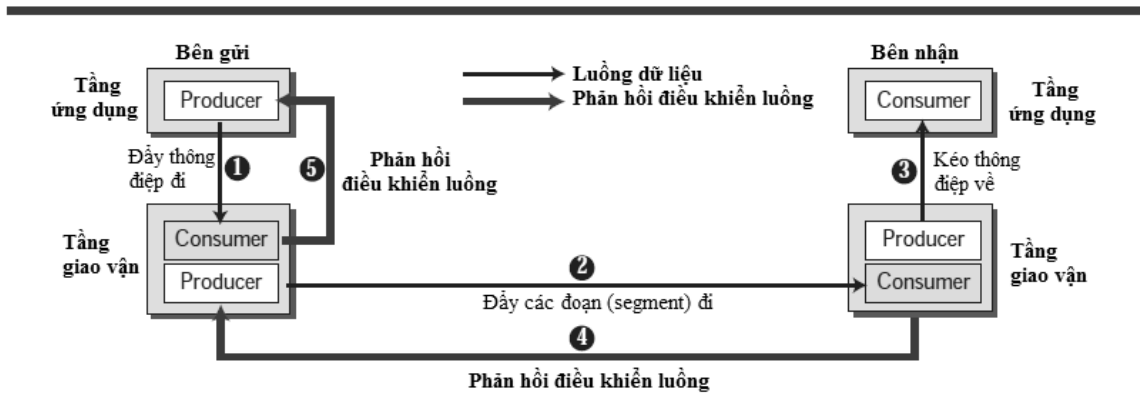
$$rwnd = \text{kích thước bộ đệm} - \text{số các byte đang đợi để được lấy về}$$

2. Cơ chế báo nhận chính trong giao thức TCP là báo nhận tích lũy, cho biết số byte tiếp theo mong muốn nhận được. Tuy nhiên, các phiên bản mới của giao thức TCP sử dụng cả hai cách báo nhận tích lũy và chọn lọc.

3.3.7 Điều khiển luồng

Điều khiển luồng cân bằng tỷ lệ giữa bên gửi và bên nhận dữ liệu. TCP phân biệt điều khiển luồng với kiểm soát lỗi. Trong phần này, chúng ta thảo luận về điều khiển luồng, bỏ qua kiểm soát lỗi. Ở đây tạm thời giả sử là kênh logic giữa bên TCP gửi và bên TCP nhận là không có lỗi.

Hình 3.11 trình bày việc truyền dữ liệu theo một chiều giữa bên gửi và bên nhận; truyền dữ liệu hai chiều có thể được suy diễn ra.



Hình 3.11 Luồng dữ liệu và phản hồi điều khiển luồng trong TCP

Hình vẽ cho thấy dữ liệu đi từ tiến trình gửi xuống TCP gửi, từ TCP gửi đến TCP nhận, và từ TCP nhận đến tiến trình nhận (đường 1, 2, và 3). Tuy nhiên, điều khiển luồng thông tin phản hồi đi từ TCP nhận tới TCP gửi và từ TCP gửi đến tiến trình gửi (đường 4 và 5). Hầu hết các cài đặt của TCP không cung cấp điều khiển luồng thông tin phản hồi từ tiến trình nhận tới TCP gửi; chúng để cho tiến trình nhận lấy dữ liệu từ TCP nhận bất cứ khi nào sẵn sàng. Nói cách khác, nhận TCP điều khiển TCP gửi; TCP gửi điều khiển tiến trình gửi.

Điều khiển luồng thông tin phản hồi từ TCP gửi đến tiến trình gửi (đường 5) đạt được đơn giản bằng cách từ chối dữ liệu từ TCP gửi khi cửa sổ của nó đầy. Điều này có nghĩa là thảo luận về điều khiển luồng ở đây tập trung vào các thông tin phản hồi được gửi từ TCP nhận tới TCP gửi (đường dẫn 4).

a. Việc mở và đóng cửa sổ

Để đạt được điều khiển luồng, TCP buộc bên gửi và bên nhận phải tự điều chỉnh kích thước cửa sổ, mặc dù kích thước của bộ đệm của cả hai bên là cố định khi kết nối được thiết lập. Cửa sổ nhận đóng (di chuyển bức tường bên trái sang bên phải) khi thêm các byte đến từ bên gửi; nó sẽ mở ra (di chuyển bức tường bên phải sang bên phải) khi các byte được lấy về từ tiến trình. Chúng tôi giả sử là nó không co lại (bức tường bên phải không di chuyển sang trái).

Việc mở, đóng và thu hẹp của cửa sổ gửi được điều khiển bởi bên nhận. Cửa sổ gửi đóng (di chuyển bức tường bên trái sang bên phải) khi một báo nhận mới cho phép nó làm như vậy. Cửa sổ gửi mở ra (di chuyển bức tường bên phải sang bên phải) khi kích thước cửa sổ nhận (*rwnd*) được cho biết bởi bên nhận cho phép nó làm như vậy. Ở đây giả sử là không xảy ra trường hợp cửa sổ co lại.

b. Việc co cửa sổ

Như đã nói trước đây, cửa sổ nhận không thể co lại. Nhưng cửa sổ gửi có thể giảm nếu bên nhận xác định một giá trị cho *rwnd* mà dẫn đến kết quả là thu hẹp cửa sổ. Một số cài đặt không cho phép thu hẹp cửa sổ gửi. Hạn chế không cho phép các bức tường bên phải của cửa sổ gửi di chuyển sang trái. Nói cách khác, bên nhận cần phải giữ mối quan hệ như sau giữa báo nhận mới và cuối cùng với giá trị *rwnd* mới và cuối cùng để ngăn chặn sự co lại của cửa sổ gửi:

$$ackNo\ mới + rwnd\ mới \geq ackNo\ cuối + rwnd\ cuối$$

Phía bên trái của bất đẳng thức là vị trí mới của bức tường bên phải trong miền không gian số thứ tự; phía bên phải là vị trí cũ của bức tường bên phải. Mỗi quan hệ cho thấy bức tường bên phải không nên di chuyển sang trái. Bất đẳng thức là một nhiệm vụ cho bên nhận để kiểm tra thông báo của nó. Tuy nhiên, lưu ý là bất đẳng thức chỉ có hiệu lực nếu $S_f < S_n$; tất cả các tính toán được thực hiện trong mô đun 2^{32} .

3.3.8 Kiểm soát lỗi trong TCP

TCP là một giao thức tầng giao vận truyền tin cậy. Điều này có nghĩa là chương trình ứng dụng phân phối dòng dữ liệu dựa trên giao thức TCP cần phân phối toàn bộ dòng đến chương trình ứng dụng ở đầu bên kia theo thứ tự, mà không có lỗi, và không có bất kỳ phần nào bị mất hoặc nhân đôi.

TCP cung cấp tính tin cậy bằng cách kiểm soát lỗi. Kiểm soát lỗi bao gồm các cơ chế phát hiện và gửi lại các đoạn bị hỏng, gửi lại các đoạn bị mất, lưu trữ các đoạn ngoài thứ tự cho đến khi đoạn mất tích đến, và phát hiện và loại bỏ các đoạn nhân đôi. Kiểm soát lỗi trong giao thức TCP đạt được thông qua việc sử dụng ba công cụ đơn giản: checksum, báo nhận, và ngưỡng thời gian (time-out).

a. Checksum

Mỗi đoạn bao gồm một trường kiểm tra, được sử dụng để kiểm tra xem một đoạn có bị hỏng hay không. Nếu đoạn bị hỏng, như bị xóa bởi một checksum không hợp lệ, thì đoạn này sẽ bị loại bỏ TCP đích và được coi là bị mất. TCP sử dụng checksum 16 bit là bắt buộc trong tất cả các đoạn.

b. Báo nhận

TCP sử dụng báo nhận để xác nhận về việc tiếp nhận đoạn dữ liệu. Các đoạn điều khiển không mang dữ liệu, nhưng vẫn sử dụng một số thứ tự, cũng được báo nhận. Đoạn ACK không bao giờ được báo nhận. Trong quá khứ, TCP chỉ sử dụng một loại báo nhận là *báo nhận tích lũy*. Ngày nay, một số cài đặt TCP cũng sử dụng *báo nhận có chọn lọc*.

- **Báo nhận tích lũy (ACK):** TCP được thiết kế để báo nhận đã nhận được các đoạn tích lũy. Bên nhận thông báo byte tiếp theo dự kiến sẽ nhận được, bỏ qua tất cả các đoạn đã nhận và được lưu trữ trong thứ tự. Điều này đôi khi được gọi là tích lũy báo nhận tích cực hay ACK. Từ "tích cực" chỉ ra rằng không có thông tin phản hồi nào được cung cấp cho các đoạn bị loại bỏ, bị mất, hoặc trùng lặp. Các trường ACK 32 bit trong tiêu đề TCP được sử dụng cho báo nhận tích lũy và giá trị của nó chỉ có hiệu lực khi bit cờ ACK được thiết lập là 1.
- **Báo nhận chọn lọc (SACK):** Ngày càng có nhiều cài đặt có thêm báo nhận có chọn lọc hay SACK. SACK không thay thế ACK, nhưng báo cáo thông tin bổ sung cho phía gửi. SACK báo cáo một khối dữ liệu là không đúng thứ tự, và cũng là một đoạn được nhân đôi, tức là nhận được nhiều hơn một lần. Tuy nhiên, vì không có quy định trong tiêu đề TCP để thêm loại thông tin này, nên SACK được cài đặt như một tùy chọn ở phần cuối của TCP header.

c. Tạo báo nhận

Khi nào thì bên nhận tạo báo nhận? Trong quá trình phát triển của TCP, có một vài quy tắc đã được định nghĩa và được sử dụng bởi một số cài đặt. Phần sau sẽ trình bày các quy tắc phổ biến nhất (thứ tự của các quy tắc không xác định tầm quan trọng của nó):

1. Khi đầu cuối A gửi một đoạn dữ liệu đến đầu cuối B, nó phải bao gồm (chứa) một báo nhận cung cấp số thứ tự tiếp theo dự kiến sẽ nhận được. Quy định này làm giảm số lượng các đoạn cần thiết và do đó làm giảm lưu lượng truy cập.
2. Khi bên nhận không có dữ liệu để gửi và nhận được một đoạn đúng theo trật tự (với số thứ tự dự kiến) và đoạn trước đã được báo nhận, thì bên nhận sẽ trì hoãn việc gửi một đoạn ACK cho đến khi đoạn khác đến hoặc đến khi hết một khoảng thời gian (thường là 500 ms). Nói cách khác, bên nhận cần phải trì hoãn việc gửi đoạn ACK nếu chỉ nhận được một đoạn theo đúng trật tự. Quy định này làm giảm lưu lượng đoạn ACK.
3. Khi một phân đoạn đến với số thứ tự được dự kiến bởi bên nhận, và đoạn trước đó theo đúng trật tự vẫn chưa được báo nhận, thì bên nhận ngay lập tức gửi một đoạn ACK. Nói cách khác, bất cứ khi nào cũng không nên để có nhiều hơn hai đoạn đến đúng trật tự mà không được báo nhận. Điều này ngăn chặn việc truyền lại các đoạn không cần thiết mà có thể dẫn đến tắc nghẽn trong mạng.
4. Khi một đoạn đến với một số thứ tự không đúng trật tự và cao hơn số dự kiến, bên nhận ngay lập tức gửi một đoạn ACK thông báo số thứ tự của đoạn dự kiến tiếp theo. Điều này dẫn đến việc *truyền lại nhanh chóng* các đoạn bị mất.
5. Khi một đoạn mất đến, bên nhận sẽ gửi một đoạn ACK để thông báo số thứ tự tiếp theo dự kiến. Điều này cho bên nhận biết đã nhận được đoạn bị thiếu.
6. Nếu một đoạn trùng lặp đến, bên nhận sẽ loại bỏ đoạn này, nhưng ngay lập tức sẽ gửi đi một báo nhận cho biết đoạn đúng trật tự tiếp theo dự kiến. Điều này giải quyết một số vấn đề khi một đoạn ACK tự biến mất.

d. Truyền lại

Trung tâm của cơ chế kiểm soát lỗi là truyền lại các đoạn. Khi một đoạn được gửi đi, nó được lưu trữ trong một hàng đợi cho đến khi nó được báo nhận. Khi bộ đếm thời gian truyền lại hết giờ hoặc khi bên gửi nhận được ba ACK trùng lặp đối với đoạn đầu tiên trong hàng đợi, đoạn đó sẽ được truyền lại.

- **Truyền lại sau khi RTO:** TCP gửi chứa một tham số là *ngưỡng thời gian truyền lại* (*retransmission time-out, RTO*) cho mỗi kết nối. Khi hết ngưỡng thời gian TCP sẽ gửi đoạn vào phần đầu của hàng đợi (đoạn có số thứ tự nhỏ nhất) và khởi động lại bộ đếm thời gian. Lưu ý ở đây giả sử là $S_f < S_n$. Phiên bản này của TCP đôi khi được gọi là *Tahoe*. Giá trị của RTO thay đổi trong TCP và được cập nhật dựa trên *round-trip time - RTT*. RTT là thời gian đoạn đi từ nguồn đến đích và có báo nhận trở lại.
- **Truyền lại sau khi nhận được ba đoạn ACK trùng lặp:** Nguyên tắc trước về truyền lại đoạn là đủ nếu giá trị của RTO không lớn. Để cho phép bên gửi để truyền lại sớm

hơn so với ngưỡng thời gian, hầu hết các cài đặt ngày nay cho phép thực hiện theo quy tắc *ba ACK trùng lặp* và truyền lại các đoạn bị mất ngay lập tức. Tính năng này được gọi là *truyền lại nhanh*, và phiên bản TCP sử dụng tính năng này được gọi là *Reno*. Trong phiên bản này, nếu ba báo nhận trùng lặp (có nghĩa là, một ACK bản gốc và ba bản sao chính xác giống hệt nhau) đến cho một đoạn, thì đoạn tiếp theo được truyền lại mà không đợi hết ngưỡng thời gian.

- **Đoạn đến ngoài thứ tự (*Out-of Order Segments*):** Ngày nay, các cài đặt TCP sẽ không loại bỏ các đoạn đến ngoài thứ tự, mà sẽ lưu chúng lại tạm thời và đánh dấu chúng là các đoạn *out-of-order* cho đến khi các đoạn còn thiếu đến. Tuy nhiên, cần lưu ý là các đoạn *out-of-order* không bao giờ được phân phối cho tiến trình. TCP phải đảm bảo rằng dữ liệu được phân phối cho tiến trình phải theo đúng thứ tự.

3.3.9 Điều khiển tắc nghẽn trong TCP

Điều khiển tắc nghẽn trong TCP dựa trên cả hai kỹ thuật vòng mở (*open-loop*) và vòng đóng (*closed-loop*). TCP sử dụng cửa sổ tắc nghẽn và chính sách tắc nghẽn để tránh tắc nghẽn, và phát hiện và giảm bớt tắc nghẽn sau khi nó đã xảy ra.

3.3.9.1 Cửa sổ tắc nghẽn

Trước đây, như đã thảo luận trong phần điều khiển luồng và các giải pháp khi bên nhận bị quá tải dữ liệu, chúng ta thấy rằng kích thước cửa sổ gửi được xác định bởi không gian bộ đệm có sẵn trong bên nhận (*rwnd*). Nói cách khác, ở đây chúng ta đã giả sử là chỉ có bên nhận mới có thể ra lệnh cho bên gửi về kích thước cửa sổ của bên gửi. Việc này hoàn toàn bỏ qua một thực thể là môi trường mạng. Nếu mạng không thể phân phối dữ liệu nhanh như nó được tạo ra bởi bên gửi, thì nó phải báo cho bên gửi làm chậm lại. Nói cách khác, ngoài bên nhận, mạng là một thực thể thứ hai để xác định kích thước của cửa sổ của bên gửi.

Bên gửi có hai thông tin: kích thước cửa sổ được thông báo cho bên gửi và kích thước cửa sổ tắc nghẽn. Kích thước thực sự của cửa sổ là giá trị nhỏ hơn trong hai giá trị này.

$$\text{Kích thước cửa sổ thực tế} = \min(\text{rwnd}, \text{cwnd})$$

Vấn đề ở đây là cần phải xác định kích thước của cửa sổ tắc nghẽn (*cwnd*) trong khoảng thời gian ngắn như nào.

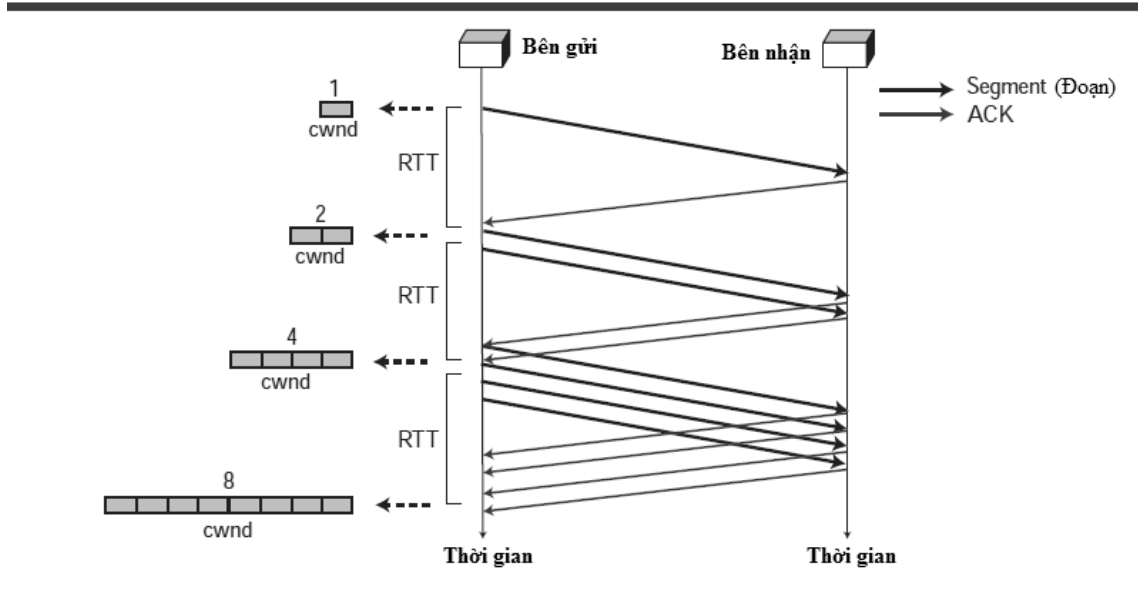
3.3.9.2 Chính sách tắc nghẽn

Chính sách chung của TCP cho việc xử lý tắc nghẽn dựa trên ba giai đoạn: *khởi động chậm*, *tránh tắc nghẽn*, và *phát hiện tắc nghẽn*. Trong giai đoạn khởi động chậm, bên gửi bắt đầu với một tốc độ truyền chậm, nhưng sẽ tăng tốc độ lên rất nhanh để đạt được một ngưỡng. Khi đã chạm ngưỡng, tỷ lệ tăng sẽ được giảm dần. Cuối cùng, nếu tắc nghẽn được phát hiện, bên gửi sẽ trở lại giai đoạn khởi động hoặc tránh tắc nghẽn, dựa trên tình trạng tắc nghẽn được phát hiện.

a. Khởi động chậm: tăng theo cấp số mũ

Thuật toán khởi động chậm dựa trên ý tưởng là kích thước của cửa sổ tắc nghẽn (*cwnd*) được bắt đầu với một *kích thước đoạn tối đa* (*maximum segment size, MSS*). MSS được xác định trong quá trình thiết lập kết nối bằng cách sử dụng một tùy chọn cùng tên. Kích thước

của cửa sổ tăng 1 MSS mỗi lần có một báo nhận đến. Như tên của nó, các thuật toán bắt đầu từ từ, nhưng phát triển theo cấp số mũ. Để thể hiện ý tưởng hãy nhìn vào hình 3.12. Ở đây, giả sử là $rwnd$ lớn hơn $cwnd$, do đó kích thước cửa sổ gửi luôn luôn bằng $cwnd$. Để đơn giản, chúng ta bỏ qua chính sách ACK bị trễ mà cho rằng mỗi đoạn được báo nhận một cách riêng lẻ.



Hình 3.12 Khởi động chậm, tăng theo cấp số mũ

Bên gửi bắt đầu với $cwnd = 1$ MSS. Điều này có nghĩa là bên gửi chỉ có thể gửi một đoạn. Sau khi ACK đầu tiên đến, kích thước của cửa sổ tắc nghẽn là tăng 1, có nghĩa là $cwnd$ giờ là 2. Lúc này hai đoạn có thể được gửi đi. Khi hai ACK đến, kích thước của cửa sổ được tăng lên 1 MSS cho mỗi ACK, có nghĩa là $cwnd$ hiện tại sẽ là 4. Như vậy 4 đoạn có thể được gửi đi. Khi bốn ACK đến, kích thước của cửa sổ tăng thêm 4, có nghĩa là $cwnd$ bây giờ là 8.

Nếu nhìn vào kích thước của $cwnd$ theo thời gian RTT, thì thấy rằng tốc độ tăng trưởng theo cấp số mũ như sau:

Bắt đầu	→	$cwnd = 1$
Sau 1 RTT	→	$cwnd = 1 \times 2 = 2 \rightarrow 2^1$
Sau 2 RTT	→	$cwnd = 2 \times 2 = 4 \rightarrow 2^2$
Sau 3 RTT	→	$cwnd = 4 \times 2 = 8 \rightarrow 2^3$

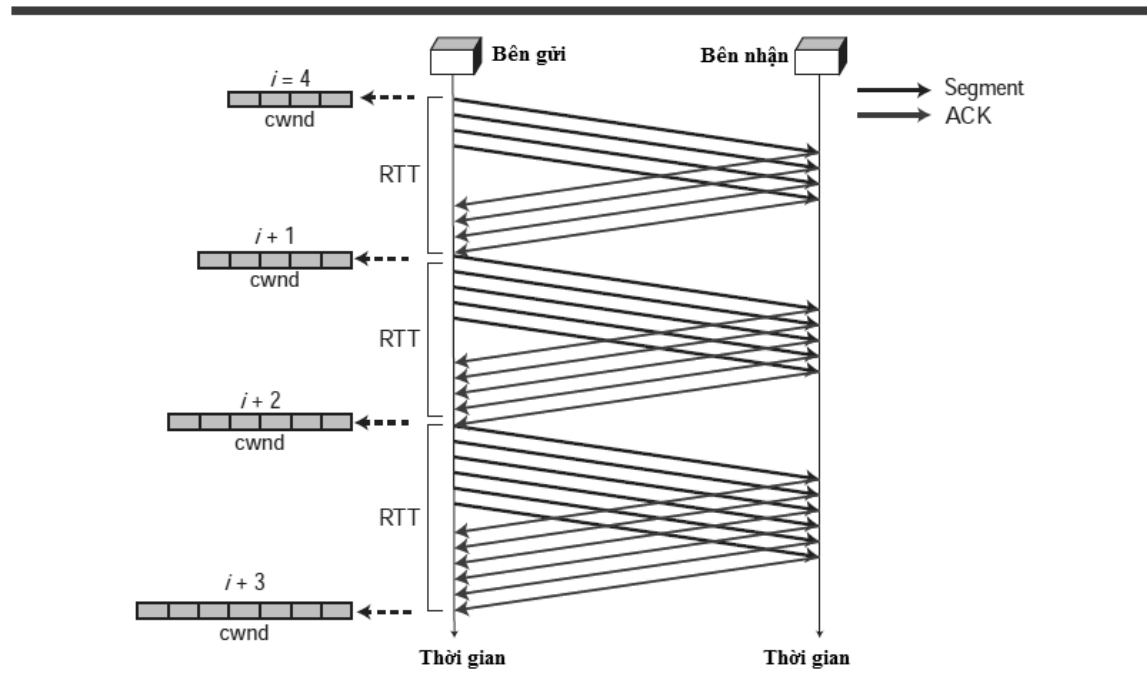
Tuy nhiên, chiến lược khởi động chậm là chậm hơn trong trường hợp báo nhận bị trễ. Cần nhớ rằng, đối với mỗi ACK, $cwnd$ chỉ tăng thêm 1 MSS. Do đó, nếu ba đoạn được báo nhận tích lũy thì kích thước của $cwnd$ chỉ tăng thêm 1 MSS, chứ không phải là 3 MSS. Sự tăng trưởng vẫn theo cấp số mũ, nhưng nó không tăng gấp 2 nữa. Với một ACK cho mỗi 2 đoạn, thì tăng trưởng là gần 1,5 lần.

Khởi động chậm không thể tiếp tục vô thời hạn mà cần phải có một ngưỡng để dừng lại quá trình này. Bên gửi theo dõi một biến có tên là $ssthresh$ (ngưỡng khởi động chậm). Khi

kích thước của cửa sổ theo byte đạt đến ngưỡng này, khởi động chậm dừng lại và bắt đầu giai đoạn tiếp theo.

b. Tránh tắc nghẽn: Tăng theo cấp số cộng

Nếu bắt đầu với các thuật toán khởi động chậm, kích thước của cửa sổ tắc nghẽn sẽ tăng theo cấp số mũ. Để tránh ùn tắc trước khi nó xảy ra, người ta phải làm chậm tốc độ tăng trưởng theo cấp số nhân này. TCP định nghĩa một thuật toán gọi là *tránh tắc nghẽn*, làm tăng cwnd theo cấp số cộng thay vì tăng theo cấp số mũ, xem hình 3.13.



Hình 3.13 Tránh tắc nghẽn, tăng theo cấp số cộng

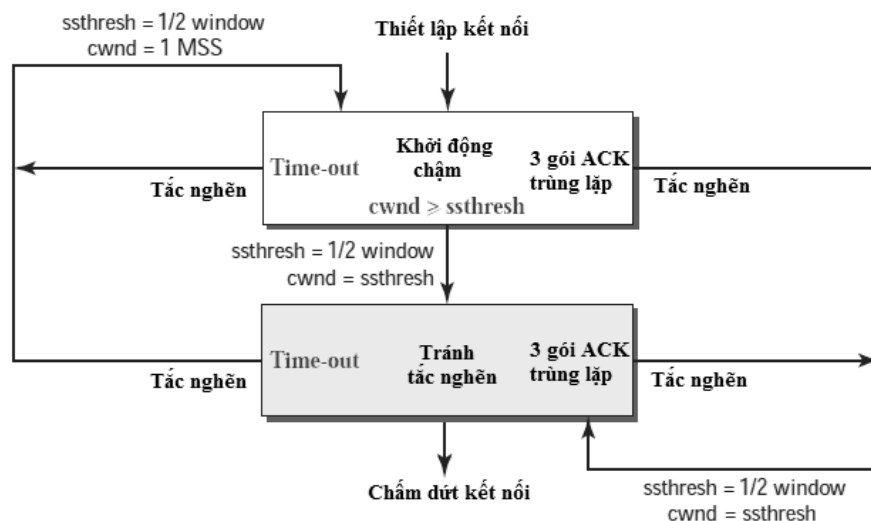
Khi kích thước của cửa sổ tắc nghẽn đạt đến ngưỡng khởi động chậm tại $cwnd = i$, thì giai đoạn khởi động chậm sẽ dừng lại và giai đoạn tăng theo cấp số cộng sẽ bắt đầu. Trong thuật toán này, mỗi lần toàn bộ "cửa sổ" của các đoạn được báo nhận, thì kích thước cửa sổ tắc nghẽn sẽ tăng thêm 1. Cửa sổ là số lượng các đoạn được truyền trong RTT. Nói cách khác, sự gia tăng dựa trên RTT, không phải dựa trên số ACK đến. Để thể hiện ý tưởng, chúng ta áp dụng thuật toán này với kịch bản tương tự như khởi động chậm. Trong trường hợp này, sau khi bên gửi nhận được báo nhận cho toàn bộ một kích thước cửa sổ các đoạn, thì kích thước cửa sổ sẽ được tăng thêm một đoạn. Kích thước của cwnd theo RTT tăng theo cấp số cộng như sau:

Bắt đầu	→	$cwnd = i$
Sau 1 RTT	→	$cwnd = i + 1$
Sau 2 RTT	→	$cwnd = i + 2$
Sau 3 RTT	→	$cwnd = i + 3$

c. Phát hiện tắc nghẽn: Giảm theo cấp số nhân

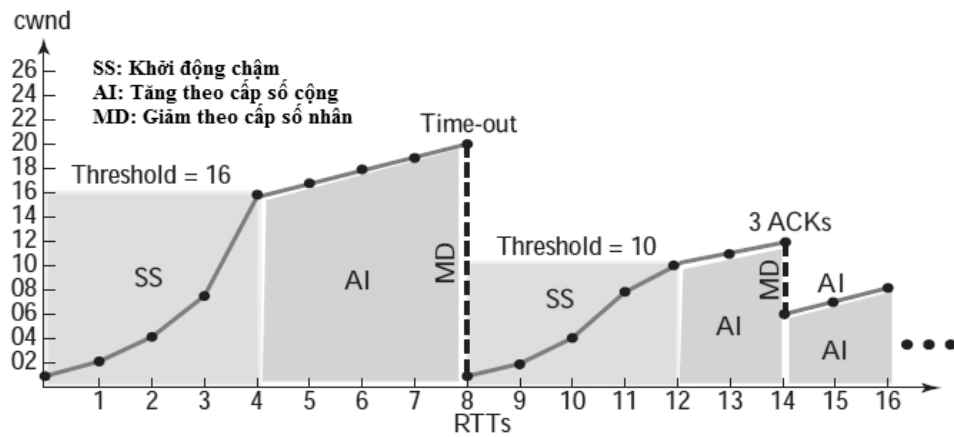
Nếu tắc nghẽn xảy ra, kích thước cửa sổ tắc nghẽn cần phải được giảm đi. Cách duy nhất để bên gửi có thể dự đoán tắc nghẽn xảy ra là truyền lại một đoạn. Đây là một giả định quan trọng được thực hiện bởi TCP. Việc truyền lại là cần thiết để khôi phục một gói tin bị thiếu mà đã được coi là bị mất của một bộ định tuyến mà có rất nhiều các gói tin được gửi đến. Nếu bộ định tuyến hoặc mạng trở nên quá tải hoặc tắc nghẽn thì sẽ bỏ qua các đoạn đi đến. Tuy nhiên, truyền lại có thể xảy ra một trong hai trường hợp: bộ định thời RTO hết thời gian hoặc khi nhận được ba ACK trùng lặp. Trong cả hai trường hợp, kích thước của các ngưỡng được giảm xuống còn một nửa (giảm theo cấp số nhân). Hầu hết các cài đặt TCP có hai phản ứng:

1. Nếu time-out xảy ra, thì khả năng lớn là sẽ xảy ra tắc nghẽn; một đoạn có thể bị bỏ rơi trong mạng và không có tin tức gì về các đoạn được gửi sau. Trong trường hợp này TCP phản ứng như sau:
 - ✓ Thiết lập giá trị ngưỡng về bằng một nửa kích thước cửa sổ hiện tại.
 - ✓ Giảm cwnd trở về còn một đoạn.
 - ✓ Bắt đầu lại giai đoạn khởi động chậm.
2. Nếu nhận được ba ACK trùng lặp, thì khả năng xảy ra tắc nghẽn ít hơn; một đoạn có thể bị bỏ rơi, nhưng một số đoạn sau vẫn đến đích an toàn do vẫn nhận được ba ACK trùng lặp. Điều này được gọi là truyền tải nhanh và phục hồi nhanh. Trong trường hợp này, TCP có phản ứng như sau:
 - ✓ Thiết lập giá trị ngưỡng về bằng một nửa kích thước cửa sổ hiện tại.
 - ✓ Đặt cwnd bằng giá trị ngưỡng (một số cài đặt thêm kích cỡ ba đoạn vào ngưỡng).
 - ✓ Bắt đầu giai đoạn tránh tắc nghẽn.



Hình 3.14 Tóm tắt chính sách quản lý tắc nghẽn trong TCP

Hình 3.14 trình bày tóm tắt chính sách quản lý tắc nghẽn của TCP và mối quan hệ giữa ba giai đoạn. Hình 3.15 là một ví dụ tắc nghẽn với kích thước cửa sổ lớn nhất khởi tạo là 32 đoạn. Ngưỡng được khởi tạo là 16 đoạn (một nửa kích thước cửa sổ lớn nhất).



Hình 3.15 Ví dụ tắc nghẽn

CHƯƠNG 4

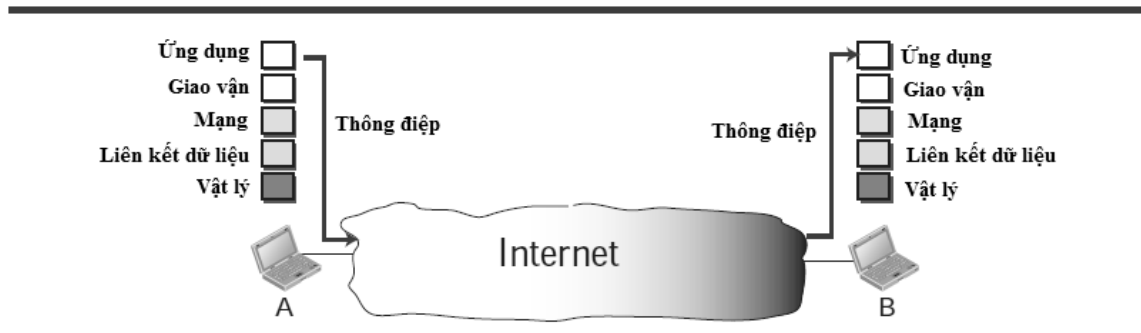
CÁC GIAO THỨC TẦNG MẠNG

Tầng mạng cung cấp nhiều dịch vụ khác nhau để thực thi chức năng của nó. Tương ứng với các dịch vụ là các giao thức. Chương này sẽ trình bày các vấn đề sau:

- Giao thức đánh địa chỉ Internet phiên bản 4 và phiên bản 6 (phiên bản 5 chưa bao giờ được thực hiện).
- Giao thức thông điệp điều khiển Internet ICMP phiên bản 4 và phiên bản 6.
- Giao thức phân giải địa chỉ ARP.
- Các giao thức định tuyến trong mạng như RIP và OSPF.

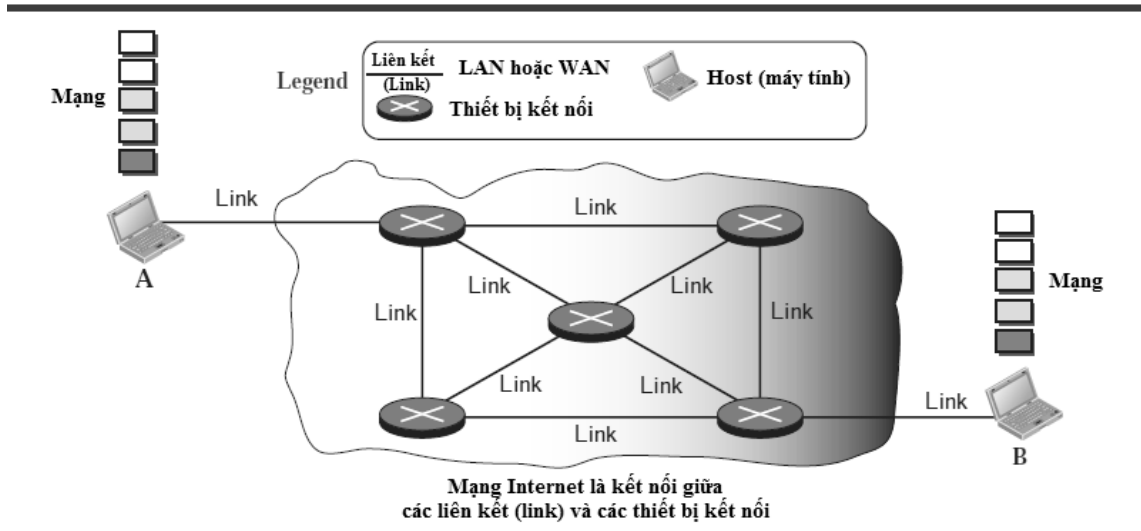
4.1 GIỚI THIỆU VỀ TẦNG MẠNG

Mạng Internet toàn cầu là một mạng lưới kết nối hàng tỷ máy tính trên thế giới lại với nhau. Khi đó, người dùng chỉ cần quan tâm đến việc một thông điệp từ tầng ứng dụng tại một máy tính này đi đến tầng ứng dụng của một máy tính khác (xem hình 4.1).



Hình 4.1. Mạng Internet như là một hộp đen

Tuy nhiên, Internet không phải là chỉ một mạng đơn lẻ mà nó được tạo thành từ nhiều mạng liên kết với nhau qua các thiết bị kết nối. Nói cách khác, Internet là một liên mạng, là tổ hợp của các mạng LAN và WAN. Để hiểu rõ hơn về vai trò của tầng mạng (hay tầng liên mạng), cần hiểu rõ hơn về các mạng LAN và WAN. Do không thể thể hiện tất cả các mạng này, nên ở đây chỉ đưa ra một ví dụ mô tả một mạng Internet nhỏ với một số mạng và các thiết bị kết nối (hình 4.2).



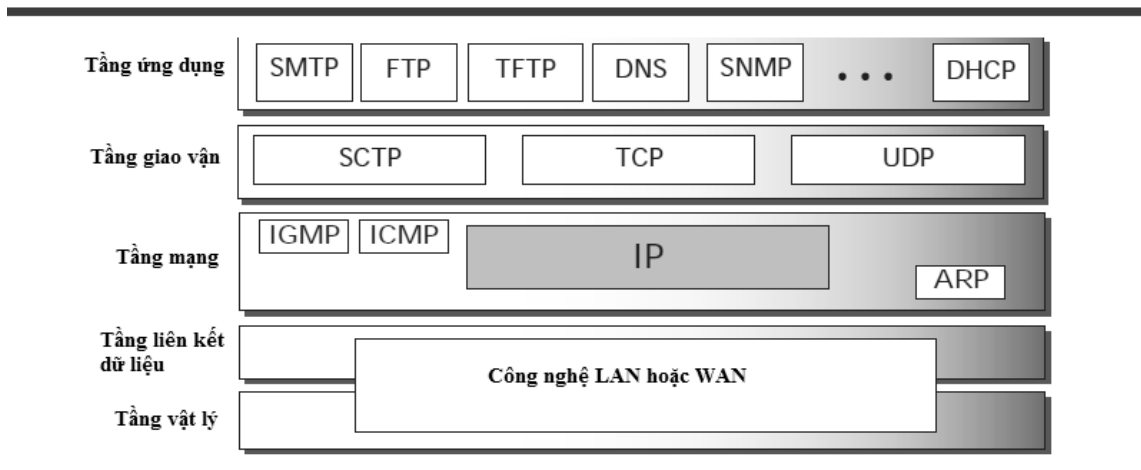
Hình 4.2 Internet là một kết nối của các mạng LAN và WAN lại với nhau

Trong mô hình này, thiết bị kết nối là một bộ định tuyến hoạt động giống như một chuyển mạch (*switch*). Khi một gói tin đến một trong các cổng (giao diện) của nó, thì gói tin này sẽ được chuyển qua cổng khác để tới chuyển mạch tiếp theo (hay đích cuối). Nói cách khác thì một tiến trình được gọi là một chuyển mạch diễn ra tại thiết bị kết nối.

4.2 IPv4 VÀ IPv6

4.2.1 Giao thức IPv4

Giao thức Internet (IP) là phương thức truyền được sử dụng bởi chồng giao thức TCP/IP tại tầng mạng. Hình 4.3 thể hiện vị trí của IP trong chồng giao thức.



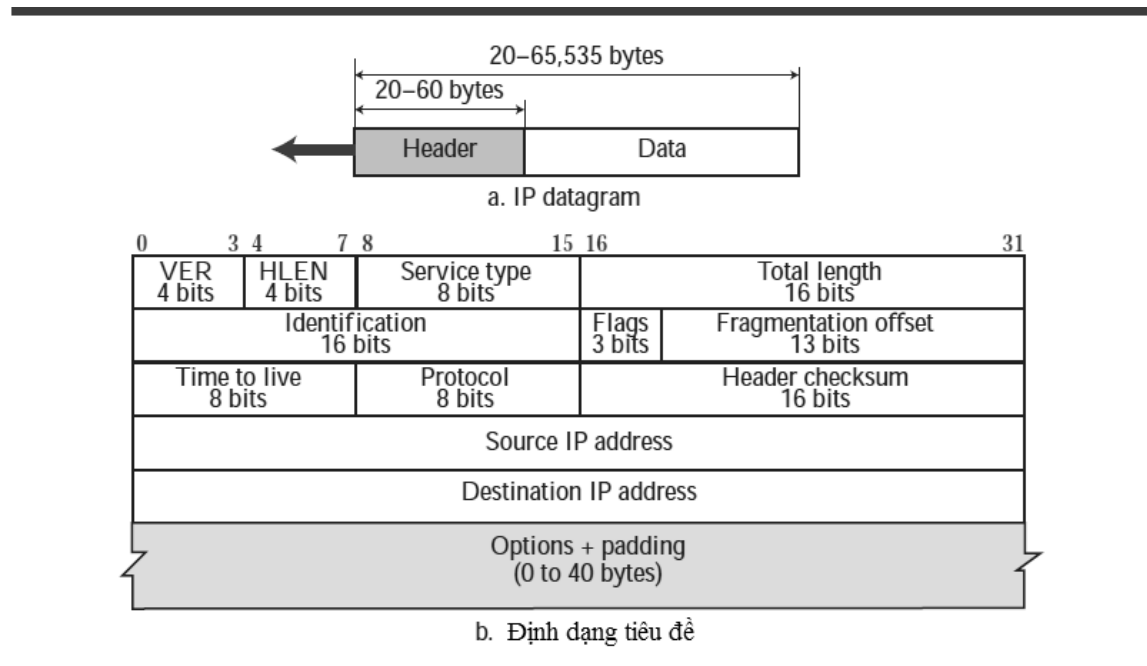
Hình 4.3. Vị trí của IP trong chồng giao thức TCP/IP

IP là một giao thức gói tin không tin cậy và không liên kết – một dịch vụ *vận chuyển với cố gắng tối đa*. Thuật ngữ cố gắng tối đa có nghĩa là các gói tin IP có thể bị hỏng, bị mất, đến không theo thứ tự, hoặc bị trì hoãn và có thể tạo ra tắc nghẽn trong mạng. Nếu tính tin cậy là quan trọng thì IP phải hoạt động cùng với một giao thức tin cậy như TCP.

IP cũng là một giao thức không liên kết trong mạng chuyển mạch gói. Điều này có nghĩa là mỗi gói tin được xử lý độc lập, và mỗi gói tin cũng có thể đi theo các tuyến đường khác nhau để tới đích. Các gói tin được gửi từ cùng một nguồn và tới cùng một đích có thể đến không theo thứ tự. Thêm nữa, gói tin cũng có thể bị hỏng hoặc mất trong quá trình truyền tin. Như vậy, IP lại phải dựa vào một giao thức mức cao hơn để xử lý các vấn đề này.

4.2.1.1 Các gói tin (datagram)

Các gói tin trong tầng mạng (Internet) được gọi là datagram. Hình 4.4 mô tả định dạng của một IP datagram. Một datagram là một gói tin có độ dài thay đổi và bao gồm 2 phần: phần tiêu đề và phần dữ liệu. Phần tiêu đề dài khoảng 20 đến 60 byte, chứa các thông tin cần thiết cho định tuyến và truyền dữ liệu.

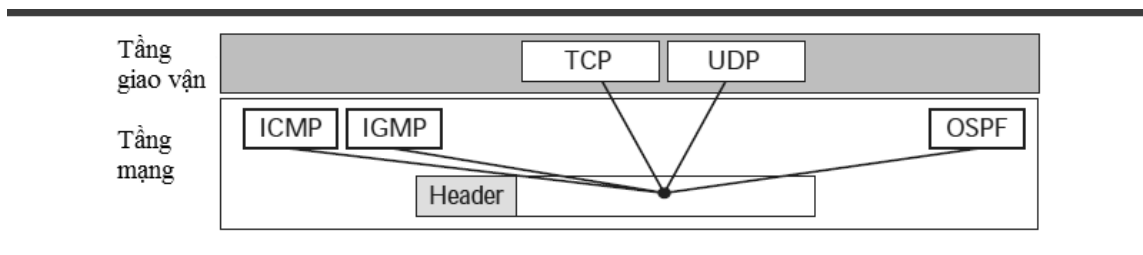


Hình 4.4 IP datagram

Các trường được mô tả như sau:

- **VER (Phiên bản):** Trường 4 bit mô tả phiên bản giao thức IP.
- **HLEN (Độ dài phần tiêu đề):** Trường 4 bit mô tả tổng độ dài của phần tiêu đề gói tin trong các dòng 4 byte.
- **Service type (Kiểu dịch vụ):** Mô tả cách gói tin cần được xử lý thế nào. Một phần dùng để xác định độ ưu tiên và phần còn lại định nghĩa kiểu dịch vụ (độ trễ thấp, thông lượng cao, v.v.). IETF đã thay đổi cách sử dụng trường 8 bit này với việc dùng nó để định nghĩa một tập các dịch vụ phân biệt.
- **Total Length (Tổng độ dài):** Trường 16 bit này xác định độ dài tổng (cả phần tiêu đề và dữ liệu) của gói tin theo đơn vị byte. Độ dài dữ liệu được xác định sau khi bỏ đi phần độ dài phần tiêu đề. Do độ dài trường này là 16 bit nên tổng độ dài gói tin IP giới hạn là $2^{16}-1$ byte.

- *Identification (Định danh)*: Trường này được sử dụng trong phân mảnh gói tin.
- *Flag (Các cờ)*: Trường này được sử dụng trong phân mảnh gói tin.
- *Fragmentation Offset (Vị trí tương đối của phân mảnh)*: Trường này được sử dụng trong phân mảnh gói tin.
- *Time to live (Thời gian sống)*: Một gói tin có thời gian sống hạn chế khi truyền qua Internet. Trường này cần khi sử dụng trong trường hợp bảng định tuyến bị hỏng và cũng nhằm hạn chế quãng đường truyền của gói tin. Ví dụ, trong trường hợp nguồn muốn hạn chế các gói tin chỉ gửi trong mạng nội bộ, nó thiết lập giá trị 1, khi gói tin đến bộ định tuyến đầu tiên thì trường này giảm xuống 0, và gói tin bị loại bỏ.
- *Protocol (Giao thức)*: Trường 8 bit này xác định giao thức mức cao sử dụng dịch vụ của lớp IP. Một gói tin IP có thể đóng gói dữ liệu từ nhiều giao thức bậc cao như TCP, UDP, ICMP, và IGMP. Trường này chỉ ra giao thức đích cuối cùng mà gói tin IP sẽ được chuyển tới. Nói cách khác do giao thức IP ghép kênh và phân kênh dữ liệu từ các giao thức mức cao khác nhau, giá trị của trường này trợ giúp tiến trình phân kênh khi gói tin đến đích cuối cùng (xem hình 4.5).



Hình 4.5 Ghép kênh dữ liệu

- *Checksum*: Lưu giá trị checksum.
- *Source IP address (Địa chỉ nguồn)*: Trường 32 bit này xác định địa chỉ IP của nguồn. Trường này phải giữ không đổi trong suốt quá trình vận chuyển gói tin IP.
- *Destination IP address (Địa chỉ đích)*: Trường 32 bit này xác định địa chỉ IP của đích. Trường này phải giữ không đổi trong suốt quá trình vận chuyển gói tin IP.

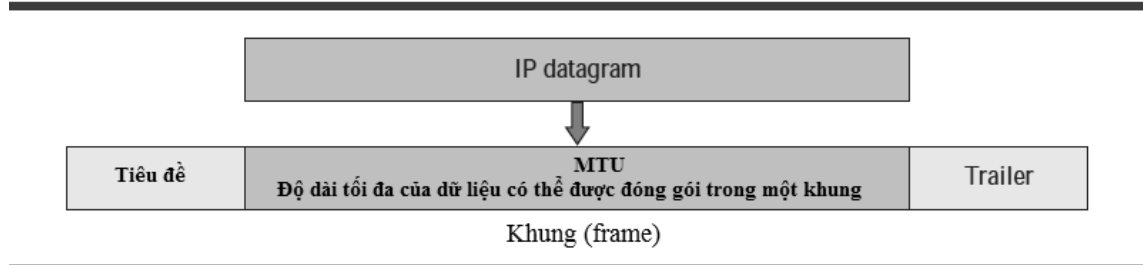
4.2.1.2 Phân mảnh

Một gói tin có thể đi qua các mạng khác nhau. Mỗi bộ định tuyến tách các gói tin IP từ frame (khung) nó nhận, xử lý, và sau đó đóng gói vào trong một frame khác. Định dạng và kích thước của *frame nhận được* phụ thuộc vào giao thức được dùng bởi các mạng vật lý mà frame được chuyển đến. Định dạng và kích thước của *frame gửi đi* phụ thuộc vào giao thức được dùng bởi các mạng vật lý mà frame sẽ được chuyển đi. Ví dụ, nếu một bộ định tuyến kết nối mạng LAN với mạng WAN, nó nhận được một frame trong định dạng LAN và gửi một frame trong định dạng WAN.

a. Maximum Transfer Unit (MTU)

Mỗi giao thức ở tầng liên kết có định dạng frame riêng. Một trong những trường có trong định dạng là *kích thước tối đa của trường dữ liệu (MTU)*. Nói cách khác, khi một gói tin

được đóng gói trong một frame, tổng kích thước của gói tin phải nhỏ hơn kích thước tối đa, được xác định bởi giới hạn của phần cứng và phần mềm sử dụng trong mạng (xem hình 4.6).



Hình 4.6 MTU (Maximum Transfer Unit)

Giá trị của MTU khác nhau trong các giao thức mạng vật lý khác nhau. Ví dụ, giá trị cho mạng LAN Ethernet là 1500 byte, cho FDDI LAN là 4352 byte, và PPP là 296 byte.

Để giao thức IP độc lập đối với mạng vật lý, các nhà thiết kế quy định chiều dài tối đa của gói tin IP là 65.535 byte. Với giao thức có kích thước MTU này, việc truyền tải sẽ hiệu quả hơn. Tuy nhiên, đối với các mạng vật lý khác, cần phải phân chia các gói tin để nó có thể đi qua các mạng này. Quá trình này được gọi là phân mảnh.

Phía nguồn thường không phân mảnh các gói tin IP. Thay vì vậy, tầng giao vận sẽ phân mảnh dữ liệu theo kích thước được cung cấp bởi giao thức IP và tầng liên kết dữ liệu.

Khi một gói tin bị phân mảnh, mỗi mảnh có phần tiêu đề riêng của mình với hầu hết các trường được lặp lại, nhưng có một số trường sẽ thay đổi. Một gói tin phân mảnh tự nó có thể tiếp tục bị phân mảnh nếu đi qua mạng có MTU nhỏ hơn. Nói cách khác, một gói dữ liệu có thể được phân mảnh nhiều lần trước khi nó đạt đến đích cuối cùng.

Gói tin có thể được phân mảnh bởi các host nguồn hoặc bất kỳ bộ định tuyến nào trên đường đi. Tuy nhiên, việc tập hợp lại các gói tin chỉ được thực hiện bởi các host đích vì mỗi mảnh là một gói tin độc lập. Trong khi đó, các gói tin bị phân mảnh có thể đi qua các tuyến đường khác nhau, và không bao giờ có thể kiểm soát hay chắc chắn về việc một gói tin phân mảnh được truyền đi trên một tuyến đường nào đó. Tất cả các phân mảnh thuộc cùng một gói tin cuối cùng sẽ đến host đích. Do vậy, việc hợp các gói tin lại tại điểm đến cuối cùng là hợp lý. Tuy nhiên, việc tập hợp lại các gói tin trong quá trình truyền tải sẽ làm suy giảm hiệu năng của hệ thống.

Khi một gói tin bị phân mảnh, các phần yêu cầu của phần tiêu đề phải được giữ lại trên tất cả các phân mảnh. Các trường tùy chọn có thể có hoặc có thể không được giữ lại. Các host hoặc các bộ định tuyến thực hiện phân mảnh gói tin cần thay đổi giá trị của ba trường: cờ, offset phân mảnh, và chiều dài tổng. Phần còn lại của các trường phải được lưu giữ lại. Tất nhiên, giá trị của checksum phải được tính toán lại sau khi phân mảnh. (*Chú ý là chỉ phần dữ liệu trong gói tin bị phân mảnh*).

b. Các trường liên quan đến việc phân mảnh

Các trường liên quan đến phân mảnh và tập hợp lại các gói tin IP là định danh, cờ, và offset phân mảnh.

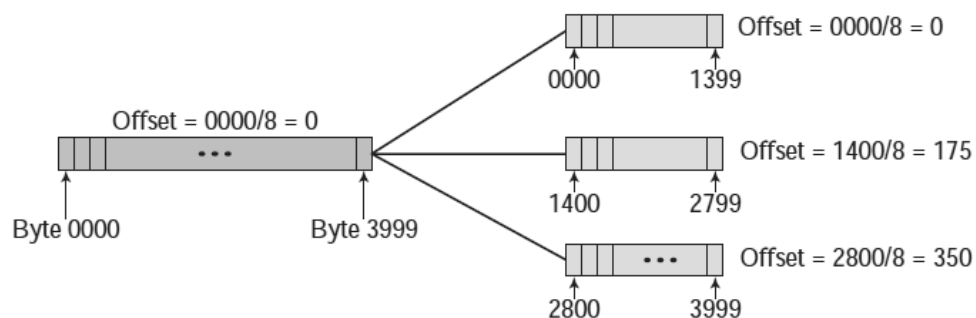
- **Identification (Định danh):** Trường 16 bit này xác định nguồn gốc máy nguồn của một gói tin. Tổ hợp của định danh và địa chỉ IP nguồn phải xác định duy nhất một gói tin khi nó rời đi khỏi máy chủ nguồn. Để đảm bảo tính duy nhất, giao thức IP sử dụng một bộ đếm để gán nhãn cho các gói dữ liệu. Bộ đếm được khởi tạo với một giá trị dương. Khi giao thức IP gửi đi một gói tin, nó sao chép giá trị hiện tại của bộ đếm đến trường định danh và tăng bộ đếm thêm một. Tính duy nhất được đảm bảo khi bộ đếm được lưu giữ trong bộ nhớ chính. Khi một gói tin bị phân mảnh, giá trị trong trường định danh được sao chép vào tất cả các phân mảnh. Nghĩa là, tất cả các phân mảnh có số định danh giống nhau, tương tự như gói tin ban đầu. Số định danh sẽ trợ giúp phía đích trong việc tập hợp lại gói tin: tất cả các phân mảnh có định danh giống nhau sẽ được hợp lại vào một gói.
- **Flag (Cờ):** Đây là một trường ba bit. Bit đầu tiên được dành riêng (không sử dụng). Bit thứ hai được gọi là bit *không được phân mảnh*. Nếu giá trị của nó là 1, máy tính không được phân mảnh gói tin. Nếu gói tin không thể truyền qua bất kỳ mạng vật lý nào, nó sẽ bị loại bỏ và hệ thống sẽ gửi một thông báo lỗi ICMP đến host nguồn. Nếu giá trị của nó là 0, các gói dữ liệu có thể được phân mảnh nếu cần. Bit thứ ba được gọi là *phân mảnh thêm*. Nếu giá trị của nó là 1, thì gói tin không phải là phân mảnh cuối cùng, phía sau còn có nhiều phân mảnh. Nếu giá trị của nó là 0, thì đây là mảnh cuối cùng hoặc phân mảnh duy nhất (xem hình 4.7).

D: Không được phân mảnh
M: Phân mảnh thêm



Hình 4.7 Trường các cờ

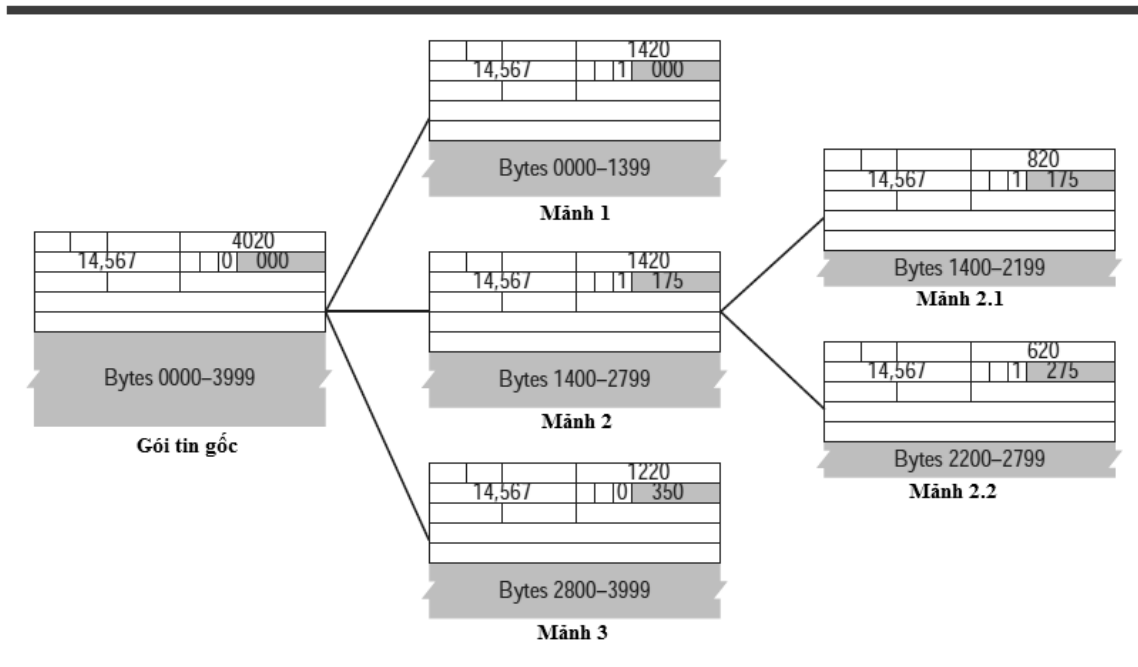
- **Fragmentation Offset (Offset phân mảnh hay Vị trí tương đối của phân mảnh):** Trường 13 bit này cho biết vị trí tương đối của phân mảnh này đối với toàn bộ gói tin. Nó là offset của dữ liệu trong gói tin gốc đo bằng đơn vị 8 byte. Hình 4.8 cho thấy một gói tin với kích thước dữ liệu 4000 byte bị phân thành ba mảnh. Các byte trong gói tin gốc được đánh số từ 0 đến 3999. Phân mảnh đầu tiên mang byte 0 đến 1399. Offset của gói tin này là $0/8 = 0$. Phân mảnh thứ hai mang byte 1400 đến 2799; giá trị offset cho phân mảnh này là $1400/8 = 175$. Cuối cùng, phân mảnh thứ ba mang byte 2800 đến 3999. Giá trị offset cho phân mảnh này là $2800/8 = 350$.



Hình 4.8 Ví dụ phân mảnh

Giá trị của offset được đo bằng đơn vị 8 byte, là do độ dài của trường offset chỉ dài 13 bit và không thể mô tả cho một chuỗi byte lớn hơn 8191. Điều này buộc host hoặc bộ định tuyến gói khi phân mảnh các gói tin phải chọn kích thước của mỗi phân mảnh để có số byte đầu tiên chia hết cho 8.

Hình 4.9 là một mô tả chi tiết hơn về phân mảnh trong hình trước. Giá trị của trường định danh là như nhau trong tất cả các phân mảnh. Đồng thời, chú ý giá trị của trường cờ với tập bit “phân mảnh thêm” cho tất cả các phân mảnh ngoại trừ mảnh cuối cùng. Ngoài ra, giá trị của trường offset cho mỗi phân mảnh cũng được hiển thị.



Hình 4.9 Ví dụ phân mảnh chi tiết

Hình 4.9 cũng cho thấy những gì sẽ xảy ra nếu một phân mảnh tự nó bị phân mảnh tiếp. Trong trường hợp này giá trị của trường offset luôn luôn là tương đối so với gói tin gốc. Ví dụ, trong hình, phân mảnh thứ hai bị phân mảnh tiếp sau đó thành hai phân mảnh 800 byte và 600 byte, nhưng offset lại cho thấy vị trí tương đối của các phân mảnh với gói tin gốc.

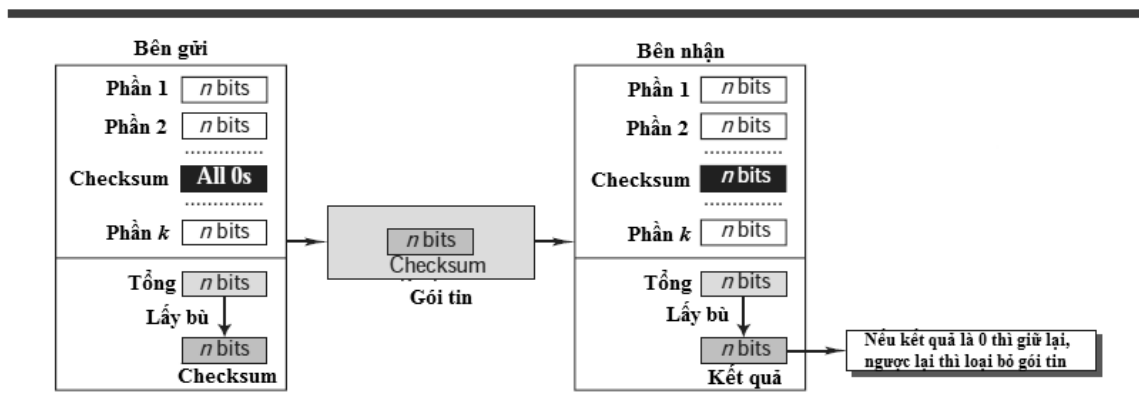
Rõ ràng là ngay cả khi mỗi phân mảnh theo một tuyến đường khác nhau và đến không theo thứ tự, các máy đích cuối cùng vẫn có thể tập hợp lại các gói tin ban đầu từ các phân mảnh đã nhận được (nếu không mảnh nào bị mất) bằng cách sử dụng chiến lược sau đây:

1. Phân mảnh đầu tiên có trường offset bằng 0.
2. Chia chiều dài của phân mảnh đầu tiên cho 8. Phân mảnh thứ hai có giá trị offset bằng với kết quả nhận được.
3. Chia tổng chiều dài của phân mảnh đầu tiên và thứ hai cho 8. Phân mảnh thứ ba có giá trị offset bằng kết quả nhận được.
4. Tiếp tục quá trình này. Phân mảnh cuối cùng có giá trị bit “phân mảnh thêm” là 0.

4.2.1.3 Checksum

Phương pháp phát hiện lỗi được sử dụng bởi hầu hết các giao thức TCP/IP được gọi là *checksum*. Checksum có thể giúp kiểm tra xem có sự sai lệch nào đã xảy ra trong quá trình truyền tải một gói tin hay không. Đây là thông tin dự phòng được thêm vào trong gói tin. Checksum được tính toán từ bên gửi và giá trị thu được này sẽ được gửi đi cùng với gói tin. Phía nhận lặp lại tính toán tương tự trên toàn bộ gói bao gồm cả checksum. Nếu kết quả là đạt yêu cầu, gói dữ liệu được chấp nhận; nếu không, nó sẽ bị từ chối.

- *Tính checksum ở bên gửi:* Tại bên gửi, phần tiêu đề gói tin được chia thành các phần n bit (n thường là 16). Những phần này được cộng lại với nhau bằng cách sử dụng thuật toán lấy bù, kết quả là một giá trị tổng cũng có độ dài n bit. Tổng này sau đó được bù (tất cả các giá trị 0 chuyển thành 1 và tất cả giá trị 1 chuyển thành 0) để tính ra checksum.
- *Checksum tại bên nhận:* Bên nhận chia gói tin nhận được thành k phần và cộng lại tất cả các phần. Sau đó thực hiện tính bù cho kết quả. Nếu kết quả cuối cùng là 0, gói dữ liệu được chấp nhận (không có lỗi trong các dữ liệu trong quá trình truyền tải và xử lý); nếu không, nó bị từ chối. Hình 4.10 cho thấy quá trình thực hiện tính toán và so sánh ở bên gửi và bên nhận.



Hình 4.10 Khái niệm checksum

- *Checksum trong gói IP:* Việc thực hiện tính checksum trong gói tin IP cũng theo nguyên tắc nêu trên. Thứ nhất, giá trị của trường được checksum được thiết lập là 0, sau đó, toàn bộ phần tiêu đề được chia thành các phần 16 bit và cộng lại với nhau. Kết quả (tổng) được tính bù và đưa vào trường checksum. Checksum trong gói tin IP chỉ sử dụng cho phần tiêu đề mà không được sử dụng cho phần dữ liệu, là do: (1) tất cả các giao thức mức cao thực hiện đóng gói dữ liệu trong gói tin IP có một trường checksum cho toàn bộ gói tin. Do đó, checksum đối với các gói tin IP không phải dùng để kiểm tra dữ liệu được đóng gói; (2) phần tiêu đề của gói tin IP thay đổi đối với từng bộ định tuyến đã đi qua, nhưng dữ liệu thì không bị đổi. Vì vậy, các checksum này chỉ dùng cho các phần đã thay đổi. Nếu ta tính kiểm soát cả dữ liệu thì mỗi bộ định tuyến sẽ phải tính toán lại checksum cho toàn bộ gói, và điều này có nghĩa là tăng thêm thời gian xử lý.

4.2.1.4 Các vấn đề về bảo mật

Giao thức IPv4, cũng như toàn bộ mạng Internet, được tạo ra khi mà người dùng Internet tin tưởng lẫn nhau. Không có bất kỳ biện pháp bảo mật nào được cung cấp cho các giao thức IPv4. Tuy nhiên, cho đến hiện tại thì tình hình đã khác, Internet đã không còn an toàn nữa. Phần này nêu một số tóm tắt về các vấn đề bảo mật trong giao thức IP cùng các giải pháp.

a. Các vấn đề bảo mật gói tin

Có ba vấn đề bảo mật đặc biệt áp dụng cho các giao thức IP là *nghe trộm gói tin*, *sửa đổi gói tin*, và *giả mạo IP*.

- *Nghe trộm gói tin:* Kẻ xâm nhập có thể chặn một gói tin IP và tạo một bản sao của nó. Nghe trộm gói tin là một tấn công thụ động, trong đó kẻ tấn công không thay đổi nội dung của gói tin. Kiểu tấn công này rất khó phát hiện vì người gửi và người nhận có thể không bao giờ biết rằng các gói tin đã bị sao chép. Mặc dù nghe trộm không thể bị ngăn chặn, nhưng việc mã hóa các gói dữ liệu có thể làm vô hiệu hóa các nỗ lực tấn công. Kẻ tấn công vẫn có thể nghe trộm dữ liệu, nhưng chúng không thể hiểu được nội dung.
- *Sửa đổi gói tin:* Kẻ tấn công chặn các gói tin, thay đổi nội dung của nó, và gửi các gói dữ liệu mới cho người nhận. Người nhận tin rằng các gói tin đến từ người gửi ban đầu. Kiểu tấn công này có thể được phát hiện bằng cách sử dụng một cơ chế toàn vẹn dữ liệu. Người nhận trước khi mở và sử dụng các nội dung của thông điệp có thể dùng cơ chế này để đảm bảo rằng các gói tin không bị thay đổi trong quá trình truyền tải.
- *Giả mạo IP:* Kẻ tấn công có thể giả vờ là người khác và tạo ra một gói tin IP mang địa chỉ nguồn của máy tính khác nào đó. Ví dụ, kẻ tấn công có thể gửi một gói tin IP cho một ngân hàng giả vờ rằng nó đến từ một trong những khách hàng. Kiểu tấn công này có thể được ngăn chặn bằng cách sử dụng một cơ chế xác thực nguồn gốc.

b. IPSec

IP ngày nay có thể được bảo vệ khỏi các cuộc tấn công đã đề cập bằng cách sử dụng một giao thức gọi là IPSec (*IP Security*). Giao thức này được sử dụng kết hợp với giao thức IP để tạo ra một dịch vụ hướng kết nối giữa hai thực thể. Khi đó chúng có thể trao đổi các gói tin IP mà không cần lo lắng về ba loại tấn công trên. IPSec cung cấp bốn dịch vụ sau:

- *Xác định các thuật toán và các khóa:* Hai thực thể muốn tạo một kênh an toàn dùng chung có thể thỏa thuận về một số thuật toán có sẵn và các khóa được sử dụng cho các mục đích bảo mật.
- *Mã hóa gói tin:* Các gói tin trao đổi giữa hai thực thể có thể được mã hóa để bảo mật bằng cách sử dụng một trong những thuật toán mã hóa và khóa chia sẻ đã thỏa thuận trong bước đầu tiên. Điều này làm cho các cuộc tấn công nghe trộm trở nên vô dụng.
- *Toàn vẹn dữ liệu:* Toàn vẹn dữ liệu đảm bảo rằng các gói tin không bị sửa đổi trong quá trình truyền tải. Nếu gói dữ liệu nhận được không vượt qua kiểm tra về tính toàn vẹn dữ liệu, nó sẽ bị loại bỏ. Điều này ngăn ngừa các loại tấn công sửa đổi gói tin.

- *Xác thực nguồn gốc:* IPsec có thể xác thực nguồn gốc của gói tin để đảm bảo rằng các gói tin không được tạo ra bởi một kẻ mạo danh. Nhờ đó ta có thể ngăn chặn các tấn công giả mạo IP như mô tả ở trên.

4.2.2 Giao thức IPv6

IPv4 có một số yếu điểm là nguyên nhân cho việc cần có một giao thức mới, giao thức Internet phiên bản 6 (IPv6). Các vấn đề như sau: cạn kiệt địa chỉ IP, các tiến trình phải thực hiện một số xử lý không cần thiết, cần thêm tùy chọn mới, cần phải hỗ trợ đa phương tiện và nhu cầu cấp bách về bảo mật.

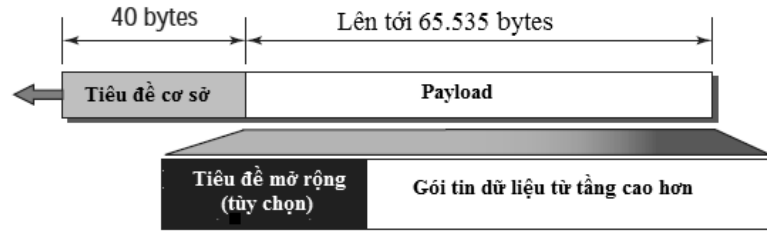
Giao thức IPv6 xử lý các vấn đề trên bằng cách cung cấp những thay đổi cơ bản sau trong giao thức:

- *Không gian địa chỉ lớn hơn:* Một địa chỉ IPv6 có độ dài 128 bit. So với địa chỉ 32 bit của IPv4, không gian địa chỉ đã được tăng rất nhiều (2^{96} lần).
- *Định dạng phần tiêu đề tốt hơn:* IPv6 sử dụng định dạng tiêu đề mới. Các tùy chọn được tách khỏi phần tiêu đề cơ sở và khi cần sẽ được chèn vào giữa phần tiêu đề cơ sở và dữ liệu lớp trên. Điều này làm đơn giản hóa và tăng tốc quá trình định tuyến bởi vì hầu hết các tùy chọn không cần phải được kiểm tra bởi các bộ định tuyến.
- *Các tùy chọn mới:* IPv6 có các tùy chọn mới để cung cấp các chức năng bổ sung.
- *Dự phòng mở rộng:* IPv6 được thiết kế để cho phép phần mở rộng của giao thức theo yêu cầu của công nghệ mới hoặc của các ứng dụng.
- *Hỗ trợ cho phân bổ tài nguyên:* Trong IPv6, trường *service type* (loại dịch vụ) đã được gỡ bỏ, nhưng hai trường mới, *traffic class* (loại lưu lượng) và *flow label* (nhãn luồng) được thêm vào, cho phép phía nguồn có thể gửi yêu cầu để xử lý đặc biệt đối với gói tin. Cơ chế này có thể được sử dụng để hỗ trợ loại lưu lượng thời gian thực như audio và video.
- *Hỗ trợ bảo mật tốt hơn:* Các tùy chọn mã hóa và xác thực trong IPv6 cung cấp bí mật và toàn vẹn của gói tin.

Hiện nay, việc áp dụng IPv6 đã bị chậm, do động lực ban đầu cho sự phát triển IPv6 (cạn kiệt địa chỉ IPv4) đã bị giảm bớt vì đã có ba giải pháp ngắn hạn: không phân lớp địa chỉ IP, sử dụng DHCP để cấp phát địa chỉ IP động, và NAT. Tuy nhiên, các dịch vụ sử dụng Internet mới nhanh chóng tăng lên, chẳng hạn như điện thoại di động IP, điện thoại IP và điện thoại di động hỗ trợ IP, có thể tạo ra nhu cầu muốn thay toàn bộ IPv4 bằng IPv6. Người ta dự đoán rằng tất cả các host trên thế giới sẽ sử dụng IPv6 trong năm 2010, nhưng điều này đã không xảy ra tới tận bây giờ.

4.2.2.1 Định dạng gói tin IPv6

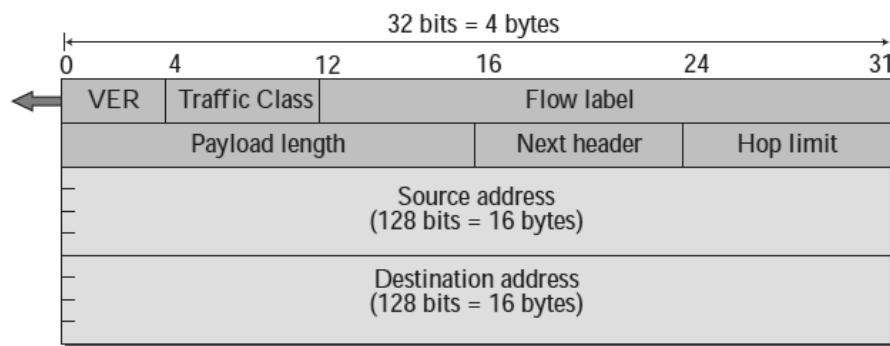
Gói tin IPv6 được thể hiện trong hình 4.11. Mỗi gói bao gồm một phần tiêu đề cơ sở bắt buộc, tiếp theo là payload. Payload bao gồm hai phần: phần tiêu đề mở rộng tùy chọn và dữ liệu từ tầng trên. Các tiêu đề cơ sở chiếm 40 byte, trong khi các tiêu đề mở rộng và dữ liệu từ các tầng trên có thể lên đến 65.535 byte thông tin.



Hình 4.11 IPv6 datagram

a. Phần tiêu đề cơ sở

Hình 4.12 mô tả phần tiêu đề cơ sở với 8 trường.



Hình 4.12 Định dạng của tiêu đề cơ sở

Các trường như sau:

- *Version (Phiên bản)*: Trường 4 bit này xác định số phiên bản IP. Đối với IPv6, giá trị là 6.
- *Traffic class (loại lưu lượng)*: Trường 8 bit này được sử dụng để phân biệt các payload khác nhau với yêu cầu vận chuyển khác nhau. Nó thay thế trường *service class* trong IPv4.
- *Flow label (nhãn luồng)*: Đây là một trường 20 bit được thiết kế để cung cấp xử lý đặc biệt đối với một luồng dữ liệu cụ thể.
- *Payload length (Độ dài payload)*: Trường độ dài payload 2 byte định nghĩa chiều dài của gói tin IP trừ phần tiêu đề cơ sở.
- *Next header (Phần tiêu đề tiếp theo)*: Trường 8 bit này xác định phần tiêu đề sau tiêu đề cơ sở trong gói tin. Tiêu đề tiếp theo là một trong các tiêu đề mở rộng tùy chọn sử dụng bởi IP hoặc tiêu đề của một gói tin được đóng gói, ví dụ như UDP hoặc TCP. Mỗi tiêu đề mở rộng cũng chứa trường này. Bảng 4.1 cho thấy các giá trị của tiêu đề tiếp theo. Trong phiên bản 4 trường này được gọi là *protocol*.

Bảng 4.1 Các mã (code) của Next Header

Code	Next Header	Code	Next Header
0	Tùy chọn Hop-by-hop	44	Phân mảnh
2	ICMP	50	Payload bảo mật mã hóa
6	TCP	51	Xác thực
17	UDP	59	Null (không có next header)
43	Định tuyến nguồn	60	Tùy chọn đích

- *Hop limit (Giới hạn hop)*: Trường 8 bit này phục vụ mục đích tương tự như trường TTL trong IPv4.
- *Source address (Địa chỉ nguồn)*: Trường địa chỉ nguồn là một địa chỉ Internet 16 byte (128 bit) xác định nguồn gốc của các gói tin.
- *Destination address (Địa chỉ đích)*: Trường địa chỉ đích là một địa chỉ Internet 16 byte (128 bit) thường xác định điểm đến cuối cùng của gói tin. Tuy nhiên, nếu định tuyến nguồn được sử dụng thì trường này chứa địa chỉ của bộ định tuyến tiếp theo.

b. Nhãn luồng (Flow label)

Lúc đầu, giao thức IP được thiết kế để làm một giao thức không liên kết. Tuy nhiên, như chúng ta đã biết, cần phải sử dụng giao thức IP làm một giao thức hướng kết nối. Công nghệ *chuyển mạch nhãn đa giao thức MPLS (Multiprotocol Label Switching)* cho phép đóng gói một gói IPv4 trong một tiêu đề MPLS sử dụng một trường nhãn. Trong phiên bản 6, *nhãn luồng* được trực tiếp thêm vào trong định dạng của gói tin IPv6, từ đó cho phép sử dụng như IPv6 theo kiểu giao thức hướng kết nối.

Đối với một bộ định tuyến, một *luồng (flow)* là một chuỗi các gói tin chia sẻ những đặc điểm tương tự, chẳng hạn như đi cùng tuyến đường, sử dụng các tài nguyên giống nhau, có cùng một kiểu bảo mật, v.v. Bộ định tuyến hỗ trợ việc xử lý nhãn luồng có một bảng nhãn luồng. Bảng này có một mục đầu vào cho mỗi nhãn luồng đang hoạt động; mỗi mục xác định các dịch vụ được yêu cầu bởi nhãn luồng tương ứng. Khi bộ định tuyến nhận một gói tin, nó thông qua bảng nhãn luồng để tìm mục tương ứng với giá trị nhãn luồng được định nghĩa trong gói. Sau đó, nó cung cấp cho các gói tin các dịch vụ được đề cập trong mục. Tuy nhiên, cần lưu ý rằng nhãn luồng tự nó không cung cấp thông tin cho các mục của bảng nhãn luồng; các thông tin được cung cấp theo các cách khác như các tùy chọn hop-by-hop hay các giao thức khác.

Trong dạng thức đơn giản nhất, một nhãn luồng có thể được sử dụng để tăng tốc độ xử lý gói tin của bộ định tuyến. Khi bộ định tuyến nhận một gói tin, thay vì tham khảo bảng định tuyến và thực hiện một thuật toán định tuyến để xác định địa chỉ của *hop* tiếp theo, nó có thể dễ dàng được tìm thấy trong một bảng nhãn luồng.

Trong dạng thức phức tạp hơn, một nhãn luồng có thể được sử dụng để hỗ trợ việc truyền tải âm thanh và hình ảnh thời gian thực. Âm thanh và hình ảnh thời gian thực, đặc biệt là ở dạng số, đòi hỏi phải có các nguồn tài nguyên như băng thông cao, bộ đệm lớn, thời gian xử lý dài, v.v. Một tiến trình có thể dành trước các nguồn tài nguyên để đảm bảo rằng dữ liệu thời gian thực sẽ không thể bị trì hoãn do thiếu tài nguyên. Việc sử dụng các dữ liệu thời gian

thực và dự phòng các nguồn lực này yêu cầu sử dụng các giao thức khác như *giao thức thời gian thực RTP (Real-Time Protocol)* và *Giao thức dự trữ tài nguyên RSVP (Resource Reservation Protocol)* thêm vào IPv6.

Để sử dụng hiệu quả nhãn luồng, có ba nguyên tắc được xác định:

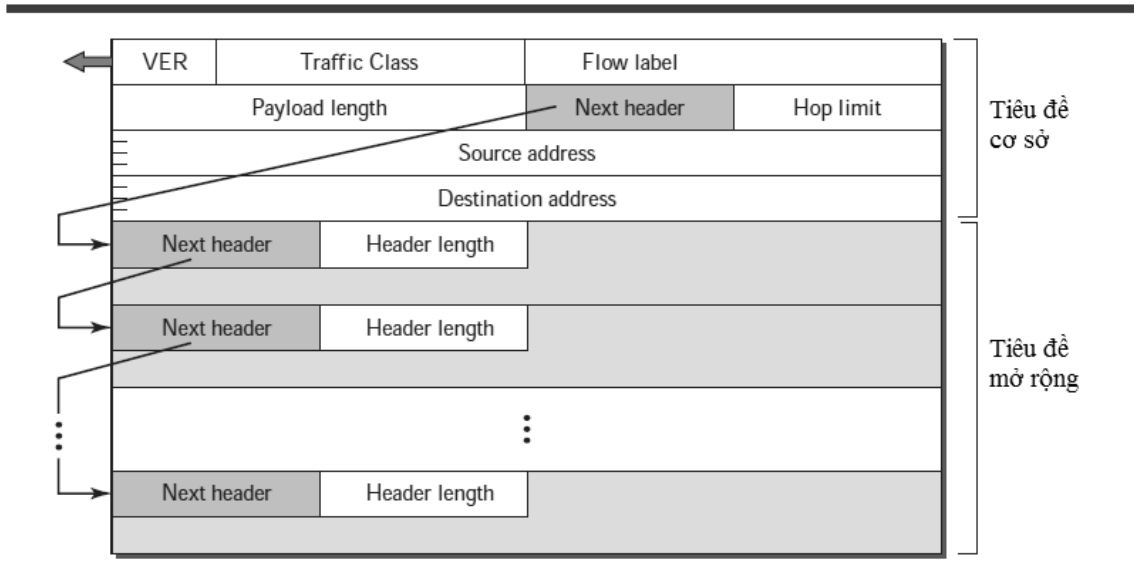
- Nhãn luồng được gán cho một gói bởi host nguồn. Nhãn là một số ngẫu nhiên từ 1 đến $2^{24} - 1$. Một nguồn không được tái sử dụng một nhãn luồng cho một luồng mới trong khi luồng hiện tại vẫn còn.
- Nếu một host không hỗ trợ các nhãn luồng, nó đặt trường này bằng 0. Nếu bộ định tuyến không hỗ trợ các nhãn luồng, nó chỉ đơn giản bỏ qua.
- Tất cả các gói thuộc cùng một luồng có cùng một nguồn, cùng một đích, cùng một độ ưu tiên và có cùng các tùy chọn.

c. So sánh giữa phần tiêu đề IPv4 và IPv6

- Trường *Header length* (độ dài phần tiêu đề) bị loại bỏ trong IPv6 bởi vì chiều dài của phần tiêu đề được cố định trong phiên bản này.
- Trường *Service type* (loại dịch vụ) bị loại bỏ trong IPv6. Trường *traffic class* (loại lưu lượng) và *Flow labe* (nhãn luồng) cùng đảm nhiệm chức năng của trường *service type*.
- Trường *Total length* (độ dài tổng) bị loại bỏ trong IPv6 và được thay thế bởi trường *Payload length* (độ dài payload).
- *Identification* (Định danh), *flag* (cờ), và trường offset bị loại bỏ khỏi phần tiêu đề cơ bản trong IPv6. Chúng được lưu trong phần tiêu đề mở rộng phân mảnh.
- Trường *TTL* được gọi là *Hop limit* (giới hạn hop) trong IPv6.
- Trường *Protocol* (giao thức) được thay thế bằng trường *Next header* (tiêu đề tiếp theo).
- *Header Checksum* bị loại bỏ bởi vì checksum được cung cấp bởi các giao thức lớp trên; do đó nó không cần thiết ở đây.
- Các trường *Option* (tùy chọn) trong IPv4 được thực hiện như các phần tiêu đề mở rộng trong IPv6.

d. Các phần tiêu đề mở rộng

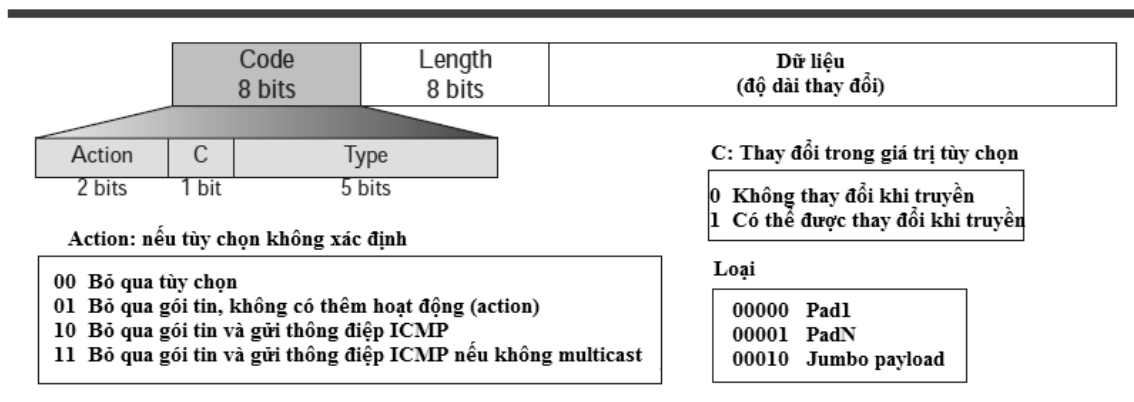
Độ dài của phần tiêu đề cơ sở cố định là 40 byte. Tuy nhiên, để cung cấp thêm nhiều chức năng cho các gói tin IP, phần tiêu đề cơ sở có thể được thêm vào phía sau gồm sáu tiêu đề mở rộng. Một số tiêu đề trong số này là tùy chọn trong IPv4. Hình 4.13 cho thấy các định dạng tiêu đề mở rộng.



Hình 4.13 Định dạng tiêu đề mở rộng

Sáu loại tiêu đề mở rộng bao gồm: tùy chọn hop-by-hop, định tuyến nguồn, phân mảnh, xác thực, payload bảo mật mã hóa (ESP), và tùy chọn đích.

- **Tùy chọn hop-by-hop:** Tùy chọn hop-by-hop được sử dụng khi nguồn cần chuyển thông tin đến tất cả các bộ định tuyến mà gói tin đi qua. Ví dụ, các bộ định tuyến phải được thông báo về chức năng quản lý, gỡ lỗi, hoặc kiểm soát nhất định. Hoặc nếu chiều dài của gói tin lớn hơn chiều dài thông thường 65.535 byte, các bộ định tuyến phải có được thông tin này. Định dạng tiêu đề của tùy chọn hop-by-hop gồm có 3 phần: trường đầu tiên (next header) xác định tiêu đề tiếp theo trong chuỗi các tiêu đề; header length (chiều dài tiêu đề) xác định số byte trong phần tiêu đề (bao gồm cả các trường tiêu đề tiếp theo); phần còn lại của tiêu đề chứa các tùy chọn khác nhau. Hiện có ba tùy chọn hop-by-hop: Pad1, PadN, và jumbo payload (hình 4.14).



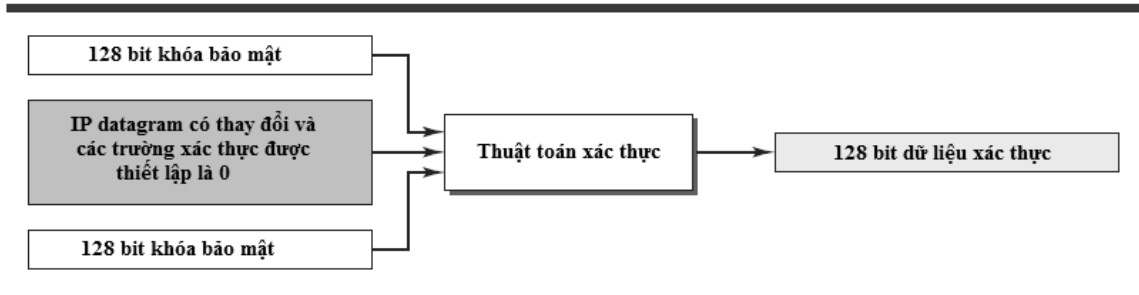
Hình 4.14 Định dạng của tùy chọn trong tiêu đề của hop-by-hop

- ✓ **Pad1:** Tùy chọn này dài 1 byte và được thiết kế cho mục đích gióng hàng (alignment). Một số tùy chọn cần phải bắt đầu từ một bit cụ thể của từ có độ dài 32 bit. Nếu một tùy chọn ngắn so với yêu cầu này đúng một byte, Pad1 sẽ được thêm vào. Pad1 không chứa trường độ dài tùy chọn hoặc trường dữ liệu tùy chọn. Nó chỉ

chứa trường mã tùy chọn với tất cả các bit thiết lập là 0 (trường *action* là 00, *change bit* là 0, và *type* là 00000). Pad1 có thể được chèn vào bất cứ nơi nào trong tiêu đề tùy chọn hop-by-hop.

- ✓ *PadN*: PadN tương tự như khái niệm Pad1. Sự khác biệt là PadN được sử dụng khi giống hàng cần dùng 2 byte hoặc nhiều hơn. Tùy chọn này bao gồm 1 byte *code* tùy chọn, 1 byte chiều dài tùy chọn, và một số byte 0 dùng để làm đệm. Giá trị của *code* tùy chọn là 1 (trường *action* là 00, *change bit* là 0, và *type* là 00001). Độ dài tùy chọn có chứa một số các byte đệm.
- ✓ *Jumbo payload*: Độ dài của payload trong gói tin IP có thể được tối đa là 65.535 byte. Tuy nhiên, nếu vì lý do nào đó ta cần một payload dài hơn, thì chúng ta có thể sử dụng tùy chọn jumbo payload để xác định độ dài dài hơn này. Tùy chọn jumbo payload phải luôn luôn bắt đầu từ một bội số của 4 byte cộng với 2 từ đoạn đầu các tiêu đề mở rộng. Tùy chọn jumbo payload bắt đầu từ $(4n + 2)$ byte, trong đó n là một số nguyên nhỏ.
- *Tùy chọn đích*: Tùy chọn đích được sử dụng khi nguồn chỉ cần chuyển thông tin đến đích. Bộ định tuyến trung gian không được phép truy cập vào thông tin này. Định dạng của tùy chọn đích giống như tùy chọn hop-by-hop. Cho đến nay, chỉ có các tùy chọn Pad1 và PadN đã được định nghĩa.
- *Định tuyến nguồn*: Tiêu đề mở rộng nguồn định tuyến kết hợp các khái niệm về *tuyến đường nguồn nghiêm ngặt* và các tùy chọn *tuyến đường nguồn mềm dẻo* của IPv4. Tiêu đề nguồn định tuyến chứa tối thiểu là 7 trường. Hai trường đầu tiên, *next header* (tiêu đề tiếp theo) và *header length* (độ dài tiêu đề), tương tự với các trường này trong tiêu đề mở rộng hop-by-hop. Trường *type* xác định loại định tuyến (mềm dẻo hoặc nghiêm ngặt). Trường *addresses left* cho biết số *hop* vẫn còn cần thiết để đạt đến đích. Trường *reserved* để dự phòng. Trường *strict/loose mask* xác định độ mềm dẻo của việc định tuyến. Nếu được thiết lập là *strict*, định tuyến phải làm theo chính xác như được chỉ ra bởi nguồn. Nếu là *loose* thì các bộ định tuyến khác có thể được đi qua ngoài những điểm đã được mô tả trong tiêu đề.
- *Phân mảnh*: Khái niệm phân mảnh tương tự như trong IPv4. Tuy nhiên, vị trí xảy ra sự phân mảnh có khác nhau. Trong IPv4, nguồn hoặc một bộ định tuyến được yêu cầu phải phân mảnh nếu kích thước của gói tin lớn hơn MTU của mạng lưới mà gói tin đi qua. Trong IPv6, chỉ có nguồn ban đầu có thể phân mảnh. Một nguồn phải sử dụng kỹ thuật *Path MTU Discovery* để tìm MTU nhỏ nhất được hỗ trợ bởi mọi mạng trên đường đi. Sau đó nguồn thực hiện phân mảnh theo cách thức này. Nếu nguồn không sử dụng kỹ thuật *Path MTU Discovery* thì nó sẽ phân mảnh gói tin thành các mảnh có kích thước 1.280 byte hoặc nhỏ hơn. Đây là kích thước tối thiểu của MTU cần thiết cho mỗi mạng được kết nối với Internet.
- *Xác thực*: Header mở rộng xác thực có 2 mục đích: xác nhận người gửi và đảm bảo tính toàn vẹn của dữ liệu. Mục tiêu đầu cần thiết bởi người nhận muốn biết chắc chắn thông điệp đến từ người gửi đúng và không phải từ một kẻ mạo danh. Mục tiêu thứ hai cần khi kiểm tra xem dữ liệu có bị thay đổi bởi *hacker* hay không trong quá trình

truyền. Định dạng của header mở rộng xác thực gồm 2 trường: SPI (*Security parameter index*) xác định thuật toán được sử dụng để xác thực; trường *authentication data* (dữ liệu xác thực) chứa các dữ liệu thực tế được tạo ra bởi các thuật toán.



Hình 4.15 Tính toán dữ liệu xác thực

Có nhiều thuật toán khác nhau có thể được sử dụng để xác thực. Hình 4.15 trình bày phương pháp tính trường dữ liệu xác thực. Người gửi gửi một khóa bí mật 128 bit, toàn bộ gói tin IP, và khóa bảo mật 128 bit một lần nữa cho các thuật toán. Những trường đó trong gói tin với giá trị thay đổi trong quá trình truyền (ví dụ, số *hop*) được thiết lập là 0. Các gói tin được truyền cho thuật toán bao gồm mở rộng tiêu đề xác thực, với các trường dữ liệu xác thực thiết lập là 0. Các thuật toán tạo ra dữ liệu xác thực được đưa vào tiêu đề mở rộng trước khi truyền gói tin.

Bên nhận hoạt động theo cách tương tự. Nó lấy khóa bí mật và các gói tin nhận được (một lần nữa, với các trường thay đổi thiết lập là không) và chuyển chúng tới các thuật toán xác thực. Nếu kết quả phù hợp với trường dữ liệu xác thực thì các gói tin IP được xác thực; nếu không, các gói tin bị loại bỏ.

- *Payload bảo mật mã hóa (Encrypted Security Payload)*: ESP là một phần mở rộng cung cấp bảo mật và bảo vệ chống lại nghe trộm. Định dạng của trường SPI là một từ 32 bit xác định kiểu mã hóa/giải mã được sử dụng. Các trường khác chứa các dữ liệu được mã hóa cùng với các tham số mở rộng cần thiết cho các thuật toán. Mã hóa có thể được thực hiện theo hai cách: *chế độ truyền tải* hoặc *chế độ đường hầm*.

e. So sánh giữa IPv4 và IPv6

Dưới đây là một so sánh nhanh giữa các tùy chọn được sử dụng trong IPv4 và các tùy chọn được sử dụng trong IPv6 (như tiêu đề mở rộng).

- Các tùy chọn không hoạt động và kết thúc của tùy chọn trong IPv4 được thay thế bằng tùy chọn Pad1 và PadN trong IPv6.
- Tùy chọn *record route* không được thực hiện trong IPv6 bởi vì không được sử dụng.
- Tùy chọn *timestamp* không được thực hiện bởi vì không được sử dụng.
- Tùy chọn *source route* được gọi là header mở rộng *source route* trong IPv6.
- Các trường phân mảnh trong phần tiêu đề cơ sở của IPv4 đã chuyển thành header mở rộng phân mảnh trong IPv6.
- Tiêu đề mở rộng xác thực là mới trong IPv6.

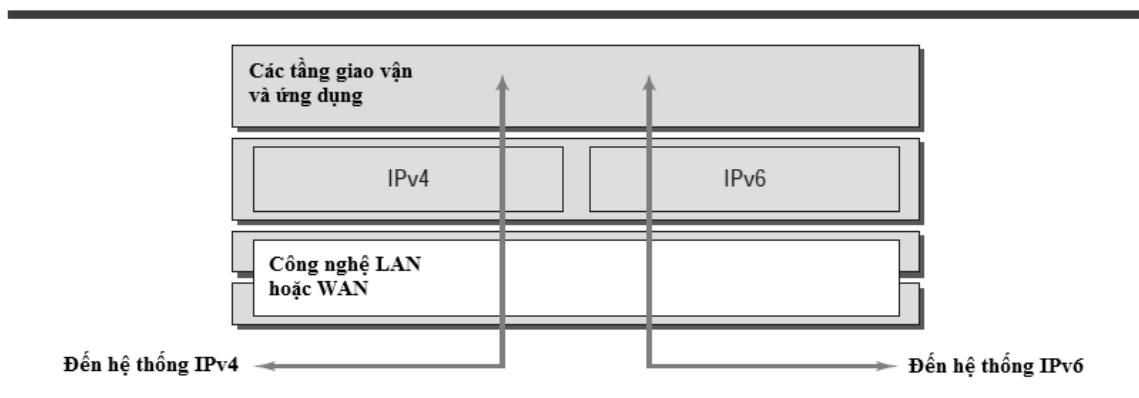
- Tiêu đề mở rộng payload bảo mật mã hóa là mới trong IPv6.

4.2.2.2 Chuyển đổi từ IPv4 sang IPv6

Do số lượng các hệ thống trên Internet là rất lớn nên quá trình chuyển đổi từ IPv4 sang IPv6 không thể diễn ra đột ngột. Cần phải có một lượng thời gian đáng kể trước khi tất cả các hệ thống trên mạng Internet có thể chuyển từ IPv4 sang IPv6. Việc chuyển đổi phải mịn để ngăn chặn bất kỳ vấn đề nào giữa hệ thống IPv4 và IPv6. Có ba chiến lược được IETF đặt ra để trợ giúp quá trình chuyển đổi: *dual stack* (chồng giao thức kép), *tunneling* (đường hầm), và *header translation* (dịch tiêu đề).

a. Dual stack (chồng giao thức kép)

Người ta khuyến nghị tất cả các host, trước khi chuyển hoàn toàn sang phiên bản 6, nên có một chồng giao thức kép. Nói cách khác, một máy phải chạy IPv4 và IPv6 đồng thời cho đến khi tất cả Internet sử dụng IPv6. Hình 4.16 cho thấy cách bố trí của một cấu hình chồng giao thức kép như vậy.

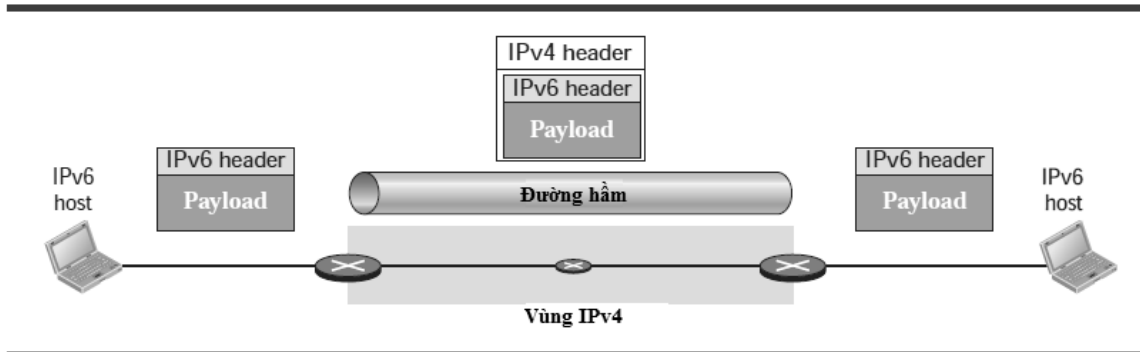


Hình 4.16 Dual stack (chồng giao thức kép)

Để xác định phiên bản sử dụng khi gửi một gói tin đến một đích, host nguồn truy vấn DNS. Nếu DNS trả về một địa chỉ IPv4, host nguồn gửi một gói tin IPv4. Nếu DNS trả về một địa chỉ IPv6, host nguồn gửi một gói tin IPv6.

b. Tunneling (đường hầm)

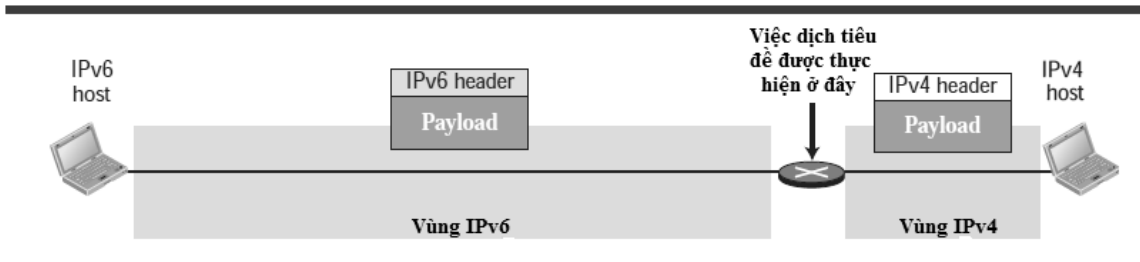
Tunneling là một chiến lược được sử dụng khi hai máy tính chạy IPv6 muốn giao tiếp với nhau, nhưng các gói tin phải đi qua một khu vực sử dụng IPv4. Để đi qua khu vực này, các gói tin phải có địa chỉ IPv4. Vì vậy, các gói tin IPv6 được đóng gói trong một gói tin IPv4 khi nó đi vào khu vực, và vỏ của gói tin được bỏ đi khi nó đi ra khỏi khu vực. Như vậy trông giống như là các gói tin IPv6 đi qua một đường hầm ở một đầu và xuất hiện ở đầu kia. Để chỉ rõ việc gói tin IPv4 đang mang dữ liệu là một gói tin IPv6, giá trị giao thức được thiết lập là 41. Tunneling được thể hiện trong hình 4.17.



Hình 4.17 Chiến lược đường hầm

c. Header translation (Dịch tiêu đề)

Việc dịch phần tiêu đề là cần thiết khi phần lớn Internet đã chuyển sang IPv6 nhưng một số hệ thống vẫn sử dụng IPv4. Bên gửi muốn sử dụng IPv6, nhưng bên nhận không hiểu IPv6. Đường hầm không hoạt động trong tình huống này bởi vì các gói tin phải được định dạng theo IPv4 để bên nhận hiểu được. Trong trường hợp này, định dạng tiêu đề phải được thay đổi hoàn toàn thông qua dịch tiêu đề. Tiêu đề của gói tin IPv6 được chuyển đổi thành tiêu đề của gói tin IPv4 (xem hình 4.18).



Hình 4.18 Chiến lược dịch tiêu đề

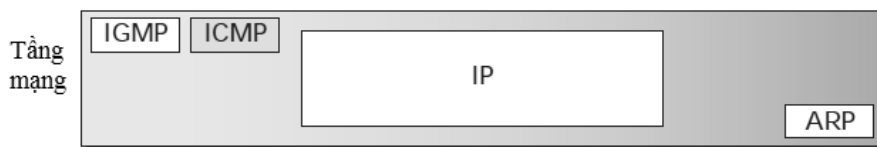
Dịch tiêu đề sử dụng ánh xạ địa chỉ để dịch một địa chỉ IPv6 thành địa chỉ IPv4. Phần sau liệt kê một số quy tắc sử dụng trong việc chuyển đổi một tiêu đề gói tin IPv6 thành một tiêu đề gói tin IPv4.

- Ánh xạ địa chỉ IPv6 được thay đổi thành một địa chỉ IPv4 bằng cách tách ra 32 bit ngoài cùng bên phải.
- Giá trị của trường ưu tiên IPv6 bị loại bỏ.
- Trường *service type* (kiểu dịch vụ) trong IPv4 được thiết lập là 0.
- Checksum cho IPv4 được tính toán và chèn vào trong trường tương ứng.
- *Flow label* của IPv6 được bỏ qua.
- Các tiêu đề mở rộng tương thích được chuyển đổi thành các tùy chọn và chèn vào trong tiêu đề IPv4. Một số có thể đã bị bỏ qua.
- Chiều dài của tiêu đề IPv4 được tính toán và đưa vào trường tương ứng.
- Chiều dài tổng của gói tin IPv4 được tính toán và đưa vào trường tương ứng.

4.3 ICMPv4 VÀ ICMPv6

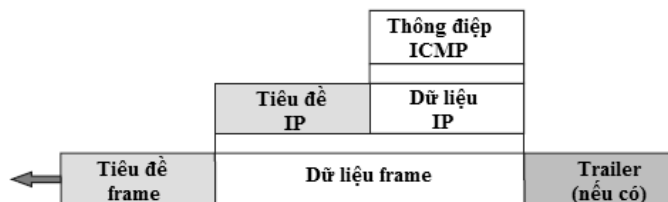
4.3.1 Giao thức ICMPv4

Giao thức IP không có cơ chế báo lỗi hoặc sửa lỗi. Trong khi có một số tình huống lỗi có thể xảy ra, như: bộ định tuyến không thể tìm thấy bộ định tuyến đích hoặc trường TTL bằng 0 (khi đó cần loại bỏ gói tin); tại đích không thể tập hợp được đầy đủ tất cả các mảnh của gói tin (do các mảnh không về đích đúng thời hạn); v.v. Giao thức IP cũng không có cơ chế quản lý truy vấn và máy chủ, ví dụ trong trường hợp cần phải xác định xem một bộ định tuyến hoặc một máy chủ còn hoạt động hay không; hoặc quản trị mạng cần thông tin từ các thiết bị này. *ICMP (Internet Control Message Protocol, giao thức thông điệp điều khiển Internet)* đã được thiết kế để bù đắp cho những thiếu sót trên. Hình 4.19 cho thấy vị trí của ICMP liên quan đến IP và các giao thức khác trong tầng mạng.



Hình 4.19 Vị trí của giao thức ICMP trong tầng mạng

ICMP là một giao thức tầng mạng. Tuy nhiên, thông điệp của nó không được truyền trực tiếp cho tầng liên kết dữ liệu mà các thông điệp trước hết sẽ đóng gói bên trong gói tin IP trước khi đi xuống các tầng thấp hơn (xem hình 4.20).



Hình 4.20 Đóng gói ICMP

Nếu giá trị của trường *protocol* (giao thức) trong các gói tin IP là 1 thì dữ liệu IP là một thông điệp ICMP.

4.3.1.1 Thông điệp ICMP

Thông điệp ICMP được chia thành hai loại chính: *thông điệp báo lỗi* và *thông điệp truy vấn*. Thông điệp báo lỗi báo cáo các vấn đề mà bộ định tuyến hoặc host (đích) có thể gặp phải khi xử lý một gói tin IP. Thông điệp truy vấn (thường xảy ra theo cặp) trợ giúp host hoặc nhà quản lý mạng có được thông tin cụ thể từ một bộ định tuyến hoặc host khác. Ví dụ, các nút có thể tìm ra các láng giềng. Ngoài ra, host có thể tìm hiểu về các bộ định tuyến trên mạng và các bộ định tuyến có thể trợ giúp một nút chuyển hướng thông điệp của nó. Bảng 4.2 liệt kê các thông điệp ICMP theo từng loại.

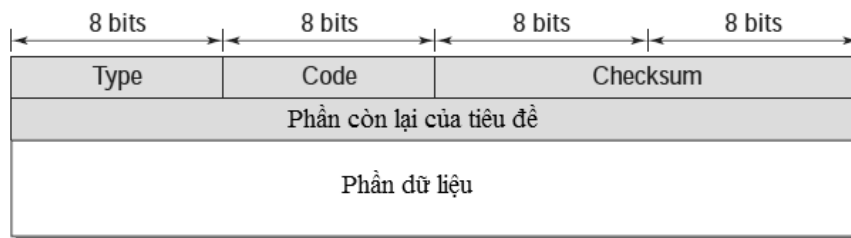
Bảng 4.2 Thông điệp ICMP

Phân loại	Mã	Thông điệp
Thông điệp báo lỗi	3	Không tới được đích
	4	Ngắt nguồn
	11	Vượt quá thời gian
	12	Có vấn đề về tham số
	5	Chuyển hướng
Thông điệp truy vấn	8 hoặc 0	Đội lại yêu cầu hoặc đáp ứng
	13 hoặc 14	Dấu thời gian yêu cầu hoặc đáp ứng

a. Định dạng thông điệp

Một thông điệp ICMP có tiêu đề 8 byte và phần dữ liệu có kích thước thay đổi. Mặc dù đối với từng loại thông điệp, định dạng chung của tiêu đề là khác nhau, nhưng 4 byte đầu tiên là giống nhau (xem hình 4.21): trường đầu tiên, *ICMP type*, xác định các loại tin nhắn; trường *code* xác định các loại thông điệp cụ thể; và trường *checksum*. Phần còn lại của tiêu đề là riêng, cụ thể cho từng loại thông điệp.

Phần dữ liệu trong các thông báo lỗi chứa thông tin cho việc tìm kiếm các gói tin ban đầu có lỗi. Trong các thông điệp truy vấn, phần dữ liệu mang thông tin bổ sung dựa trên kiểu của truy vấn.

**Hình 4.21 Định dạng chung của thông điệp ICMP****b. Các thông điệp báo lỗi**

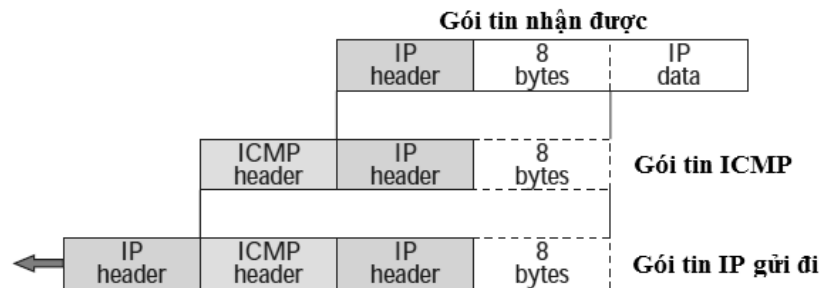
Một trong những trách nhiệm chính của ICMP là báo cáo lỗi. Mặc dù công nghệ đã sản xuất được các phương tiện truyền thông ngày càng đáng tin cậy, nhưng lỗi vẫn còn tồn tại và cần phải được xử lý. Giao thức IP, như đã thảo luận trong phần trước, là một giao thức không đáng tin cậy. Điều này có nghĩa là kiểm tra lỗi và kiểm soát lỗi không phải là mối quan tâm của IP. ICMP được thiết kế, một phần để bù đắp cho thiếu sót này. Tuy nhiên, ICMP không sửa lỗi mà đơn giản chỉ là báo cáo lỗi. Việc sửa lỗi được dành cho các giao thức tầng cao hơn. Thông báo lỗi luôn luôn được gửi đến nguồn ban đầu vì thông tin về tuyến đường có sẵn trong các gói tin chỉ là các địa chỉ IP nguồn và đích. ICMP sử dụng địa chỉ IP nguồn để gửi các thông báo lỗi về nguồn (khởi đầu) của gói tin.

Năm loại lỗi được xử lý là: không tới được đích, ngắt nguồn, vượt quá thời gian, có vấn đề về tham số, và chuyển hướng.

Những điểm quan trọng về thông báo lỗi ICMP là:

- Không có thông điệp lỗi ICMP được tạo ra để đáp ứng với một gói tin mang một thông điệp lỗi ICMP.
- Không có thông điệp lỗi ICMP được tạo ra cho một gói tin bị phân mảnh không phải là đoạn đầu tiên.
- Không có thông điệp lỗi ICMP được tạo ra cho một gói tin có địa chỉ multicast.
- Không có thông điệp lỗi ICMP được tạo ra cho một gói tin có địa chỉ đặc biệt như 127.0.0.0 hoặc 0.0.0.0.

Lưu ý rằng tất cả các thông điệp lỗi chứa một phần dữ liệu bao gồm tiêu đề IP của gói tin ban đầu cộng với 8 byte đầu tiên của dữ liệu trong gói đó. Tiêu đề gói tin ban đầu được thêm vào để cung cấp cho nguồn (nơi nhận được thông báo lỗi) các thông tin về chính gói đó. 8 byte dữ liệu được thêm bởi vì 8 byte đầu tiên cung cấp thông tin về số cổng (UDP và TCP) và số thứ tự (TCP). Thông tin này là cần thiết để các nguồn có thể thông báo cho các giao thức (TCP hoặc UDP) về lỗi. ICMP tạo thành một gói tin báo lỗi, sau đó gói này được đóng gói trong một gói tin IP (xem hình 4.22).



Hình 4.22 Nội dung của trường dữ liệu cho các thông điệp báo lỗi

➤ **Không tới được đích (Destination-unreachable):**

Khi một bộ định tuyến không thể định tuyến một gói tin hoặc một host không thể truyền đi một gói tin, gói tin sẽ bị loại bỏ và bộ định tuyến hay host gửi thông điệp “không tới được đích” trở lại host nguồn tạo ra gói tin. Hình 4.23 cho thấy định dạng của thông điệp “không tới được đích”.

Type: 3	Code: 0 to 15	Checksum
Phần chưa được sử dụng (Tất cả các bit đều là 0)		
Phần gói tin IP datagram nhận được bao gồm tiêu đề IP cộng với 8 byte đầu tiên của dữ liệu gói tin		

Hình 4.23 Định dạng thông điệp không tới được đích

Trường *code* cho loại lỗi này chỉ ra lý do loại bỏ các gói tin:

- ✓ **Code 0:** Mạng không thể truy cập, có thể là do lỗi phần cứng.
- ✓ **Code 1:** Host không thể truy cập, có thể là do lỗi phần cứng.

- ✓ **Code 2:** Giao thức không thể truy cập. Một gói tin IP có thể mang dữ liệu thuộc về các giao thức cấp cao hơn như UDP, TCP, và OSPF. Nếu host đích nhận được một gói tin cần phải được chuyển đi, ví dụ, đến giao thức TCP, nhưng giao thức TCP không hoạt động tại thời điểm này, thì một thông điệp code 2 được gửi đi.
- ✓ **Code 3:** Cổng không thể truy cập. Chương trình ứng dụng (tiền trình) gói tin được dự định trước không hoạt động tại thời điểm này.
- ✓ **Code 4:** Phân mảnh là cần thiết, nhưng trường DF (*do not fragment*) của gói tin đã được thiết lập. Nói cách khác, bên gửi gói tin chỉ ra rằng gói dữ liệu không bị phân mảnh, nhưng định tuyến là không thể nếu không thực hiện phân mảnh.
- ✓ **Code 5:** Định tuyến nguồn không thể hoàn thành. Nói cách khác, một hoặc nhiều thiết bị định tuyến được xác định trong tùy chọn định tuyến nguồn không thể truy cập.
- ✓ **Code 6:** Mạng đích không rõ. Cái này khác code 0. Trong code 0, bộ định tuyến biết rằng mạng đích tồn tại, nhưng nó là không thể truy cập vào lúc này. Đối với code 6, bộ định tuyến không có thông tin về mạng đích.
- ✓ **Code 7:** Host đích không rõ. Cái này khác với code 1. Với code 1, bộ định tuyến biết rằng host đích tồn tại, nhưng nó không thể truy cập vào lúc này. Đối với code 7, bộ định tuyến không hề biết về sự tồn tại của host đích.
- ✓ **Code 8:** Host nguồn đang bị cô lập.
- ✓ **Code 9:** Truyền thông với mạng đích bị cấm về mặt hành chính.
- ✓ **Code 10:** Truyền thông với host đích bị cấm về mặt hành chính.
- ✓ **Code 11:** Mạng không thể truy cập đối với các kiểu xác định của dịch vụ. Cái này khác với code 0. Ở đây bộ định tuyến có thể định tuyến các gói tin nếu nguồn đã yêu cầu một loại dịch vụ có sẵn.
- ✓ **Code 12:** Host không thể truy cập cho các loại hình quy định của dịch vụ. Cái này khác với code 1. Ở đây bộ định tuyến có thể định tuyến các gói tin nếu nguồn đã yêu cầu một loại dịch vụ có sẵn.
- ✓ **Code 13:** Host không thể truy cập vì người quản trị đã đặt một bộ lọc đối với nó.
- ✓ **Code 14:** Host không thể truy cập vì quyền ưu tiên của host bị vi phạm. Thông điệp này được gửi bởi một bộ định tuyến để chỉ ra rằng ưu tiên đã yêu cầu không được phép cho đích đến.
- ✓ **Code 15:** Host không thể truy cập bởi vì ưu tiên của nó đã bị bỏ. Thông điệp này được tạo ra khi các nhà khai thác mạng áp đặt một mức tối thiểu về quyền ưu tiên cho hoạt động của mạng, trong khi các gói tin được gửi đi với một độ ưu tiên thấp hơn.

➤ **Ngắt nguồn (Source Quench):**

Giao thức IP là một giao thức không liên kết. Không có liên lạc giữa các host nguồn dùng để tạo gói tin, các thiết bị định tuyến để chuyển tiếp gói tin, và các máy chủ đích để xử

lý gói tin. Một trong những hậu quả của việc thiếu vắng liên lạc là sự thiếu kiểm soát luồng và kiểm soát tắc nghẽn.

Thông điệp ngắt nguồn trong ICMP được thiết kế để thêm một kiểu điều khiển luồng và điều khiển tắc nghẽn cho IP. Khi một bộ định tuyến hoặc host loại bỏ một gói do tắc nghẽn, nó sẽ gửi một thông điệp ngắt nguồn tới bên gửi gói tin. Thông báo này có hai mục đích. Đầu tiên, nó thông báo cho nguồn rằng gói tin đã được loại bỏ. Thứ hai, nó cảnh báo nguồn rằng có ùn tắc ở đâu đó trong đường truyền và đó là nguồn gốc nên làm chậm (ngắt) tiến trình gửi. Định dạng thông điệp ngắt nguồn được thể hiện trong hình 4.24.

Type: 4	Code: 0	Checksum
Phần chưa được sử dụng (Tất cả các bit đều là 0)		
Phần gói tin IP datagram nhận được bao gồm tiêu đề IP cộng với 8 byte đầu tiên của dữ liệu gói tin		

Hình 4.24 Định dạng thông điệp ngắt nguồn

➤ **Vượt quá thời gian:**

Thông điệp vượt quá thời gian được tạo ra trong hai trường hợp:

- *Thứ nhất*, bộ định tuyến sử dụng bảng định tuyến để tìm *hop* kế tiếp (bộ định tuyến tiếp theo) nhận gói. Nếu có sai sót trong một hoặc nhiều bảng định tuyến thì gói tin có thể di chuyển theo một vòng lặp hoặc một chu trình, đi từ một bộ định tuyến tới bộ định tuyến tiếp theo hoặc đi qua một loạt các thiết bị định tuyến không ngừng. Trong mỗi gói chứa một trường thời gian sống TTL để kiểm soát tình trạng này. Khi một gói tin tới một bộ định tuyến, giá trị của trường này được giảm đi 1. Khi giá trị TTL đạt đến 0, sau khi thực hiện giảm giá trị, bộ định tuyến loại sẽ loại bỏ gói tin. Tuy nhiên, khi gói tin bị loại bỏ, thì một thông báo vượt quá thời gian phải được gửi bằng bộ định tuyến tới nguồn ban đầu.
- *Thứ hai*, thông điệp vượt quá thời gian cũng được tạo ra khi tất cả các phân mảnh tạo nên một thông điệp không đến host đích trong một thời hạn nhất định. Khi phân mảnh đầu tiên tới, host đích khởi động bộ đếm thời gian. Nếu tất cả các mảnh không đến khi hết thời gian, đích sẽ loại bỏ tất cả các mảnh và gửi một thông điệp vượt thời gian cho bên gửi ban đầu.

Hình 4.25 cho thấy định dạng của thông điệp vượt thời gian. Code 0 được sử dụng khi các gói tin bị loại bỏ bởi bộ định tuyến do giá trường TTL là 0. Code 1 được sử dụng khi các mảnh đến của một gói tin bị loại bỏ bởi vì một số mảnh đã không đến đúng thời hạn.

Type: 11	Code: 0 hoặc 1	Checksum
Phần chưa được sử dụng (Tất cả các bit đều là 0)		
Phần gói tin IP datagram nhận được bao gồm tiêu đề IP cộng với 8 byte đầu tiên của dữ liệu gói tin		

Hình 4.25 Định dạng thông điệp vượt quá thời gian

➤ **Có vấn đề về tham số:**

Bất kỳ sự mơ hồ nào trong phần tiêu đề của một gói tin đều có thể gây ra các vấn đề nghiêm trọng khi gói tin truyền qua mạng Internet. Nếu một bộ định tuyến hoặc host đích phát hiện ra một giá trị không rõ ràng hoặc còn thiếu trong bất kỳ trường nào của gói tin, nó loại bỏ gói tin và gửi thông báo tham số có vấn đề tới nguồn.

Hình 4.26 cho thấy định dạng của thông điệp tham số có vấn đề. Trường code trong trường hợp này xác định lý do loại bỏ gói tin:

- **Code 0:** Có một lỗi hoặc có gì đó không rõ ràng trong một trường tiêu đề. Trong trường hợp này, giá trị tại các trường con trỏ (*pointer*) sẽ trỏ đến byte có vấn đề. Ví dụ, nếu giá trị là 0, thì byte đầu tiên là một trường không hợp lệ.
- **Code 1:** Phần bắt buộc trong một tùy chọn bị mất. Trong trường hợp này, con trỏ không được sử dụng.

Type: 12	Code: 0 hoặc 1	Checksum
Pointer	Phần chưa được sử dụng (Tất cả các bit đều là 0)	
Phần gói tin IP datagram nhận được bao gồm tiêu đề IP cộng với 8 byte đầu tiên của dữ liệu gói tin		

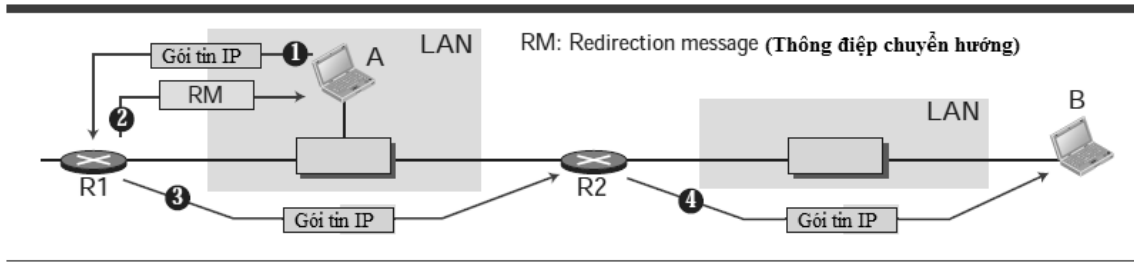
Hình 4.26 Định dạng thông điệp có vấn đề về tham số

➤ **Chuyển hướng:**

Khi một bộ định tuyến muốn gửi một gói tin sang mạng khác, nó cần phải biết địa chỉ IP của bộ định tuyến phù hợp tiếp theo. Điều này cũng tương tự nếu người gửi là một host. Cả hai thiết bị định tuyến và host sau đó phải có một bảng định tuyến để tìm địa chỉ của bộ định tuyến tiếp theo. Bộ định tuyến tham gia vào quá trình cập nhật định tuyến và chúng có nghĩa vụ phải cập nhật liên tục. Định tuyến là quá trình động.

Tuy nhiên, để hiệu quả, host không tham gia vào quá trình cập nhật định tuyến vì trong Internet, số lượng host nhiều hơn hẳn số lượng thiết bị định tuyến. Việc cập nhật bảng định tuyến của các host một cách tự động sẽ tạo ra lưu lượng vô cùng lớn. Host thường sử dụng định tuyến tĩnh. Khi một host xuất hiện, bảng định tuyến của nó có một số giới hạn các chỉ mục. Nó thường chỉ biết địa chỉ IP của một bộ định tuyến mặc định. Vì lý do này, khi host gửi một gói dữ liệu sang một mạng khác, nó có thể đi tới bộ định tuyến sai. Trong trường hợp này, bộ định tuyến nhận được gói tin sẽ chuyển tiếp gói tin đến bộ định tuyến chính xác. Tuy nhiên, để cập nhật bảng định tuyến của host, nó sẽ gửi một thông điệp chuyển hướng đến host.

Khái niệm về *chuyển hướng* được thể hiện trong hình 4.27. Host A muốn gửi một gói dữ liệu tới host B. Bộ định tuyến R2 rõ ràng là sự lựa chọn định tuyến hiệu quả nhất, nhưng host A không chọn bộ định tuyến R2. Thay vì vậy thì gói tin đi đến R1. R1, sau khi tham khảo bảng của nó, phát hiện ra rằng các gói tin phải nên đi đến R2. Nó sẽ gửi các gói tin đến R2 và, đồng thời, gửi một thông điệp chuyển hướng đến host A. Host A của bảng định tuyến có thể được cập nhật tại thời điểm này.



Hình 4.27 Khái niệm chuyển hướng

Định dạng của thông điệp chuyển hướng được thể hiện trong hình 4.28. Lưu ý rằng địa chỉ IP của bộ định tuyến phù hợp (đích) được chỉ ra trong hàng thứ hai.

Type: 5	Code: 0 to 3	Checksum
Địa chỉ IP của bộ định tuyến đích		
Phần gói tin IP datagram nhận được bao gồm tiêu đề IP cộng với 8 byte đầu tiên của dữ liệu gói tin		

Hình 4.28 Định dạng thông điệp chuyển hướng

Mặc dù thông điệp chuyển hướng được xem như là một thông điệp thông báo lỗi nhưng nó khác với các thông báo lỗi khác. Bộ định tuyến không loại bỏ các gói tin mà trong trường hợp này gói tin sẽ được gửi đến bộ định tuyến thích hợp. Trường *code* cho thông điệp chuyển hướng như sau:

- **Code 0:** Chuyển hướng cho một tuyến đường mạng cụ thể.
- **Code 1:** Chuyển hướng cho một tuyến đường host cụ thể.
- **Code 2:** Chuyển hướng cho một tuyến đường mạng cụ thể dựa trên một loại dịch vụ nhất định.
- **Code 3:** Chuyển hướng cho một tuyến đường host cụ thể dựa trên một loại dịch vụ nhất định.

c. Các thông điệp truy vấn

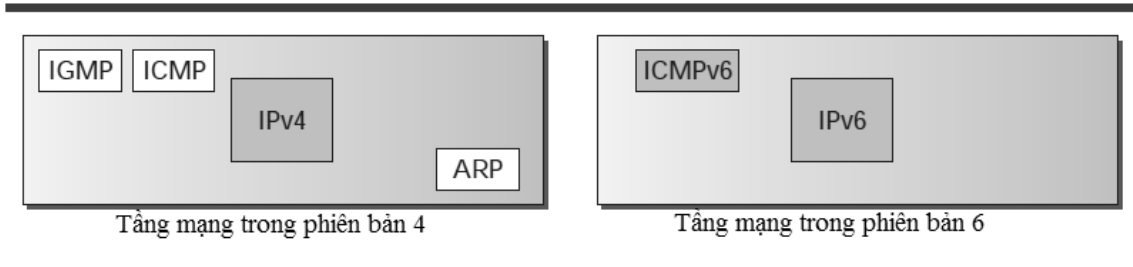
Ngoài báo cáo lỗi, ICMP cũng có thể dự đoán một số vấn đề về mạng. Điều này được thực hiện thông qua các thông điệp truy vấn. Một nhóm năm cặp thông điệp khác nhau đã được thiết kế cho mục đích này, nhưng ba trong số các cặp này đã bị bỏ. Hiện giờ chỉ có hai cặp được sử dụng: echo yêu cầu và trả lời, và timestamp yêu cầu và phát lại. Với loại thông điệp ICMP này, một nút gửi một thông điệp và được nút đích trả lời bằng một thông điệp có định dạng cụ thể.

d. Checksum: Trong ICMP, checksum được tính trên toàn bộ thông điệp (cả phần tiêu đề và phần dữ liệu).

4.3.2 Giao thức ICMPv6

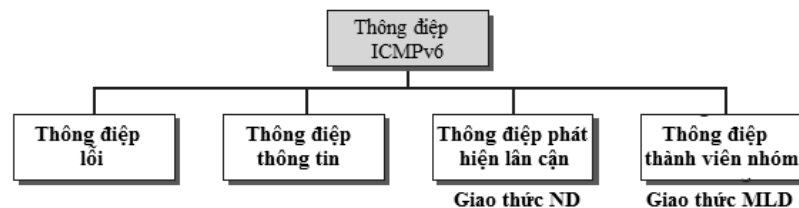
Một giao thức đã được sửa đổi trong phiên bản 6 của chồng giao thức TCP/IP là ICMP. Phiên bản *Internet Control Message Protocol version 6* (ICMPv6) mới này, có chiến lược và mục đích giống như phiên bản 4. Tuy nhiên ICMPv6 phức tạp hơn ICMPv4: một số giao thức

độc lập trong phiên bản 4 giờ là một phần của ICMPv6 và một số thông điệp mới đã được thêm vào. Hình 4.29 so sánh tầng mạng của phiên bản 4 với phiên bản 6. Các giao thức ICMP, ARP (*Address Resolution protocol*), và IGMP (*Internet Group Management Protocol*) trong phiên bản 4 được kết hợp thành một giao thức duy nhất, ICMPv6.



Hình 4.29 So sánh tầng mạng trong phiên bản 4 và phiên bản 6

ICMPv6, giống như ICMPv4, là giao thức hướng thông điệp. Nó sử dụng các thông điệp để báo lỗi, lấy thông tin, thăm dò lân cận, hoặc quản lý truyền thông multicast. Tuy nhiên, một vài giao thức khác được thêm vào ICMPv6 để xác định các chức năng và biên dịch các thông điệp. Có một vài chiến lược khác nhau để nhóm các thông điệp ICMPv6. Một số các thông điệp như: thông điệp ICMPv6, thông điệp ND hoặc thông điệp MLD. Các thông điệp ICMPv6 được phân loại theo chức năng và vai trò. Trong phần mô tả mỗi loại sẽ đề cập và mô tả các giao thức tương ứng đã được thêm vào cho các chức năng và giải thích các thông điệp. Để có được phân loại này, tất cả các thông điệp có cùng một loại định dạng và các loại thông điệp đều được xử lý bởi các giao thức ICMPv6. Các giao thức khác như ND (*Neighbor Discovery*) và MLD (*Multicast Listener Discovery*) hoạt động dưới giao thức ICMPv6. Hình 4.30 thể hiện cách phân loại các thông điệp ICMPv6.

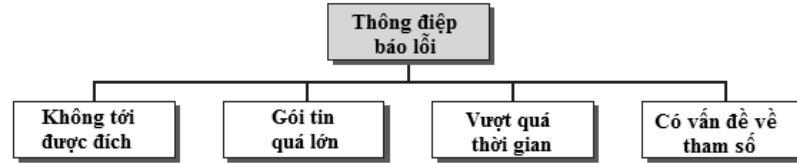


Hình 4.30 Phân loại các thông điệp ICMPv6

Hình 4.30 cho thấy hai nhóm thông điệp được gửi và nhận dưới sự kiểm soát của giao thức ND hoặc giao thức MLD.

4.3.2.1 Các thông điệp báo lỗi

Như trong thảo luận về phiên bản 4, một trong những trách nhiệm chính của ICMP là báo cáo lỗi. Trong phiên bản 6, bốn loại lỗi được xử lý: không tới được đích, gói tin quá lớn, vượt quá thời gian, và các vấn đề tham số (xem hình 4.31). Lưu ý rằng thông điệp ngắt nguồn, được sử dụng để kiểm soát tắc nghẽn trong phiên bản 4, đã bị loại bỏ trong phiên bản này bởi vì các trường ưu tiên và trường nhãn luồng trong IPv6 được dùng để xử lý tắc nghẽn. Thông điệp chuyển hướng đã chuyển từ loại báo lỗi sang loại tìm kiếm lân cận (láng giềng), do vậy thông điệp này như một phần của thông điệp phát hiện lân cận.



Hình 4.31 Các thông điệp báo lỗi

ICMPv6 tạo một gói tin báo lỗi, sau đó đóng gói trong một gói tin IPv6. Tiếp đó gói tin được gửi đến nguồn ban đầu của gói tin lỗi.

4.3.2.2 Các thông điệp thông tin

Hai trong số các thông điệp ICMPv6 có thể được phân loại như thông điệp thông tin: thông điệp *echo request* và *echo response*. Các thông điệp *echo request* và *echo response* được thiết kế để kiểm tra xem hai thiết bị trên mạng Internet có thể giao tiếp với nhau hay không. Một host hoặc bộ định tuyến có thể gửi một thông điệp *echo request* sang host khác; host nhận hoặc bộ định tuyến có thể trả lời bằng cách sử dụng thông điệp *echo response*.

4.3.2.3 Các thông điệp phát hiện lân cận

Một số thông điệp trong ICMPv4 đã được định nghĩa lại trong ICMPv6 để xử lý các vấn đề phát hiện lân cận. Một số thông điệp mới cũng được thêm vào để cung cấp phần mở rộng. Vấn đề quan trọng nhất là định nghĩa của hai giao thức mới xác định rõ chức năng của các thông điệp nhóm: các giao thức *Neighbor-Discovery (ND)* và *Inverse-Neighbor-Discovery (IND)*. Hai giao thức được sử dụng bởi các nút (host hoặc bộ định tuyến) trên cùng liên kết (mạng) cho ba mục đích chính:

1. Host sử dụng giao thức ND để tìm bộ định tuyến lân cận để chuyển tiếp các gói tin.
2. Nút sử dụng giao thức ND để tìm các địa chỉ lớp liên kết của các lân cận (các nút trên cùng một mạng).
3. Nút sử dụng giao thức IND để tìm các địa chỉ IPv6 của lân cận.

4.3.2.4 Các thông điệp thành viên nhóm

Việc quản lý xử lý phân phối multicast trong IPv4 được giao cho giao thức IGMPv3. Trong IPv6, trách nhiệm này được trao cho giao thức *Multicast Listener Delivery (MLD)*. MLDv1 là bản sao của IGMPv2; MLDv2 là bản sao của IGMPv3 (tham khảo tài liệu RFC 3810). Ý tưởng cũng giống như IGMPv3, nhưng các kích thước và định dạng của thông điệp đã được thay đổi để phù hợp với kích thước lớn hơn địa chỉ multicast trong IPv6. Giống như IGMPv3, MLDv2 có hai loại thông điệp: thông điệp *membership-query* và thông điệp *membership-report*. Thông điệp *membership-query* có thể được chia thành ba loại: *general*, *group-specific*, và *group-and-source specific*.

4.4 GIAO THỨC ARP (Address Resolution Protocol)

4.4.1 Ánh xạ địa chỉ

Internet được xây dựng nên từ sự kết hợp của các mạng vật lý kết nối với nhau bởi liên mạng các thiết bị như bộ định tuyến. Các gói tin bắt đầu từ một host nguồn có thể đi qua các mạng vật lý khác nhau trước khi đạt đến đích.

Các host và thiết bị định tuyến được nhận ra ở cấp độ mạng nhờ các địa chỉ logic của chúng. Một địa chỉ logic là một địa chỉ liên mạng. Địa chỉ này là duy nhất, sử dụng trên toàn cầu. Nó được gọi là một địa chỉ logic bởi vì nó thường được thực hiện trong phần mềm. Mọi giao thức khi làm việc với các mạng cần sử dụng các địa chỉ logic. Các địa chỉ logic trong chồng giao thức TCP/IP được gọi là địa chỉ IP và dài 32 bit.

Tuy nhiên, các gói tin phải đi qua mạng vật lý để tới các host và thiết bị định tuyến. Ở cấp độ vật lý, host và thiết bị định tuyến được nhận ra bởi địa chỉ vật lý của chúng. Một địa chỉ vật lý là một địa chỉ cục bộ. Địa chỉ này được gọi là địa chỉ vật lý bởi vì nó thường (nhưng không phải luôn luôn) được thực hiện trong phần cứng. Ví dụ về các địa chỉ vật lý là địa chỉ MAC 48 bit trong giao thức Ethernet, được in trên NIC (*Network Interface Card*) và được cài đặt trong các host hoặc bộ định tuyến.

Địa chỉ vật lý và địa chỉ logic là hai định danh khác nhau. Chúng ta cần cả hai vì một mạng vật lý như Ethernet có thể có hai giao thức khác nhau tại tầng mạng như IP và IPX (Novell) cùng một lúc. Tương tự như vậy, một gói tin trong một tầng mạng như IP có thể đi qua các mạng vật lý khác nhau như Ethernet và LocalTalk (Apple).

Điều này có nghĩa là vận chuyển một gói tin đến một host hoặc một bộ định tuyến đòi hỏi hai mức độ giải quyết: logic và vật lý. Chúng ta cần phải có khả năng ánh xạ một địa chỉ logic thành địa chỉ vật lý tương ứng của nó và ngược lại. Việc này có thể được thực hiện bằng cách sử dụng ánh xạ động hay tĩnh.

a. Ánh xạ tĩnh

Ánh xạ tĩnh có nghĩa là tạo một bảng liên kết một địa chỉ logic với một địa chỉ vật lý. Bảng này được lưu trữ trong các máy trên mạng. Mỗi máy biết địa chỉ IP của máy khác nhưng không biết địa chỉ vật lý có thể tìm trong bảng. Điều này có một số hạn chế vì địa chỉ vật lý có thể thay đổi theo những cách sau:

1. Một máy có thể thay đổi NIC của nó, dẫn đến có một địa chỉ vật lý mới.
2. Trong một số mạng LAN, như LocalTalk, địa chỉ vật lý thay đổi mỗi khi máy tính được bật.
3. Máy tính di động có thể di chuyển đến từ một mạng vật lý khác, dẫn đến sự thay đổi về địa chỉ vật lý của nó.

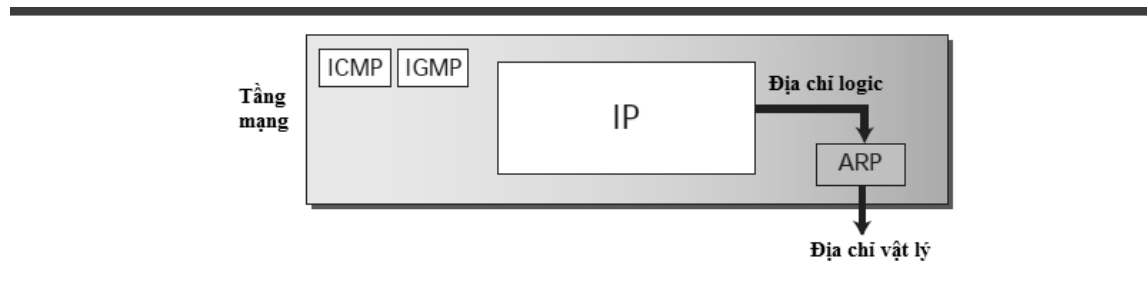
Để có thể phù hợp với những thay đổi này, một bảng ánh xạ tĩnh phải được cập nhật định kỳ, việc này có thể ảnh hưởng đến hiệu suất mạng.

b. Ánh xạ động

Trong ánh xạ động, mỗi khi một máy biết địa chỉ logic của máy khác, nó có thể sử dụng một giao thức để tìm địa chỉ vật lý. Hai giao thức được thiết kế để thực hiện ánh xạ động là *giao thức phân giải địa chỉ ARP (Address Resolution Protocol)* và *giao thức phân giải địa chỉ ngược RARP (Reverse Address Resolution Protocol)*. ARP ánh xạ một địa chỉ logic thành một địa chỉ vật lý; RARP ánh xạ một địa chỉ vật lý thành một địa chỉ logic. Do RARP đã được thay thế bằng giao thức khác nên trong phần này sẽ chỉ thảo luận về giao thức ARP.

4.4.2 Giao thức ARP

Bất cứ khi nào một host hoặc một bộ định tuyến cần gửi một gói tin IP đến một host hoặc bộ định tuyến khác, nó cần phải biết được địa chỉ logic (IP) của bên nhận. Tuy nhiên, gói tin IP lại được đóng gói trong một khung (frame) để có thể đi qua các mạng vật lý. Điều này có nghĩa là bên gửi cần địa chỉ vật lý của bên nhận hay cần phải có một ánh xạ gán một địa chỉ logic thành một địa chỉ vật lý. Hình 4.32 cho thấy vị trí của ARP trong chồng giao thức TCP/IP. ARP nhận một địa chỉ logic của giao thức IP, ánh xạ địa chỉ tới địa chỉ vật lý tương ứng và chuyển nó tới tầng liên kết dữ liệu.



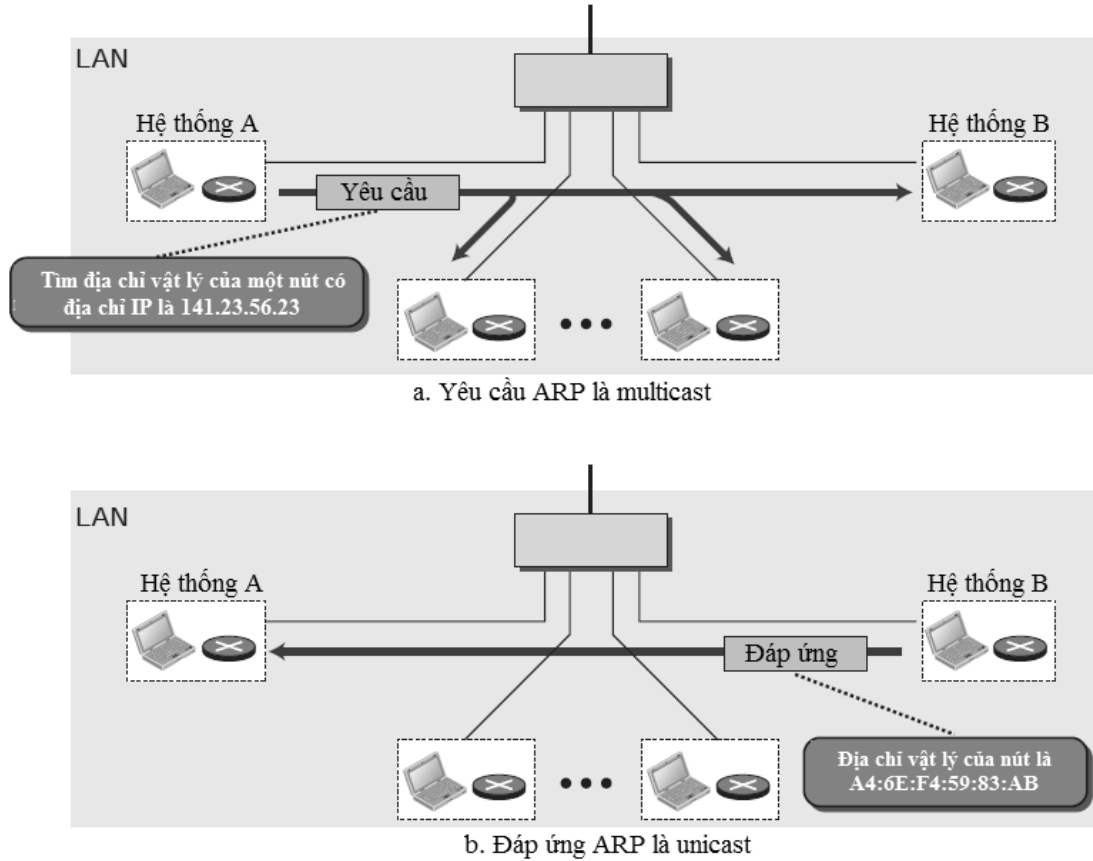
Hình 4.32 Vị trí của ARP trong chồng giao thức TCP/IP

ARP liên kết một địa chỉ IP với địa chỉ vật lý của nó. Trên một mạng vật lý điển hình, chẳng hạn như một mạng LAN, mỗi thiết bị trên một liên kết được xác định bởi một địa chỉ vật lý hoặc địa chỉ trạm, thường được in trên NIC. Bất cứ khi nào một host, hoặc một bộ định tuyến cần tìm địa chỉ vật lý của một host hoặc bộ định tuyến trên mạng của nó, nó sẽ gửi một gói tin truy vấn ARP. Các gói tin bao gồm địa chỉ vật lý và địa chỉ IP của bên gửi, và địa chỉ IP của bên nhận. Bởi vì bên gửi không biết địa chỉ vật lý của bên nhận nên truy vấn sẽ được quảng bá (*multicast*) trên mạng (xem hình 4.33).

Mỗi host hoặc bộ định tuyến trên mạng tiếp nhận và xử lý các gói tin truy vấn ARP, nhưng chỉ máy nhận nhận ra địa chỉ IP của nó và gửi lại một gói tin trả lời ARP. Gói tin trả lời có chứa IP và địa chỉ vật lý của bên nhận. Các gói tin được truyền *unicast* (riêng) trực tiếp cho bên yêu cầu sử dụng địa chỉ vật lý nhận được từ gói tin truy vấn.

Trong hình 4.33a, hệ thống bên trái (A) có một gói tin cần phải được chuyển giao cho một hệ thống khác (B) với địa chỉ IP 141.23.56.23. Hệ thống A cần truyền gói tin đến tầng liên kết dữ liệu của nó để truyền đi thực sự, nhưng nó không biết địa chỉ vật lý của bên nhận. Nó sẽ sử dụng dịch vụ của ARP bằng cách yêu cầu giao thức ARP gửi một gói tin ARP yêu cầu quảng bá để tìm địa chỉ vật lý của hệ thống với địa chỉ IP là 141.23.56.23.

Gói tin này được nhận bởi tất cả các hệ thống trên mạng vật lý, nhưng chỉ có hệ thống B sẽ trả lời nó (như thể hiện trong hình 4.33b). Hệ thống B sẽ gửi một gói tin trả lời ARP bao gồm địa chỉ vật lý của nó. Sau đó hệ thống A có thể gửi tất cả các gói dữ liệu đến đích bằng cách sử dụng địa chỉ vật lý đã nhận được.



Hình 4.33 Hoạt động của ARP

a. Định dạng gói tin ARP

Hình 4.34 cho thấy định dạng của một gói ARP. Các trường như sau:

- **Hardware type (Kiểu phần cứng):** Đây là một trường 16 bit xác định loại mạng mà ARP đang chạy. Mỗi mạng LAN được gán một số nguyên dựa trên kiểu của nó. Ví dụ, Ethernet được gán là loại 1. ARP có thể được sử dụng trên bất kỳ mạng vật lý nào.
- **Protocol type (Loại giao thức):** Đây là một trường 16 bit định nghĩa giao thức. Ví dụ, giá trị của trường này trong giao thức IPv4 là 0800₁₆. ARP có thể được sử dụng với các giao thức cấp cao hơn.
- **Hardware length (Độ dài phần cứng):** Đây là một trường 8 bit xác định độ dài của địa chỉ vật lý sử dụng đơn vị byte. Ví dụ, Ethernet sử dụng giá trị là 6.
- **Protocol length (Độ dài giao thức):** Đây là một trường 8 bit xác định chiều dài của địa chỉ logic trong byte. Ví dụ, giá trị là 4 đối với giao thức IPv4.

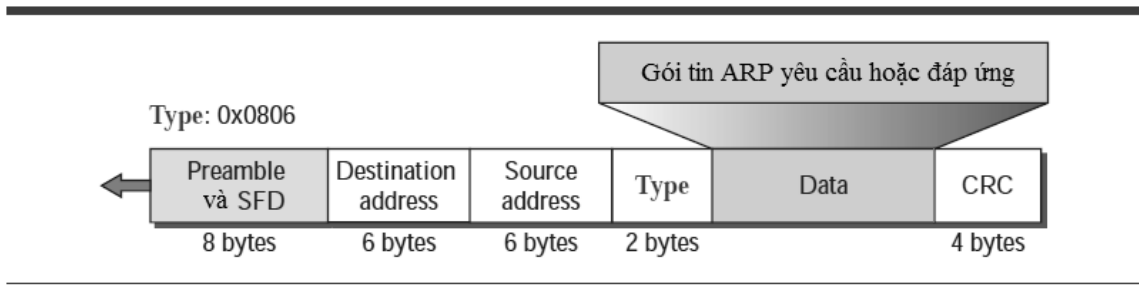
Hardware Type		Protocol Type
Hardware length	Protocol length	Operation Request 1, Reply 2
Sender hardware address (Ví dụ, 6 byte cho Ethernet)		
Sender protocol address (Ví dụ, 4 byte cho IP)		
Target hardware address (Ví dụ, 6 byte cho Ethernet) (Không điền trong Yêu cầu)		
Target protocol address (Ví dụ, 4 byte cho IP)		

Hình 4.34 Gói tin ARP

- *Operation (Hoạt động)*: Đây là một trường 16 bit xác định loại gói tin. Hai loại gói tin được định nghĩa: ARP request 1 (yêu cầu), ARP reply 2 (đáp ứng).
- *Sender hardware address (Địa chỉ phần cứng bên gửi)*: Đây là một trường có chiều dài thay đổi xác định địa chỉ vật lý của bên gửi. Ví dụ, đối với Ethernet trường này dài 6 byte.
- *Sender protocol address (Địa chỉ giao thức bên gửi)*: Đây là một trường có chiều dài thay đổi xác định địa chỉ logic của bên gửi (ví dụ, IP). Đối với giao thức IP, trường này dài 4 byte.
- *Target hardware address (Địa chỉ phần cứng đích)*: Đây là một trường có chiều dài thay đổi xác định địa chỉ vật lý đích. Ví dụ, đối với Ethernet trường này dài 6 byte. Đối với một thông điệp ARP request, trường này là tất cả số 0 vì bên gửi không biết địa chỉ vật lý của phía đích.
- *Target protocol address (Địa chỉ giao thức đích)*: Đây là một trường chiều dài thay đổi xác định địa chỉ hợp lý logic của máy đích (ví dụ, IP). Đối với giao thức IPv4, trường này dài 4 byte.

b. Đóng gói

Một gói tin ARP được đóng gói trực tiếp vào một khung (frame) liên kết dữ liệu. Ví dụ, trong hình 4.35 một gói tin ARP được đóng gói trong một khung Ethernet. Lưu ý rằng trường *type* cho biết dữ liệu trong khung là một gói tin ARP.



Hình 4.35 Đóng gói gói tin ARP

c. Hoạt động

Phần này trình bày cách thức ARP hoạt động trên một mạng Internet điển hình. Phần đầu mô tả các bước liên quan, sau đó, sẽ thảo luận về bốn trường hợp trong đó một host hoặc bộ định tuyến cần phải sử dụng ARP.

Có bảy bước liên quan đến một quá trình ARP:

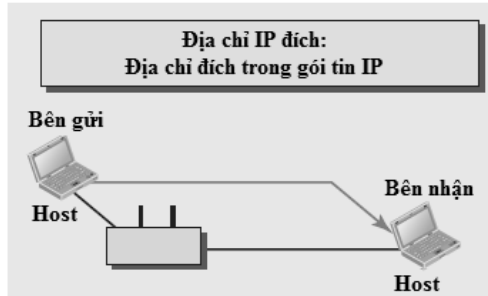
1. Bên gửi cần biết địa chỉ IP đích.
2. IP yêu cầu ARP tạo ra một thông điệp ARP request, thêm vào địa chỉ vật lý bên gửi, địa chỉ IP bên gửi và địa chỉ IP đích. Trường địa chỉ vật lý đích được điền đầy toàn số 0.
3. Thông điệp này được truyền tới tầng liên kết dữ liệu mà nó được đóng gói trong một khung bằng cách sử dụng địa chỉ vật lý của bên gửi là địa chỉ nguồn và địa chỉ quảng bá vật lý là địa chỉ đích.
4. Tất cả các host và các bộ định tuyến đều nhận được khung. Bởi vì khung chứa một địa chỉ đích quảng bá nên tất cả các trạm loại bỏ thông điệp và chuyển nó tới ARP. Tất cả các máy (ngoại trừ đích) sẽ hủy các gói tin. Máy đích nhận diện được địa chỉ IP.
5. Máy đích trả lời với một thông điệp ARP reply có chứa địa chỉ vật lý của nó. Thông điệp này là unicast.
6. Bên gửi nhận được tin nhắn trả lời. Lúc này nó biết được địa chỉ vật lý của máy đích.
7. Gói tin IP mang dữ liệu cho máy đích sẽ được đóng gói trong một khung và truyền unicast đến đích.

Sau đây là bốn trường hợp khác nhau mà các dịch vụ của ARP có thể được sử dụng (xem hình 4.36).

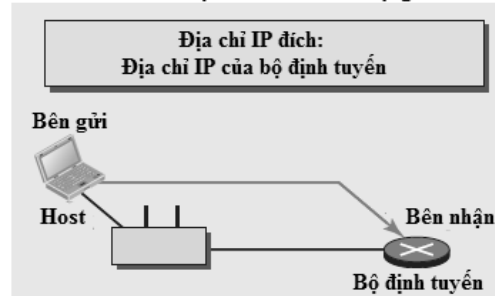
1. Bên gửi là một host và muốn gửi một gói tin đến host khác trên cùng mạng. Trong trường hợp này, địa chỉ logic phải được ánh xạ tới một địa chỉ vật lý là địa chỉ IP đích trong tiêu đề gói tin.
2. Bên gửi là một host và muốn gửi một gói tin đến một host trên mạng khác. Trong trường hợp này, host nhìn vào bảng định tuyến của nó và tìm thấy địa chỉ IP của hop tiếp theo (bộ định tuyến) cho điểm đến này. Nếu nó không có bảng định tuyến, nó sẽ tìm kiếm địa chỉ IP của bộ định tuyến mặc định. Địa chỉ IP của bộ định tuyến trở thành địa chỉ logic cần phải được ánh xạ tới một địa chỉ vật lý.

3. Bên gửi là một bộ định tuyến đã nhận được một gói tin dành cho một host trên mạng khác. Nó kiểm tra bảng định tuyến của nó và tìm thấy địa chỉ IP của bộ định tuyến tiếp theo. Địa chỉ IP của bộ định tuyến tiếp theo trở thành địa chỉ logic cần phải được ánh xạ tới một địa chỉ vật lý.
4. Bên gửi là một bộ định tuyến đã nhận được gói tin dành cho một host trong cùng mạng. Địa chỉ IP đích của gói tin cần phải được ánh xạ thành một địa chỉ vật lý.

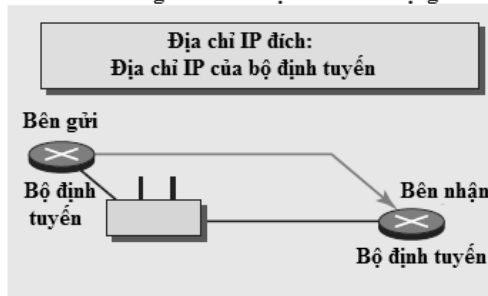
Trường hợp 1: Một host cần gửi một gói tin đến một host khác trên cùng mạng.



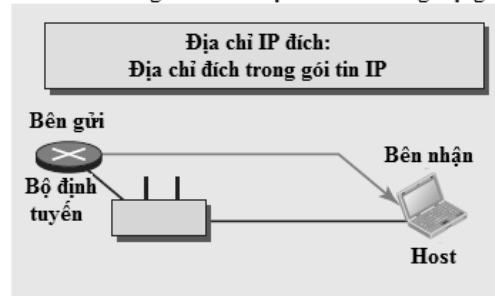
Trường hợp 2: Một host cần gửi một gói tin đến một host khác trên mạng khác



Trường hợp 3: Một bộ định tuyến cần gửi một gói tin đến một host trên cùng mạng



Trường hợp 4: Một bộ định tuyến cần gửi một gói tin đến một host trên cùng mạng



Hình 4.36 Bốn trường hợp sử dụng ARP

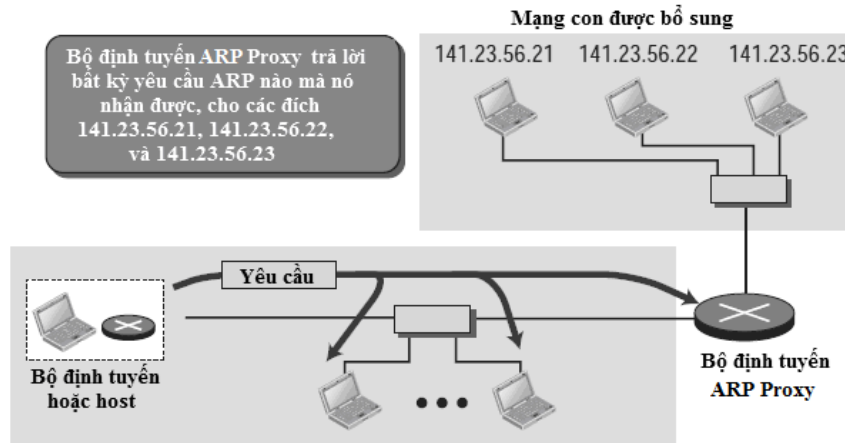
d. ARP Proxy

ARP Proxy được sử dụng để tạo ra hiệu ứng *subnetting*. ARP proxy là một ARP đóng vai trò đại diện cho một tập hợp các host. Bất cứ khi nào bộ định tuyến chạy ARP proxy nhận được yêu cầu ARP tìm kiếm địa chỉ IP của một trong các host, bộ định tuyến sẽ gửi một ARP trả lời thông tin phản cứng địa chỉ vật lý của nó. Sau khi bộ định tuyến nhận được các gói tin IP, nó sẽ gửi các gói tin đến host hoặc bộ định tuyến thích hợp.

Hình 4.37 trình bày một ví dụ: ARP được cài đặt trên host bên phải sẽ chỉ trả lời một yêu cầu ARP với địa chỉ IP đích của 141.23.56.23.

Tuy nhiên, người quản trị có thể cần phải tạo ra một mạng con mà không thay đổi toàn bộ hệ thống để nhận ra các địa chỉ mạng con. Giải pháp là thêm một bộ định tuyến chạy một ARP proxy. Trong trường hợp này, các bộ định tuyến hoạt động thay mặt cho tất cả các host được cài đặt trên mạng con. Khi nhận được một yêu cầu ARP với địa chỉ IP mục tiêu trùng với địa chỉ của một trong những máy (141.23.56.21, 141.23.56.22 và 141.23.56.23), nó sẽ gửi

một ARP trả lời và thông báo địa chỉ phần cứng của nó như là địa chỉ phần cứng đích. Khi bộ định tuyến nhận được các gói tin IP, nó sẽ gửi các gói tin đến host thích hợp.



Hình 4.37 ARP Proxy

4.5 CÁC GIAO THỨC ĐỊNH TUYẾN: RIP VÀ OSPF

RIP (Routing Information Protocol) và *OSPF (Open Shortest Path First)* là hai giao thức định tuyến unicast phổ biến nhất trên Internet.

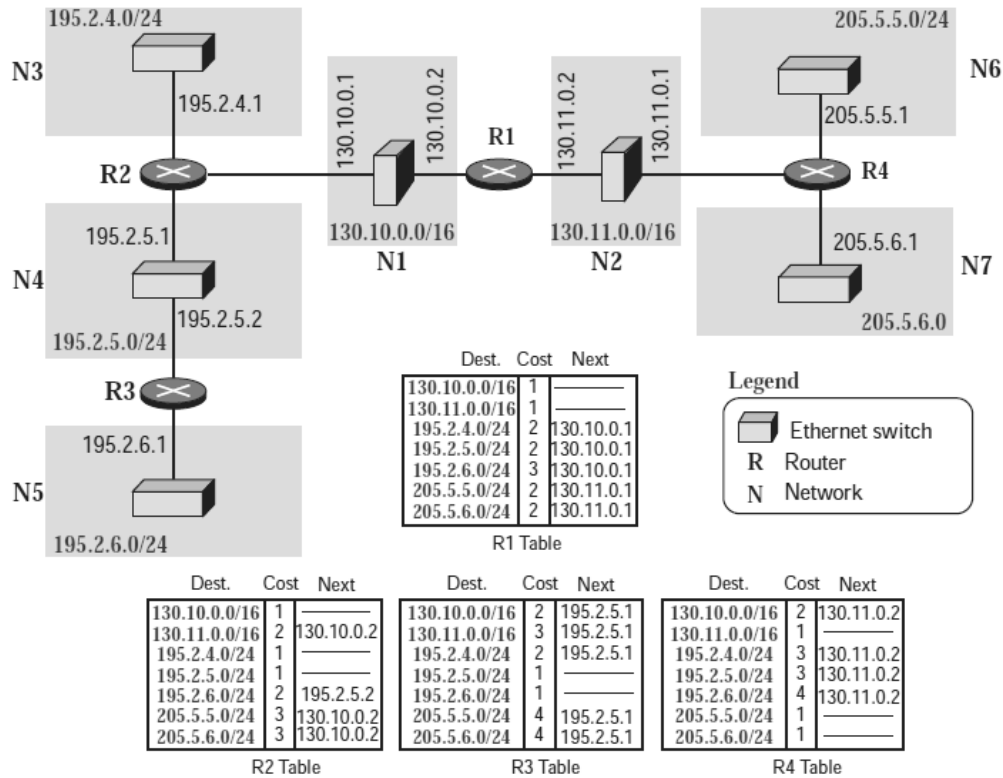
4.5.1 RIP (Routing Information Protocol)

RIP là một giao thức định tuyến ngoại vùng (*intra-domain*) được sử dụng trong một hệ thống tự trị (*Autonomous system, AS*). Đây là một giao thức rất đơn giản dựa trên định tuyến vector khoảng cách. RIP thực hiện định tuyến vector khoảng cách trực tiếp với một số xem xét sau:

1. Hệ thống tự trị làm việc với các bộ định tuyến và các mạng (các liên kết), hay còn được gọi là một nút.
2. Đích trong một bảng định tuyến là một mạng, có nghĩa là cột đầu tiên xác định địa chỉ mạng.
3. Ma trận được sử dụng bởi RIP rất đơn giản; khoảng cách được định nghĩa là số lượng liên kết (mạng) có thể được sử dụng để đi đến đích. Vì lý do này, ma trận trong RIP được gọi là *hop count* (đếm số hop).
4. Giới hạn số hop là 16, nghĩa là tất cả đường đi trong hệ thống tự trị sử dụng RIP không thể có nhiều hơn 15 hop.
5. Cột nút tiếp theo xác định địa chỉ của bộ định tuyến mà gói tin sẽ được gửi đi để đến được đích của nó.

Hình 4.38 cho thấy một hệ thống tự trị với bảy mạng và bốn bộ định tuyến. Bảng của mỗi bộ định tuyến cũng được hiển thị. Xem bảng định tuyến cho R1: Bảng này có bảy mục hiển thị cách thức để đi đến mỗi mạng trong hệ thống tự trị. Bộ định tuyến R1 được kết nối trực tiếp đến mạng 130.10.0.0 và 130.11.0.0, nghĩa là không có chỉ mục *hop* tiếp theo cho hai mạng này. Để gửi một gói tin đến một trong ba mạng ở phía bên trái, bộ định tuyến R1 cần

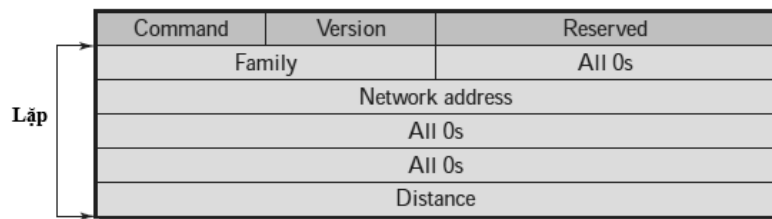
phải chuyển những gói tin đến R2. Các mục nút tiếp theo cho ba mạng này là giao diện của bộ định tuyến R2 với địa chỉ IP 130.10.0.1. Để gửi một gói tin đến hai mạng ở phía bên phải, bộ định tuyến R1 cần phải gửi gói tin đến giao diện của bộ định tuyến R4 với địa chỉ IP 130.11.0.1. Các bảng khác có thể được giải thích tương tự.



Hình 4.38 Ví dụ một miền sử dụng RIP

a. Định dạng thông điệp RIP

Định dạng thông điệp RIP được mô tả trong hình 4.39.



Hình 4.39 Định dạng thông điệp RIP (thông điệp phản hồi)

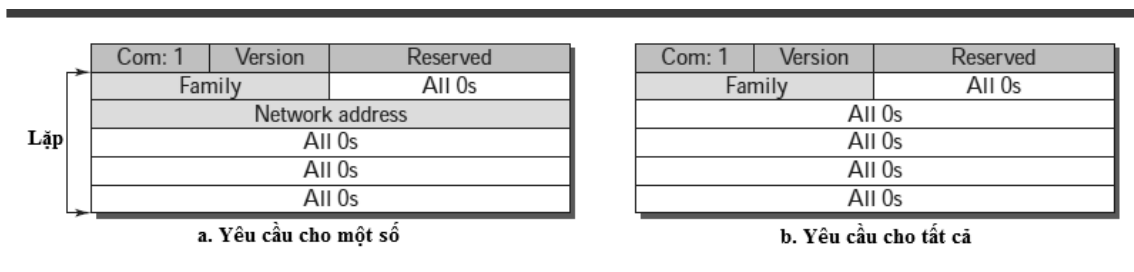
- Command:** Trường 8 bit này quy định các loại thông điệp: yêu cầu (1) hoặc phản hồi (2).
- Version:** Trường 8 bit này xác định các phiên bản (1 hoặc 2).
- Family:** Trường 16 bit này xác định họ giao thức được sử dụng. Đối với giao thức TCP/IP giá trị là 2.

- **Network address:** Trường địa chỉ xác định địa chỉ của mạng đích. RIP đã phân bổ 14 byte cho trường này được áp dụng với bất kỳ giao thức. Tuy nhiên, IP hiện đang sử dụng chỉ có 4 byte. Phần còn lại của địa chỉ được điền đầy các số 0.
- **Distance:** Trường 32 bit này xác định số hop (chi phí) từ các bộ định tuyến quảng cáo cho mạng đích. Lưu ý rằng một phần của thông điệp được lặp lại cho từng mạng đích (được gọi là mục).

b. Thông điệp yêu cầu và phản hồi

RIP có hai loại thông điệp: yêu cầu và phản hồi.

- **Thông điệp yêu cầu:** Thông điệp yêu cầu được gửi bởi một bộ định tuyến mới hoạt động hoặc một bộ định tuyến có một số mục bị time-out (hết thời gian). Có thể yêu cầu truy vấn về một mục cụ thể hoặc tất cả các mục (xem hình 4.40).



Hình 4.40 Thông điệp yêu cầu

- **Thông điệp phản hồi:** là thông điệp được dùng để trả lời cho một yêu cầu (chứa thông tin về các đích được chỉ định trong yêu cầu tương ứng), hoặc là thông điệp được gửi định kỳ mỗi 30 giây hoặc khi có sự thay đổi trong bảng định tuyến (cập nhật thông tin). Hình 4.39 là định dạng thông điệp phản hồi.

c. Bộ định thời trong RIP

RIP sử dụng ba bộ định thời để hỗ trợ hoạt động của nó: *bộ đếm thời gian định kỳ* (Periodic: 25-35s) kiểm soát việc gửi thông điệp; *bộ đếm thời gian hết hạn* (Expiration: 180s) kiểm soát giá trị của một tuyến đường, và *bộ đếm thời gian thu gom rác* (Garbage collection: 120s) sẽ quảng bá về lỗi của một tuyến đường.

- **Bộ đếm thời gian định kỳ:** Bộ đếm thời gian định kỳ kiểm soát quảng cáo của các thông điệp cập nhật thường xuyên. Mặc dù giao thức xác định bộ đếm thời gian này phải được thiết lập 30 giây nhưng trong thực tế người ta sử dụng một số ngẫu nhiên từ 25 đến 35 giây. Điều này nhằm ngăn chặn các nỗ lực đồng bộ hóa, và do đó gây ra quá tải trên Internet, khi các bộ định tuyến cập nhật cùng một lúc. Mỗi bộ định tuyến có một bộ đếm thời gian định kỳ được thiết lập một số từ 25 đến 35 một cách ngẫu nhiên. Nó thực hiện đếm ngược, khi tới 0, thông báo cập nhật được gửi đi, và bộ đếm thời gian lại được thiết lập ngẫu nhiên một lần nữa.
- **Bộ đếm thời gian hết hạn:** Bộ đếm thời gian hết hạn kiểm soát tính hợp lệ của một tuyến đường. Khi một bộ định tuyến nhận được thông tin cập nhật cho một tuyến đường, bộ đếm thời gian hết hạn được thiết lập thành 180s cho tuyến đường cụ thể.

Mỗi khi nhận một bản cập nhật mới cho các tuyến đường, bộ đếm thời gian được thiết lập lại. Trong những tình huống bình thường quá trình được thực hiện mỗi 30 giây. Tuy nhiên, nếu có một vấn đề trên Internet và không có bản cập nhật nhận được trong khoảng thời gian 180s đã xác định, tuyến đường được coi là hết hạn và số hop của tuyến đường được thiết lập là 16, có nghĩa là không thể truy cập đích đến. Mỗi tuyến đường có bộ đếm thời gian hết hạn của riêng nó.

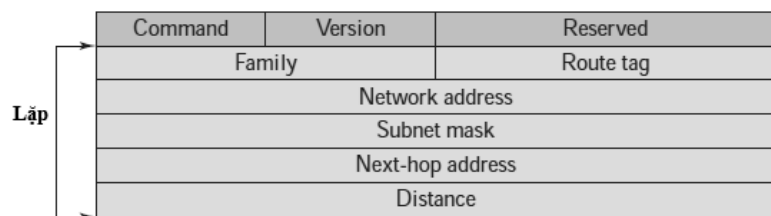
- **Bộ đếm thời gian gom rác:** Khi thông tin về một tuyến đường trở nên không hợp lệ, các bộ định tuyến không xóa bỏ tuyến đường từ bảng của nó ngay lập tức. Thay vào đó, nó tiếp tục quảng cáo tuyến đường với giá trị là 16. Đồng thời, một bộ đếm thời gian gọi là bộ đếm thời gian gom rác được thiết lập 120s cho tuyến đó. Khi giá trị đạt đến 0, tuyến đường được loại bỏ khỏi bảng. Bộ đếm thời gian này cho phép các thiết bị lân cận nhận thức được sự vô hiệu của một tuyến đường trước khi loại bỏ.

d. RIP phiên bản 2

RIP phiên bản 2 được thiết kế để khắc phục một số thiếu sót của phiên bản 1. Các nhà thiết kế của phiên bản 2 đã không tăng cường độ dài của thông điệp cho mỗi mục. Họ chỉ thay thế các trường này trong phiên bản 1, những trường được chèn toàn số 0, cho giao thức TCP/IP với một số trường mới.

Hình 4.41 cho thấy định dạng của một thông điệp RIP phiên bản 2. Các trường mới của thông điệp này như sau:

- **Route tag:** Trường này mang thông tin như số hệ thống tự trị. Nó có thể được sử dụng để cho phép RIP nhận được thông tin từ một giao thức định tuyến ngoại vùng (interdomain).
- **Subnet mask:** Đây là một trường 4 byte mang mặt nạ mạng con (hoặc tiền tố). Điều này có nghĩa rằng RIP2 hỗ trợ giải quyết vấn đề đánh địa chỉ không phân lớp và CIDR.
- **Next-hop address:** Trường này hiển thị địa chỉ của *hop* tiếp theo. Điều này đặc biệt hữu ích nếu hai hệ thống độc lập chia sẻ cùng một mạng (ví dụ, mạng backbone). Sau đó, các thông điệp có thể xác định các bộ định tuyến mà gói tin sẽ đi đến tiếp trong cùng hoặc khác hệ thống tự trị.



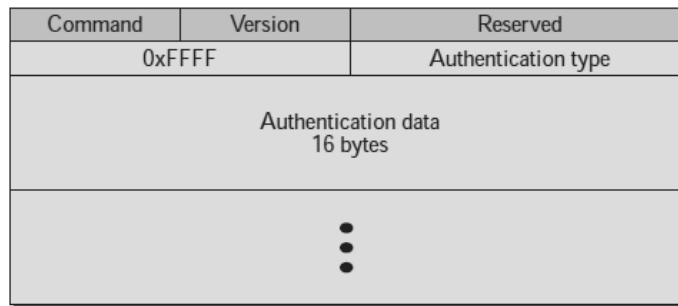
Hình 4.41 Định dạng RIP phiên bản 2

Có lẽ sự khác biệt quan trọng nhất giữa hai phiên bản của RIP là đánh địa chỉ phân lớp và không phân lớp. RIPv1 sử dụng có phân lớp. Mục duy nhất trong định dạng thông điệp là địa chỉ mạng (với một mặt nạ mặc định). RIPv2 bổ sung thêm một trường cho mặt nạ mạng con, có thể được sử dụng để xác định chiều dài tiền tố mạng. Điều này có nghĩa là trong phiên

bản này, chúng ta có thể sử dụng đánh địa chỉ không phân lớp. Một nhóm các mạng có thể được kết hợp thành một tiền tố và được quảng cáo chung.

e. Xác thực

Xác thực được thêm vào để bảo vệ chống lại các thông điệp quảng cáo trái phép. Không có trường mới nào được thêm vào gói tin; thay vào đó, mục đầu tiên của thông điệp được dành cho thông tin xác thực. Để chỉ ra rằng một mục là thông tin xác thực mà không phải thông tin định tuyến, giá trị $FFFF_{16}$ được đặt vào trường *family* (xem hình 4.42). Trường thứ hai là *authentication type* (loại xác thực), xác định các giao thức được sử dụng để xác thực, và trường thứ ba là *authentication data* chứa dữ liệu xác thực thực tế.



Hình 4.42 Xác thực

f. Multicasting (quảng bá)

Phiên bản 1 của RIP sử dụng quảng bá để gửi thông điệp RIP cho các lân cận. Bằng cách này, tất cả các bộ định tuyến trên mạng và các host đều nhận được gói tin. Mặt khác, RIP phiên bản 2 sử dụng địa chỉ multicast cho tất cả các bộ định tuyến để chỉ gửi các tin nhắn RIP tới các bộ định tuyến RIP trong mạng.

g. Đóng gói

Tin nhắn RIP được đóng trong gói dữ liệu người dùng UDP. Thông điệp RIP không chứa trường nào chỉ ra độ dài của thông điệp, mà nó có thể được xác định từ các gói tin UDP. Cổng thường gán cho RIP trong UDP là cổng 520.

4.5.2 OSPF (Open Shortest Path First)

OSPF là một giao thức định tuyến nội vùng (*intradomain*) dựa trên định tuyến trạng thái liên kết (*link state*). Phạm vi của nó cũng là một hệ thống tự trị.

a. Các vùng (hay khu vực)

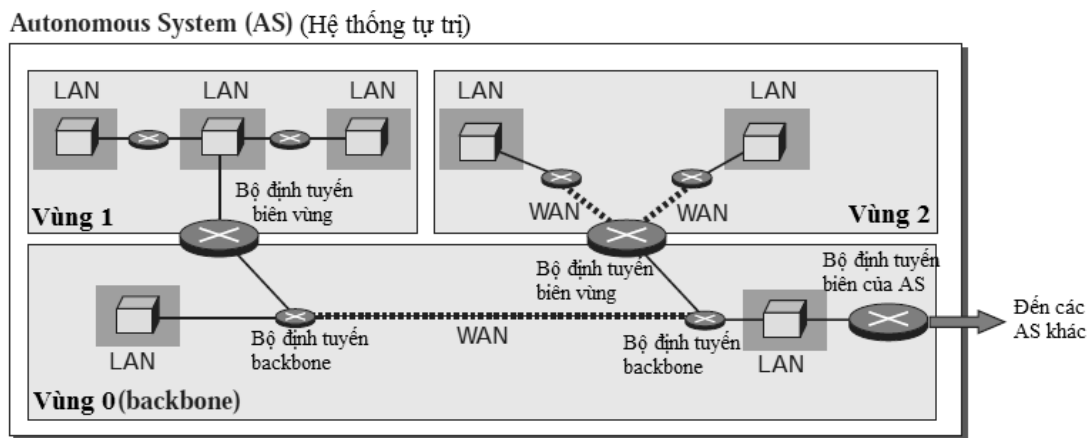
Để thực hiện định tuyến một cách hiệu quả và kịp thời, OSPF chia một hệ thống tự trị thành các vùng. Một vùng là một tập hợp các mạng, host và thiết bị định tuyến, tất cả nằm trong một hệ thống tự trị. Một hệ thống tự trị có thể được chia thành nhiều vùng khác nhau. Tất cả các mạng bên trong một vùng phải được kết nối.

Bộ định tuyến bên trong một vùng gửi thông tin định tuyến toàn bộ vùng. Tại biên của một vùng, các bộ định tuyến đặc biệt gọi là các bộ định tuyến biên vùng sẽ tổng hợp các thông

tin về vùng này và gửi cho các vùng khác. Giữa các vùng bên trong một hệ thống tự trị là một vùng đặc biệt được gọi là xương sống và tất cả các vùng bên trong một hệ thống tự trị phải được kết nối với xương sống. Nói cách khác, xương sống hoạt động như là một vùng chính và các vùng khác là các khu vực thứ cấp. Tuy nhiên, điều này không có nghĩa là các bộ định tuyến trong vùng không thể được kết nối với nhau. Các bộ định tuyến bên trong xương sống được gọi là bộ định tuyến xương sống. Lưu ý rằng một bộ định tuyến xương sống cũng có thể là một bộ định tuyến biên vùng.

Nếu vì một số vấn đề nào đó, kết nối giữa các xương sống và một vùng bị hỏng, thì một liên kết ảo giữa các bộ định tuyến phải được tạo ra bởi người quản trị để giúp cho các chức năng của xương sống liên tục được thực hiện.

Mỗi vùng có một định danh vùng. Định danh vùng của xương sống là 0. Hình 4.43 cho thấy một hệ thống tự trị và các vùng của nó.



Hình 4.43 Các vùng trong một hệ thống tự trị

b. Độ đo

Giao thức OSPF cho phép người quản trị chỉ định một chi phí, được gọi là độ đo, cho mỗi tuyến đường. Độ đo có thể dựa vào loại hình dịch vụ (chậm trễ tối thiểu, thông lượng tối đa, v.v.). Trên thực tế, mỗi bộ định tuyến có thể có nhiều bảng định tuyến, mỗi bảng dựa trên một loại dịch vụ khác nhau.

c. Loại liên kết

Trong OSPF, thuật ngữ kết nối được gọi là một liên kết. Có 4 loại liên kết là: point-to-point (điểm-điểm), transient (tạm thời), stub, và virtual (ảo).

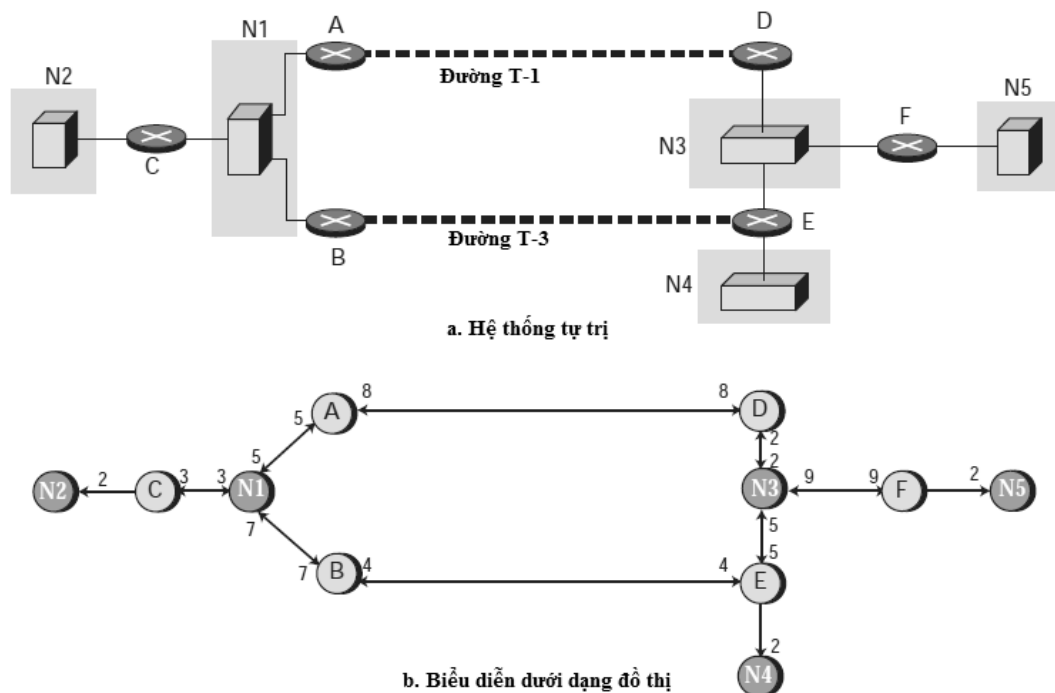
- **Liên kết Point-to-Point:** Một liên kết Point-to-point nối hai thiết bị định tuyến mà không cần bất kỳ host nào khác hoặc bộ định tuyến ở giữa. Nói cách khác, mục đích của liên kết (mạng) chỉ là để kết nối hai thiết bị định tuyến. Một ví dụ về loại liên kết này là hai bộ định tuyến nối với nhau bằng một đường dây điện thoại hoặc một T-line.
- **Liên kết transient:** Một liên kết transient là một mạng với một số thiết bị định tuyến gắn với nó. Dữ liệu có thể vào qua bất kỳ bộ định tuyến nào và ra qua bất kỳ bộ định

tuyến nào. Ví dụ cho kiểu liên kết này tất cả các mạng LAN và một số WAN với hai hoặc nhiều bộ định tuyến.

- **Liên kết stub:** Một liên kết stub là một mạng được kết nối với chỉ một bộ định tuyến. Các gói dữ liệu đi vào và đi ra mạng đều thông qua bộ định tuyến này. Đây là một trường hợp đặc biệt của mạng transient.
- **Liên kết virtual:** Khi sự liên kết giữa hai bộ định tuyến bị hỏng, người quản trị có thể tạo ra một liên kết ảo giữa chúng bằng cách sử dụng một đường dài hơn có thể đi qua một số bộ định tuyến.

d. Biểu diễn đồ thị

Một AS có thể được biểu diễn dưới dạng một đồ thị trong OSPF. Hình 4.44 cho thấy một AS nhỏ với bảy mạng và sáu bộ định tuyến, trong đó có hai mạng là point-to-point (không có định danh). A, B, C, D, E, F là các bộ định tuyến, và N1, N2, N3, N4, N5 là các mạng (đại diện bởi các bộ định tuyến được chỉ định), và OSPF coi cả hai loại này đều như là các nút.

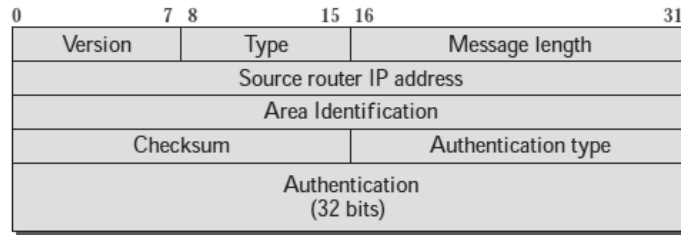


Hình 4.44 Ví dụ một AS và biểu diễn đồ họa của nó trong OSPF

e. Gói tin OSPF

OSPF sử dụng năm loại gói tin khác nhau: *hello*, *database description*, *link state*, *request*, *link state update*, và *link state acknowledgment*.

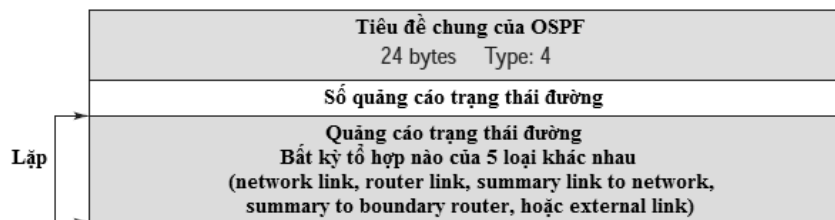
Tất cả các gói OSPF có cùng tiêu đề chung (xem hình 4.45). Các trường tiêu đề như sau:



Hình 4.45 Tiêu đề chung OSPF

- *Version (phiên bản)*: Trường 8 bit này xác định các phiên bản của giao thức OSPF. Hiện tại là phiên bản 2
- *Type (loại)*: Trường 8 bit này xác định kiểu của gói tin. Như chúng ta đã đề cập, có năm loại, với giá trị 1-5 để xác định các loại.
- *Message length (độ dài thông điệp)*: Trường 16 bit này xác định độ dài của cả thông điệp bao gồm tiêu đề.
- *Source router IP address (địa chỉ IP của bộ định tuyến nguồn)*: Trường 32 bit này xác định địa chỉ IP của bộ định tuyến gửi gói tin.
- *Area identification (định danh vùng)*: Trường 32 bit này xác định các khu vực có sự định tuyến.
- *Checksum*: Trường này được sử dụng để phát hiện lỗi trên toàn bộ gói không bao gồm trường authentication type và authentication data.
- *Authentication type (loại xác thực)*: Trường 16 bit này xác định giao thức xác thực được sử dụng trong vùng này. Tại thời điểm này, hai loại chứng thực được định nghĩa: 0 cho không có gì và 1 cho mật khẩu.
- *Authentication (xác thực)*: Trường 64 bit. Đây là giá trị thực tế của dữ liệu xác thực. Trong tương lai, khi thêm các kiểu xác thực được định nghĩa, trường này sẽ là kết quả của tính toán xác thực. Còn hiện tại, nếu kiểu xác thực là 0, trường này được điền đầy các số 0. Nếu là 1, trường này mang một mật khẩu 8 ký tự.

Phần sau sẽ trình bày cụ thể về gói tin *Link State Update (cập nhật trạng thái liên kết)*, là trung tâm của hoạt động OSPF. Gói tin này được sử dụng bởi một bộ định tuyến để quảng cáo các trạng thái liên kết của nó. Định dạng chung của gói cập nhật trạng thái liên kết được thể hiện trong hình 4.46.



Hình 4.46 Gói tin cập nhật trạng thái đường

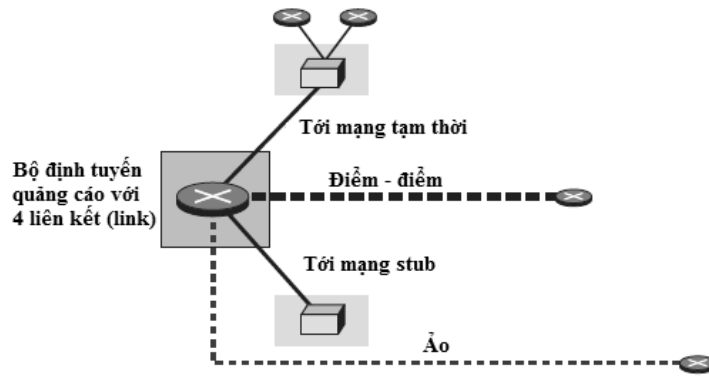
Mỗi gói cập nhật có thể chứa các LSA (*Link state advertisement*) khác nhau (gồm 5 loại như hình 4.46). Tất cả năm loại có tiêu đề chung giống nhau. Tiêu đề chung này được thể hiện trong hình 4.47 và được mô tả dưới đây:

Link state age	Reserved	E	T	Link state type
Link state ID				
Advertising router				
Link state sequence number				
Link state checksum	Length			

Hình 4.47 Tiêu đề chung của LSA

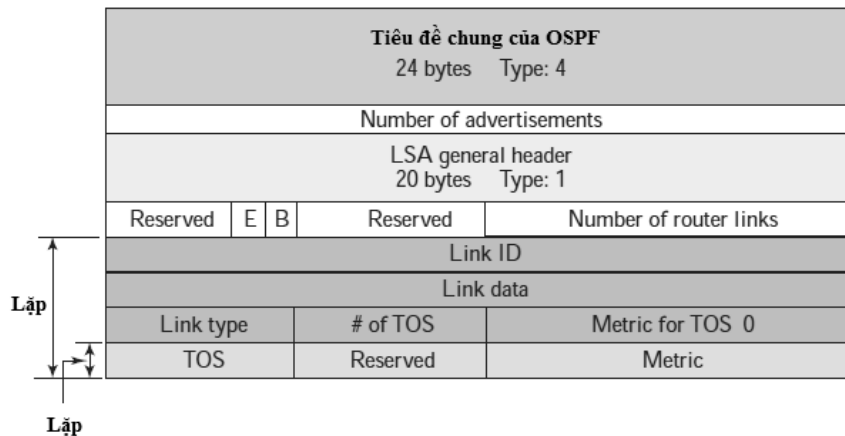
- *Link stage age*: Trường này cho biết thời gian (số giây) trôi qua kể từ thông điệp này lần đầu tiên được tạo ra. Nhớ lại rằng loại thông điệp này đi từ bộ định tuyến tới bộ định tuyến (kiểu ngập lụt). Khi một bộ định tuyến tạo ra thông điệp, giá trị của trường này là 0. Khi mỗi bộ định tuyến lần lượt chuyển tiếp thông điệp này, nó ước tính thời gian vận chuyển và thêm vào giá trị tích lũy của trường này.
- *E flag*: Nếu cờ 1 bit này được thiết lập là 1, có nghĩa là khu vực này là một khu vực stub. Một khu vực stub là một khu vực chỉ có một đường được kết nối với khu vực xương sống.
- *T flag*: Nếu cờ 1 bit này được thiết lập là 1, có nghĩa là các bộ định tuyến có thể xử lý nhiều loại dịch vụ.
- *Link stage type*: Trường này xác định loại LSA. Có năm loại quảng cáo khác nhau: (1) network link (liên kết mạng), (2) router link (liên kết bộ định tuyến), (3) summary link to network (tóm tắt liên kết tới mạng), (4) summary to boundary router (tóm tắt bộ định tuyến biên), và (5) external link (liên kết mở rộng).
- *ID link stage*: Giá trị của trường này phụ thuộc vào loại liên kết: với loại 1 thì đây là địa chỉ IP của bộ định tuyến; với loại 2 thì là địa chỉ IP của bộ định tuyến được chỉ định; với loại 3 là địa chỉ của mạng; với loại 4 là địa chỉ IP của bộ định tuyến AS biên; và với loại 5 là địa chỉ của các mạng bên ngoài.
- *Advertising router*: Đây là địa chỉ IP của bộ định tuyến quảng cáo thông điệp.
- *Link state sequence number*: Đây là một số thứ tự được gán cho mỗi thông điệp cập nhật trạng thái liên kết.
- *Link state checksum*: Đây không phải là checksum bình thường. Giá trị của trường này được tính bằng cách sử dụng checksum Fletcher, dựa trên toàn bộ gói tin ngoại trừ trường age.
- *Length*: Xác định chiều dài của toàn bộ gói theo byte.

Liên kết bộ định tuyến LSA xác định các liên kết của một bộ định tuyến thực sự. Bộ định tuyến thực sự sử dụng quảng cáo này công bố thông tin về tất cả các liên kết của nó và những gì ở phía bên kia của liên kết (hàng xóm). Hình 4.48 mô tả một liên kết bộ định tuyến.



Hình 4.48 Liên kết bộ định tuyến

Các liên kết bộ định tuyến LSA quảng cáo tất cả các liên kết của một bộ định tuyến (bộ định tuyến thực sự). Định dạng của gói tin liên kết định tuyến được thể hiện trong hình 4.49.



Hình 4.49 Định dạng gói tin liên kết bộ định tuyến LSA

Bảng 4.3 Các loại liên kết, định danh liên kết và dữ liệu liên kết

Loại liên kết	Định danh liên kết	Dữ liệu liên kết
Loại 1: Điểm-điểm	Địa chỉ của bộ định tuyến lân cận	Số giao diện
Loại 2: Tạm thời	Địa chỉ của bộ định tuyến được chỉ định	Địa chỉ bộ định tuyến
Loại 3: Stub (sơ khai)	Địa chỉ mạng	Mật nã mạng
Loại 4: Ảo	Địa chỉ của bộ định tuyến lân cận	Địa chỉ bộ định tuyến

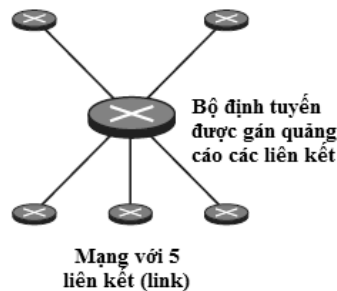
Các trường của liên kết bộ định tuyến LSA như sau:

- **Link ID:** Giá trị của trường này phụ thuộc vào loại hình liên kết. Bảng 4.3 cho thấy các định danh liên kết khác nhau dựa trên loại liên kết.
- **Link data:** Trường này cho biết thêm thông tin về liên kết. Một lần nữa, giá trị này phụ thuộc vào loại của liên kết (xem bảng 4.3).

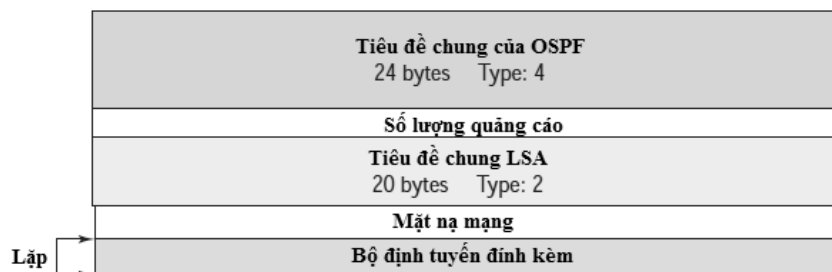
- *Link type*: Bốn loại liên kết khác nhau được xác định dựa vào loại mạng mà bộ định tuyến được kết nối (xem bảng 4.3).
- *Number of types of service (TOS)*: Trường này xác định số lượng loại hình dịch vụ thông báo cho mỗi liên kết.
- *Metric for TOS 0*: Trường này xác định các thông số cho loại dịch vụ mặc định (TOS 0).
- *TOS*: Trường này xác định loại hình dịch vụ.
- *Metric*: Trường này xác định độ đo cho TOS tương ứng.

Liên kết mạng LSA xác định các liên kết của một mạng. Một bộ định tuyến được chỉ định, thay mặt cho mạng transient, phân phối loại gói liên kết mạng LSP này. Gói tin thông báo sự tồn tại của tất cả các bộ định tuyến kết nối với mạng (xem hình 4.50). Định dạng của gói quảng cáo liên kết mạng được hiển thị trong hình 4.51. Các trường của liên kết mạng LSA như sau:

- *Network mask*: Trường này xác định mặt nạ mạng.
- *Attached router*: Trường lặp lại này xác định địa chỉ IP của tất cả các bộ định tuyến kèm theo.



Hình 4.50 Liên kết mạng



Hình 4.51 Định dạng gói tin quảng cáo liên kết mạng

CHƯƠNG 5

CÁC GIAO THỨC AN TOÀN MẠNG INTERNET

Chương này trình bày các vấn đề an toàn thông tin trên mạng Internet, bao gồm các nội dung sau:

- An ninh tầng mạng
- An ninh tầng giao vận
- An ninh tầng ứng dụng
- Firewall

5.1 AN NINH TẦNG MẠNG

Có ba lý do cần bảo mật tại tầng mạng: thứ nhất, không phải tất cả các chương trình client/server đều được bảo vệ ở tầng ứng dụng; thứ hai, không phải tất cả các chương trình client/server ở tầng ứng dụng đều sử dụng các dịch vụ của TCP (để được bảo vệ bởi tầng giao vận), mà một số chương trình sử dụng dịch vụ của UDP; thứ ba, nhiều ứng dụng, chẳng hạn như các giao thức định tuyến, trực tiếp sử dụng dịch vụ của IP thì lại cần các dịch vụ an ninh tại tầng IP.

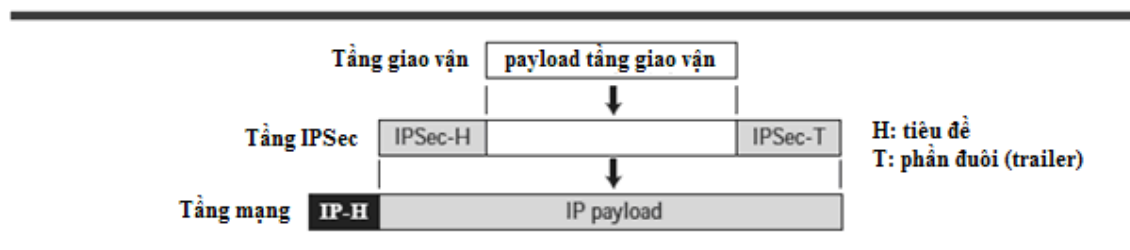
IP Security (IPSec) là một tập hợp các giao thức được thiết kế bởi *Internet Engineering Task Force* (IETF) để đảm bảo an toàn cho một gói tin ở cấp độ mạng. IPSec giúp tạo ra các gói tin xác thực và bí mật cho lớp IP.

5.1.1 Hai chế độ hoạt động của IPSec

IPSec hoạt động theo một trong hai chế độ khác nhau là *chế độ truyền tải* và *chế độ đường hầm*.

a. Chế độ truyền tải

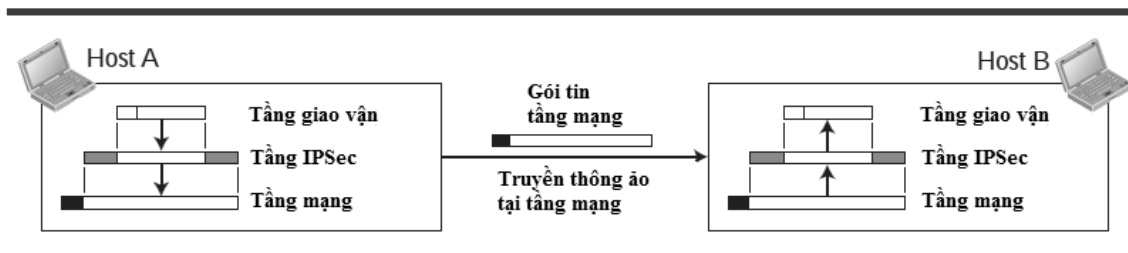
Trong chế độ truyền tải, IPSec bảo vệ những thông tin được cung cấp từ tầng giao vận chuyển đến tầng mạng. Nói cách khác, chế độ truyền tải bảo vệ payload được đóng gói trong tầng mạng, như trong hình 5.1.



Hình 5.1 IPSec trong chế độ truyền tải

Chú ý là chế độ truyền tải không bảo vệ tiêu đề IP. Nói cách khác, chế độ truyền tải không bảo vệ toàn bộ gói tin IP; nó chỉ bảo vệ gói dữ liệu từ tầng giao vận (payload tầng IP). Trong chế độ này, tiêu đề IPsec (và phần đuôi, trailer) được bổ sung vào thông tin đến từ tầng giao vận, sau đó, thêm vào phần tiêu đề IP.

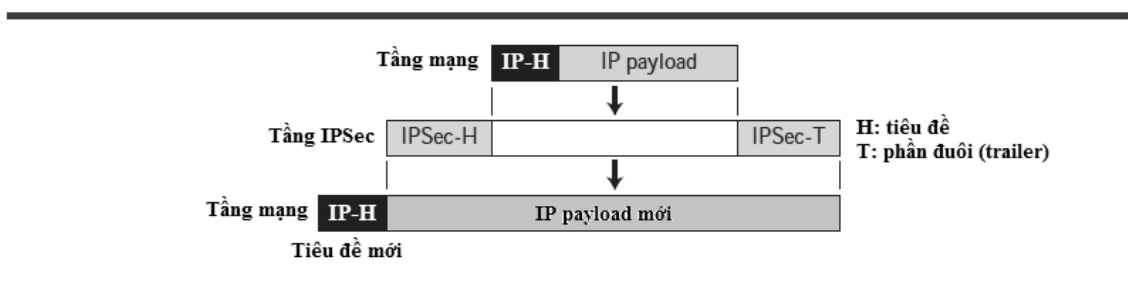
Chế độ truyền tải thường được sử dụng khi cần bảo vệ dữ liệu host-to-host (end-to-end). Host gửi sử dụng IPSec để xác thực và/hoặc mã hóa payload chuyển giao từ tầng giao vận. Host nhận sử dụng IPSec để kiểm tra xác thực và/hoặc giải mã các gói tin IP và cung cấp cho tầng giao vận (xem hình 5.2).



Hình 5.2 Chế độ truyền tải

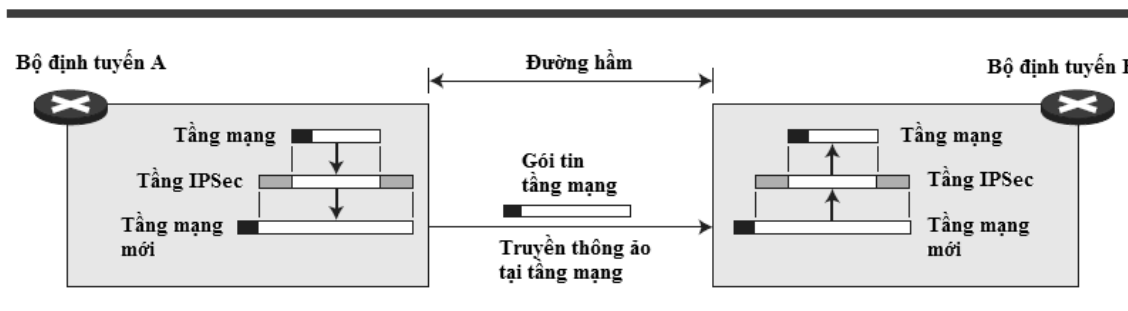
b. Chế độ đường hầm (Tunnel Mode)

Trong chế độ đường hầm, IPSec bảo vệ toàn bộ gói tin IP. Nó lấy một gói tin IP, bao gồm cả tiêu đề, áp dụng phương pháp bảo mật IPSec cho toàn bộ gói tin, và sau đó thêm một tiêu đề IP mới, như thể hiện trong hình 5.3.



Hình 5.3 IPSec trong chế độ đường hầm

Các tiêu đề IP mới có thông tin khác với tiêu đề IP gốc. Chế độ đường hầm thường được sử dụng giữa hai thiết bị định tuyến, giữa một host và một bộ định tuyến, hoặc giữa một bộ định tuyến và một host, như trong hình 5.4. Toàn bộ gói tin ban đầu được bảo vệ khỏi sự xâm nhập giữa người gửi và người nhận, như là toàn bộ gói tin đi qua một đường hầm tương tự.



Hình 5.4 Chế độ đường hầm

c. So sánh hai chế độ

Trong chế độ truyền tải, IPSec nằm giữa tầng giao vận và tầng mạng. Trong chế độ đường hầm, lưu lượng là từ tầng mạng đến IPSec và sau đó trở lại tầng mạng một lần nữa. Hình 5.5 so sánh hai chế độ.



Hình 5.5 So sánh chế độ truyền tải và chế độ đường hầm

5.1.2 Hai giao thức an ninh của IPSec

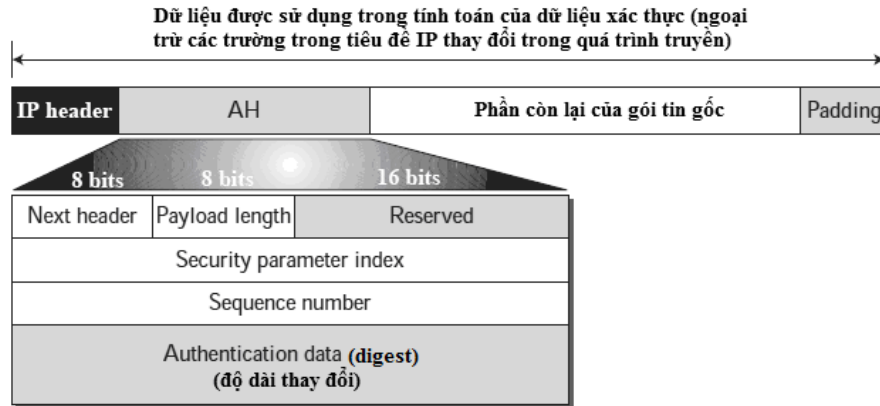
IPSec định nghĩa hai giao thức, *Authentication Header (AH)* và *Encapsulating Security Payload (ESP)*, để cung cấp xác thực và/hoặc mã hóa cho các gói tin ở mức IP.

a. Authentication Header (AH)

Giao thức *Authentication Header (AH)* được thiết kế để xác thực host nguồn và để đảm bảo tính toàn vẹn của payload mang trong các gói tin IP. Giao thức sử dụng một hàm băm và khóa đối xứng (bí mật) để tạo ra một digest (thông điệp tóm tắt); thông điệp tóm tắt được đưa vào trong tiêu đề xác thực. AH sau đó được đặt ở vị trí thích hợp tùy theo chế độ (truyền tải hoặc đường hầm). Hình 5.6 cho thấy các trường và vị trí của tiêu đề xác thực trong chế độ truyền tải.

Khi một gói tin IP mang một tiêu đề xác thực, giá trị ban đầu trong trường giao thức của tiêu đề IP được thay thế bằng giá trị 51. Một trường trong tiêu đề xác thực (trường next header) giữ giá trị ban đầu của trường protocol (loại payload được mang bởi gói tin IP). Việc bổ sung một tiêu đề xác thực được thực hiện theo các bước sau:

1. Tiêu đề xác thực được thêm vào payload với trường dữ liệu xác thực được thiết lập là 0.
2. Padding có thể được thêm vào để làm cho chiều dài tổng chẵn đối với một thuật toán băm cụ thể.
3. Băm dựa trên gói tổng. Tuy nhiên, chỉ có những trường của tiêu đề IP không bị thay đổi trong quá trình truyền thì mới được tính thông điệp tóm tắt (dữ liệu xác thực).
4. Dữ liệu xác thực được chèn vào trong tiêu đề xác thực.
5. Tiêu đề IP được thêm vào sau khi thay đổi giá trị của trường protocol thành 51.



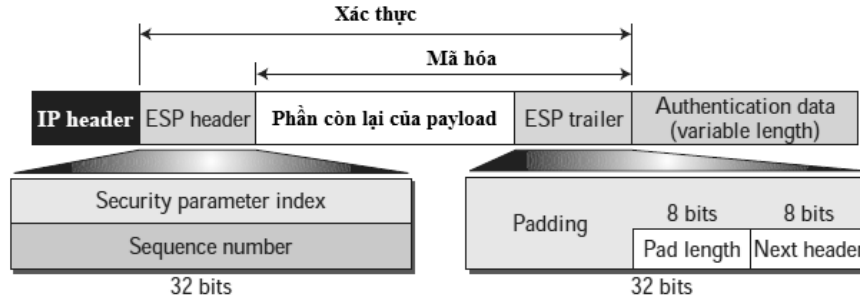
Hình 5.6 Giao thức AH (Authentication Header)

Mô tả ngắn gọn của từng trường như sau:

- *Next header*: Trường 8 bit này xác định loại payload mang bởi gói tin IP (như TCP, UDP, ICMP, hoặc OSPF).
- *Payload length*: Gồm 8 bit. Thực tế, trường này không xác định độ dài của payload mà nó xác định độ dài của tiêu đề xác thực theo bội của 4 byte, (nhưng không bao gồm 8 byte đầu tiên).
- *Security Parameter Index*: Trường SPI 32 bit đóng vai trò của một bộ định danh mạch ảo và là như nhau cho tất cả các gói được gửi trong một kết nối gọi là Security Association.
- *Sequence number*: Một số thứ tự 32 bit cung cấp thông tin cho một chuỗi các gói dữ liệu. Các số thứ tự giúp ngăn chặn việc phát lại. Chú ý là các số thứ tự không lặp đi lặp lại ngay cả khi một gói tin được truyền lại. Một số thứ tự không chồng nhau sau khi nó đạt đến 2^{32} và khi đó cần phải thiết lập kết nối mới.
- *Authentication data*: Cuối cùng, các trường dữ liệu xác thực là kết quả của việc áp dụng một hàm băm cho toàn bộ gói tin IP ngoại trừ các trường bị thay đổi trong quá trình vận chuyển (ví dụ, thời gian sống).

b. Encapsulating Security Payload (ESP)

Giao thức AH không cung cấp bảo mật, chỉ có xác thực nguồn và toàn vẹn dữ liệu. Sau đó IPSec xác định một giao thức thay thế là *Encapsulating Security Payload (ESP)*, cung cấp xác thực nguồn gốc, tính toàn vẹn và bảo mật. ESP thêm một tiêu đề và phần đuôi (trailer). Chú ý là dữ liệu xác thực của ESP được thêm vào cuối của gói tin, làm cho tính toán dễ dàng hơn. Hình 5.7 cho thấy vị trí của tiêu đề và trailer của ESP.



Hình 5.7 Encapsulating Security Payload (ESP)

Khi một gói tin IP mang một tiêu đề và trailer ESP, giá trị của trường giao thức trong tiêu đề IP là 50. Một trường bên trong trailer ESP (trường tiêu đề tiếp theo) chứa giá trị ban đầu của trường giao thức (các loại payload được mang bởi các gói tin IP, chẳng hạn như TCP hoặc UDP). Thủ tục ESP có các bước sau:

1. Trailer ESP được thêm vào payload.
2. Payload và trailer được mã hóa.
3. Tiêu đề ESP được thêm vào.
4. Tiêu đề ESP, payload, và trailer ESP được sử dụng để tạo ra dữ liệu xác thực.
5. Dữ liệu xác thực được thêm vào cuối của trailer ESP.
6. IP header được thêm vào sau khi thay đổi giá trị giao thức thành 50.

Các trường cho tiêu đề và trailer như sau:

- *SPI*: Trường 32 bit này tương tự như trường định nghĩa cho giao thức AH.
- *Sequence number*: Trường số thứ tự 32 bit cũng tương tự như trường định nghĩa cho giao thức AH.
- *Padding*: Trường này có độ dài thay đổi (0 đến 255 byte) gồm các số 0 đóng vai trò là lớp đệm.
- *Pad length*: Trường 8 bit này xác định số lượng byte đệm. Các giá trị từ 0 đến 255; hiếm khi có giá trị tối đa.
- *Next header*: Trường 8 bit này tương tự như định nghĩa trong giao thức AH. Mục đích giống như của trường giao thức trong IP header trước khi đóng gói.
- *Authentication header*: Cuối cùng, trường dữ liệu xác thực là kết quả của việc áp dụng một phương thức xác thực cho các phần của gói tin. Chú ý sự khác biệt giữa dữ liệu xác thực trong AH và ESP. Trong AH, một phần của tiêu đề IP được bao gồm trong tính toán của dữ liệu xác thực, còn trong ESP thì không.

c. IPv4 và IPv6:

IPSec hỗ trợ cả IPv4 và IPv6. Tuy nhiên trong IPv6, AH và ESP là một phần của tiêu đề mở rộng.

d. So sánh AH với ESP

Giao thức ESP được thiết kế sau khi giao thức AH được sử dụng. ESP làm bất cứ điều gì AH làm và có thêm các chức năng bổ sung (bảo mật). Như vậy, hiện nay, có thể không cần sử dụng giao thức AH. Tuy nhiên, AH đã được đưa vào trong một số sản phẩm thương mại, nên AH sẽ vẫn là một phần của Internet cho đến khi các sản phẩm này được loại bỏ.

5.1.3 Dịch vụ cung cấp bởi IPSec

Hai giao thức, AH và ESP, có thể cung cấp một số dịch vụ bảo mật cho các gói tin ở tầng mạng. Bảng 5.1 cho thấy danh sách các dịch vụ của mỗi giao thức.

Bảng 5.1 Các dịch vụ của IPSec

Các dịch vụ	AH	ESP
Điều khiển truy nhập	Có	Có
Xác thực thông điệp (toàn vẹn thông điệp)	Có	Có
Xác thực thực thể (xác thực nguồn dữ liệu)	Có	Có
Bảo mật	Không	Có
Bảo vệ tấn công phát lại	Có	Có

- **Điều khiển truy cập:** IPSec cung cấp điều khiển truy cập gián tiếp bằng cách sử dụng một cơ sở dữ liệu *Security Association (SA)*. Khi một gói tin đến đích, và Security Association không được thiết lập cho gói này, thì gói tin bị loại bỏ.
- **Toàn vẹn thông điệp:** Thông điệp được bảo vệ tính toàn vẹn trong cả AH và ESP. Một thông điệp tóm tắt (digest) của dữ liệu được tạo ra và được gửi bởi bên gửi phải được kiểm tra bởi bên nhận.
- **Xác thực thực thể:** Security Association và keyed-hash digest của dữ liệu được gửi bởi bên gửi sẽ xác thực bên gửi dữ liệu trong cả AH và ESP.
- **Tính bí mật:** Mã hóa của thông điệp trong ESP cung cấp tính bí mật, còn AH thì không. Do vậy, nếu cần đến tính bí mật thì nên sử dụng ESP thay vì AH.
- **Bảo vệ tấn công phát lại:** Trong cả hai giao thức, kỹ thuật tấn công phát lại được ngăn chặn bằng cách sử dụng số thứ tự và một cửa sổ trượt. Mỗi tiêu đề IPSec có chứa một số thứ tự duy nhất khi Security Association được thiết lập. Số bắt đầu từ 0 và tăng cho đến khi giá trị đạt $2^{32}-1$. Khi số thứ tự đạt mức tối đa, nó được lập lại là 0 và, đồng thời, Security Association cũ sẽ bị xóa và sau đó thiết lập cái mới. Để ngăn chặn xử lý gói dữ liệu trùng lặp, IPSec uỷ quyền sử dụng một cửa sổ kích thước cố định bên nhận. Kích thước cửa sổ được xác định bởi bên nhận với giá trị mặc định là 64.

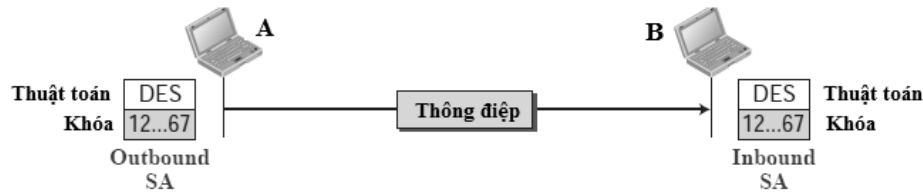
5.1.4 Security Association (SA)

Security Association là một khía cạnh rất quan trọng của IPSec. IPSec đòi hỏi một mối quan hệ logic, được gọi là Security Association (SA), giữa hai máy. Phần sau sẽ thảo luận về ý tưởng và sau đó là cách thức SA được sử dụng trong IPSec.

a. Ý tưởng của SA

Một SA là một hợp đồng giữa hai bên giúp tạo ra một kênh an toàn giữa chúng. Giả sử là A cần giao tiếp một hướng với B. Nếu A và B chỉ quan tâm đến khía cạnh bảo mật an ninh, họ có thể tạo ra một khóa bí mật để chia sẻ với nhau. Có thể nói rằng có hai SA giữa A và B;

một SA ra và một SA vào. Mỗi SA lưu trữ giá trị của khóa trong một biến và tên của thuật toán mã hóa/giải mã trong một biến khác. A sử dụng thuật toán và khóa để mã hóa một thông điệp tới B; B sử dụng thuật toán và khóa khi anh ta cần phải giải mã thông điệp nhận được từ A. Hình 5.8 cho thấy một SA đơn giản.



Hình 5.8 Một SA đơn giản

Các SA có thể được tham gia nhiều hơn nếu hai bên cần bảo đảm tính toàn vẹn và xác thực thông điệp. Mỗi liên kết cần dữ liệu khác như thuật toán cho toàn vẹn thông điệp, khóa, và các thông số khác. Nó có thể phức tạp hơn nhiều nếu các bên cần phải sử dụng các thuật toán cụ thể và các thông số cụ thể cho các giao thức khác nhau, chẳng hạn như IPSec AH hoặc IPSec ESP.

b. Cơ sở dữ liệu SA (SAD)

SA có thể rất phức tạp. Điều này đặc biệt đúng nếu A muốn gửi tin nhắn cho nhiều người và B cần phải nhận được tin nhắn từ nhiều người. Ngoài ra, mỗi trang web cần phải có SA trong và ngoài để cho phép giao tiếp hai chiều. Nói cách khác, cần một tập hợp các SA có thể được thu thập vào một cơ sở dữ liệu. Cơ sở dữ liệu này được gọi là cơ sở dữ liệu SA (SAD). Cơ sở dữ liệu có thể được coi như là một bảng hai chiều với mỗi hàng xác định một SA duy nhất. Thông thường, có hai SAD là *inbound* (trong) và *outbound* (ngoài). Hình 5.9 trình bày khái niệm SAD inbound hoặc SAD outbound cho mỗi thực thể.

Chỉ mục	SN	OF	ARW	AH/ESP	LT	Mode	MTU
< SPI, DA, P >							
< SPI, DA, P >							
< SPI, DA, P >							
< SPI, DA, P >							

Cơ sở dữ liệu SA

Ký hiệu:

SPI: Security Parameter Index (Chỉ mục tham số bảo mật)
 DA: Destination Address (Địa chỉ nguồn)
 AH/ESP: Thông tin cho mỗi thực thể
 P: Protocol (Giao thức)
 Mode: IPSec Mode Flag

SN: Sequence Number (Số tuần tự)
 OF: Overflow Flag
 ARW: Anti-Replay Window
 LT: Lifetime (Thời gian sống)
 MTU: Path MTU

Hình 5.9 Cơ sở dữ liệu SA

Khi một host cần gửi một gói tin phải mang một tiêu đề IPSec, host cần phải tìm mục tương ứng trong SAD outbound để tìm thông tin cho việc bảo mật cho gói tin. Tương tự như vậy, khi một host nhận được một gói mang tiêu đề IPSec, host cần phải tìm mục tương ứng trong SAD inbound để tìm thông tin kiểm tra sự an toàn của gói tin. Việc tìm kiếm này phải cụ thể theo nghĩa là bên nhận cần phải chắc chắn về thông tin chính xác được sử dụng để xử lý các gói tin. Mỗi mục trong một SAD trong được lựa chọn bằng cách sử dụng ba chỉ số: chỉ số tham số bảo mật (một số 32 bit định nghĩa SA tại điểm đến), địa chỉ đích, và giao thức (AH hoặc ESP).

c. Chính sách bảo mật

Một khía cạnh quan trọng của IPSec là chính sách bảo mật (Security Policy - SP), trong đó xác định loại bảo mật áp dụng cho một gói khi nó được gửi đi hoặc khi nó đã đến. Trước khi sử dụng SAD, host phải xác định chính sách định trước cho các gói tin.

d. Cơ sở dữ liệu chính sách bảo mật

Mỗi host đang sử dụng giao thức IPSec cần phải lưu một cơ sở dữ liệu chính sách bảo mật (SPD). Một lần nữa, cần một SPD inbound và một SPD outbound. Mỗi mục trong SPD có thể được truy cập bằng cách sử dụng một chỉ số sextuple: địa chỉ nguồn, địa chỉ đích, tên, giao thức, cổng nguồn và cổng đích, như thể hiện trong hình 5.10.

Chỉ số	Chính sách
< SA, DA, Name, P, SPort, DPort >	
< SA, DA, Name, P, SPort, DPort >	
< SA, DA, Name, P, SPort, DPort >	
< SA, DA, Name, P, SPort, DPort >	

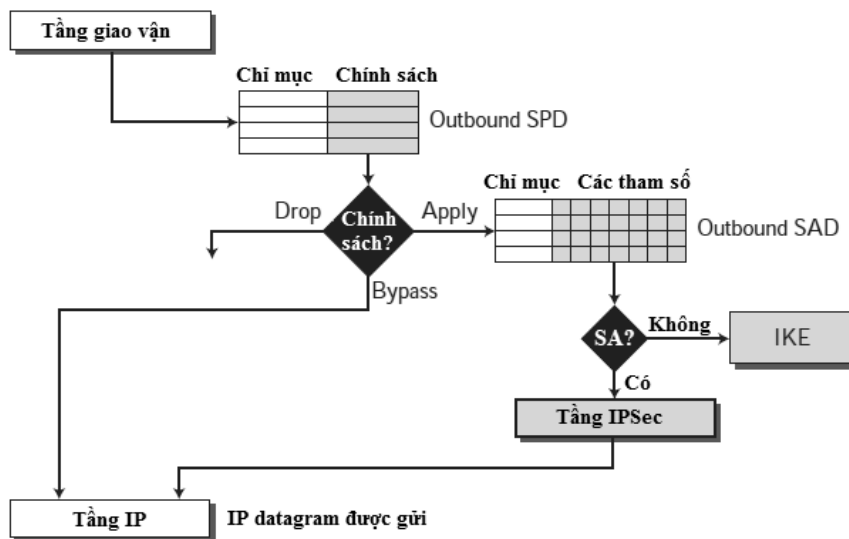
Ký hiệu:

SA: Source Address (Địa chỉ nguồn)	SPort: Source Port (Cổng nguồn)
DA: Destination Address (Địa chỉ đích)	DPort: Destination Port (Cổng đích)
P: Protocol (Giao thức)	

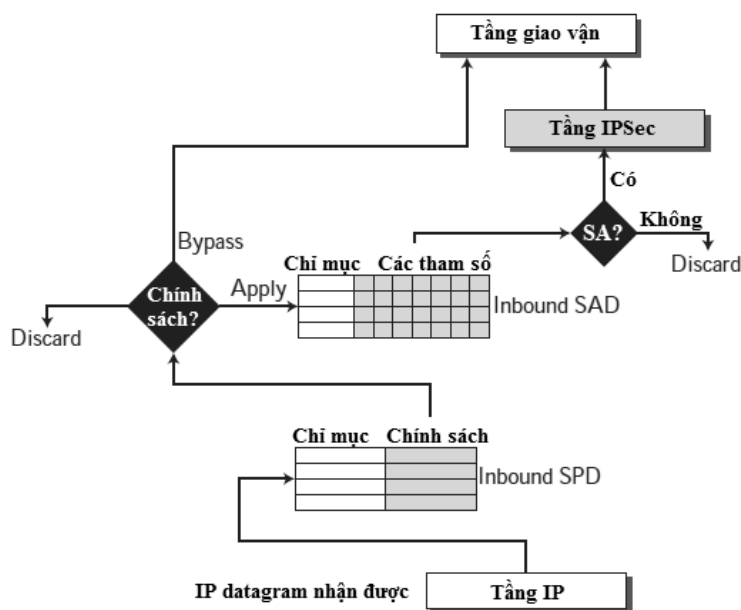
Hình 5.10 Cơ sở dữ liệu chính sách bảo mật (SPD)

Nguồn và địa chỉ đích có thể là unicast, multicast, hoặc địa chỉ wildcard. Tên thường xác định một thực thể DNS. Giao thức là AH hoặc ESP. Cổng nguồn và đích là địa chỉ cổng cho tiến trình chạy ở host nguồn và host đích.

- **Outbound SPD:** Khi một gói tin được gửi đi, SPD outbound sẽ được tham khảo. Hình 5.11 cho thấy việc xử lý một gói tin của người gửi. Đầu vào cho SPD outbound là chỉ số sextuple; đầu ra là một trong ba trường hợp sau: drop (gói không thể được gửi), bypass (bỏ qua tiêu đề an ninh), và apply (áp dụng bảo mật theo quy định của SAD, nếu không có SAD thì tạo ra).
- **Inbound SPD:** Khi một gói tin đến, SPD inbound sẽ được tham khảo. Mỗi mục trong SPD trong cũng được truy cập bằng cách sử dụng cùng một chỉ số sextuple. Hình 5.12 cho thấy việc xử lý một gói tin của bên nhận. Các đầu vào cho SPD inbound là chỉ số sextuple; đầu ra là một trong ba trường hợp sau: discard (loại bỏ gói), bypass (bỏ qua sự an toàn và cung cấp các gói tin đến tầng giao vận), và apply (áp dụng chính sách sử dụng SAD).



Hình 5.11 Xử lý outbound

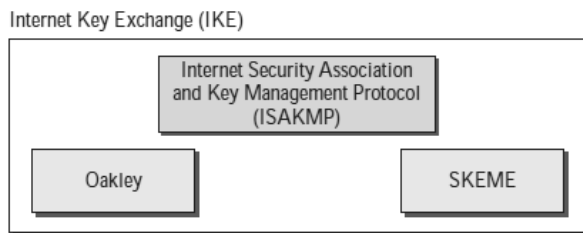


Hình 5.12 Xử lý inbound

5.1.5 Internet Key Exchange (IKE)

Internet Key Exchange (IKE) là một giao thức được thiết kế để tạo ra các SA cả inbound và outbound. Khi một máy ngang hàng cần gửi một gói tin IP, nó sẽ tham khảo cơ sở dữ liệu chính sách bảo mật (SPD) để xem nếu liệu có một SA cho loại lưu lượng đó hay không. Nếu không có SA, IKE được gọi sẽ tạo ra.

IKE là một giao thức phức tạp dựa trên ba giao thức khác: Oakley, SKEME, và ISAKMP, như thể hiện trong hình 5.13.

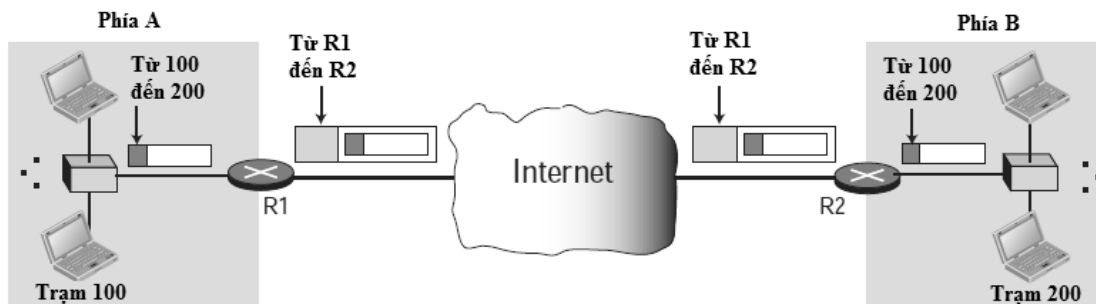


Hình 5.13 Các thành phần của IKE

Giao thức Oakley được phát triển bởi Hilarie Orman, là một giao thức tạo khóa. SKEME, được thiết kế bởi Hugo Krawczyk, là một giao thức trao đổi khóa. Giao thức này sử dụng mã hóa khóa công khai để xác thực thực thể trong một giao thức trao đổi khóa. Internet Security Association và Key Management Protocol (ISAKMP) là một giao thức được thiết kế bởi Cơ quan An ninh Quốc gia (NSA) dùng để thực hiện trao đổi quy định trong IKE. Giao thức này định nghĩa một số gói dữ liệu, giao thức, và các thông số cho phép trao đổi IKE sẽ diễn ra trong các thông điệp theo chuẩn, được định dạng để tạo ra SA.

5.1.6 Mạng riêng ảo (VPN)

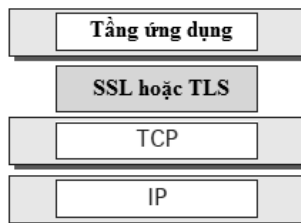
Một trong những ứng dụng của IPsec là các mạng riêng ảo. Một mạng riêng ảo (VPN) là một công nghệ đang được phổ biến trong các tổ chức lớn sử dụng Internet toàn cầu cho thông tin liên lạc nội bộ và liên tổ chức, nhưng yêu cầu tính riêng tư trong thông tin liên lạc trong nội bộ tổ chức của họ. VPN là một mạng riêng nhưng ảo. Là riêng vì nó đảm bảo sự riêng tư bên trong tổ chức. Nó là ảo bởi vì nó không sử dụng mạng WAN riêng thật sự; mạng là công cộng về vật lý nhưng là mạng ảo riêng. Hình 5.14 cho thấy ý tưởng về một mạng riêng ảo. Bộ định tuyến R1 và R2 sử dụng công nghệ VPN để đảm bảo sự riêng tư cho tổ chức. Công nghệ VPN sử dụng giao thức IPsec ESP trong chế độ đường hầm. Một gói tin, bao gồm tiêu đề, được đóng gói trong một gói ESP. Các bộ định tuyến ở biên giới của các khu vực gửi tin sử dụng địa chỉ IP riêng của mình và địa chỉ của các bộ định tuyến tại khu vực đích trong gói tin mới. Các mạng công cộng (Internet) có trách nhiệm mang các gói tin từ R1 đến R2. Người ngoài không thể giải mã các nội dung của gói hoặc các địa chỉ nguồn và đích. Giải mã diễn ra tại R2, để thấy địa chỉ đích của gói tin và vận chuyển nó.



Hình 5.14 Mạng riêng ảo VPN

5.2 AN NINH TẦNG GIAO VẬN

Hai giao thức để đảm bảo an ninh ở tầng giao vận chiếm ưu thế hiện nay là Secure Sockets Layer (SSL) và Transport Layer Security (TLS). TLS thực sự là một phiên bản IETF của giao thức trước. Phần này sẽ thảo luận về SSL, còn TLS hoàn toàn tương tự. Hình 5.15 cho thấy vị trí của SSL và TLS trong mô hình Internet.



Hình 5.15 Vị trí của SSL và TLS trong mô hình mạng Internet

Một trong những mục tiêu của các giao thức này là cung cấp xác thực cho server và client, bảo đảm tính bí mật và toàn vẹn dữ liệu. Chương trình tầng ứng dụng client/server, chẳng hạn như HTTP, sử dụng các dịch vụ của TCP, có thể đóng gói dữ liệu trong các gói tin SSL. Nếu server và client có khả năng chạy các chương trình SSL (hoặc TLS), thì client có thể sử dụng https URL: // ... thay vì http: // ... để cho phép các thông điệp HTTP được đóng gói trong các gói tin SSL (hoặc TLS). Ví dụ, số thẻ tín dụng có thể được chuyển giao một cách an toàn thông qua Internet cho người mua sắm trực tuyến.

5.2.1 Kiến trúc SSL

SSL được thiết kế để cung cấp dịch vụ bảo mật và nén dữ liệu tạo ra từ tầng ứng dụng. Thông thường, SSL có thể nhận dữ liệu từ bất kỳ giao thức tầng ứng dụng nào, nhưng thường là giao thức HTTP. Dữ liệu nhận được từ các ứng dụng được nén (tùy chọn), ký và mã hóa. Dữ liệu sau đó được đưa đến một giao thức tầng giao vận đáng tin cậy như TCP. Netscape đã phát triển SSL năm 1994, phiên bản 2 và 3 đã được phát hành trong năm 1995. Phần này sẽ thảo luận SSLv3.

SSL cung cấp một số dịch vụ trên dữ liệu nhận được từ các tầng ứng dụng.

- *Phân mảnh*: Đầu tiên, SSL chia dữ liệu thành các khối 214 byte hoặc ít hơn.
- *Nén*: Mỗi mảnh của dữ liệu được nén bằng cách sử dụng một trong những phương pháp nén không giảm chất lượng thông qua thương lượng giữa client và server. Dịch vụ này là tùy chọn.
- *Toàn vẹn dữ liệu*: Để duy trì tính toàn vẹn của dữ liệu, SSL sử dụng một hàm băm có khóa để tạo ra một mã xác thực thông điệp MAC (Message authentication code).
- *Bảo mật*: Để cung cấp bảo mật, dữ liệu gốc và MAC được mã hóa bằng cách sử dụng mật mã khóa đối xứng.
- *Đóng khung*: Một tiêu đề được thêm vào payload được mã hóa. Payload sau đó được đưa đến một giao thức tầng giao vận đáng tin cậy.

a. Thuật toán trao đổi khóa

Để trao đổi các thông điệp xác thực và bảo mật, cả client và server cần những bí mật mật mã (cryptographic secrets). Tuy nhiên, để tạo ra bí mật này, một bí mật pre-master phải được

thiết lập giữa hai bên. SSL xác định một số phương pháp quan trọng để thiết lập trao đổi bí mật pre-master này.

b. Thuật toán mã hóa/giải mã

Client và server cũng cần phải cùng đồng ý một bộ thuật toán mã hóa và giải mã.

c. Các thuật toán băm

SSL sử dụng thuật toán băm để cung cấp toàn vẹn thông điệp (xác thực thông điệp). Một số thuật toán băm cũng đã được xác định cho mục đích này.

d. Cipher Suite

Sự kết hợp các thuật toán trao đổi, băm và mã hóa khóa đã xác định một bộ thuật toán mã hóa cho mỗi phiên SSL.

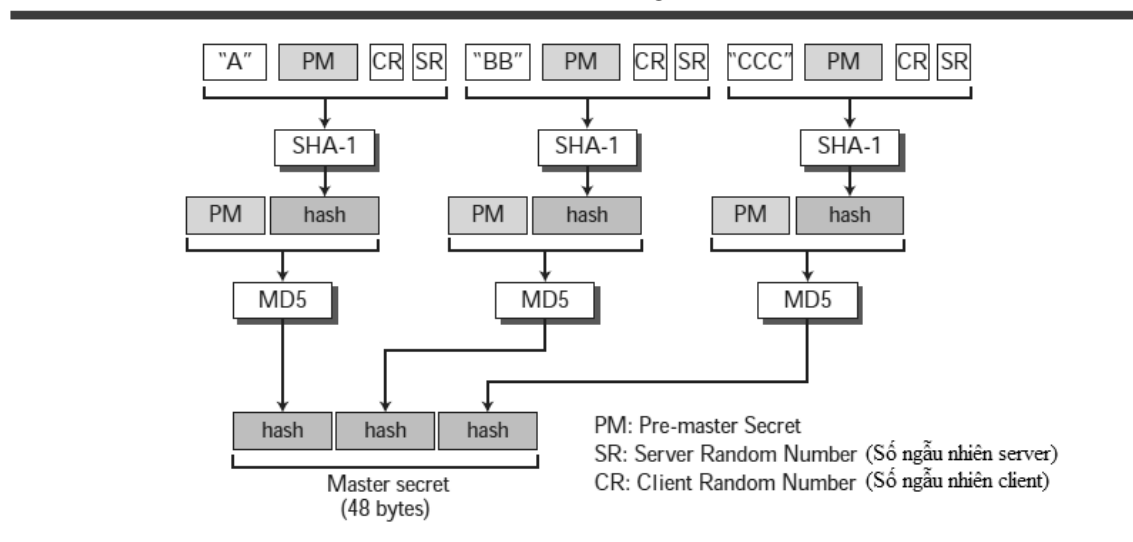
e. Các thuật toán nén

Nén là tùy chọn trong SSL. Không có thuật toán nén cụ thể được xác định. Vì vậy, một hệ thống có thể sử dụng bất cứ thuật toán nén nào nó muốn.

f. Sinh tham số mật mã

Để có được tính toàn vẹn và bí mật cho thông điệp, SSL cần sáu bí mật mật mã, bốn khóa và hai IV (véc-tơ khởi tạo). Client cần một khóa để xác thực thông điệp, một khóa để mã hóa, và một IV làm khởi đầu trong tính toán. Các server cũng cần như vậy. SSL yêu cầu các khóa cho một hướng phải khác với khóa cho hướng khác để nếu có tấn công theo một hướng thì hướng khác sẽ không bị ảnh hưởng. Các thông số được tạo ra bằng cách sử dụng thủ tục sau đây:

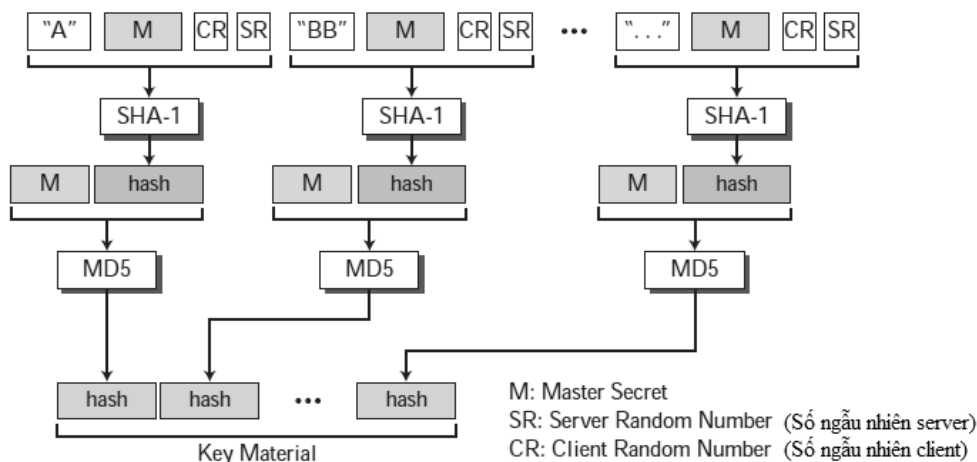
1. Client và server trao đổi hai số ngẫu nhiên; một được tạo ra bởi client và một do server.
2. Các client và server trao đổi một bí mật pre-master, sử dụng một trong các thuật toán trao đổi khóa xác định trước.
3. Một bí mật master 48 byte được tạo ra từ các bí mật pre-master bằng cách áp dụng hai hàm băm (SHA-1 và MD5), như thể hiện trong hình 5.16.



Hình 5.16 Tính master secret từ pre-master secret

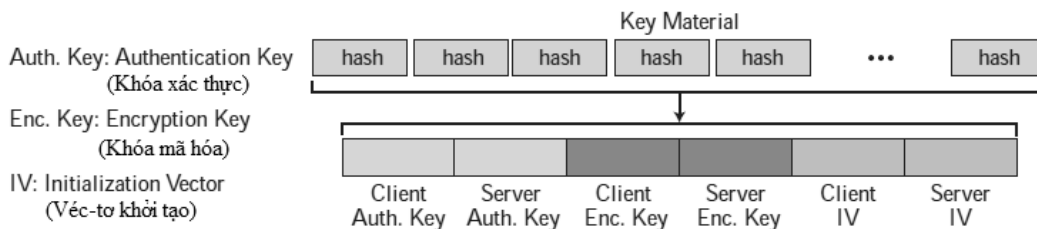
4. Bí mật master được sử dụng để tạo key material có độ dài thay đổi bằng cách áp dụng một tập các hàm băm giống nhau và thêm vào với các hằng số khác nhau, như thể hiện trong hình 5.17. Mô-đun này được lặp đi lặp lại cho đến khi các key material với

nhiều kích thước được tạo ra. Chú ý là chiều dài của khối key material phụ thuộc vào bộ mật mã được lựa chọn và kích thước của các khóa cần thiết cho bộ mật mã này.



Hình 5.17 Tính key material từ master secret

5. Sáu secret khác nhau được tách ra từ key material, như trong hình 5.18.



Hình 5.18 Trích xuất các secret mật mã từ key material

g. Phiên và kết nối

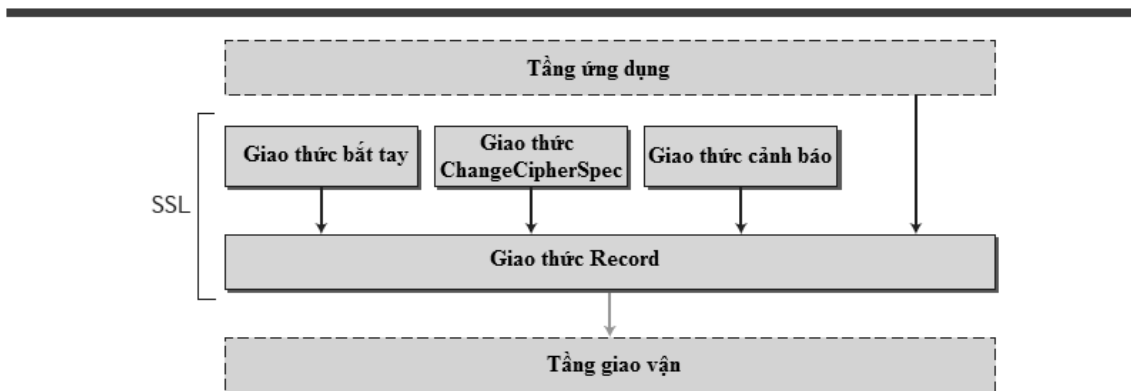
SSL phân biệt kết nối và phiên. Một phiên là một liên kết giữa một client và một server. Sau khi một phiên được thiết lập, hai bên có thông tin phổ biến như: định danh phiên, chứng nhận xác thực mỗi bên (nếu cần thiết), phương pháp nén (nếu cần), bộ mã hóa, và một bí mật master được sử dụng để tạo ra khóa cho việc mã hóa xác thực thông điệp.

Đối với hai thực thể trao đổi dữ liệu, việc thành lập một phiên là cần thiết, nhưng chưa đủ; chúng cần phải tạo ra một kết nối với nhau. Hai thực thể trao đổi hai số ngẫu nhiên và sử dụng bí mật master để tạo ra các khóa và các thông số cần thiết cho việc trao đổi các thông điệp liên quan đến xác thực và bảo mật.

Một phiên có thể bao gồm nhiều kết nối. Một kết nối giữa hai bên có thể được chấm dứt và tái lập trong cùng một phiên. Khi một kết nối được chấm dứt, hai bên cũng có thể chấm dứt phiên, nhưng điều này là không bắt buộc. Phiên có thể bị đình chỉ và tiếp tục một lần nữa.

5.2.2 Bốn giao thức SSL

SSL định nghĩa bốn giao thức trong hai lớp, như thể hiện trong hình 5.19.

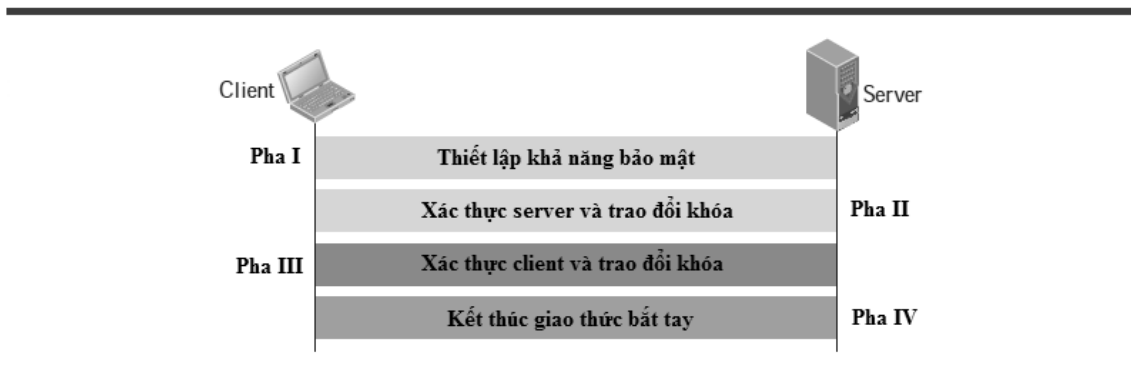


Hình 5.19 Bốn giao thức SSL

Giao thức Record là vật mang. Nó mang thông điệp từ ba giao thức khác cũng như các dữ liệu đến từ tầng ứng dụng. Các thông điệp từ giao thức Record là các payload đến tầng giao vận, thông thường là TCP. *Giao thức bắt tay* cung cấp các thông số bảo mật cho giao thức Record. Nó thiết lập một bộ mã hóa và cung cấp các khóa cùng các thông số bảo mật. Nó cũng xác nhận server cho client và client cho server nếu cần. *Giao thức ChangeCipherSpec* được sử dụng để báo hiệu sự sẵn sàng của các bí mật mật mã. *Giao thức cảnh báo* được sử dụng để báo cáo các tình trạng bất thường.

a. *Giao thức bắt tay*

Giao thức này sử dụng các thông điệp để thương lượng bộ mã hóa để xác thực server cho client và client cho server, khi cần, và trao đổi thông tin để xây dựng những bí mật mật mã. Việc bắt tay được thực hiện trong bốn pha, như thể hiện trong hình 5.20.



Hình 5.20 Giao thức bắt tay

b. *Giao thức ChangeCipherSpec*

Chúng ta đã thấy rằng việc đàm phán của bộ thuật toán mã hóa và việc sinh ra mật mã bí mật được hình thành dần trong giao thức bắt tay. Câu hỏi đặt ra bây giờ là: Khi nào hai bên có thể sử dụng các tham số bí mật? SSL quy định rằng các bên không thể sử dụng các thông số, hay các bí mật cho đến khi chúng đã gửi hoặc nhận được một thông điệp đặc biệt, thông điệp *ChangeCipherSpec*, được trao đổi trong giao thức bắt tay và quy định tại giao thức *ChangeCipherSpec*. Lý do là vấn đề không chỉ gửi hoặc nhận tin nhắn. Bên gửi và bên nhận cần hai trạng thái, không phải là một. Một trạng thái là pending, dùng để theo dõi những thông số và các bí mật. Trạng thái khác là trạng thái hoạt động, nắm giữ các thông số và bí mật được

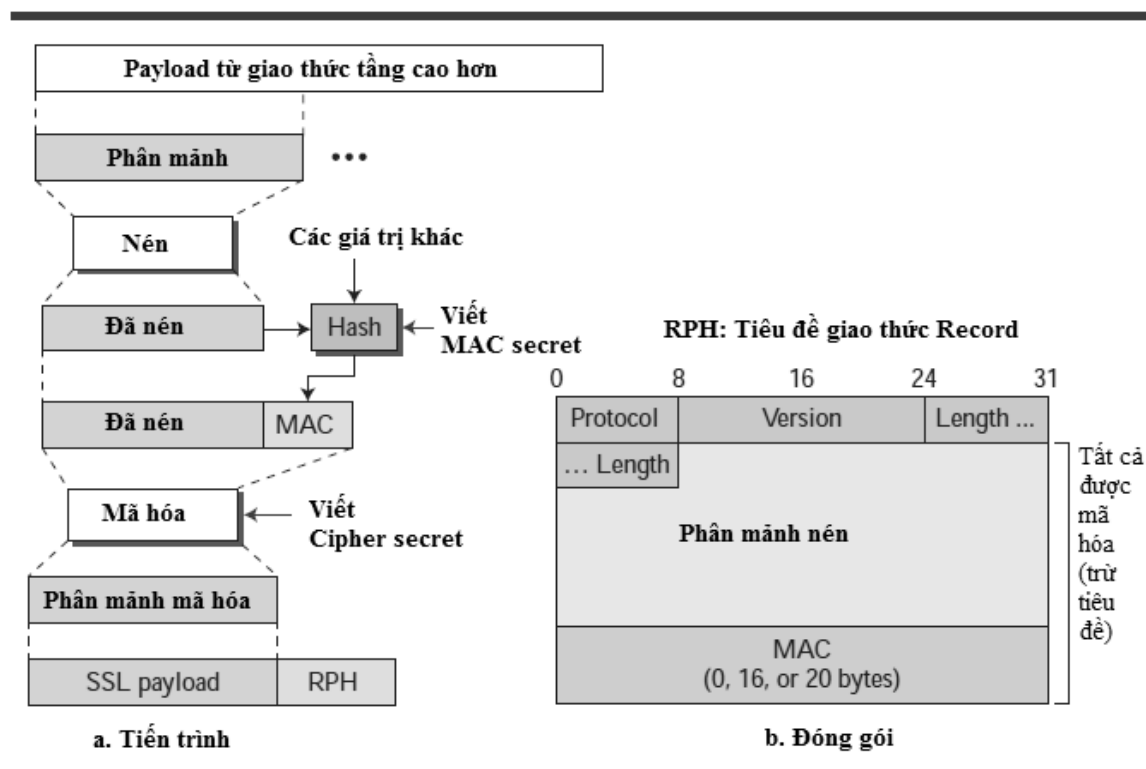
sử dụng bởi giao thức Record để ký/xác minh hoặc mã hóa/giải mã thông điệp. Ngoài ra, mỗi trạng thái có hai bộ giá trị: trong (inbound) và ngoài (outbound).

c. Giao thức cảnh báo

SSL sử dụng giao thức cảnh báo để báo các lỗi và điều kiện bất thường. Nó chỉ sử dụng một thông điệp mô tả vấn đề và mức độ của nó (cảnh cáo hoặc nguy hiểm).

d. Giao thức Record

The giao thức Record mang thông điệp từ các tầng trên (giao thức bắt tay, ChangeCipherSpec, cảnh báo, hoặc tầng ứng dụng). Thông điệp này bị phân mảnh và nén nếu cần; một MAC được thêm vào thông điệp được nén bằng cách sử dụng thuật toán băm đã đàm phán. Phân mảnh nén và MAC được mã hóa bằng cách sử dụng thuật toán mã hóa đã thương lượng. Cuối cùng, tiêu đề SSL được thêm vào thông điệp mã hóa. Hình 5.21 cho thấy quá trình này ở bên gửi. Quá trình ở bên nhận được đảo ngược.



Hình 5.21 Xử lý được thực hiện bởi giao thức Record

5.3 AN NINH TẦNG ỨNG DỤNG

Phần này thảo luận về hai giao thức cung cấp dịch vụ bảo mật cho e-mail: Pretty Good Privacy (PGP) và Secure/Multipurpose Internet Mail Extension (S/MIME).

5.3.1 An ninh cho e-mail

Việc gửi e-mail là một hoạt động một lần, bản chất của hoạt động này khác với trong hai phần trước. Trong IPSec hoặc SSL, giả sử rằng hai bên tạo ra một phiên làm việc ở giữa và trao đổi dữ liệu theo cả hai hướng. Trong e-mail, không có phiên nào tồn tại. Hai người A và B không thể tạo ra một phiên. A gửi một thông điệp cho B; đôi khi sau đó, B đọc tin và có thể

có hoặc không gửi tin trả lời. Những gì A gửi cho B là hoàn toàn độc lập với những gì B gửi cho A.

a. Các thuật toán mật mã

Nếu e-mail là một hoạt động một lần, làm thế nào có thể người gửi và người nhận đồng ý về một giải thuật mã hóa để sử dụng cho bất mật e-mail? Nếu không có phiên và không có tín hiệu bắt tay đàm phán các thuật toán mã hóa/giải mã và băm, làm thế nào có thể nhận biết được thuật toán nào mà người gửi đã chọn cho từng mục đích?

Để giải quyết vấn đề, giao thức định nghĩa một tập hợp các thuật toán cho mỗi hoạt động mà người dùng sử dụng trong hệ thống của mình. A thêm tên (hoặc định danh) của các thuật toán đã sử dụng trong e-mail. Ví dụ, A có thể chọn DES để mã hóa/giải mã và MD5 để băm. Khi A gửi một thông điệp cho B, A cho các định danh tương ứng cho DES và MD5 trong thông điệp của mình. B nhận được tin nhắn và trích xuất các định danh đầu tiên. Sau đó, B biết thuật toán nào sử dụng để giải mã và thuật toán nào cho băm.

b. Bí mật mật mã

Vấn đề tương tự đối với các thuật toán mã hóa cũng xuất hiện đối với những bí mật mật mã (khóa). Nếu không có đàm phán, làm thế nào hai bên có thể thiết lập bí mật giữa họ? Các giao thức bảo mật e-mail ngày nay yêu cầu mã hóa/giải mã được thực hiện bằng cách sử dụng một thuật toán khóa đối xứng và khóa bí mật một lần gửi đi với thông điệp. A có thể tạo ra một khóa bí mật và gửi nó với thông điệp gửi cho B. Để bảo vệ khóa bí mật do việc chặn bởi người C, các khóa bí mật được mã hóa với khóa công khai của B. Nói cách khác, sẽ mã hóa chính khóa bí mật.

c. Chứng nhận

Rõ ràng là một số thuật toán khóa công khai phải được sử dụng để bảo mật e-mail. Ví dụ, cần phải mã hóa khóa bí mật hoặc ký tin nhắn. Để mã hóa một khóa bí mật, A cần khóa công khai của B; để kiểm tra tin nhắn đã được ký, B cần khóa công khai của A. Vì vậy, để gửi một tin nhắn xác thực và bí mật nhỏ, cần phải có hai khóa công khai. Làm thế nào A có thể yên tâm về khóa công khai của B, và làm thế nào B có thể yên tâm với khóa công khai của A? Mỗi giao thức bảo mật e-mail có một phương pháp khác nhau xác nhận khóa.

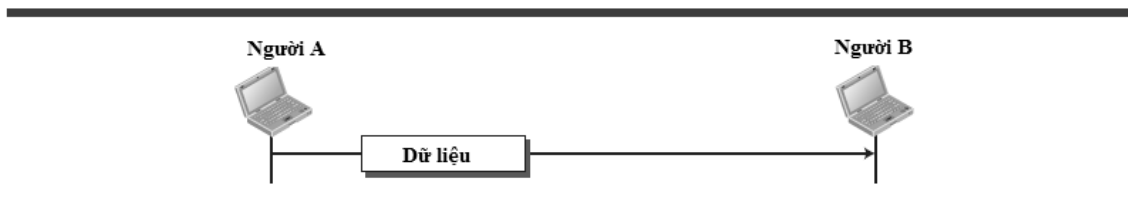
5.3.2 Pretty Good Privacy (PGP)

Pretty Good Privacy (PGP) được phát minh bởi Phil Zimmermann để cung cấp e-mail với tính bí mật, tính toàn vẹn và xác thực. PGP có thể được sử dụng để tạo ra một tin nhắn e-mail an toàn.

a. Các kịch bản

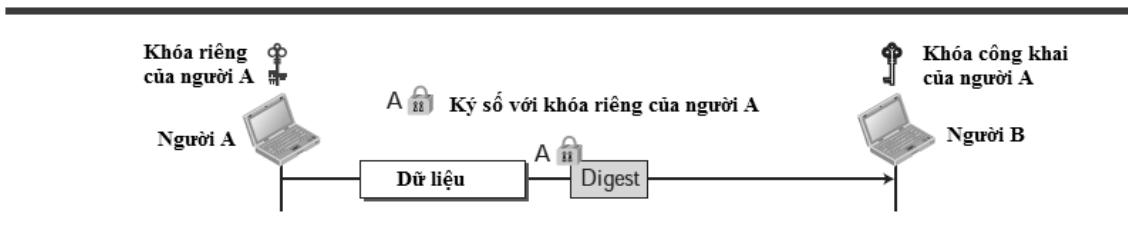
Để hiểu được ý tưởng chung của PGP, phần sau sẽ đi từ kịch bản đơn giản tới kịch bản phức tạp trong quá trình xử lý.

- **Bản rõ:** Kịch bản đơn giản nhất là gửi tin nhắn e-mail trong bản rõ như trong hình 5.22. Không có tính bí mật hay toàn vẹn cho thông điệp trong kịch bản này.



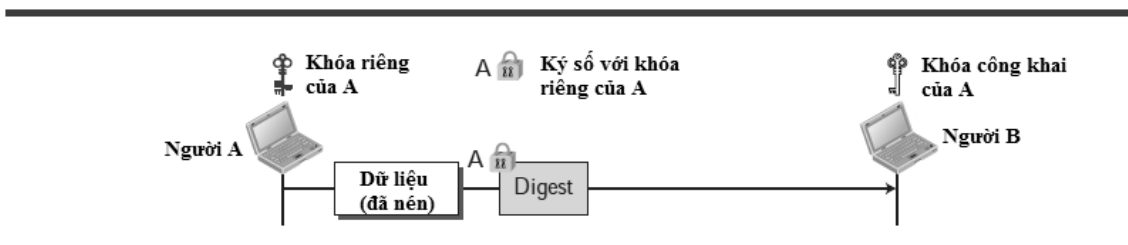
Hình 5.22 Thông điệp bản rõ

- *Toàn vẹn thông điệp:* Sự cải tiến tiếp theo là để cho A ký tin nhắn. A tạo ra một thông điệp tóm tắt của tin nhắn và ký nó với khóa riêng của mình (hình 5.23). Khi B nhận được thông báo, B xác nhận thông điệp bằng cách sử dụng khóa công khai của A. Có hai khóa cần thiết cho kịch bản này. A cần phải biết khóa riêng của mình; B cần biết khóa công khai của A.



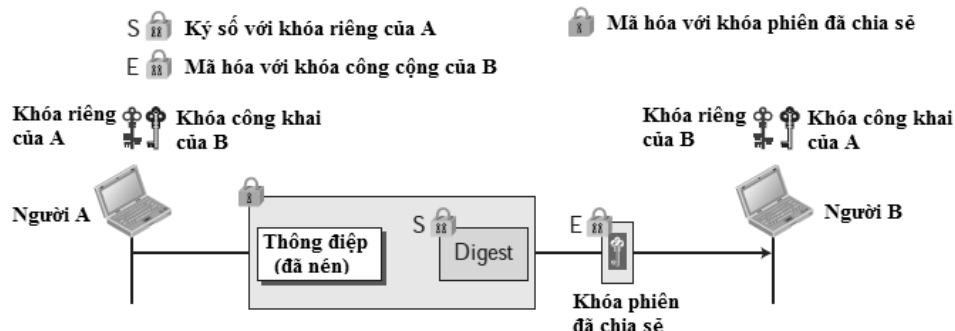
Hình 5.23 Một thông điệp xác thực

- *Nén:* Một cải tiến nữa là nén thông điệp để làm cho gói dữ liệu nhỏ gọn hơn. Sự cải thiện này không có lợi ích về bảo mật, nhưng nó giúp giảm bớt lưu lượng truy cập. Hình 5.24 cho thấy kịch bản mới.



Hình 5.24 Một thông điệp nén

- *Bí mật với khóa phiên một lần:* (Hình 5.25) Như đã biết, tính bí mật trong một hệ thống e-mail có thể đạt được bằng cách sử dụng mã hóa thông thường với một khóa phiên một lần. A có thể tạo ra một khóa phiên, sử dụng khóa phiên mã hóa tin nhắn và thông điệp tóm tắt, và gửi khóa riêng của mình cùng với thông điệp. Tuy nhiên, để bảo vệ khóa phiên, A mã hóa nó với khóa công khai của B. Khi B nhận được gói tin, đầu tiên B giải mã khóa, sử dụng khóa riêng của mình để loại bỏ khóa. Sau đó, B sử dụng khóa phiên để giải mã phần còn lại của thông điệp. Sau khi giải nén phần còn lại của thông điệp, B tạo ra một thông điệp tóm tắt của thông điệp và kiểm tra để xem liệu nó có bằng thông điệp tóm tắt được gửi bởi A không. Nếu bằng nhau, tin nhắn là xác thực.



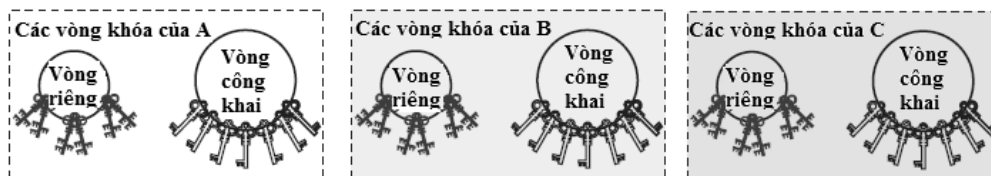
Hình 5.25 Một thông điệp bảo mật

- **Chuyển đổi mã:** Một dịch vụ khác được cung cấp bởi PGP là chuyển đổi mã. Hầu hết các hệ thống e-mail cho phép tin nhắn chỉ chứa các ký tự ASCII. Để dịch các ký tự khác không có trong bộ mã ASCII, PGP sử dụng chuyển đổi Radix-64.

PGP cho phép phân đoạn tin nhắn sau khi nó đã được chuyển đổi sang Radix-64 để làm cho các đơn vị truyền có kích cỡ đồng đều, được cho phép bởi các giao thức e-mail cơ bản.

b. Vòng khóa

Trong tất cả các kịch bản trước đó, chúng ta giả định rằng A cần phải gửi một tin nhắn chỉ cho B, nhưng không phải luôn luôn là như vậy. A có thể cần phải gửi tin nhắn cho nhiều người; A cần vòng khóa. Trong trường hợp này, A cần một vòng khóa công khai, với mỗi khóa thuộc từng người mà A cần phải đáp ứng (gửi hoặc nhận tin nhắn). Ngoài ra, các nhà thiết kế PGP quy định một vòng khóa riêng/khóa công khai. Lý do là A có thể muốn thay đổi các cặp khóa của mình theo thời gian. Một lý do khác là A có thể cần phải đáp ứng với các nhóm người khác nhau (bạn bè, đồng nghiệp, v.v.). A có thể muốn sử dụng một cặp khóa khác nhau cho mỗi nhóm. Vì vậy, mỗi người dùng cần phải có hai bộ vòng: một vòng khóa riêng/công khai và một vòng khóa công khai của người khác. Hình 5.26 cho thấy một nhóm gồm ba người, mỗi người có một chiếc vòng gồm các cặp khóa riêng/khóa công khai và đồng thời có một vòng khóa công khai thuộc về những người khác trong nhóm.



Hình 5.26 Các vòng khóa trong PGP

Ví dụ A có vài cặp khóa riêng/khóa công khai của mình và khóa công khai thuộc về người khác. Lưu ý rằng tất cả mọi người có thể có nhiều hơn một khóa công khai. Hai trường hợp có thể xảy ra.

1. A cần phải gửi một tin nhắn cho người khác trong nhóm.
 - ✓ A sử dụng khóa riêng của mình để ký tóm tắt.
 - ✓ A sử dụng khóa công khai của người nhận để mã hóa một khóa phiên mới được tạo ra.
 - ✓ A mã hóa tin nhắn và ký tóm tắt với khóa phiên được tạo ra.

2. A nhận được một tin nhắn từ một người khác trong cộng đồng.
 - ✓ A sử dụng khóa riêng của mình để giải mã khóa phiên.
 - ✓ A sử dụng khóa phiên để giải mã thông điệp và tóm tắt.
 - ✓ A sử dụng khóa công khai để xác minh tóm tắt.

c. Các thuật toán PGP

PGP định nghĩa một tập các thuật toán khóa bất đối xứng và đối xứng, hàm băm mật mã, và các phương pháp nén.

d. Các ứng dụng của PGP

PGP đã được sử dụng rộng rãi cho các e-mail cá nhân.

5.3.3 Secure/Multipurpose Internet Mail Extension (S/MIME)

Một dịch vụ bảo mật khác được thiết kế cho thư điện tử là Secure/Multipurpose Internet Mail Extension (S/MIME). Giao thức này là một giải pháp nâng cao của giao thức Multipurpose Internet Mail Extension (MIME).

Để xác định cách các dịch vụ an ninh, chẳng hạn như tính bí mật và toàn vẹn, có thể được thêm vào các loại nội dung MIME, S/MIME đã định nghĩa *cú pháp tin nhắn mật mã* (*Cryptographic Message Syntax*, CMS). Cú pháp trong mỗi trường hợp xác định các phương thức mã hóa chính xác cho từng loại nội dung. Sau đây sẽ mô tả các kiểu tin nhắn (tham khảo RFC 3369 và RFC 3370).

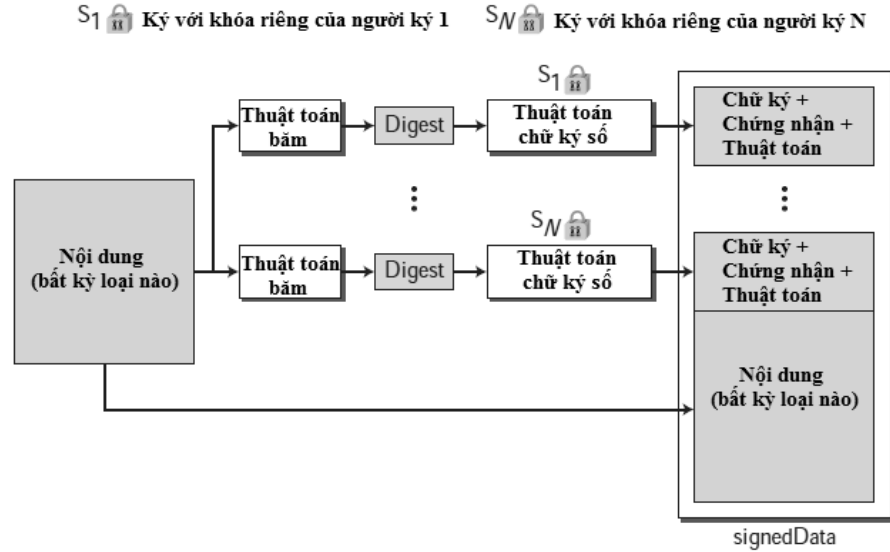
a. Kiểu nội dung dữ liệu:

Đây là một chuỗi tùy ý. Đối tượng được tạo ra được gọi là dữ liệu.

b. Kiểu nội dung dữ liệu được ký

Kiểu này chỉ cung cấp tính toàn vẹn cho dữ liệu. Nó chứa kiểu bất kỳ và 0 hoặc một số giá trị chữ ký. Kết quả mã hóa là một đối tượng được gọi là dữ liệu được ký. Hình 5.27 cho thấy quá trình tạo ra một đối tượng loại này. Sau đây là các bước trong quá trình:

1. Đối với mỗi người ký, một thông điệp tóm tắt được tạo ra từ nội dung bằng cách sử dụng thuật toán băm cụ thể được lựa chọn bởi người ký đó.
2. Mỗi thông điệp tóm tắt được ký với khóa riêng của người ký.
3. Nội dung, giá trị chữ ký, chứng nhận, và các thuật toán được thu thập để tạo ra đối tượng signedData.

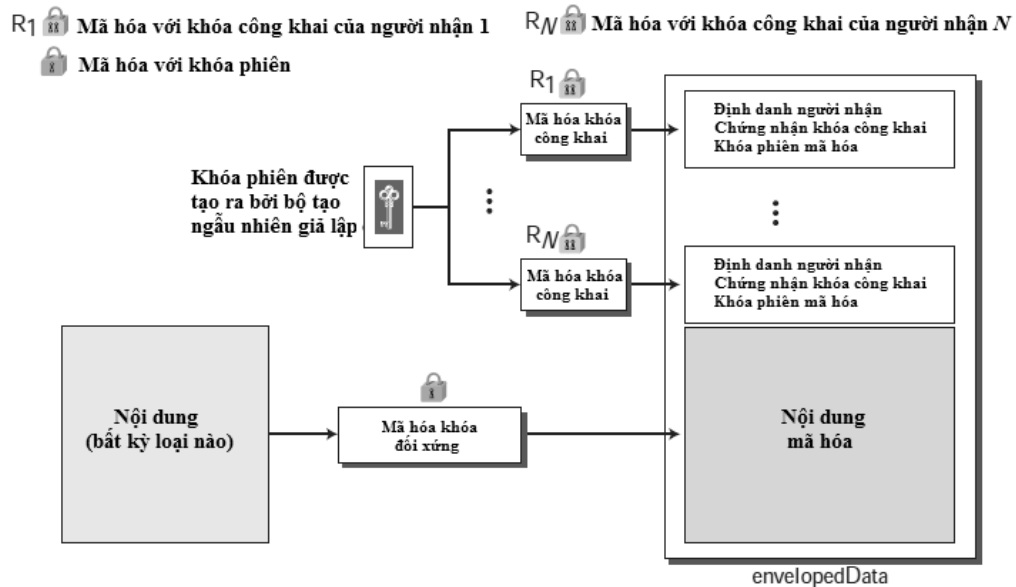


Hình 5.27 Kiểu nội dung dữ liệu được ký

c. Kiểu nội dung dữ liệu được đóng gói

Loại này được sử dụng để cung cấp bảo mật cho tin nhắn. Nó chứa bất kỳ loại nào và 0 hoặc một số khóa được mã hóa và chứng chỉ. Kết quả mã hóa là một đối tượng được gọi là envelopedData. Hình 5.28 cho thấy quá trình tạo ra một đối tượng loại này.

1. Khóa phiên giả ngẫu nhiên được tạo ra cho các thuật toán khóa đối xứng được sử dụng.
2. Đối với mỗi người nhận, một bản sao của khóa phiên được mã hóa với khóa công khai của mỗi người nhận.
3. Nội dung được mã hóa bằng thuật toán đã xác định và khóa phiên đã tạo.
4. Nội dung mã hóa, khóa phiên mã hóa, thuật toán đã sử dụng, và các chứng chỉ được mã hóa bằng cách sử dụng Radix-64.

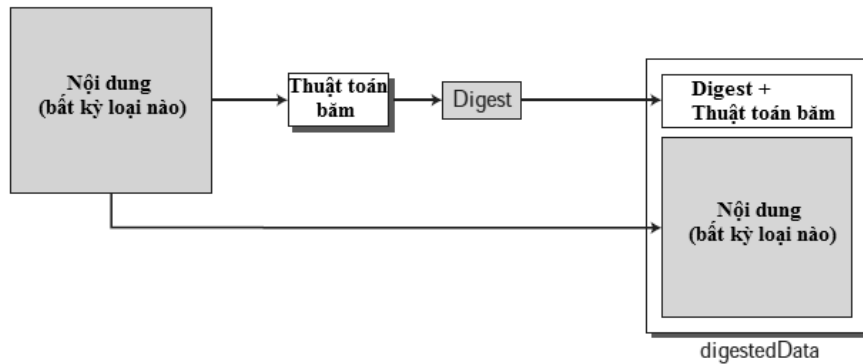


Hình 5.28 Kiểu nội dung dữ liệu được đóng gói

d. Kiểu nội dung dữ liệu thông điệp tóm tắt

Loại này được sử dụng để cung cấp tính toàn vẹn cho tin nhắn. Kết quả thường được sử dụng làm nội dung cho các loại nội dung dữ liệu được đóng gói. Kết quả mã hóa là một đối tượng được gọi là digestedData. Hình 5.29 cho thấy quá trình tạo ra một đối tượng loại này.

1. Thông điệp tóm tắt được tính từ nội dung.
2. Thông điệp tóm tắt, thuật toán, và nội dung được thêm vào với nhau để tạo ra đối tượng digestedData.



Hình 5.29 Kiểu nội dung dữ liệu thông điệp tóm tắt (digest)

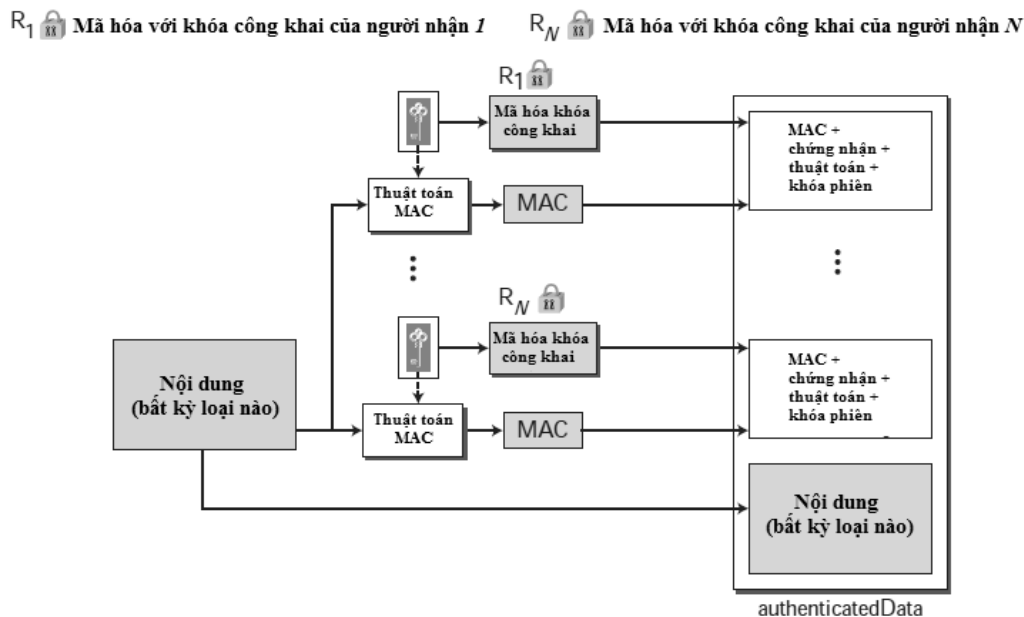
e. Kiểu nội dung dữ liệu mã hóa

Loại này được sử dụng để tạo ra một phiên bản mã hóa của bất kỳ loại nội dung nào. Mặc dù điều này có vẻ giống như kiểu nội dung dữ liệu được đóng gói, nhưng kiểu nội dung dữ liệu mã hóa không có người nhận. Nó có thể được sử dụng để lưu trữ dữ liệu được mã hóa thay vì chuyển nó. Quá trình này rất đơn giản; người dùng sử dụng bất kỳ khóa nào (thường là từ mật khẩu) và bất kỳ thuật toán nào mã hóa nội dung. Nội dung mã hóa được lưu trữ mà không bao gồm khóa hoặc thuật toán. Đối tượng được tạo ra gọi là encryptedData.

f. Kiểu nội dung dữ liệu xác thực

Loại này được sử dụng để cung cấp sự xác thực của dữ liệu. Đối tượng được gọi là authenticatedData (hình 5.30).

1. Sử dụng một bộ tạo ngẫu nhiên giả lập, tạo khóa MAC cho mỗi người nhận.
2. Khóa MAC được mã hóa với khóa công khai của người nhận.
3. Một MAC được tạo ra cho nội dung.
4. Nội dung, MAC, thuật toán, và các thông tin khác được thu thập cùng nhau để tạo ra đối tượng authenticatedData.



Hình 5.30 Kiểu nội dung dữ liệu xác thực

g. Quản lý khóa

Việc quản lý khóa trong S/MIME là một sự kết hợp của quản lý khóa sử dụng bởi X.509 và PGP. S/MIME sử dụng các chứng nhận khóa công khai được ký bởi các cơ quan chứng nhận theo quy định của X.509. Tuy nhiên, người dùng có trách nhiệm duy trì việc xác minh chữ ký theo quy định của PGP.

h. Thuật toán mật mã

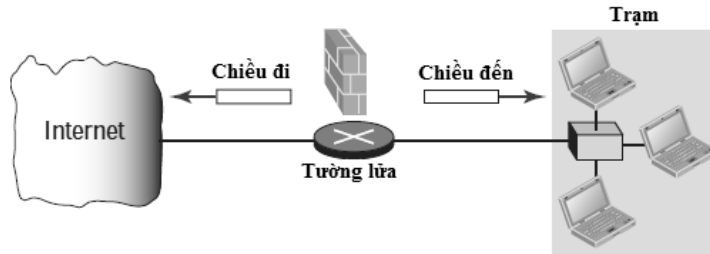
S/MIME định nghĩa một số thuật toán mật mã. Trong phạm vi tài liệu này sẽ không trình bày chi tiết về các thuật toán mật mã.

i. Các ứng dụng của S/MIME

Trong tương lai, S/MIME có thể sẽ trở thành sự lựa chọn công nghiệp để cung cấp an ninh cho e-mail thương mại.

5.4 TƯỜNG LỬA

Tất cả các biện pháp an ninh trước không thể ngăn chặn một người C nào đó gửi tin nhắn có hại cho hệ thống. Để kiểm soát quyền truy cập vào một hệ thống cần phải có tường lửa. Tường lửa là một thiết bị (thường là một bộ định tuyến hoặc một máy tính) được cài đặt giữa mạng nội bộ của một tổ chức và phần còn lại của Internet. Nó được thiết kế để chuyển tiếp một số gói dữ liệu và lọc những gói tin khác (không thực hiện chuyển tiếp). Hình 5.31 cho thấy một bức tường lửa.

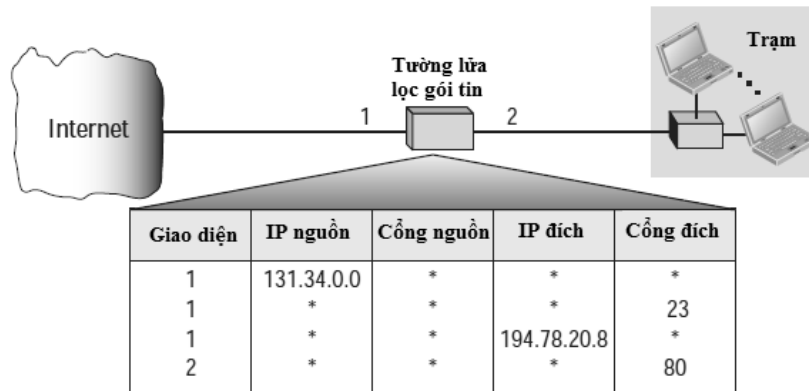


Hình 5.31 Tường lửa

Ví dụ, tường lửa có thể lọc tất cả các gói dữ liệu đến một host cụ thể hoặc một máy chủ cụ thể như HTTP. Tường lửa cũng có thể được sử dụng để từ chối truy cập đến một host hoặc một dịch vụ cụ thể trong tổ chức. Tường lửa thường được phân thành 2 loại là *tường lửa lọc gói tin* hoặc *tường lửa proxy*.

5.4.1 Tường lửa lọc gói tin (Packet-Filter Firewall)

Tường lửa có thể được sử dụng như một bộ lọc gói tin. Nó có thể chuyển tiếp hoặc chặn các gói tin dựa trên thông tin trong các tiêu đề tầng mạng và tầng giao vận: địa chỉ IP nguồn và địa chỉ IP đích, địa chỉ cổng nguồn và địa chỉ cổng đích, và loại giao thức (TCP hoặc UDP). Một tường lửa lọc gói là một bộ định tuyến có sử dụng một bảng lọc để quyết định liệu gói tin cần bị loại bỏ hay không (không được chuyển tiếp). Hình 5.32 cho thấy ví dụ về một bảng lọc cho loại tường lửa này.



Hình 5.32 Tường lửa lọc gói tin

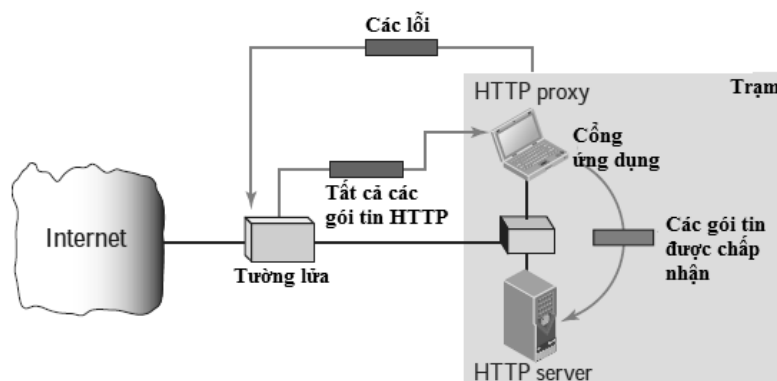
Các gói tin sau đây được lọc:

1. Gói tin đến từ mạng 131.34.0.0 sẽ bị chặn (cảnh báo an ninh). Ở đây, * (dấu sao) có nghĩa là "bất kỳ".
2. Gói tin đến bất kỳ máy chủ TELNET nội bộ nào (port 23) sẽ bị chặn.
3. Gói tin đến host nội bộ 194.78.20.8 sẽ bị chặn (khi tổ chức muốn host này chỉ để dùng nội bộ).
4. Gói tin gửi đi đến một máy chủ HTTP (cổng 80) sẽ bị chặn (khi tổ chức không muốn nhân viên lướt Web).

5.4.2 Tường lửa proxy

Tường lửa lọc gói tin dựa trên các thông tin có sẵn trong các tiêu đề tầng mạng và tầng giao vận (IP và TCP/UDP). Tuy nhiên, đôi khi cần phải lọc thông điệp dựa trên thông tin có sẵn trong các thông điệp đó (ở tầng ứng dụng). Ví dụ, giả sử một tổ chức muốn thực hiện các chính sách sau đây liên quan đến trang web của mình: chỉ có những người sử dụng Internet đã thiết lập quan hệ kinh doanh với công ty trước đây mới có thể truy cập và việc truy cập tới những người dùng khác phải bị chặn. Trong trường hợp này, việc sử dụng tường lửa lọc gói tin là không khả thi vì nó không thể phân biệt các gói tin khác nhau đi đến cổng TCP 80 (HTTP). Kiểm tra phải được thực hiện ở tầng ứng dụng (sử dụng các URL).

Một giải pháp là cài đặt một máy tính proxy (đôi khi được gọi là cổng ứng dụng), đứng giữa máy tính của khách hàng và máy tính của công ty. Khi client của người dùng gửi một thông điệp, cổng ứng dụng sẽ chạy một tiến trình server để nhận yêu cầu. Server sẽ mở các gói tin ở mức ứng dụng và phát hiện ra liệu yêu cầu là hợp pháp hay không. Nếu có, server hoạt động như một tiến trình client và gửi thông điệp tới server thực sự trong công ty. Nếu không, thông điệp bị loại bỏ và một thông báo lỗi sẽ được gửi đến người sử dụng bên ngoài. Bằng cách này, các yêu cầu của người dùng bên ngoài được lọc dựa trên nội dung ở tầng ứng dụng. Hình 5.33 cho thấy một cài đặt cổng ứng dụng cho HTTP.



Hình 5.33 Tường lửa proxy

TÀI LIỆU THAM KHẢO

- [1] Behrouz A. Forouzan, *TCP/IP Protocol Suite*, McGraw-Hill, 2010.
- [2] Javvin Technologies, *Network Protocols Handbook*, Javvin Technologies Inc., 2004-2005.
- [3] James F Kurose and Keith W. Ross, *Computer networking - A Top-Down Approach Featuring the Internet*, Addison-Wesley, 2004.
- [4] Douglas E. Comer, *Internetworking with TCP/IP - Volume One (6th Edition)*, Prentice Hall, 2013.
- [5] Kevin R. Fall, W. Richard Stevens, *TCP/IP Illustrated, Volume 1: The Protocols (2nd Edition)*, Addison-Wesley, 2011.
- [6] Laura Chappell, Gerald Combs, *Wireshark (R) 101: Essential Skills for Network Analysis (Wireshark Solutions)*, Chappell Univerisy, 2013.
- [7] Barrett, Daniel, J., Silverman, Richard, E., and Byrnes, Robert, G., *SSH: The Secure Shell*, Sebastopol, CA: O'Reilly, 2005.
- [8] Bishop, Matt, *Introduction to Computer Security*. Reading, MA: Addison-Wesley, 2005.