

ĐỀ CƯƠNG CUỐI KỲ

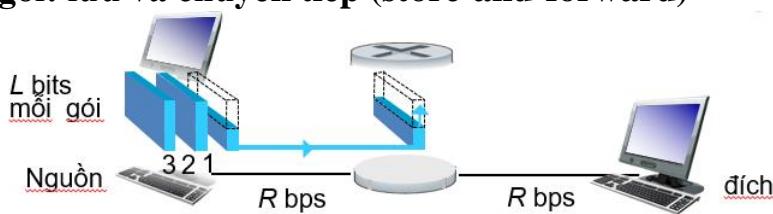
Môn: Nhập môn mạng máy tính

Chương 1.

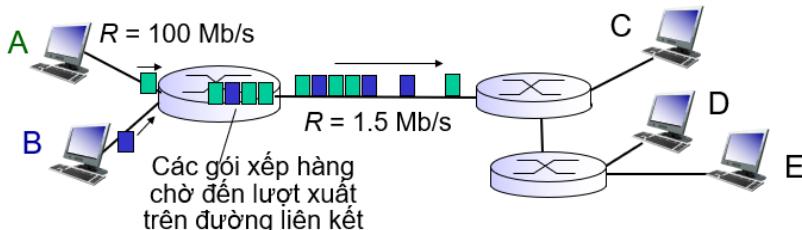
(10% - 4 câu)

1. Độ trễ, phân loại và tính toán (Slide 25, 26, 43, 44, 45)

Chuyển mạch gói: lưu và chuyển tiếp (store-and-forward)



- Mất L/R giây để truyền tải L -bit gói trong đường liên kết tại tốc độ R bps.
 - Lưu và chuyển tiếp:** toàn bộ các gói phải đến bộ định tuyến trước khi nó có thể được truyền tải trên đường liên kết tiếp theo.
- Ví dụ số về one-hop:
- $L = 7.5 \text{ Mbits}$
 - $R = 1.5 \text{ Mbps}$
 - $\text{Độ trễ truyền tải one-hop} = 5 \text{ sec}$
- Độ trễ giữa 2 đầu cuối (end-end delay) = $2L/R$ (giả sử không có độ trễ lan truyền)



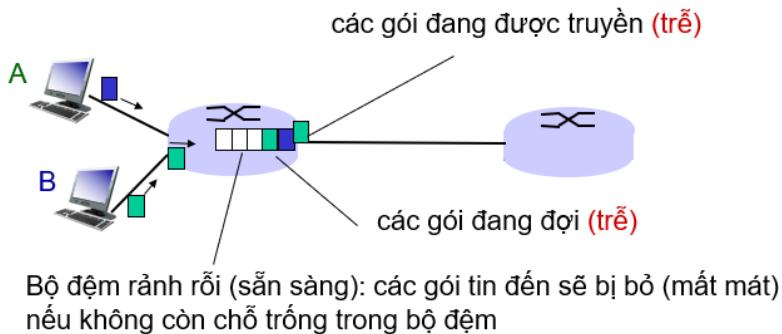
Xếp hàng và sự mất mát:

- Nếu tốc độ đến (theo bit) đến đường liên kết vượt quá tốc độ truyền dẫn của đường liên kết trong một khoảng thời gian:
 - Các gói sẽ xếp hàng và đợi để được truyền tải trên đường liên kết
 - Các gói có thể bị bỏ (bị mất) nếu bộ nhớ (bộ đệm) bị đầy

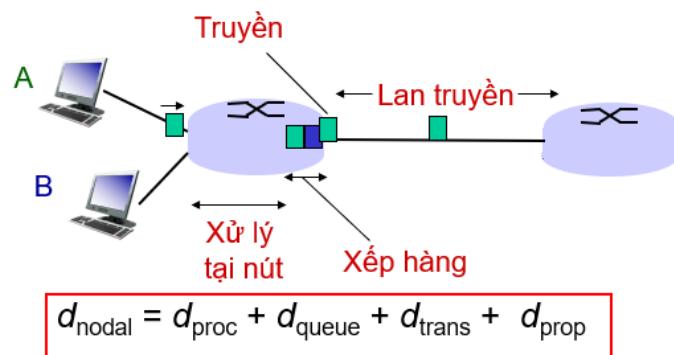
Sự mất mát và độ trễ xảy ra như thế nào?

Các gói tin đợi trong bộ đệm của bộ định tuyến (router)

- Tốc độ đến của các gói tin đến đường liên kết (tạm thời) vượt quá khả năng của đường liên kết đầu ra
- Các gói tin đợi và chờ đến lượt



Bốn nguồn gây ra chậm trễ gói tin



dproc: Xử lý tại nút

- Kiểm tra các bit lỗi
- Xác định đường ra
- Thông thường < msec

dqueue: Độ trễ xếp hàng

- Thời gian đợi tại cổng ra cho việc truyền dữ liệu
- Phụ thuộc vào mức độ tắc nghẽn của bộ định tuyến

dtrans: Trễ do truyền:

- L: chiều dài gói (bits)
- R: băng thông đường liên kết (bps)
- $d_{\text{trans}} = L/R$

dprop: Trễ do lan truyền:

- d: độ dài của đường liên kết vật lý
- s: tốc độ lan truyền trong môi trường (thiết bị, dây dẫn) ($\sim 2 \times 10^8$ m/sec)
- $d_{\text{prop}} = d/s$

**dtrans and dprop rất khác nhau*

2. Mô hình mạng (TCP/IP, OSI) (Slide 60-62): thứ tự các tầng lớp, đơn vị dữ liệu của các tầng và thiết bị tại các tầng

Chồng giao thức Internet

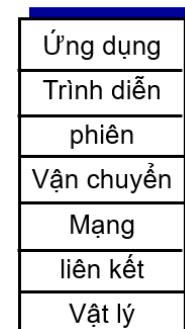
- **Ứng dụng (application):** hỗ trợ các ứng dụng mạng (*Đơn vị dữ liệu: Data*)
 - FTP, SMTP, HTTP
- **Vận chuyển (transport):** chuyển dữ liệu từ tiến trình này đến tiến trình kia (process-process) (*Đơn vị dữ liệu: Segment*)
 - TCP, UDP
- **Mạng (network):** định tuyến những gói dữ liệu từ nguồn tới đích (*Đơn vị dữ liệu: Packet*)
 - IP, các giao thức định tuyến
- **Liên kết (data link):** chuyển dữ liệu giữa các thành phần mạng lân cận (*Đơn vị dữ liệu: Frame*)
 - Ethernet, 802.111 (WiFi), PPP
- **Vật lý (physical):** các bit “trên đường dây” (*Đơn vị dữ liệu: Bit*)



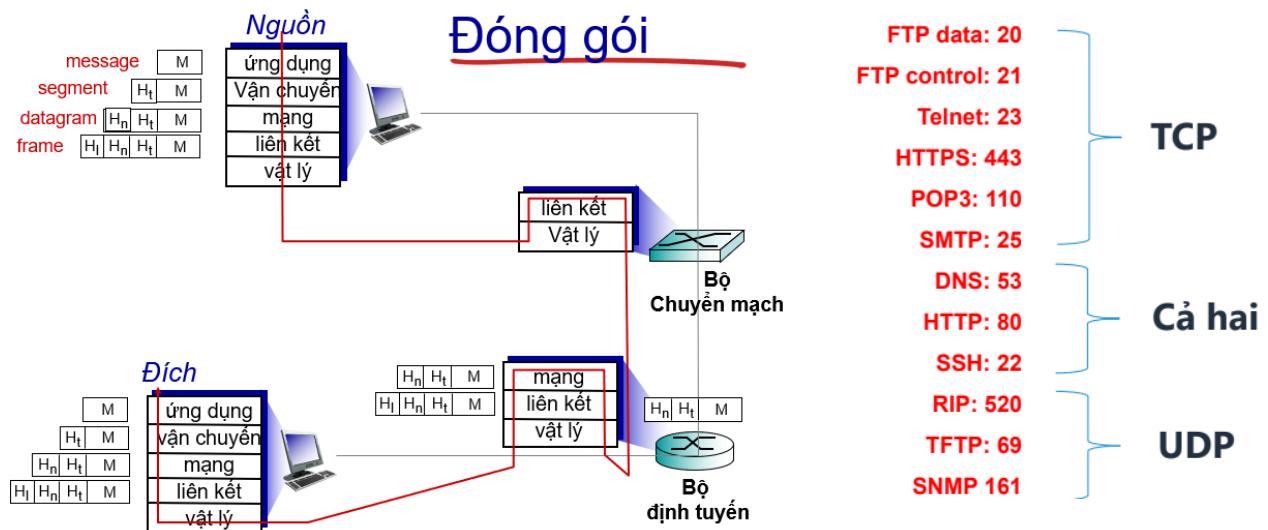
Mô hình tham chiếu ISO/OSI

Có thêm:

- **Trình diễn (presentation):** cho phép các ứng dụng giải thích ý nghĩa của dữ liệu, ví dụ mã hóa, nén, những quy ước chuyên biệt (*Đơn vị dữ liệu: Data*)
- **Phiên (session):** sự đồng bộ hóa, khả năng chịu lỗi, phục hồi sự trao đổi dữ liệu (*Đơn vị dữ liệu: Data*)
- Chồng giao thức Internet “thiếu” những lớp này!
 - Những dịch vụ này, nếu cần, phải được thực hiện trong lớp ứng dụng (application layer)
 - Cần hay không?



Đóng gói & Port của một số giao thức

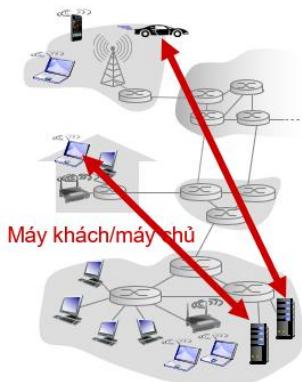


Chương 2.

(20% - 8 câu)

1. Các kiến trúc (Client-server và Peer to Peer) (Slide 7,8)

Kiến trúc máy khách-máy chủ



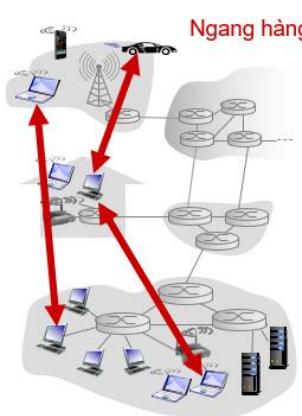
Máy chủ (server):

- Máy luôn hoạt động
- Địa chỉ IP cố định
- Tổ chức thành các trung tâm dữ liệu để mở rộng quy mô

Máy khách (client):

- Giao tiếp với máy chủ
- Có thể kết nối không liên tục
- Có thể thay đổi địa chỉ IP
- Không giao tiếp trực tiếp với các máy khách khác

Kiến trúc P2P (ngang hàng)

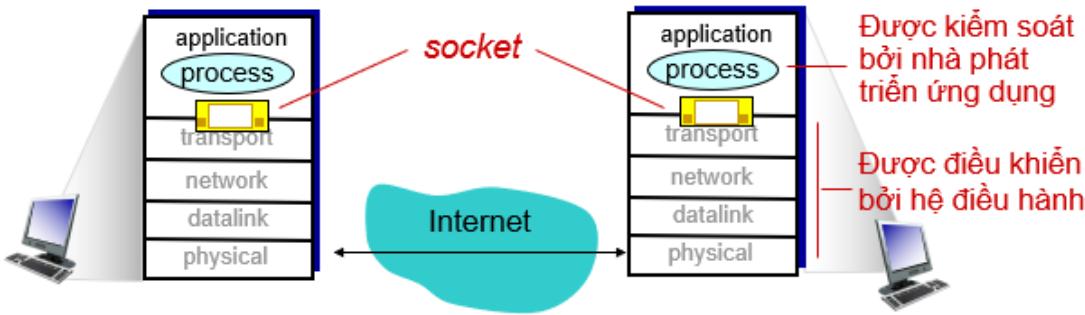


- Không có máy chủ
- Các hệ thống đầu cuối bất kỳ (peer) truyền thông trực tiếp với nhau
 - Các peer yêu cầu dịch vụ từ các bên khác và cung cấp dịch vụ ngược lại cho các bên khác
 - **Có khả năng tự mở rộng – các peer mới cung cấp thêm dịch vụ mới, cũng như có thêm nhu cầu mới về dịch vụ**
 - Các peer được kết nối không liên tục và có thể thay đổi địa chỉ IP
 - Quản lý phức tạp

2. Socket (Slide 10-16)

Sockets

- Tiến trình gửi/nhận thông điệp đến/từ socket của nó
- Socket tương tự như cửa ra vào
 - Tiến trình gửi đầy thông điệp ra khỏi cửa
 - Tiến trình gửi dựa trên hạ tầng vận chuyển bên kia của cánh cửa để phân phối thông điệp đến socket tại tiến trình nhận



Xác định tiến trình

- Để nhận thông điệp, tiến trình phải có **định danh**
- Thiết bị hệ thống đầu cuối có địa chỉ IP 32-bit duy nhất
- Q:** địa chỉ IP của hệ thống đầu cuối mà trên tiến trình đang chạy trên đó có đủ để xác định tiến trình đó hay không?
 - A:** không, có nhiều tiến trình có thể đang được chạy trên cùng một hệ thống đầu cuối
- Định danh (identifier)** bao gồm cả địa chỉ IP và **số cổng (port number)** được liên kết với tiến trình trên hệ thống đầu cuối.
- Ví dụ về số cổng:
 - Máy chủ Web: 80
 - Máy chủ thư điện tử: 25
- Để gửi thông điệp HTTP đến Web máy chủ gaia.cs.umass.edu :

 - Địa chỉ IP:** 128.119.245.12
 - Số cổng:** 80

- Còn nữa...

Giao thức lớp **Ứng dụng: định nghĩa**

- Các loại thông điệp được trao đổi**
 - e.g., yêu cầu (request), đáp ứng (response)
- Cú pháp thông điệp:**
 - Các trường trong thông điệp và cách mà các trường được định nghĩa
- Ngữ nghĩa của thông điệp**
 - Ý nghĩa của thông tin trong các trường
- Các quy tắc (rules)** khi nào và cách mà các tiến trình gửi và đáp ứng các thông điệp
- Các giao thức mở:**
 - Được định nghĩa trong RFCs
 - Cung cấp khả năng tương tác cho các ứng dụng thuộc các nhà phát triển khác nhau
 - Vd: HTTP, SMTP
- Các giao thức độc quyền:**
 - Vd: Skype

Dịch vụ vận chuyển nào mà ứng dụng cần?

Toàn vẹn dữ liệu (data integrity)

- Một số ứng dụng (ví dụ truyền tập tin, web...) yêu cầu độ tin cậy 100% khi truyền dữ liệu
- Các ứng dụng khác (ví dụ audio) có thể chịu được một số mất mát.

Định thời (timing)

- Một số ứng dụng (ví dụ, thoại Internet, game tương tác) yêu cầu độ trễ thấp để đạt được “hiệu quả” thực thi

Thông lượng (throughput)

- Một số ứng dụng (vd: đa phương tiện) yêu cầu thông lượng tối thiểu để đạt được “hiệu quả” thực thi
- Các ứng dụng khác (“ứng dụng mềm dẻo”) có thể dùng bất kỳ thông lượng nào cũng được

An ninh

- Mã hóa, toàn vẹn dữ liệu, ...

Các yêu cầu dịch vụ vận chuyển: các ứng dụng phổ biến

| Ứng dụng | Mất dữ liệu | Thông lượng | Độ nhạy thời gian |
|---------------------------------|-------------|---|-------------------|
| Truyền tập tin | Không | Mềm dẻo | Không |
| Thư điện tử | Không | Mềm dẻo | Không |
| Web documents | Không | Mềm dẻo | Không |
| Audio/video theo thời gian thực | Chịu lỗi | Audio: 5kbps-1Mbps Video: 10kbps-5Mbps | Có, 100' s msec |
| Audio/video đã lưu | Chịu lỗi | Như trên | Có, vài giây |
| Game tương tác | Chịu lỗi | Trên một vài kbps | Có, 100' s msec |
| Nhắn tin | Không | Mềm dẻo | Có và không |

Các dịch vụ thuộc giao thức vận chuyển trên Internet

Dịch vụ TCP:

- **Truyền tải có đảm bảo (reliable transport)** giữa tiến trình gửi và nhận
- **Điều khiển luồng thông tin (flow control):** bên gửi sẽ không gửi vượt khả năng bên nhận
- **Điều khiển tắc nghẽn (congestion control):** điều tiết bên gửi khi mạng quá tải
- **Không hỗ trợ:** định thời, bảo đảm thông lượng tối thiểu, bảo mật
- **Hướng kết nối (connection-oriented):** yêu cầu thiết lập kết nối giữa tiến trình máy khách và máy chủ trước khi truyền

Dịch vụ UDP:

- **Truyền dữ liệu không đảm bảo (unreliable data transfer)** giữa tiến trình gửi và nhận
- **Không hỗ trợ:** độ tin cậy, điều khiển luồng, điều khiển tắc nghẽn, định thời, bảo đảm thông lượng, bảo mật, và thiết lập kết nối.

Q: Tại sao phải quan tâm? Tại sao có UDP?

Ứng dụng Internet: Các giao thức lớp ứng dụng

| Ứng dụng | Giao thức lớp Ứng dụng | Giao thức dưới lớp Vận chuyển |
|------------------------|---|----------------------------------|
| Thư điện tử | SMTP [RFC 2821] | TCP |
| Remote terminal access | Telnet [RFC 854] | TCP |
| Web | HTTP [RFC 2616] | TCP |
| Truyền tập tin | FTP [RFC 959] | TCP |
| Streaming multimedia | HTTP (e.g., YouTube), RTP [RFC 1889] | TCP or UDP |
| Thoại Internet | SIP, RTP, đặc quyền (e.g., Skype) | TCP or UDP |

3. HTTP (*Slide 20-27, 30-32, 34-38*)

Tổng quan HTTP

HTTP: hypertext transfer protocol

- Giao thức lớp ứng dụng của Web
- Mô hình client/server
 - **Client:** trình duyệt gửi yêu cầu, nhận phản hồi (dùng giao thức HTTP) và “hiển thị” các đối tượng Web
 - **Server:** Web máy chủ gửi (dùng giao thức HTTP) các đối tượng để trả lời yêu cầu

Dùng TCP:

- Máy khách khởi tạo kết nối TCP (tạo socket) đến cổng 80 của máy chủ
- Máy chủ chấp nhận kết nối TCP từ máy khách
- Các thông điệp HTTP (thông điệp thuộc giao thức lớp ứng dụng) được trao đổi giữa trình duyệt (HTTP client) và máy chủ web (HTTP server)
- Kết nối TCP được đóng



HTTP “không lưu trạng thái”

- Máy chủ không duy trì thông tin về các yêu cầu trước đó của máy khách

Các kết nối HTTP

HTTP không bền vững

- Chỉ tối đa một đối tượng được gửi qua kết nối TCP
 - Kết nối sau đó sẽ bị đóng
- Tải nhiều đối tượng yêu cầu nhiều kết nối

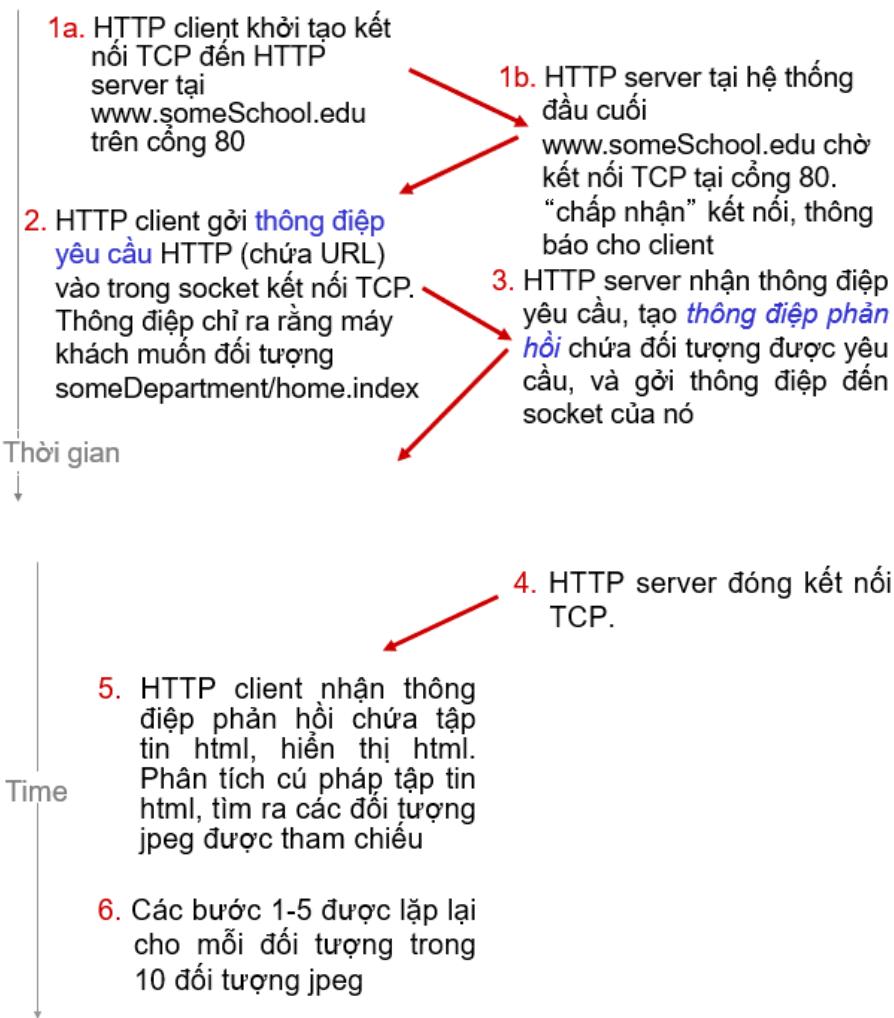
HTTP bền vững

- Nhiều đối tượng có thể được gửi qua một kết nối TCP giữa máy khách và máy chủ

HTTP không bền vững

Giả sử người dùng vào URL như sau:
www.someSchool.edu/someDepartment/home.index

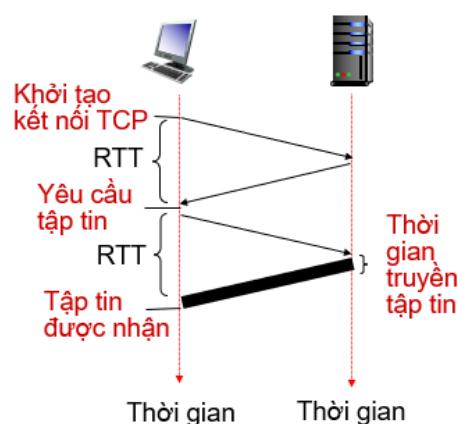
(chứa text,
tham chiếu đến 10
hình jpeg)



RTT (round-trip time): thời gian để cho một gói tin nhỏ đi từ máy khách đến máy chủ và quay ngược lại

Thời gian đáp ứng HTTP:

- Một RTT để khởi tạo kết nối TCP
- Một RTT cho yêu cầu HTTP và vài byte đầu tiên của phản hồi HTTP được trả về
- Thời gian truyền tập tin
- Thời gian đáp ứng HTTP không bền vững = $2\text{RTT} + \text{thời gian truyền tập tin}$



HTTP bền vững

Vấn đề với HTTP không bền vững:

- Yêu cầu 2 RTTs cho mỗi đối tượng
- Tốn tài nguyên khi Hệ điều hành xử lý mỗi kết nối TCP
- Các trình duyệt thường mở các kết nối TCP song song để lấy các đối tượng được tham chiếu

HTTP bền vững:

- Máy chủ để kết nối mở sau khi gửi phản hồi
- Các thông điệp HTTP tiếp theo giữa cùng máy khách/máy chủ được gửi trên kết nối đã mở ở trên
- Máy khách gửi các yêu cầu ngay khi nó gặp một đối tượng tham chiếu
- Chỉ cần một RTT cho tất cả các đối tượng được tham chiếu

Thông điệp yêu cầu HTTP

- Hai loại thông điệp HTTP: yêu cầu (**request**), phản hồi (**response**)
- **Thông điệp yêu cầu HTTP:**
 - ASCII (dạng thức con người có thể đọc được)



Các phương thức

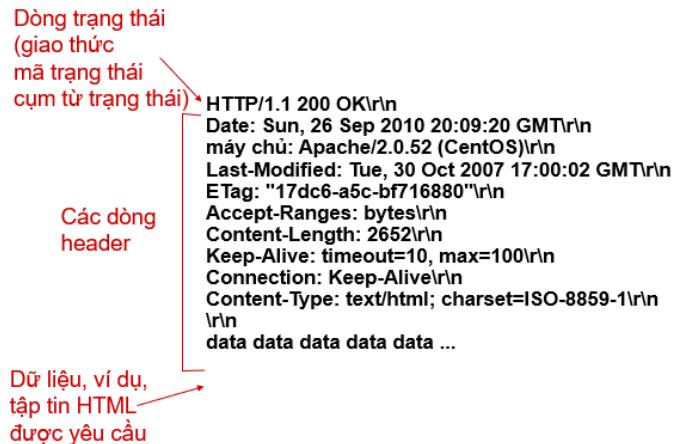
HTTP/1.0:

- GET
- POST
- HEAD
 - Yêu cầu máy chủ loại bỏ đối tượng được yêu cầu ra khỏi thông điệp phản hồi

HTTP/1.1:

- GET, POST, HEAD
- PUT
 - Tải tập tin trong thân thực thể đến đường dẫn được xác định trong trường URL
- DELETE
 - Xóa tập tin được chỉ định trong trường URL

Thông điệp phản hồi HTTP



Các mã trạng thái phản hồi HTTP

- Mã trạng thái xuất hiện trong dòng đầu tiên trong thông điệp đáp ứng từ máy chủ tới máy khách
- Một số mã mẫu:

200 OK

- Yêu cầu thành công, đối tượng được yêu cầu sau ở trong thông điệp này

301 Moved Permanently

- Đối tượng được yêu cầu đã được di chuyển, vị trí mới được xác định sau trong trường Location:

304 Not Modified

- Nếu header yêu cầu bao gồm tham số 'if modified since', mã trạng thái này sẽ được trả về, trong trường hợp file không thay đổi kể từ ngày đó.

400 Bad Request

- Máy chủ không hiểu thông điệp yêu cầu

404 Not Found

- Thông tin được yêu cầu không tìm thấy trên máy chủ này

502 Bad Gateway

505 HTTP Version Not Supported

Trạng thái người dùng-máy chủ: cookies

Nhiều Website dùng cookies

4 thành phần:

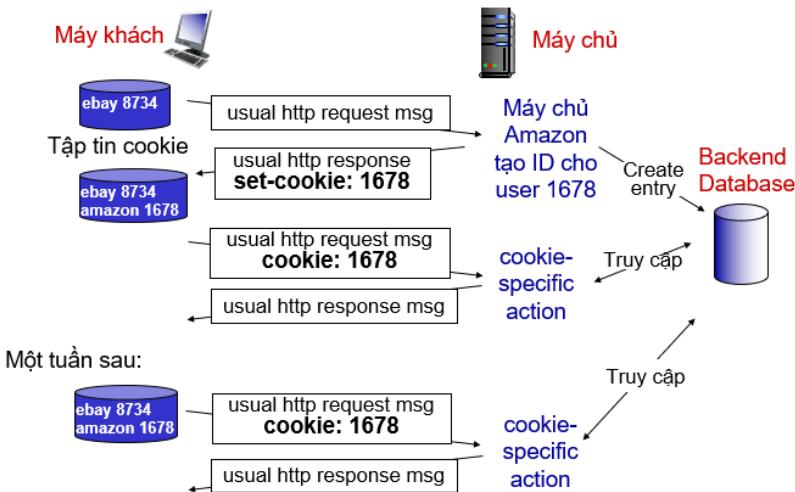
- Dòng đầu cookie (cookie header line) của thông điệp phản hồi HTTP
- Cookie header line trong thông điệp yêu cầu HTTP kế tiếp
- Tập tin cookie được lưu trữ trên máy người dùng, được quản lý bởi trình duyệt của người dùng
- Cơ sở dữ liệu tại Web site

Ví dụ:

- Susan thường truy cập Internet từ một PC
- Vào trang thương mại điện tử lần đầu tiên

- Khi yêu cầu khởi tạo HTTP đến trang web đó, thì trang đó tạo:
 - ID duy nhất
 - Một bản ghi trong cơ sở dữ liệu cho ID đó

Cookies: lưu trữ “trạng thái”



Cookies

Cookie có thể được sử dụng cho:

- Cấp phép
- Giỏ mua hàng
- Các khuyến cáo
- Trạng thái phiên làm việc của user (Web thư điện tử)

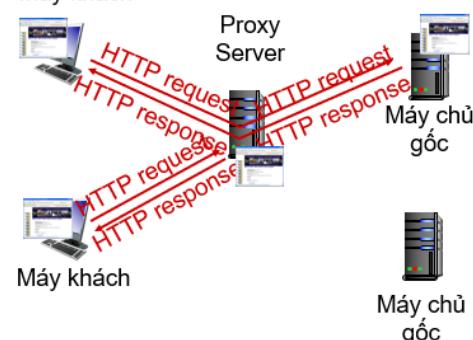
Làm thế nào để giữ “trạng thái”:

- Các thời điểm kết thúc giao thức: duy trì trạng thái tại người gửi/nhận thông qua nhiều giao dịch
- Cookies: các thông điệp HTTP mang trạng thái

Web caches (web đệm) (proxy server)

Mục tiêu: thỏa mãn yêu cầu của máy khách không cần liên quan đến máy chủ nguồn

- User thiết lập trình duyệt: truy cập Web thông qua cache
- Trình duyệt gửi tất cả yêu cầu HTTP đến cache
 - Đối tượng có trong cache: cache trả về đối tượng
 - Ngược lại cache yêu cầu đối tượng từ máy chủ gốc, sau đó trả đối tượng đó cho máy khách



Thông tin thêm về Web caching

- Cache hoạt động như cả máy khách và máy chủ
 - Máy chủ đối với máy khách yêu cầu thông tin

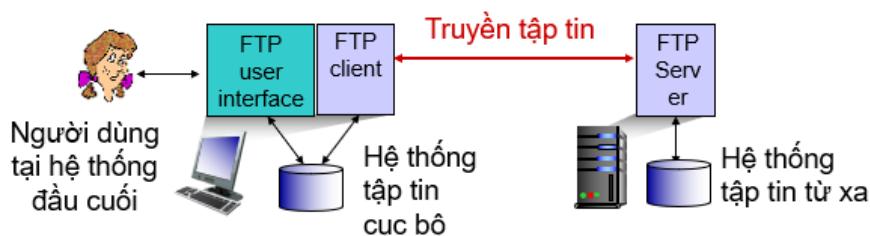
- Máy khách đối với máy chủ cung cấp
- Thông thường cache được cài đặt bởi ISP (trường đại học, công ty, ISP riêng)

Tại sao dùng Web caching?

- Giảm thời gian đáp ứng cho yêu cầu của máy khách
- Giảm lưu lượng trên đường liên kết truy cập ra Internet của một tổ chức
- Internet có rất nhiều cache: cho phép những nhà cung cấp nội dung với lượng tài nguyên “nghèo nàn” vẫn cung cấp nội dung một cách hiệu quả (chia sẻ tập tin P2P cũng vậy)

4. FTP các khái niệm (Slide 45, 46)

FTP: giao thức truyền tập tin



- Truyền tập tin đến/từ máy ở xa
- Mô hình máy khách/máy chủ
 - **Máy khách:** phía khởi tạo phiên truyền (đến/từ máy ở xa)
 - **Máy chủ:** máy ở xa
- FTP: RFC 959
- FTP server: cổng 21

FTP: kết nối điều khiển và kết nối dữ liệu riêng biệt

- FTP máy khách liên hệ với FTP máy chủ tại cổng 21, dùng TCP
- Máy khách được cấp phép trên kết nối điều khiển
- Máy khách duyệt thư mục từ xa, bằng cách gửi các lệnh trên kết nối điều khiển
- Khi máy chủ nhận lệnh truyền tập tin, **máy chủ** mở kết nối dữ liệu TCP thứ 2 (để truyền tập tin) đến máy khách
- Sau khi truyền một tập tin, máy chủ đóng kết nối dữ liệu
- Máy chủ mở kết nối dữ liệu TCP khác để truyền tập tin khác
- Kết nối điều khiển: "*out of band*" (*ngoại tuyến*)
- FTP máy chủ duy trì “trạng thái”: thư mục hiện tại, xác thực trước đó



5. Giao thức về thư điện tử SMTP, POP (Slide 49-52, 57-59)

Thư điện tử

Ba thành phần chính:

- User agents

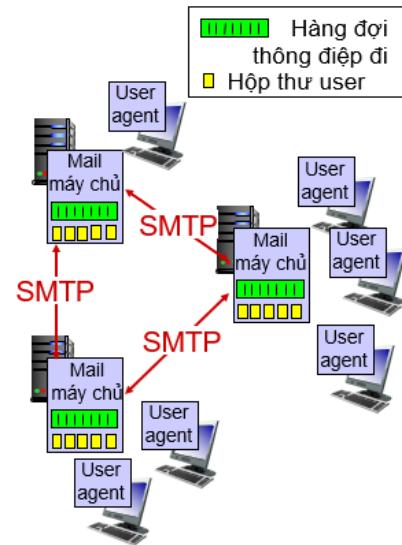
- Máy chủ thư điện tử (mail server)
- Simple mail transfer protocol: SMTP

User Agent

- Còn gọi là “mail reader”
- Soạn thảo, sửa đổi, đọc các thông điệp thư điện tử
- Ví dụ Outlook, Thunderbird, iPhone mail máy khách
- Các thông điệp đi và đến được lưu trên máy chủ

Máy chủ thư điện tử:

- Hộp thư (mailbox) chứa thông điệp đến user
- Hàng thông điệp (message queue) của các thông điệp mail ra ngoài (chuẩn bị gửi)
- Giao thức SMTP được dùng để gửi các thông điệp thư điện tử giữa các mail server
 - Client là máy chủ thư điện tử gửi
 - Server là máy chủ thư điện tử nhận

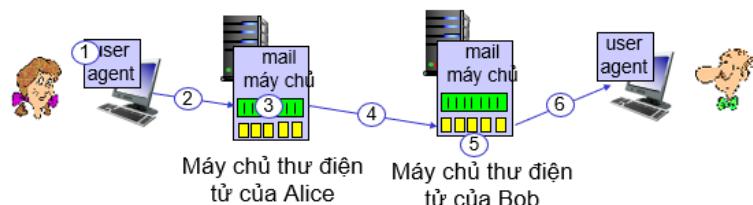


SMTP [RFC 2821]

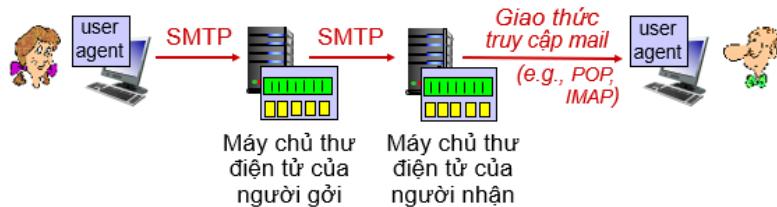
- Sử dụng TCP để truyền thông điệp thư điện tử một cách tin cậy từ máy khách đến cổng 25 máy chủ
- Truyền trực tiếp: máy chủ gửi thư đến máy chủ nhận
- 3 giai đoạn truyền
 - Bắt tay
 - Truyền thông điệp
 - Đóng
- Tương tác lệnh/phản hồi (như HTTP, FTP)
 - Lệnh: văn bản ASCII
 - Phản hồi: mã và cụm trạng thái
- Thông điệp phải ở dạng mã ASCII 7 bit

Tình huống: Alice gửi thông điệp đến Bob

- 1) Alice dùng một UA để soạn thảo thông điệp “gửi đến” bob@someschool.edu
- 2) UA của Alice gửi thông điệp đến máy chủ thư điện tử của cô ta; thông điệp được đặt trong hàng đợi
- 3) Phần máy khách của SMTP máy chủ mở kết nối TCP với máy chủ thư điện tử của Bob
- 4) Phần máy khách của SMTP máy chủ gửi thông điệp của Alice trên kết nối TCP
- 5) Máy chủ thư điện tử của Bob đặt thông điệp đó trong hộp thư của Bob
- 6) Bob kích hoạt user agent của anh ta để đọc thông điệp

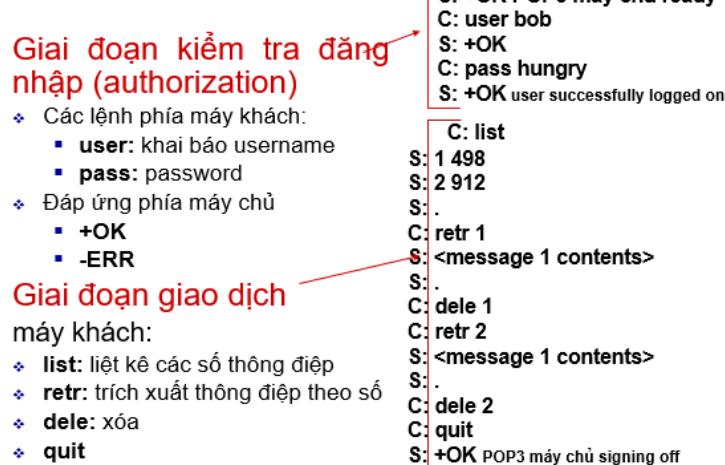


Các giao thức truy cập Mail



- **SMTP:** truyền dẫn/lưu trữ thư vào máy chủ của người nhận
- Giao thức truy cập thư: trích xuất từ máy chủ
 - **POP:** Post Office Protocol [RFC 1939]: xác thực, tải thư về
 - **IMAP:** Internet Mail Access Protocol [RFC 1730]: nhiều tính năng hơn, bao gồm cả các thao tác thay đổi các thông điệp đang được lưu trên máy chủ
 - **HTTP:** gmail, Hotmail, Yahoo! Mail...

Giao thức POP3



POP3 và IMAP

Tìm hiểu thêm về POP3

- Ví dụ trên sử dụng chế độ “tải xuống và xóa” của POP3
 - Bob không thể đọc lại thư điện tử nếu anh ta thay đổi máy khách
- Chế độ “tải xuống-và-giữ” của POP3: sao chép các thông điệp trên các máy khách khác nhau
- POP3 không giữ trạng thái của các phiên làm việc

IMAP

- Giữ tất cả các thông điệp ở một nơi: tại máy chủ
- Cho phép người dùng tổ chức, sắp xếp các thông điệp trong các thư mục
- Giữ trạng thái của người dùng trong suốt phiên làm việc:
 - Các tên của các thư mục và ánh xạ giữa các ID của thông điệp và tên của thư mục

6. DNS (Slide 61-70)

Hệ thống tên miền (DNS: domain name system)

Con người: nhiều cách nhận dạng:

- Số an sinh xã hội, tên, số hộ chiếu

Các hệ thống đầu cuối Internet, bộ định tuyến:

- Địa chỉ IP (32 bit) – được dùng cho định địa chỉ gói tin
- “tên”, ví dụ www.yahoo.com – được dùng bởi con người

Q: làm cách nào để ánh xạ giữa địa chỉ IP và tên, và ngược lại?

Hệ thống tên miền (Domain Name System):

- Cơ sở dữ liệu phân tán được thực hiện theo tổ chức phân cấp của nhiều máy chủ DNS
- Giao thức lớp ứng dụng: các hệ thống đầu cuối, các máy chủ tên miền trao đổi để phân giải tên (dịch địa chỉ \Leftrightarrow tên)
 - Lưu ý: chức năng trong phần lõi Internet, được thực hiện như là giao thức lớp ứng dụng
 - Sự phức tạp ở “biên” của mạng”

DNS: các dịch vụ, cấu trúc

Các dịch vụ DNS

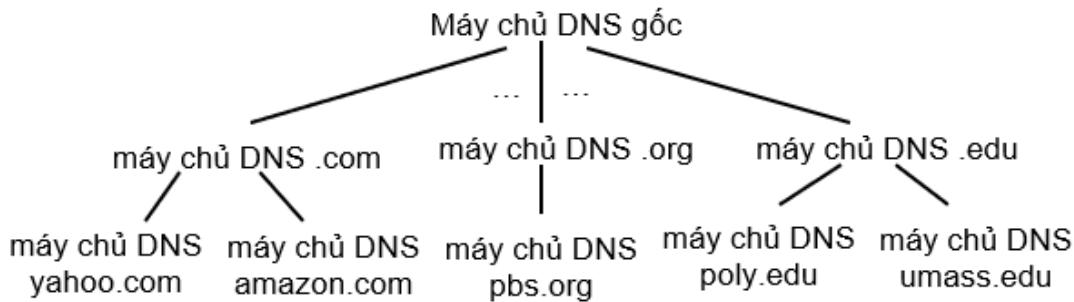
- Dịch tên máy ra địa chỉ IP
- Bí danh máy
 - Lưu các tên gốc, bí danh tương ứng
- Bí danh máy chủ thư điện tử
- Cân bằng tải
 - Các bản sao cho web server: nhiều địa chỉ IP tương ứng cho 1 tên miền

Tại sao không tập trung hóa DNS?

- Một điểm chịu lỗi
- Lưu lượng
- Cơ sở dữ liệu tập trung cách xa nơi yêu cầu
- Bảo trì

A: **không biến đổi được quy mô!**

DNS: cơ sở dữ liệu phân cấp, phân tán

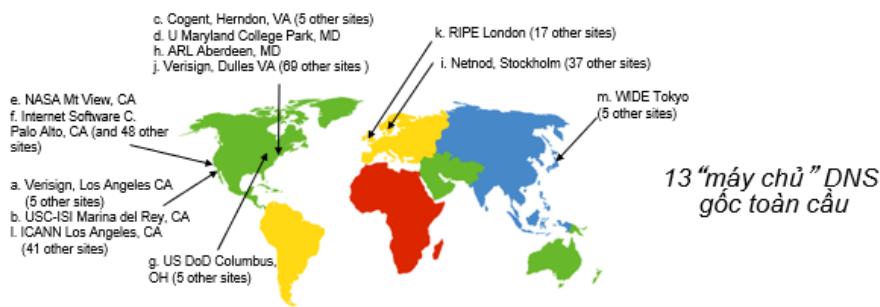


Máy khách muốn địa chỉ IP của www.amazon.com:

- Máy khách truy vấn máy chủ gốc (root) để tìm DNS máy chủ quản lý vùng “.com”
- Máy khách truy vấn máy chủ DNS “.com” tìm DNS máy chủ quản lý vùng “amazon.com”
- Máy khách truy vấn máy chủ DNS “amazon.com” để lấy địa chỉ IP của www.amazon.com

DNS: các máy chủ tên miền gốc

- Được máy chủ tên miền cục bộ liên lạc để hỏi khi không thể phân giải tên miền
- Máy chủ tên miền gốc:
 - Liên lạc với máy chủ tên miền có thẩm quyền (authoritative DNS server) nếu ánh xạ tên không xác định
 - Lấy ánh xạ
 - Trả về ánh xạ đến máy chủ tên miền cục bộ



TLD, máy chủ có thẩm quyền

Các máy chủ miền cấp cao nhất (top-level domain (TLD) servers):

- Chịu trách nhiệm cho tên miền com, org, net, edu, aero, jobs, museums, và tất cả các tên miền cấp cao nhất của quốc gia, như là: uk, fr, ca, jp
- Công ty Network Solutions quản lý máy chủ chứa các thông tin của vùng .com TLD
- Tổ chức Educause quản lý .edu TLD

Các DNS máy chủ có thẩm quyền:

- Các tổ chức sở hữu các DNS máy chủ riêng nhằm cung cấp các tên được cấp phép và ánh xạ địa chỉ IP cho các hệ thống đầu cuối được đặt tên của tổ chức đó
- Có thể được quản lý bởi tổ chức hoặc nhà cung cấp dịch vụ

Máy chủ tên miền cục bộ

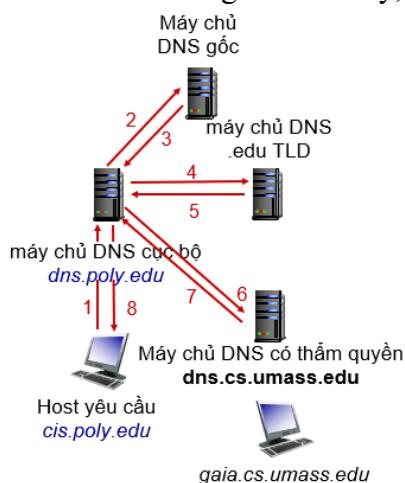
- Không hoàn toàn theo cấu trúc phân cấp
- Mỗi ISP (ISP, công ty, trường đại học) có một máy chủ cục bộ như vậy
 - Còn được gọi là “default name server”
- Khi một hệ thống đầu cuối tạo một truy vấn DNS, truy vấn được gửi đến DNS máy chủ cục bộ của nó
 - Có bộ nhớ đệm cục bộ (local cache) chứa thông tin về các cặp tên-địa chỉ gần đây (nhưng có thể hết hạn!)
 - Hoạt động như một proxy, chuyển truy vấn vào trong hệ thống DNS phân cấp

Ví dụ phân giải tên miền DNS

- Hệ thống đầu cuối tại cis.poly.edu muốn biết địa chỉ IP của gaia.cs.umass.edu

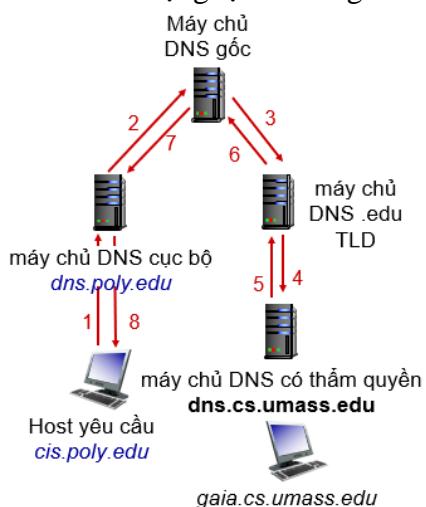
Truy vấn tuần tự:

- Máy chủ được hỏi sẽ trả lời với tên của máy chủ quản lý vùng liên quan
- “tôi không biết tên này, nhưng hãy hỏi thêm thông tin từ máy chủ này”



Truy vấn đệ quy:

- Đây trách nhiệm phân giải tên cho máy chủ tên miền được hỏi
- Tải nặng tại các tầng trên của hệ thống phân cấp?



DNS: caching, cập nhật các bản ghi

- Một khi máy chủ tên miền biết về 1 ánh xạ tên-địa chỉ, nó sẽ *lưu tạm* ánh xạ đó
 - Các mục cache hết hạn (sẽ bị xóa) sau một khoảng thời gian (TTL)
 - Thông tin trong các máy chủ TLD thường được lưu tạm trong các máy chủ tên miền cục bộ
 - Do đó các máy chủ tên gốc không bị truy cập thường xuyên
- Các mục được lưu tạm có thể hết hạn
 - Nếu cập nhật thông tin tên-địa chỉ IP thay đổi, có thể các máy khác trên Internet không biết được cho đến khi tất cả TTL hết hạn.

- Cơ chế cập nhật/thông báo được đề xuất trong bộ chuẩn IETF
 - RFC 2136

Các bản ghi DNS

DNS: cơ sở dữ liệu phân tán lưu trữ các bản ghi thông tin (resource records - RR)

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

type=A

- **name** là tên host
- **value** là địa chỉ IP

type=NS

- **Name** là tên miền (e.g., foo.com)
- **value** là tên của máy chủ tên miền có thẩm quyền quản lý tên miền này

type=CNAME

- **name** là bí danh của một tên “gốc” (tên thực)
- VD: www.ibm.com tên thực là **máy chủ east.backup2.ibm.com**
- **value** là tên gốc

type=MX

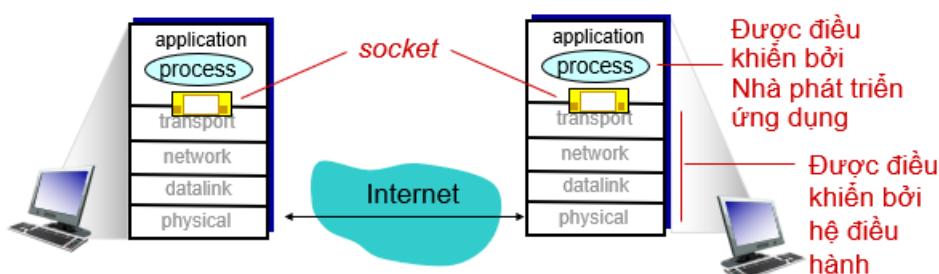
- **value** là tên của máy chủ thư điện tử được liên kết với **name**

7 .Lập trình Socket (Slide 96-105): Phân biệt được đâu là chương trình viết cho server/client, TCP/UDP)

Lập trình Socket

Mục tiêu: tìm hiểu cách xây dựng các ứng dụng máy khách/máy chủ liên lạc bằng sockets

socket: một cánh cửa giữa tiến trình ứng dụng và giao thức vận chuyển giữa 2 đầu cuối



Hai loại socket cho hai dịch vụ transport:

- **UDP:** chuyên gói tin không đảm bảo
- **TCP:** chuyên luồng tin có đảm bảo (stream-oriented)

Ứng dụng ví dụ:

1. Máy khách nhận một dòng các ký tự (dữ liệu) từ bàn phím của nó và gửi dữ liệu đó đến máy chủ.
2. Máy chủ nhận được dữ liệu đó và chuyển đổi các ký tự sang chữ hoa.
3. Máy chủ gửi dữ liệu đã được sửa đổi cho máy khách ở trên.
4. Máy khách nhận được dữ liệu đã bị sửa đổi và hiển thị dòng đó lên màn hình của nó.

Lập trình Socket với **UDP**

UDP: không “kết nối” giữa máy khách và máy chủ

- Không bắt tay trước khi gửi dữ liệu
- Bên gửi chỉ rõ địa chỉ IP đích và số cổng cho mỗi gói (packet)
- Bên nhận lấy địa chỉ IP và số cổng của bên gửi từ gói nhận được

UDP: dữ liệu được truyền có thể bị mất hoặc được nhận không đúng thứ tự

Quan điểm ứng dụng:

- UDP cung cấp cơ chế truyền các nhóm bytes (“datagrams”) không tin cậy giữa máy khách và máy chủ

Sự tương tác socket máy khách/máy chủ: UDP



Ứng dụng ví dụ: UDP máy khách

Python UDP client

```

Bao gồm thư viện socket  
của Python → from socket import *
serverName = 'hostname'
serverPort= 12000
clientSocket = socket(socket.AF_INET,
                      socket.SOCK_DGRAM)
message = raw_input('Input lowercase sentence.:')
clientSocket.sendto(message,(serverName,
                             serverPort))
modifiedMessage, serverAddress =
clientSocket.recvfrom(2048)
print modifiedMessage
clientSocket.close()

```

Tạo socket UDP
cho máy chủ →

Nhận thông điệp từ bàn phím
người dùng →

Đính kèm tên máy
chủ, cổng đến thông
điệp; gửi vào socket →

Đọc các ký tự trả lời
từ socket vào chuỗi →

In ra chuỗi được
nhận và đóng socket →

Ứng dụng ví dụ : UDP server

Python UDP server

```

from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("", serverPort))
print "The server is ready to receive"
while 1:
    message, clientAddress =
    serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage,
    clientAddress)

```

Lập trình Socket với *TCP*

Máy khách phải liên lạc với máy chủ

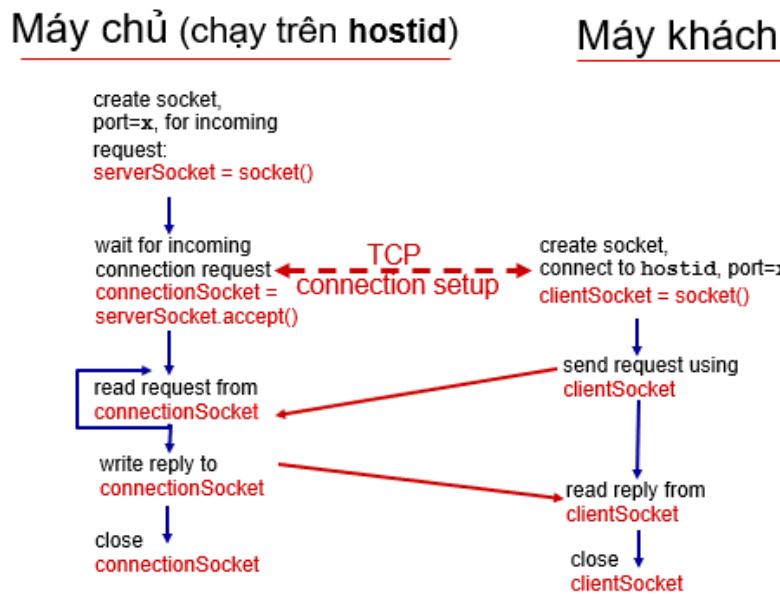
- Tiền trình máy chủ phải được chạy trước
- Máy chủ phải tạo socket để mời máy khách đến liên lạc

Máy khách tiếp xúc máy chủ bằng:

- Tạo socket TCP, xác định địa chỉ IP, số cổng của tiền trình máy chủ
- **Khi máy khách tạo socket:** máy khách TCP thiết lập kết nối đến máy chủ TCP
- Khi được máy khách liên lạc, **máy chủ TCP tạo socket mới** cho tiền trình máy chủ để trao đổi với máy khách đó
 - Cho phép máy chủ nói chuyện với nhiều máy khách
 - Số cổng nguồn được dùng để phân biệt các máy khách (xem tiếp chương 3)

Quan điểm ứng dụng: TCP cung cấp việc truyền các byte đáng tin cậy và theo thứ tự giữa máy khách và máy chủ

Tương tác socket máy khách/máy chủ: TCP



Ứng dụng ví dụ : TCP client

Python TCP client

```

from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence.')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()

```

Tạo TCP socket cho máy chủ, cảng 12000

Không cần định kèm tên máy chủ, cảng

Ví dụ ứng dụng: TCP máy chủ

Python TCPmáy chủ

```

from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(("",serverPort))
serverSocket.listen(1)
print 'The server is ready to receive'
while 1:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
connectionSocket.close()

```

Tạo socket TCP
chào đón → Máy chủ bắt đầu lắng nghe các yêu cầu TCP đến
Lắp mái mái → Máy chủ đợi accept() chờ yêu cầu đến, socket mới được tạo trả về
Đọc các byte từ socket (nhưng không đọc địa chỉ như UDP)
Đóng kết nối đến máy khách này(nhưng không đóng socket chào đón)

Chương 3.

(20% - 8 câu)

1. Multiplexing và Demultiplexing (Slide 8): Xem ảnh máy cái port và biết nó đi đâu về đâu

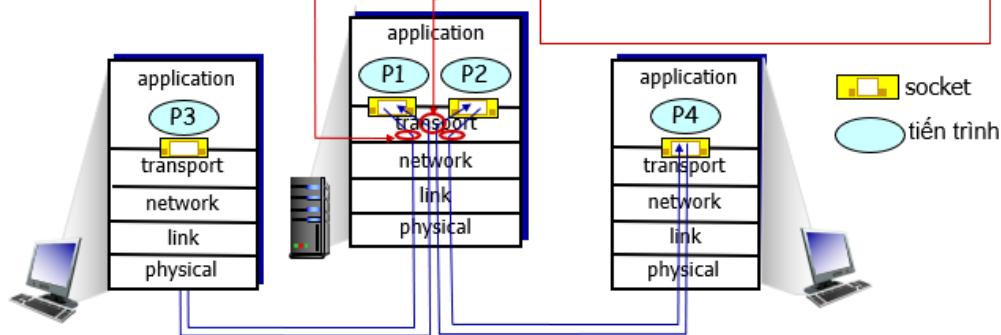
Multiplexing/demultiplexing

Multiplexing tại bên gửi:

xử lý dữ liệu từ nhiều socket, thêm thông tin header về tầng Vận chuyển vào segment (được sử dụng sau cho demultiplexing)

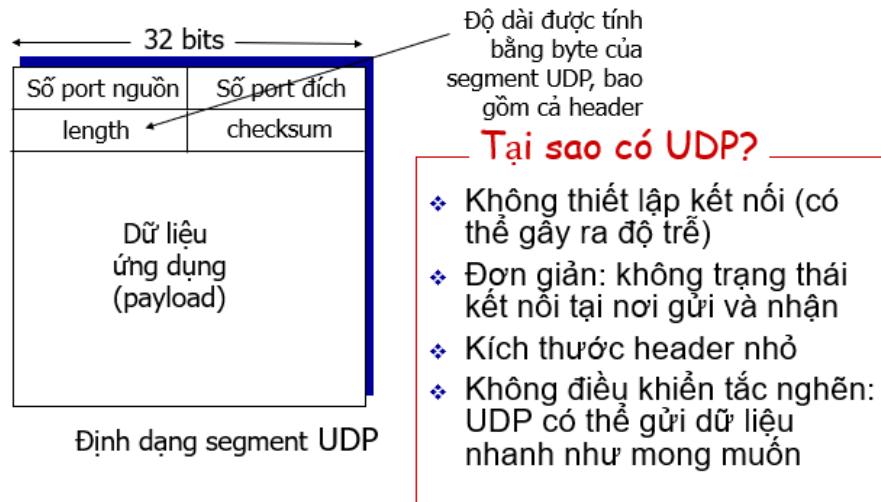
demultiplexing tại bên nhận:

sử dụng thông tin trong header để chuyển segment vừa nhận vào đúng socket



2. UDP header và tính checksum (Slide 17-19)

UDP: segment header



UDP checksum

Mục tiêu: dò tìm “các lỗi” (các bit cờ được bật) trong các segment đã được truyền

Bên gửi:

- Xét nội dung của segment, bao gồm các trường của header, là chuỗi các số nguyên 16-bit
- checksum: tổng bù 1 của các chuỗi số 16 bit trong nội dung segment
- Bên gửi đặt giá trị checksum vào trường checksum UDP

Bên nhận:

- Tính toán checksum của segment đã nhận
- Kiểm tra giá trị trên có bằng với giá trị trong trường checksum hay không:
 - NO – có lỗi xảy ra
 - YES – không có lỗi. Nhưng có thể còn lỗi khác nữa không? Xem phần sau....

Internet checksum: ví dụ

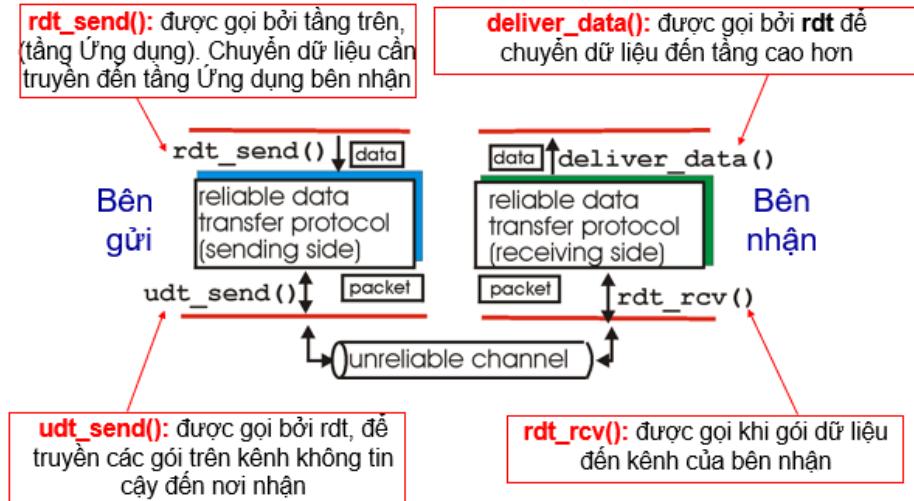
Ví dụ: cộng 2 số nguyên 16 bit

$$\begin{array}{r}
 \begin{array}{r}
 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
 \end{array} \\
 \hline
 \text{bit dư} \quad \boxed{1} & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
 \hline
 \text{tổng} & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
 \text{checksum} & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
 \end{array}$$

Lưu ý: khi cộng các số, bit nháy ở phía cao nhất cần được thêm vào kết quả

3. Các nguyên lý truyền dữ liệu tin cậy : Có khoảng 1-2 câu trong đề thi. Phân biệt được giống và khác nhau giữa các version. Máy rtd. VD nhìn vô cái hình biết version máy. Tính toán hiệu suất k cần quan tâm. (Slide 24-43, 50, 54). Bài tập bao nhiêu trạng thái, ...

Truyền dữ liệu tin cậy: bắt đầu



Truyền dữ liệu tin cậy: bắt đầu

Chúng ta sẽ:

- Từng bước phát triển truyền dữ liệu tin cậy (rdt) bên phía người gửi và nhận
- Chỉ xem xét chuyển dữ liệu theo 1 hướng
 - Nhưng điều khiển thông tin sẽ theo cả 2 hướng!
- Sử dụng finite state machines (FSM) để xác định bên gửi và nhận

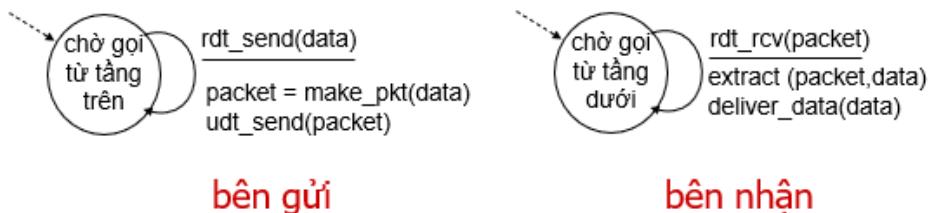


rdt1.0: truyền tin cậy trên 1 kênh tin cậy

- Kênh cơ bản tin cậy hoàn toàn (underlying channel perfectly reliable)
 - không có bit lỗi
 - không mất mát gói
- Các FSMs riêng biệt cho bên gửi và nhận:

- Bên gửi gửi dữ liệu vào kênh cơ bản (underlying channel)

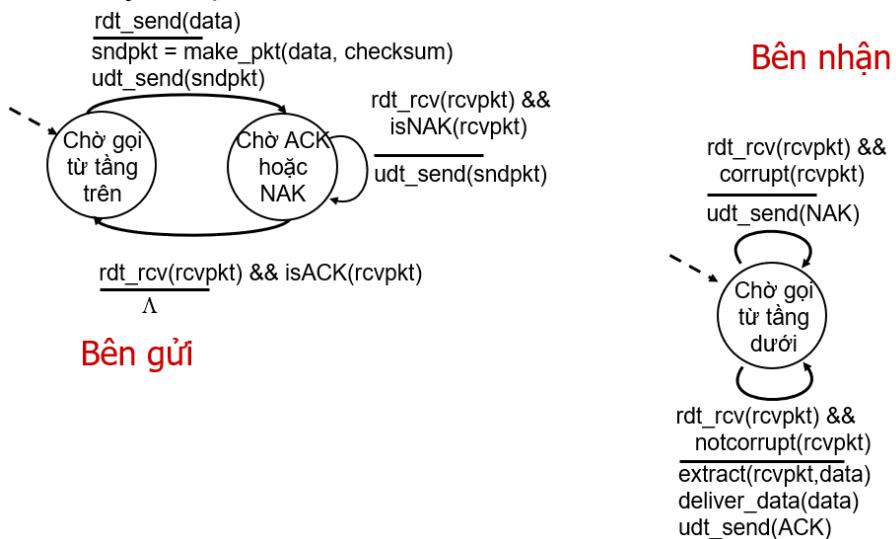
Bên nhận đọc dữ liệu từ kênh cơ bản (underlying channel)



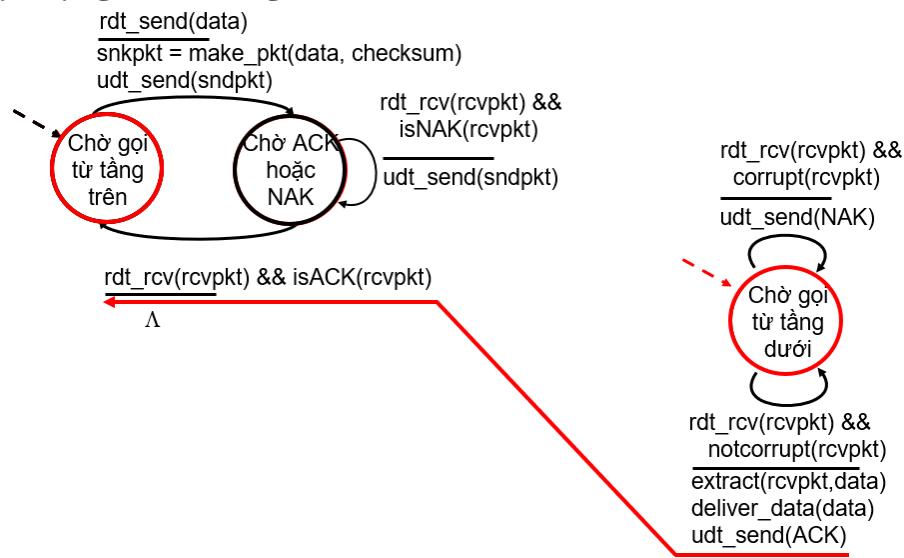
rdt2.0: kênh với các lỗi

- Kênh cơ bản có thể đảo các bit trong packet
 - checksum để kiểm tra các lỗi
- Câu hỏi: làm sao khôi phục các lỗi:
 - *acknowledgements (ACKs)*: bên nhận thông báo cho bên gửi rằng packet được nhận thành công (OK)
 - *negative acknowledgements (NAKs)*: bên nhận thông báo cho bên gửi rằng packet đã bị lỗi
 - Bên gửi truyền lại gói nào được xác nhận là NAK
- Các cơ chế mới trong rdt2.0 (sau rdt1.0):
 - Phát hiện lỗi
 - Phản hồi: các thông điệp điều khiển (ACK, NAK) từ bên nhận đến bên gửi

rdt2.0: đặc điểm kỹ thuật FSM



rdt2.0: hoạt động khi không lỗi



rdt2.0 có lỗi hỏng nghiêm trọng!

Điều gì xảy ra nếu ACK/NAK bị hỏng?

- ❖ Bên gửi sẽ không biết điều gì đã xảy ra ở bên nhận!
- ❖ Không thể đơn phương truyền lại: có thể trùng lặp

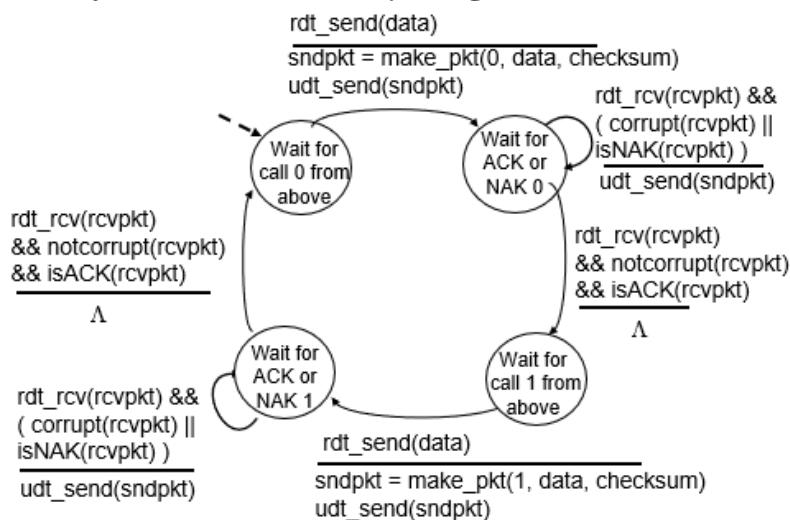
Xử lý trùng lặp:

- ❖ Bên gửi truyền lại packet hiện thời nếu ACK/NAK bị hỏng
- ❖ Bên gửi thêm **số thứ tự** vào trong mỗi packet (**sequence number**)
- ❖ Bên nhận hủy packet bị trùng lặp

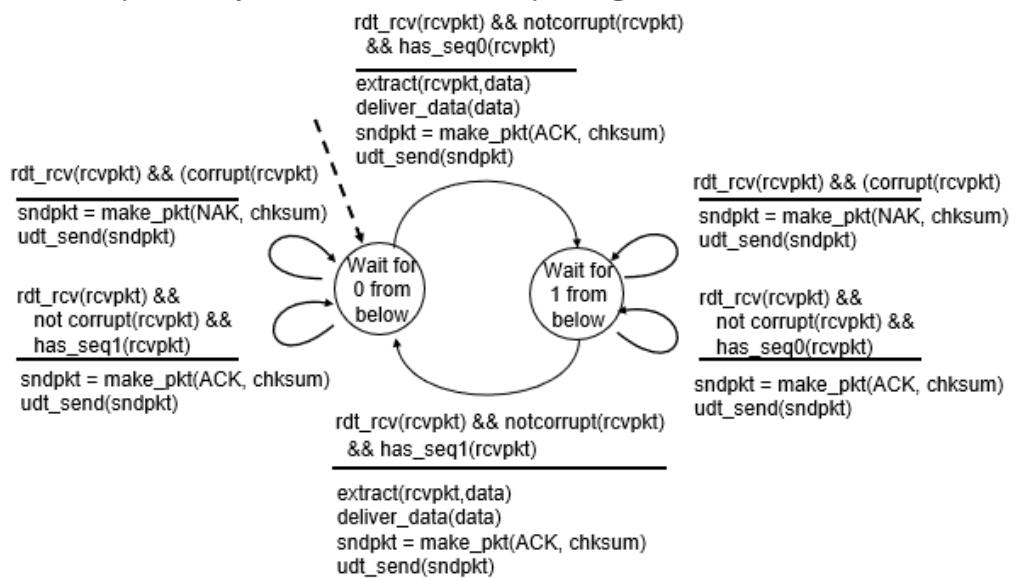
Stop and wait

Bên gửi gửi một packet, sau đó chờ phản hồi từ bên nhận

rdt2.1: bên gửi, xử lý các ACK/NAK bị hỏng



rdt2.1: bên nhận, xử lý các ACK/NAK bị hỏng



rdt2.1: thảo luận

Bên gửi:

- Số thứ tự (seq #) được thêm vào packet
- 2 số thứ tự (0,1) là đủ. Tại sao?
- Phải kiểm tra có hay không ACK/NAK vừa nhận bị hỏng
- Số trạng thái tăng lên 2 lần
 - Trạng thái phải “nhớ” xem packet “mong đợi” có số thứ tự là 0 hay 1

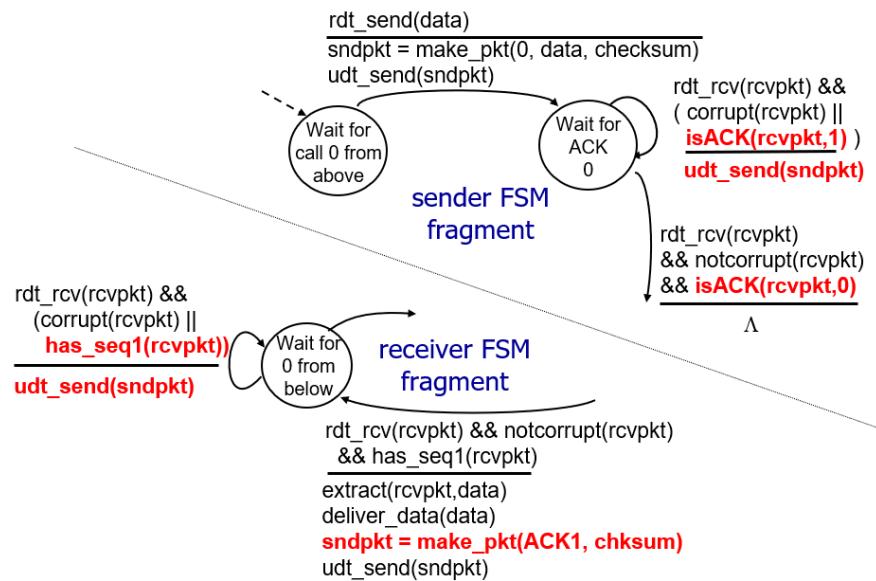
Bên nhận:

- Phải kiểm tra gói vừa nhận có trùng hay không
 - Trạng thái chỉ rõ có hay không 0 hoặc 1 là số thứ tự của gói được mong chờ
- Chú ý: bên nhận có thể không biết ACK/NAK vừa rồi có được bên gửi nhận tốt hay không

rdt2.2: một giao thức không cần NAK

- Chức năng giống như rdt2.1, chỉ dùng các ACK
- Thay cho NAK, bên nhận gửi ACK cho gói cuối cùng được nhận thành công
 - Bên nhận phải ghi rõ số thứ tự của gói vừa được ACK
- ACK bị trùng tại bên gửi dẫn tới kết quả giống như hành động của NAK: *truyền lại gói vừa rồi*

rdt2.2: các phần bên nhận và gửi



rdt3.0: các kênh với lỗi và mất mát

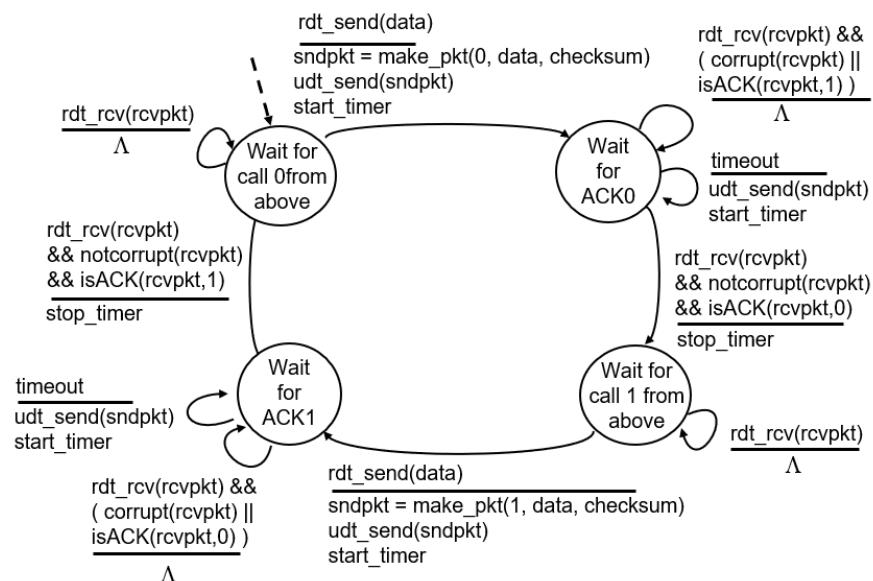
Giả định mới: kênh truyền cũng có thể làm mất gói (dữ liệu, các ACK)

- checksum, số thứ tự, các ACK, việc truyền lại sẽ hỗ trợ... nhưng không đủ

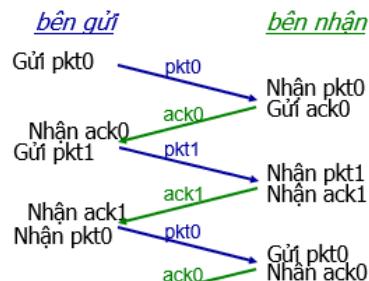
Cách tiếp cận: bên gửi chờ ACK trong khoảng thời gian “hợp lý”

- Truyền lại nếu không nhận được ACK trong khoảng thời gian này
- Nếu gói (hoặc ACK) chỉ trễ (không mất):
 - Việc truyền lại sẽ gây trùng, nhưng số thứ tự đã xử lý trường hợp này
 - Bên nhận phải xác định số thứ tự của gói vừa gửi ACK
- Yêu cầu bộ định thì đếm lùi

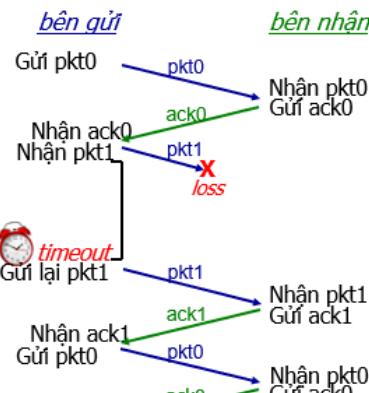
rdt3.0 bên gửi



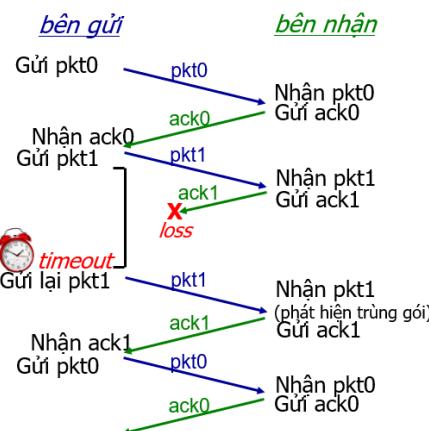
Hành động của rdt3.0



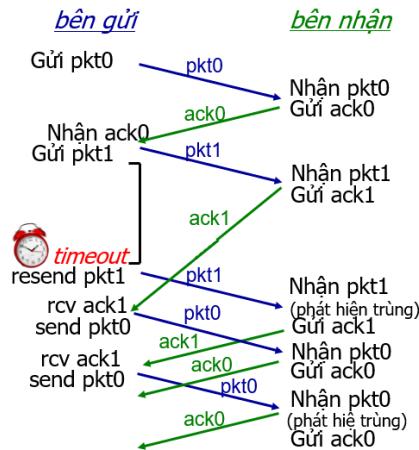
(a) Không mất mát



(b) Mất gói



(c) Mất ACK



(d) Thời gian chờ quá ngắn / delayed ACK

Hiệu suất của rdt3.0

- rdt3.0 làm việc được, nhưng đánh giá hiệu suất hơi rắc rối
- Ví dụ: đường link 1 Gbps, trẽ lan truyền giữa 2 đầu cuối là 15 ms, gói 8000 bit:

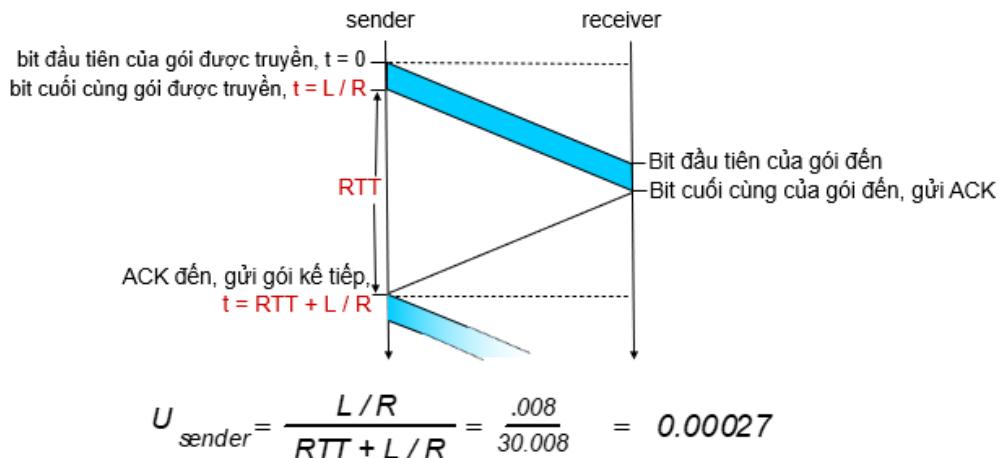
$$D_{truyền} = \frac{L}{R} = \frac{8000 \text{ bits}}{10^9 \text{ bits/sec}} = 8 \text{ microsecs}$$

- U_{sender}: utilization – khoảng thời gian mà bên gửi gửi được dữ liệu

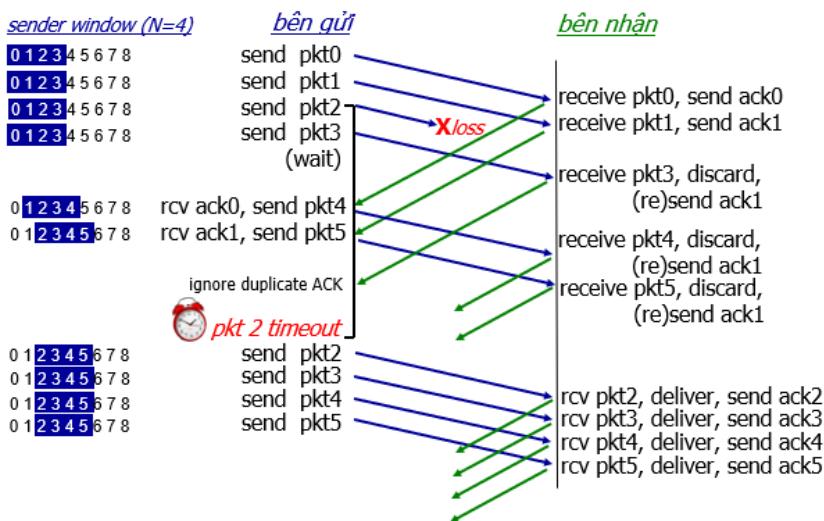
$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{.008}{30.008} = 0.00027$$

- Nếu RTT=30 msec, gói 1KB mỗi 30 msec: thông lượng 33kB/sec trên đường link 1Gbps
- Giao thức mạng hạn chế việc sử dụng các tài nguyên vật lý!

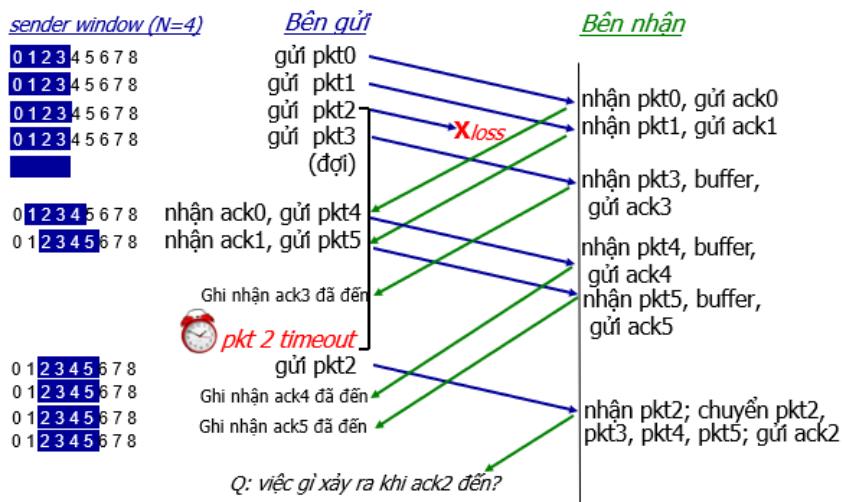
rdt3.0: hoạt động “stop-and-wait”



Hoạt động GBN

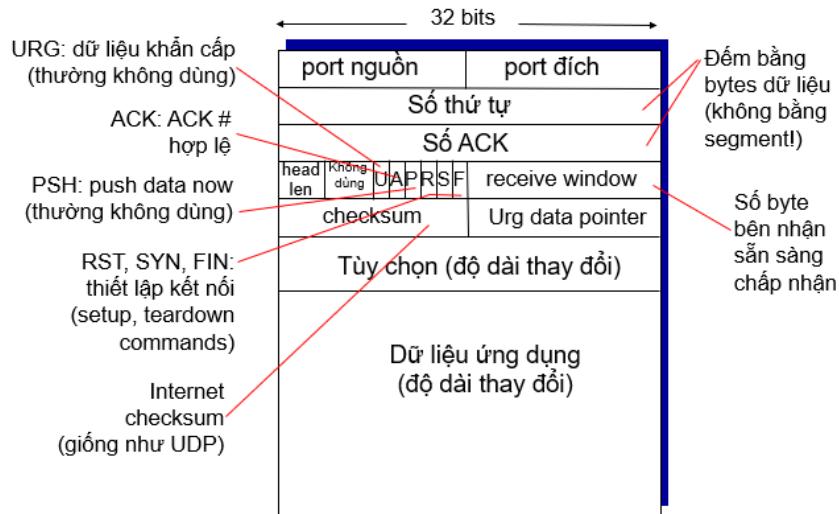


Hành động của lặp lại có lựa chọn



4. TCP header (Slide 58-60)

Cấu trúc segment TCP



Số thứ tự TCP và ACK

Các số thứ tự:

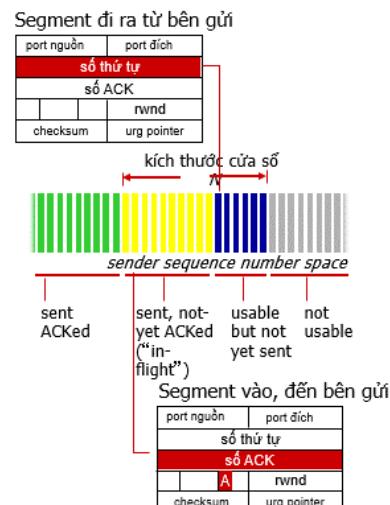
- Dòng byte “đánh số” byte đầu tiên trong dữ liệu của segment

Các ACK:

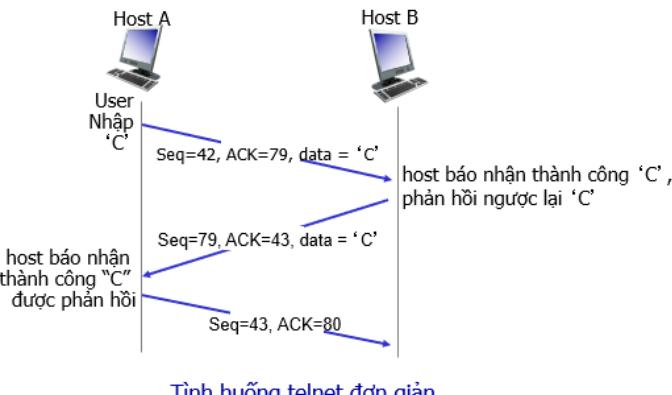
- số thứ tự của byte kế tiếp được mong đợi từ phía bên kia
- ACK tích lũy

Hỏi: làm thế nào để bên nhận xử lý các segment không theo thứ tự

- Trả lời: TCP không đề cập, tùy thuộc người thực hiện



Số thứ tự TCP và ACK



5. Tính toán RTT và Time out. Biết cách tính với công thức (Slide 61-63)

TCP round trip time và timeout

Hỏi: làm cách nào để thiết lập giá trị TCP timeout?

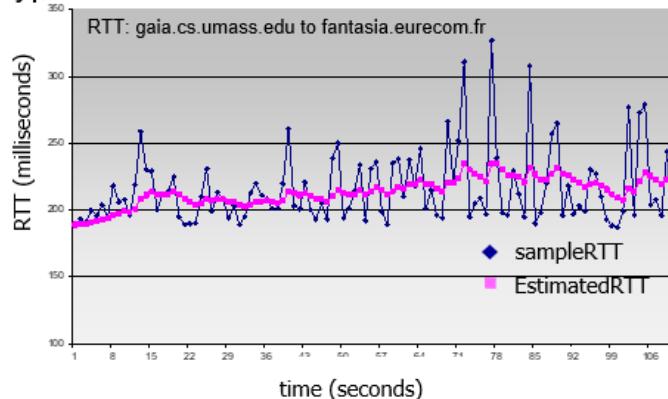
- ❖ Dài hơn RTT
 - Nhưng RTT thay đổi
- ❖ Quá ngắn: timeout sớm, truyền lại không cần thiết
- ❖ Quá dài: phản ứng chậm đối với việc mất mát gói

Q: làm cách nào để ước lượng RTT?

- ❖ **SampleRTT:** thời gian được đo từ khi truyền segment đến khi báo nhận ACK
 - Lần đi việc truyền lại
- ❖ **SampleRTT** sẽ thay đổi, muốn RTT được ước lượng “mượt hơn”
 - Đo lường trung bình của một số giá trị vừa xảy ra, không chỉ **SampleRTT** hiện tại

$$\text{EstimatedRTT} = (1 - \alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$$

- ❖ Đường trung bình dịch chuyển hàm mũ (exponential weighted moving average)
- ❖ ảnh hưởng của mẫu đã xảy ra sẽ làm giảm tốc độ theo cấp số nhân
- ❖ typical value: $\alpha = 0.125$



❖ Khoảng thời gian timeout (timeout interval):
EstimatedRTT cộng với “biên an toàn”

- Sự thay đổi lớn trong **EstimatedRTT** → an toàn biên lớn hơn
- ❖ Ước lượng độ lệch SampleRTT từ EstimatedRTT:

$$\text{DevRTT} = (1 - \beta) * \text{DevRTT} + \beta * |\text{SampleRTT} - \text{EstimatedRTT}|$$

(typically, $\beta = 0.25$)

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$$

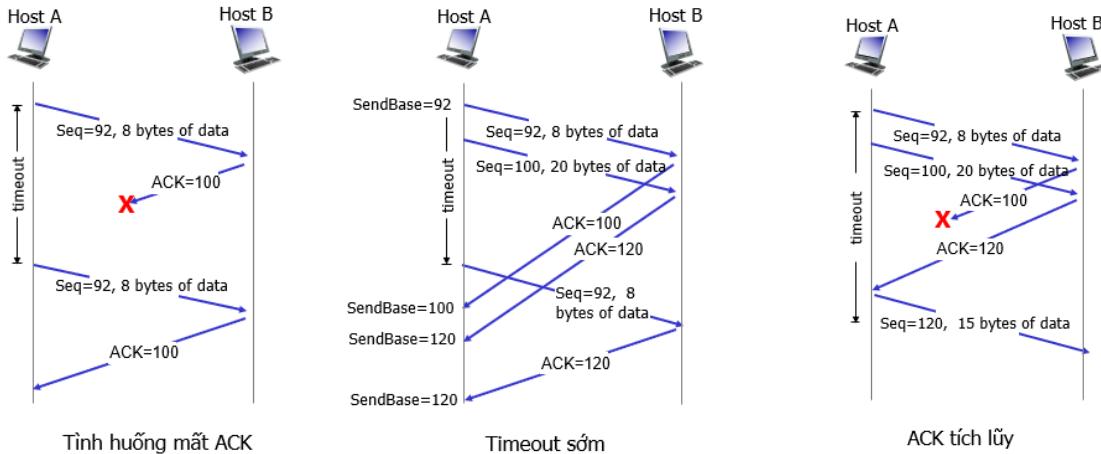


↑
estimated RTT

↑
“biên an toàn”

6. Tình huống truyền lại (Slide 68-69)

TCP: tình huống truyền lại



7. TCP truyền lại nhanh (Slide 71-72)

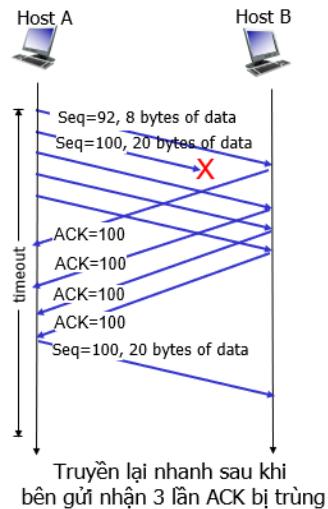
TCP truyền lại nhanh

- ❖ Chu kỳ time-out thường tương đối dài:
 - Độ trễ dài trước khi gửi lại gói bị mất
- ❖ Phát hiện các segment bị mất thông qua các ACKs trùng.
 - Bên gửi thường gửi nhiều segment song song
 - Nếu segment bị mất, thì sẽ có khả năng có nhiều ACK trùng.

TCP truyền lại nhanh

Nếu bên gửi nhận 3 ACK của cùng 1 dữ liệu ("3 ACK trùng"), thì gửi lại segment chưa được ACK với số thứ tự nhỏ nhất

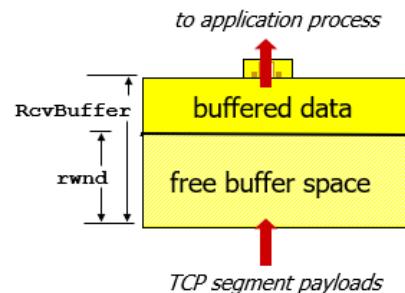
- Có khả năng segment không được ACK đã bị mất, vì thế không đợi đến thời gian timeout



8. TCP điều khiển luồng (Slide 75)

TCP điều khiển luồng

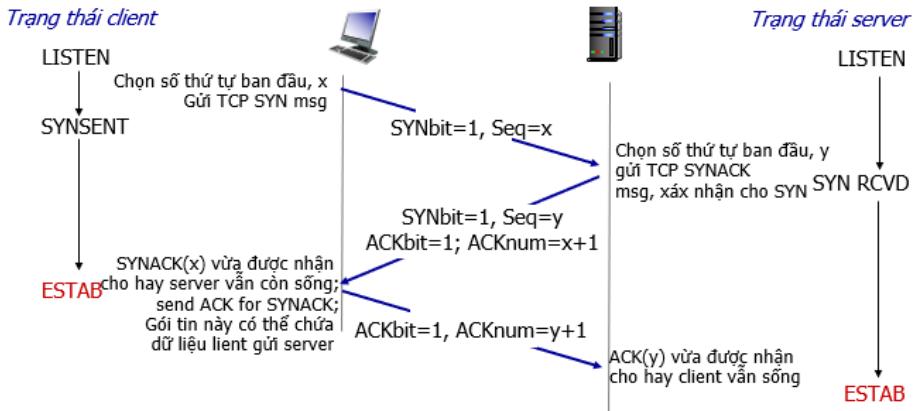
- Bên nhận "thông báo" không gian bộ nhớ đệm còn trống bằng cách thêm giá trị **rwnd** trong TCP header của các segment từ bên nhận đến bên gửi
 - Kích thước của **RcvBuffer** được thiết đặt thông qua các tùy chọn của socket (thông thường mặc định là 4096 byte)
 - Nhiều hệ điều hành tự động điều chỉnh **RcvBuffer**
- Bên gửi giới hạn khối lượng dữ liệu gửi mà không cần ACK bằng giá trị **rwnd** của bên nhận
- Bảo đảm bộ đệm bên nhận sẽ không bị tràn



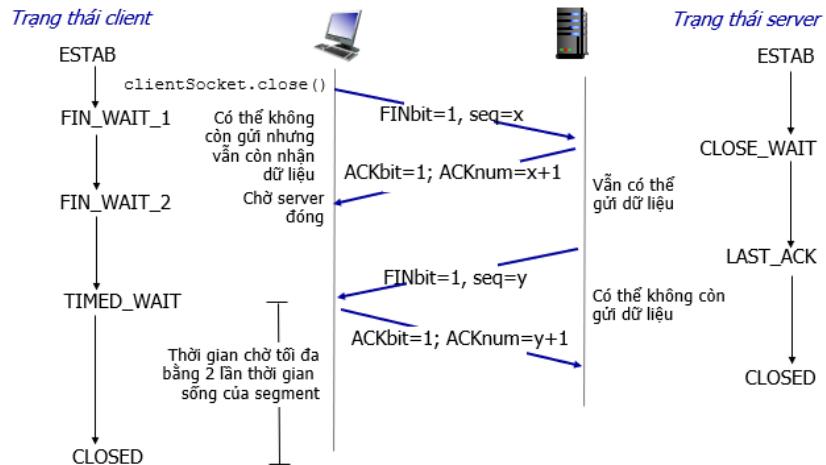
Bộ đệm phía bên nhận

9. Quản lý kết nối (Slide 80, 83)

TCP bắt tay 3 lần (3-way handshake)



TCP: đóng kết nối

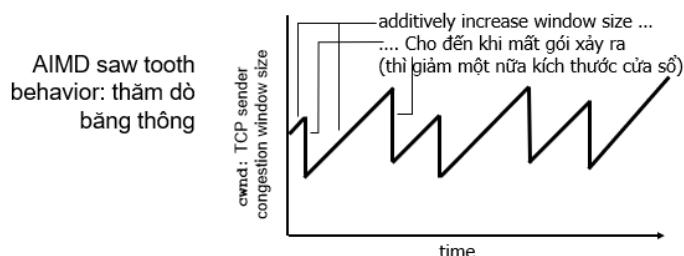


10. Điều khiển tắc nghẽn (Slide 99-105)

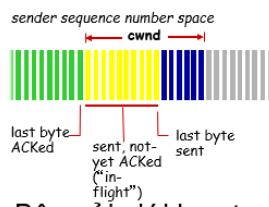
TCP điều khiển tắc nghẽn: tăng theo cấp số cộng, giảm theo cấp số nhân

Hướng tiếp cận: bên gửi tăng tốc độ truyền (kích thước cửa sổ), thăm dò băng thông có thể sử dụng, cho đến khi mất mát gói xảy ra

- **tăng theo cấp số cộng (additive increase):** tăng cwnd (congestion window) lên 1 MSS sau mỗi RTT cho đến khi mất gói xảy ra
- **giảm theo cấp số nhân (multiplicative decrease):** giảm một nửa cwnd sau khi mất gói xảy ra



TCP điều khiển tắc nghẽn: chi tiết



TCP tốc độ gửi:

- ❖ *Ước lượng:* khối lượng byte gửi (cwnd) đợi ACK trong khoảng thời gian RTT

- ❖ Bên gửi giới hạn truyền tải:

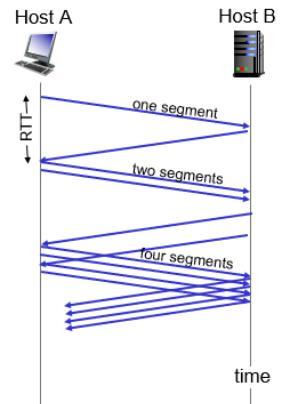
$$\text{LastByteSent} - \text{LastByteAcked} \leq \text{cwnd}$$

$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

- ❖ **cwnd** thay đổi, chức năng nhận biết tắc nghẽn trên mạng

TCP Slow Start

- Khi kết nối bắt đầu, tăng tốc độ theo cấp số nhân cho đến sự kiện mất gói đầu tiên xảy ra:
 - initially **cwnd** = 1 MSS
 - Gấp đôi **cwnd** mỗi RTT
 - Được thực hiện bằng cách tăng **cwnd** cho mỗi ACK nhận được
- **Tóm lại:** tốc độ ban đầu chậm, nhưng nó sẽ tăng lên theo cấp số nhân



TCP: phát hiện, phản ứng khi mất gói

- Mất gói được chỉ ra bởi timeout:
 - **cwnd** được thiết lập 1 MSS;
 - Sau đó kích thước cửa sổ sẽ tăng theo cấp số nhân (như trong slow start) đến ngưỡng, sau đó sẽ tăng tuyến tính
- Mất gói được xác định bởi 3 ACK trùng nhau: TCP RENO
 - Các ACK trùng lặp chỉ ra mạng vẫn có khả năng truyền
 - **cwnd** bị cắt một nửa sau đó tăng theo tuyến tính
- TCP TAHOE luôn luôn thiết lập **cwnd** bằng 1 (timeout hoặc 3 ack trùng nhau)

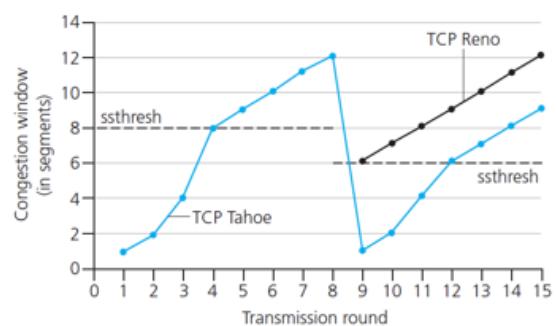
TCP: chuyển từ slow start qua CA

Hỏi: khi nào tăng cấp lũy thừa nên chuyển qua tuyến tính?

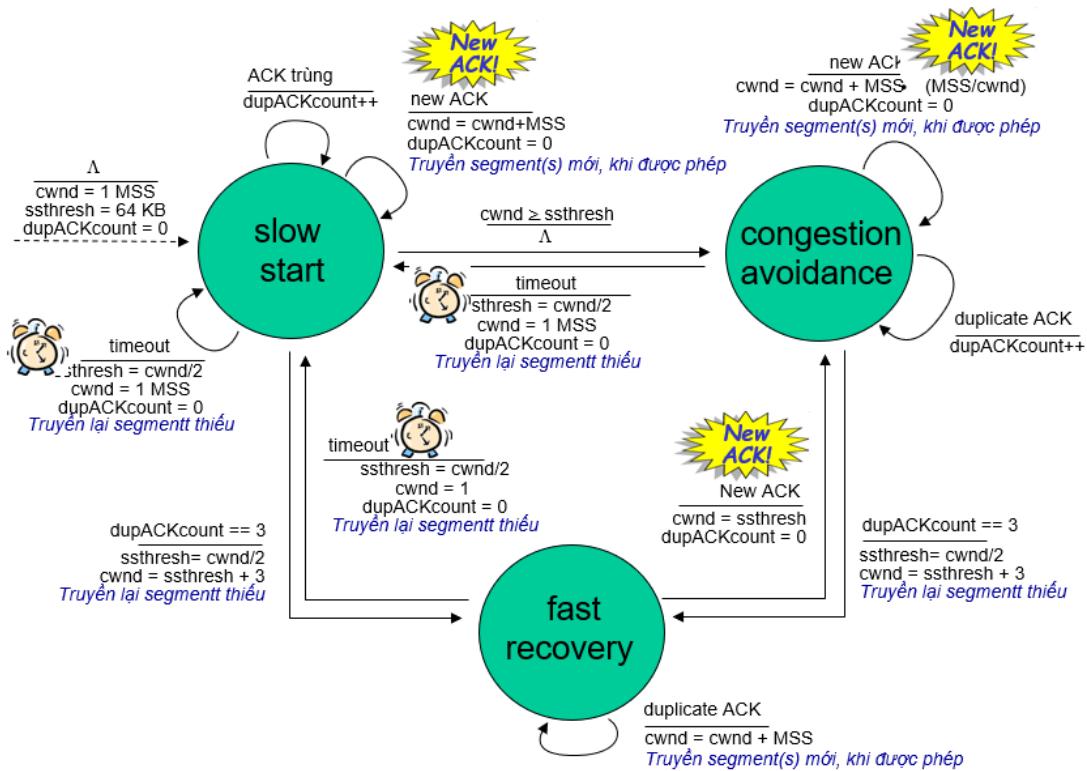
Trả lời: khi **cwnd** được $1/2$ giá trị của nó trước thời gian timeout.

Thực hiện:

- **ssthresh** thay đổi
- Khi mất gói, **ssthresh** được thiết lập về chỉ $1/2$ của **cwnd** trước khi mất gói

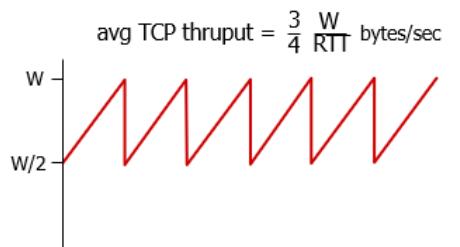


Tóm tắt: TCP điều khiển tắc nghẽn



TCP thông lượng (throughput)

- Thông lượng trung bình của TCP như là chức năng của kích thước cửa sổ và RTT?
 - Bỏ qua slow start, giả sử dữ liệu luôn luôn được gửi
- W: kích thước cửa sổ (được đo bằng byte) khi mất gói xảy ra
 - Kích thước cửa sổ trung bình (# in-flight bytes) là $\frac{3}{4} W$
 - Thông lượng trung bình là $\frac{3}{4}W$ mỗi RTT



Chương 4.

(30% - 12 câu)

1. Phân biệt được mạng mạch ảo và mạng mạch gói, mục 4.2 (Slide 11-16)

Dịch vụ connection-oriented (hướng kết nối) và connection-less (phi kết nối)

- Mạng *datagram* cung cấp dịch vụ *phi kết nối* tại tầng Mạng
- Mạng *mạch ảo (virtual-circuit network)* cung cấp dịch vụ *hướng kết nối* tại tầng Mạng
- Tương tự như các dịch vụ kết nối định hướng và không định hướng của tầng Vận chuyển, nhưng:
 - *Dịch vụ:* kết nối giữa 2 máy đầu cuối
 - *Không lựa chọn:* hệ thống mạng chỉ cung cấp 1 trong 2 dịch vụ
 - *Triển khai:* bên trong phần lõi của mạng

Các mạch ảo (Virtual circuits)

“các hoạt động trên đường đi từ nguồn tới đích tương tự như mạng điện thoại”

- Hiệu quả hoạt động tốt
- Các hoạt động của mạng dọc theo đường đi từ nguồn tới đích

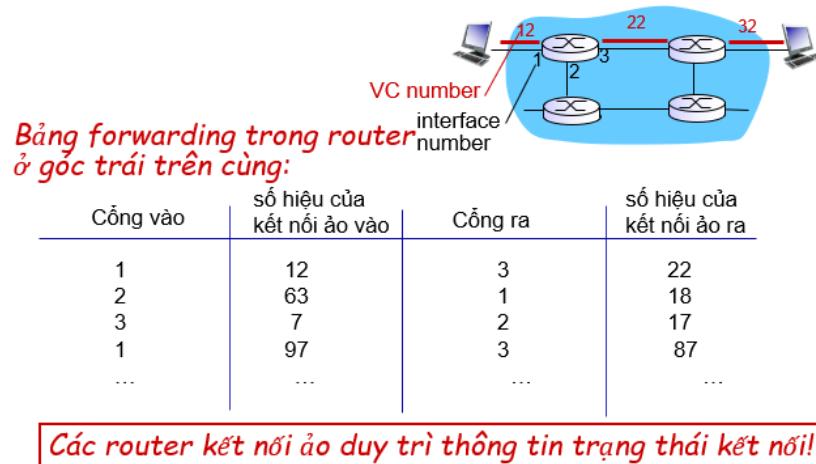
- Thiết lập cuộc gọi mỗi cuộc gọi trước khi dữ liệu có thể truyền. Ngắt kết nối sau khi kết thúc cuộc gọi.
- Mỗi packet mang số nhận dạng của kết nối ảo (VC identifier) (không phải là địa chỉ của host đích)
- Mỗi router trên đường đi từ nguồn tới đích duy trì trạng thái cho mỗi kết nối đi qua nó.
- Đường kết nối, các tài nguyên router (băng thông, bộ nhớ đệm) có thể được cấp phát cho từng kết nối ảo (các tài nguyên dành riêng => dịch vụ có thể dự đoán trước)

Triển khai kết nối ảo (VC)

Một kết nối ảo bao gồm:

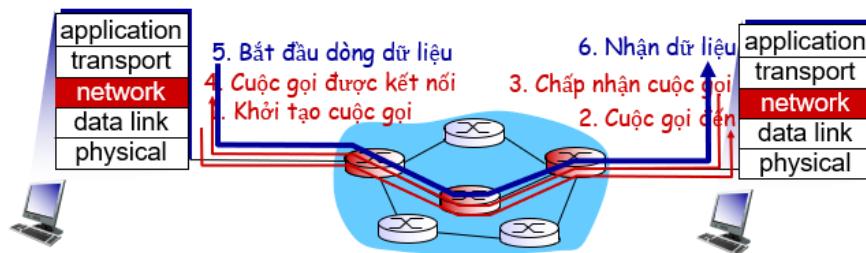
1. *Đường đi (path)* từ nguồn tới đích
 2. *Số hiệu nhận dạng kết nối ảo (VC numbers)*, mỗi số cho mỗi kết nối dọc theo đường đi
 3. *Các mục trong các bảng forwarding* ở trong các router dọc theo đường đi
- Gói thuộc về 1 kết nối ảo mang số nhận dạng của kết nối ảo đó (không dùng địa chỉ đích)
 - Số nhận dạng kết nối ảo có thể thay đổi trên mỗi đoạn kết nối
 - Số nhận dạng mới của kết nối ảo được cấp phát từ bảng forwarding

Bảng forwarding của kết nối ảo



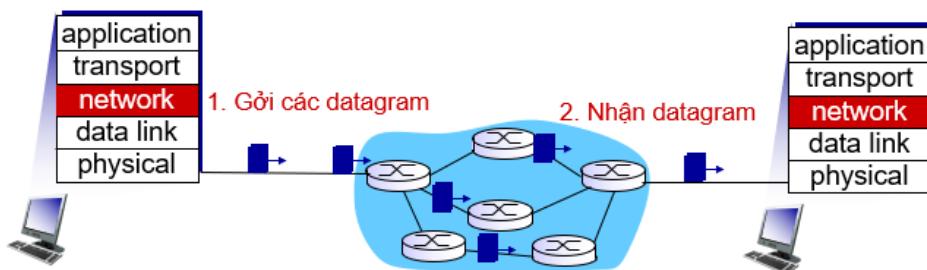
Các mạch ảo: các giao thức gửi tín hiệu

- Được dùng để thiết lập, duy trì kết nối ảo
- Được dùng trong ATM, frame-relay, X.25
- Không được sử dụng trong Internet ngày nay



Mạng chuyển gói (Datagram network)

- Không thiết lập cuộc gọi tại tầng Mạng
- Các router: không giữ trạng thái về các kết nối giữa 2 điểm cuối
 - Không có khái niệm “kết nối” ở tầng Mạng
- Vận chuyển các gói dùng địa chỉ máy đích



2. So trùng prefix dài nhất (Slide 19)

So trùng phần đầu dài nhất (Longest prefix matching)

So trùng phần đầu dài nhất

Khi tìm kiếm 1 công ra trong bảng forwarding cho địa chỉ đích, so phần đầu dài nhất trùng giữa địa chỉ trong bảng và địa chỉ đích.

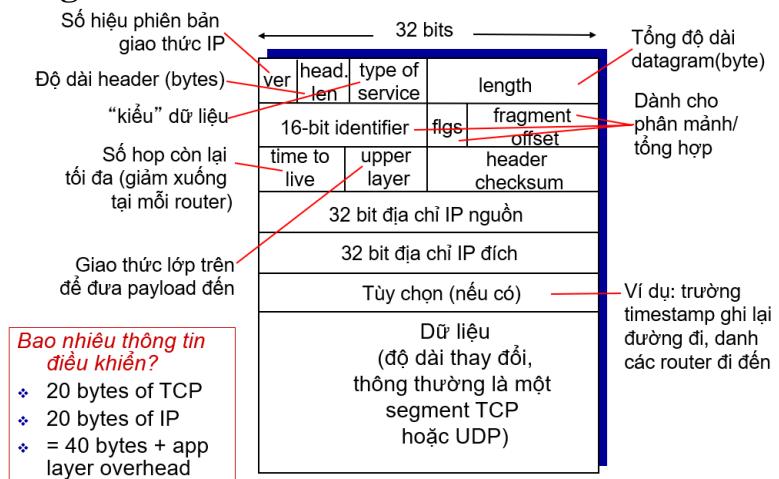
| Dãy địa chỉ đích | Link interface |
|---------------------------------------|----------------|
| 11001000 00010111 00010110 **** ***** | 0 |
| 11001000 00010111 00011000 **** ***** | 1 |
| 11001000 00010111 00011011 **** ***** | 2 |
| otherwise | 3 |

Ví dụ:

DA: 11001000 00010111 00010110 10100001 Interface nào?
DA: 11001000 00010111 00011000 10101010 Interface nào?

3. IP header (Slide 34)

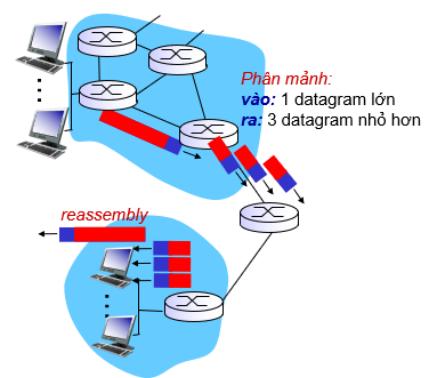
Định dạng IP datagram

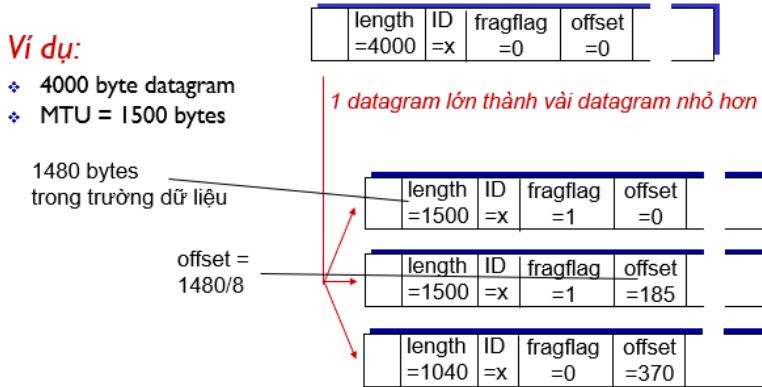


4. Phân mảnh và tổng hợp (Slide 35-36)

Phân mảnh và tổng hợp IP

- Các đoạn kết nối mạng có MTU (max.transfer size) – frame lớn nhất có thể truyền trên kết nối
 - Các kiểu kết nối khác nhau có các MTU khác nhau
- Các gói IP datagram lớn được chia (“fragmented”) bên trong mạng
 - 1 datagram thành 1 vài datagram
 - “tổng hợp” chỉ được thực hiện ở đích cuối cùng
 - Các bit của IP header được sử dụng để xác định, xếp thứ tự các fragment liên quan





5. Địa chỉ IP : Các lớp địa chỉ, địa chỉ dành riêng, địa chỉ mạng, địa chỉ host, địa chỉ broadcast, địa chỉ default gateway, chia mạng con.

Số câu hỏi cho phần này nhiều. Chiếm 6-8 câu.

6. Giao thức DHCP, cấp IP động (Slide 45-48)

DHCP: Dynamic Host Configuration Protocol

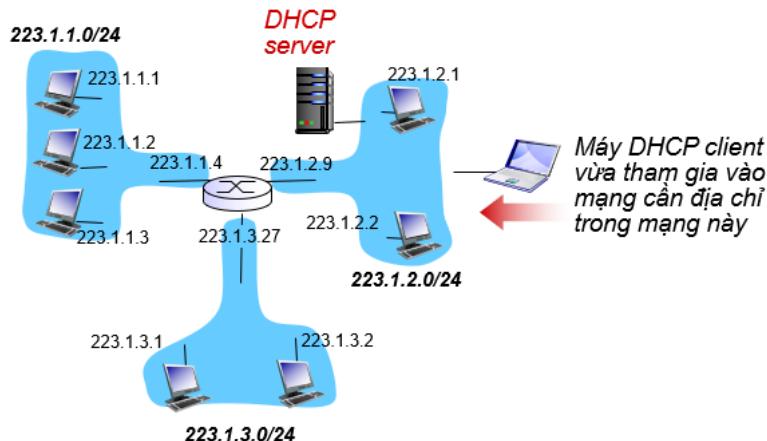
Mục tiêu: cho phép host (máy) tự động lấy địa chỉ IP của nó từ server trong mạng khi host đó tham gia vào mạng

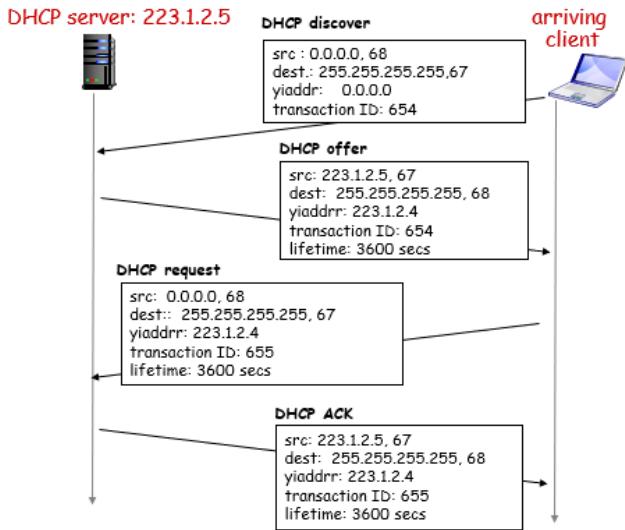
- Có thể giới hạn địa chỉ IP mà host đó vừa được cấp
- Cho phép tái sử dụng các địa chỉ IP (chỉ giữ địa chỉ trong khi được kết nối)
- Hỗ trợ cho người dùng di động muốn tham gia vào mạng (trong thời gian ngắn)

Tổng quan DHCP:

- Host gửi thông điệp “**DHCP discover**” [tùy chọn] cho tất cả các máy trong mạng (gửi broadcast)
- DHCP server nhận và trả lời bằng thông điệp “**DHCP offer**” [tùy chọn]
- Host yêu cầu địa chỉ IP: thông điệp “**DHCP request**”
- DHCP server gửi địa chỉ: thông điệp “**DHCP ack**”

Ngữ cảnh DHCP client-server





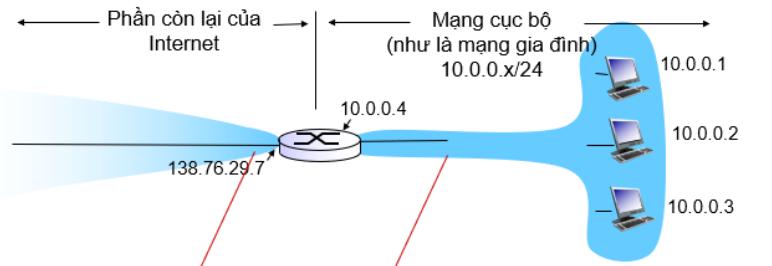
DHCP: cung cấp nhiều thông tin

DHCP không chỉ trả về địa chỉ IP được chỉ định trên subnet, mà nó còn có thể trả về nhiều thông tin như sau:

- Địa chỉ của router ở cửa ngõ kết nối ra ngoài mạng của client (default gateway)
- Tên và địa chỉ IP của DNS sever
- Network mask (cho biết phần của mạng và phần host của địa chỉ IP)

7. NAT (Slide 56-59)

NAT: network address translation



Tất cả datagram đi ra khỏi mạng cục bộ có cùng một địa chỉ IP NAT là: 138.76.29.7, với các số hiệu cổng nguồn khác nhau

Các datagram với nguồn hoặc đích trong mạng này có địa chỉ 10.0.0.x/24 cho nguồn, đích (như thông thường)

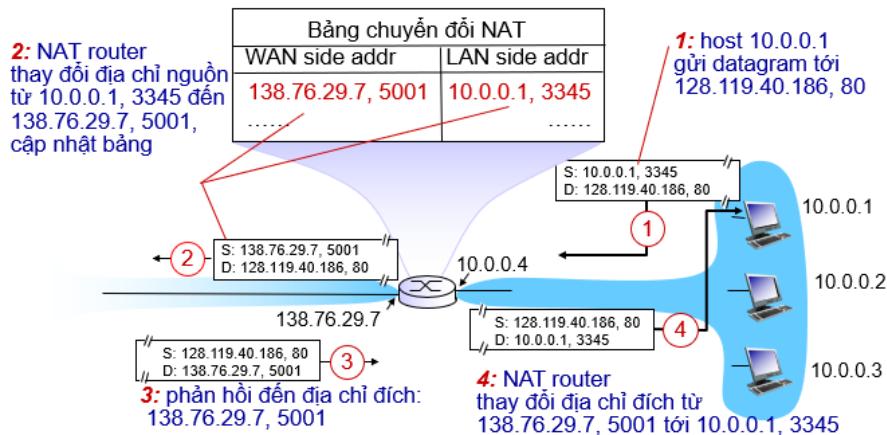
Động lực: mạng cục bộ chỉ dùng 1 địa chỉ IP đối với thế giới bên ngoài:

- Không cần thiết dùng 1 vùng địa chỉ từ ISP: chỉ cần 1 địa chỉ IP cho tất cả các thiết bị
- Có thể thay đổi các địa chỉ IP của các thiết bị trong mạng cục bộ mà không cần thông báo cho thế giới bên ngoài
- Có thể thay đổi ISP mà không cần thay đổi địa chỉ IP của các thiết bị trong mạng nội bộ

- Bên ngoài không nhìn thấy và không biết địa chỉ rõ ràng của các thiết bị bên trong mạng cục bộ (tăng cường bảo mật)

Triển khai: NAT router phải:

- Với các datagram đi ra: thay thế (địa chỉ IP nguồn, số hiệu cổng nguồn) của mọi datagram đi ra bên ngoài bằng (địa chỉ IP NAT, số hiệu port mới)
 - ... Các client/server ở xa sẽ dùng địa chỉ đó (địa chỉ IP NAT, số hiệu port mới) như là địa chỉ đích
- Ghi nhớ (trong bảng chuyển đổi NAT) mọi cặp chuyển đổi (địa chỉ IP nguồn, số hiệu port) sang (địa chỉ IP NAT, số hiệu port mới)
- Với các datagram đi đến: thay thế (địa chỉ IP NAT, số hiệu port mới) trong các trường đích của mọi datagram đến với giá trị tương ứng (địa chỉ IP và số hiệu cổng nguồn) trong bảng NAT



8. Giao thức ICMP (Slide 65-66)

ICMP: Internet Control Message Protocol

- Được sử dụng bởi các host và router để truyền thông tin tầng Mạng
 - Thông báo: host, network, port, giao thức không có thực
 - Các dạng gói tin echo request/reply (được dùng bởi ping)
- Tầng Mạng “trên” IP:
 - Các thông điệp ICMP được gửi trong các IP datagram
- Thông điệp ICMP: loại, mã cộng thêm 8 byte đầu tiên của IP datagram gây ra lỗi

| Loại | mã | Mô tả |
|------|----|---|
| 0 | 0 | echo reply (ping) |
| 3 | 0 | dest. network unreachable |
| 3 | 1 | dest host unreachable |
| 3 | 2 | dest protocol unreachable |
| 3 | 3 | dest port unreachable |
| 3 | 6 | dest network unknown |
| 3 | 7 | dest host unknown |
| 4 | 0 | source quench (congestion control - not used) |
| 8 | 0 | echo request (ping) |
| 9 | 0 | route advertisement |
| 10 | 0 | router discovery |
| 11 | 0 | TTL expired |
| 12 | 0 | bad IP header |

Traceroute và ICMP

- Nguồn gửi một chuỗi các segment UDP đến đích
 - Cái đầu tiên có TTL =1

- Cái thứ 2 có TTL=2, tương tự.
- Không quan tâm số port
- Khi datagram thứ n đến router thứ n:
 - Router hủy datagram
 - Và gửi đến nguồn một thông điệp ICMP (loại 11, mã 0)
 - Thông điệp ICMP bao gồm tên và địa chỉ IP của router
- Khi thông điệp ICMP đến, nguồn ghi lại RTTs

Tiêu chuẩn dùng:

- Gói UDP đến host đích
- Dịch trả về thông điệp ICMP “port không có thực”(loại 3, mã 3)
- Nguồn dùng

9. Thuật toán định tuyến Link-state (Slide 79-83) Máy độ phức tạp hong cần

Thuật toán routing Link-State

Thuật toán Dijkstra

- Tất cả các node (router/đỉnh trên đồ thị) đều biết chi phí kết nối, cấu trúc mạng
 - Được thực hiện thông qua gửi quảng bá các trạng thái kết nối (“link state broadcast”)
 - Tất cả các nodes có cùng thông tin với nhau
- Tính toán đường đi có chi phí thấp nhất từ 1 node (‘nguồn’) đến tất cả các node khác
 - Cho trước bảng *forwarding* của node đó
- Lặp lại: sau k lần lặp lại, biết được đường đi có chi phí thấp nhất tới k đích

Ký hiệu:

- $c(x,y)$: chi phí kết nối từ node x đến y; $c(x,y) = \infty$ nếu giữa x và y không có kết nối
- $D(v)$: giá trị chi phí hiện tại của đường đi từ nguồn tới đích v
- $p(v)$: node liền kề trước v nằm trên đường đi từ nguồn tới v
- N' : tập các node mà chi phí đường đi thấp nhất đã được xác định

Thuật toán Dijkstra

```

1 Khởi tạo:
2  $N' = \{u\}$ 
3 for tất cả các v
4   if v liền kề với u
5     then  $D(v) = c(u,v)$ 
6   else  $D(v) = \infty$ 
7
8 Lặp
9 tim w chưa tồn tại trong  $N'$  có  $D(w)$  nhỏ nhất
10 thêm w vào tập  $N'$ 
11 cập nhật lại  $D(v)$  cho tất cả kề với w và không có trong  $N'$  :
12  $D(v) = min( D(v), D(w) + c(w,v) )$ 
13 /* chi phí mới đến v là chính nó hoặc chi phí đường đi ngắn nhất
14 cộng với chi phí từ w đến v*/
15 Cho đến khi tất cả các node trong  $N'$ 

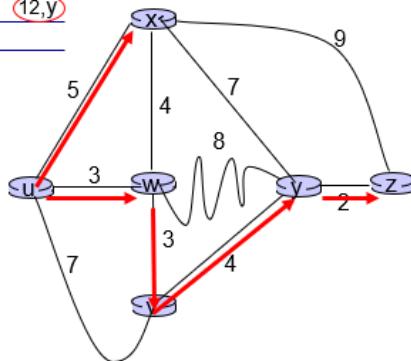
```

Thuật toán Dijkstra : ví dụ

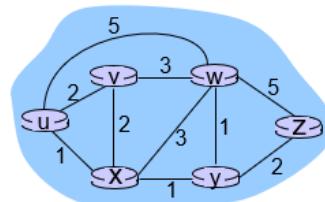
| Bước | N' | D(v) p(v) | D(w) p(w) | D(x) p(x) | D(y) p(y) | D(z) p(z) |
|------|--------|--------------|--------------|--------------|--------------|--------------|
| 0 | u | 7,u | 3,u | 5,u | ∞ | ∞ |
| 1 | uw | 6,w | 5,u | 11,w | ∞ | |
| 2 | uwx | 6,w | | 11,w | 14,x | |
| 3 | uwxv | | | 10,v | 14,x | |
| 4 | uwxvy | | | | 12,y | |
| 5 | uwxvzy | | | | | |

Ghi chú:

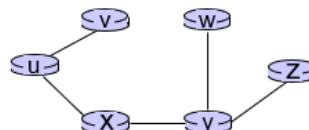
- Xây dựng cây đường đi ngắn nhất bằng cách lần theo các node liền kề trước đó
- Các đường có chi phí bằng nhau có thể tồn tại (có thể bị chia tùy tiện)



| Bước | N' | D(v),p(v) | D(w),p(w) | D(x),p(x) | D(y),p(y) | D(z),p(z) |
|------|--------|-----------|-----------|-----------|-----------|-----------|
| 0 | u | 2,u | 5,u | 1,u | ∞ | ∞ |
| 1 | ux | 2,u | 4,x | | 2,x | ∞ |
| 2 | uxy | 2,u | 3,y | | 4,y | |
| 3 | uxyv | | 3,y | | 4,y | |
| 4 | uxyvw | | | | | 4,y |
| 5 | uxyvwz | | | | | |



Kết quả cây đường đi ngắn nhất từ u:



Kết quả bảng forwarding trong u:

| Đích đến | link |
|----------|-------|
| v | (u,v) |
| x | (u,x) |
| y | (u,x) |
| w | (u,x) |
| z | (u,x) |

10. Thuật toán Distance vector (Slide 86-92)

Thuật toán Distance vector

Công thức Bellman-Ford (dynamic programming)

cho

$d_x(y) :=$ chi phí của đường đi có chi phí ít nhất từ x
tới y

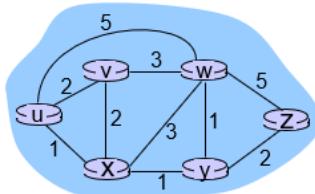
thì

$$d_x(y) = \min_v \{ c(x,v) + d_v(y) \}$$

Chi phí từ neighbor v tới đích y
Chi phí tới neighbor v

\min được thực hiện trên tất cả các neighbor v của x

Bellman-Ford ví dụ



Rõ ràng, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

Biểu thức B-F cho kết quả:

$$\begin{aligned} d_u(z) &= \min \{ c(u,v) + d_v(z), \\ &\quad c(u,x) + d_x(z), \\ &\quad c(u,w) + d_w(z) \} \\ &= \min \{ 2 + 5, \\ &\quad 1 + 3, \\ &\quad 5 + 3 \} = 4 \end{aligned}$$

Node có giá trị tối thiểu là trạm (hop) kế tiếp trong
đường đi ngắn nhất, được sử dụng trong bảng
forwarding

Thuật toán Distance vector

- $D_x(y) =$ ước lượng chi phí thấp nhất từ x đến y
 - x lưu lại distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x:
 - Biết chi phí đến mỗi neighbor v: $c(x,v)$
 - Lưu lại distance vectors của các neighbor của nó. Cho mỗi neighbor v, x lưu lại $\mathbf{D}_v = [D_v(y): y \in N]$

Ý tưởng chính:

- Mỗi node định kỳ gửi ước lượng distance vector của nó cho các neighbor
- Khi x nhận ước lượng DV mới từ neighbor, nó cập nhật DV cũ của nó dùng công thức B-F:

$$D_x(y) \leftarrow \min_v \{ c(x,v) + D_v(y) \} \text{ với mỗi node } y \in N$$

- Dưới những điều kiện tự nhiên, giá trị ước lượng $D_x(y)$ sẽ hội tụ tới chi phí thực sự nhỏ nhất $d_x(y)$

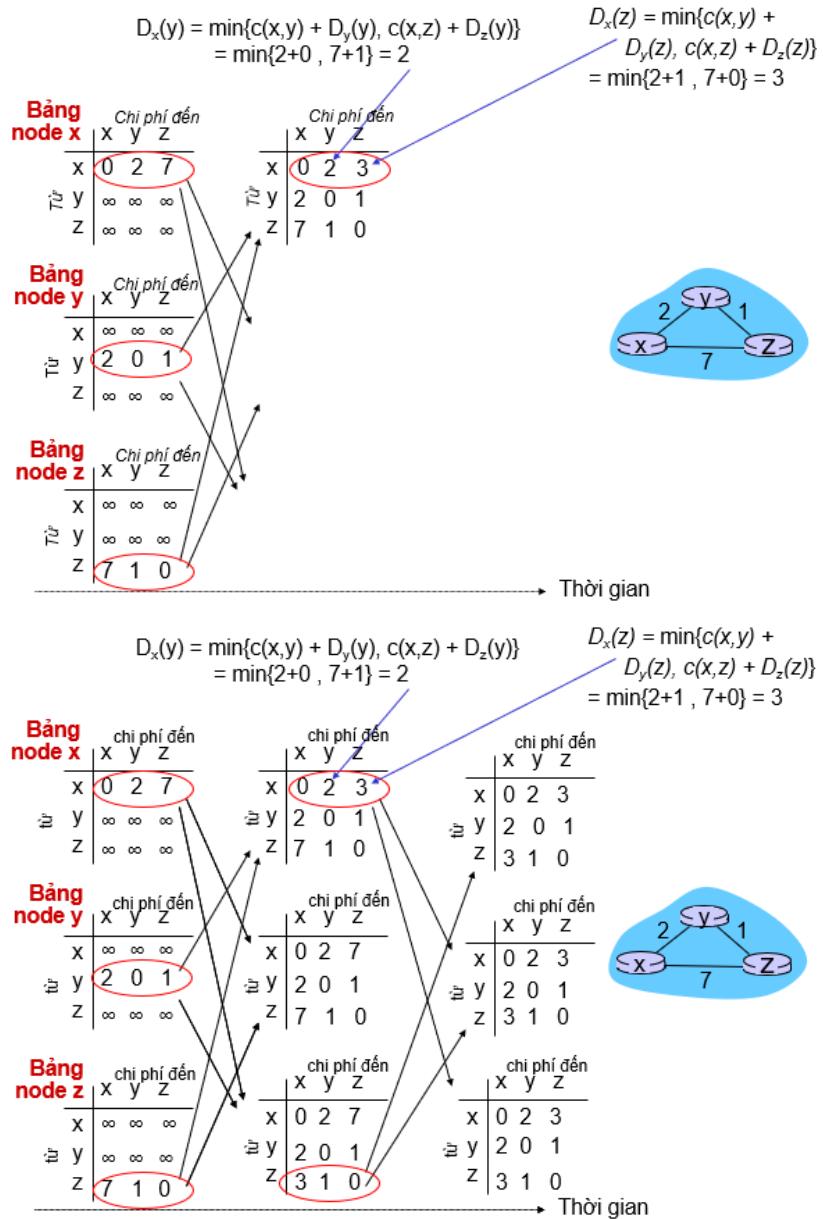
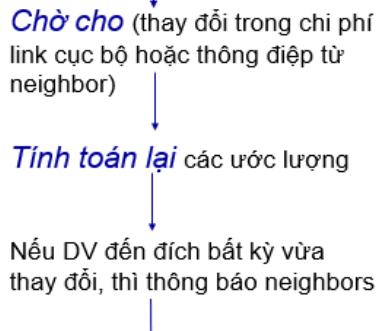
Lặp, không đồng bộ: mỗi lặp cục bộ tại 1 node được gây ra bởi:

- Chi phí kết nối cục bộ thay đổi
- Nhận được DV thông báo cập nhật từ node lân cận (neighbor)

Phân bố:

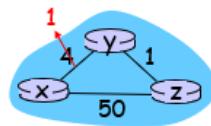
- Mỗi node thông báo đến các node lân cận *chỉ khi* DV của nó thay đổi
 - Các node lân cận sau đó thông báo đến các node lân cận của nó nếu cần thiết

Mỗi node:



Chi phí kết nối thay đổi:

- Node phát hiện sự thay đổi chi phí kết nối cục bộ
- Cập nhật thông tin định tuyến, tính toán lại distance vector
- Nếu DV thay đổi, thì thông báo cho neighbor



**“tín
tốt
đi
nhanh”**

t_0 : y phát hiện sự thay đổi chi phí kết nối, và cập nhật DV của nó, thông báo đến các neighbor của nó.

t_1 : z nhận được cập nhật từ y, và cập nhật bảng của nó, tính chi phí mới thấp nhất đến x, gởi DV của nó đến các neighbor của nó.

t_2 : y nhận được cập nhật của z, và cập nhật bảng distance của nó. Các chi phí thấp nhất của y *không* thay đổi, vì vậy y không gởi thông điệp đến z.

11. Định tuyến có cấu trúc (Slide 98-100)

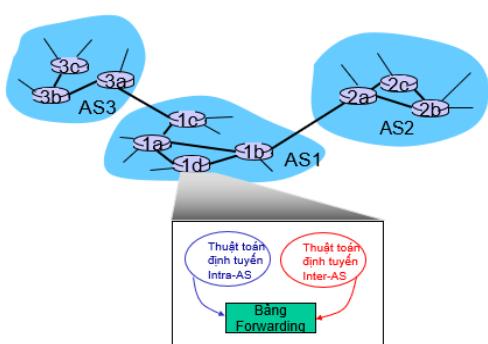
Định tuyến có cấu trúc

- Các router được gom vào các vùng tự trị “autonomous systems” (AS)
- Các router trong cùng AS chạy cùng giao thức định tuyến với nhau
 - Giao thức **định tuyến “intra-AS”**
 - Các router trong các AS khác nhau có thể chạy các giao thức định tuyến intra-AS khác nhau

Gateway router:

- Tại “biên” (“edge”) của AS của nó
- Có liên kết đến router trong AS khác

Kết nối các AS



- Bảng forwarding được cấu hình bởi cả thuật toán định tuyến intra- và inter-AS
 - Intra-AS thiết lập các mục cho các đích đến nội mạng
 - Inter-AS & intra-AS thiết lập các mục cho các đích đến ngoại mạng

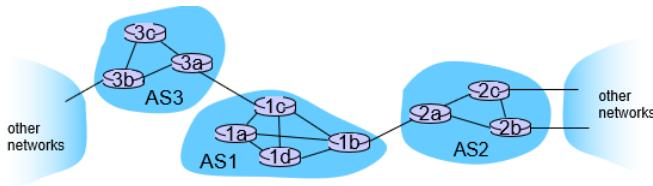
Tác vụ Inter-AS

- Giả sử router trong AS1 nhận được datagram với đích đến nằm ngoài AS1:
 - router nên chuyển packet đến router gateway, nhưng là cái nào?

AS1 phải:

- Học các đích đến nào có thể tới được thông qua AS2 và AS3
- Lan truyền thông tin này đến tất cả các router trong AS1

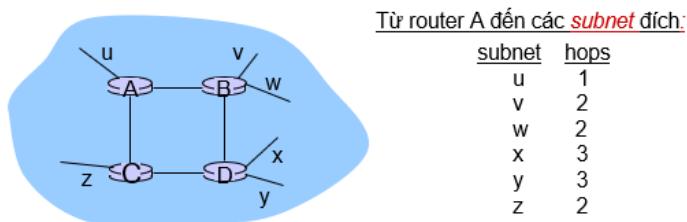
Công việc của định tuyến inter-AS!



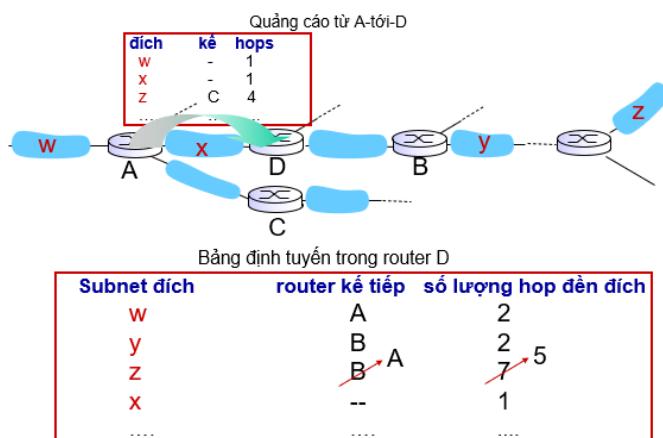
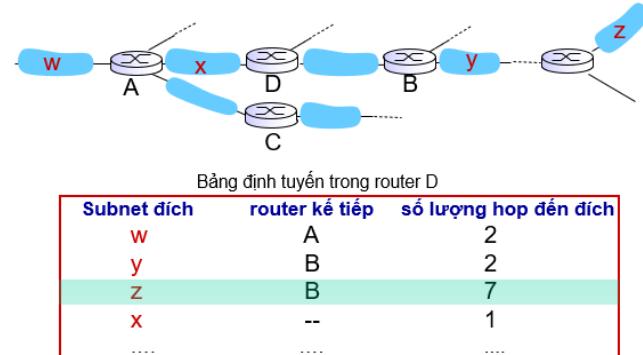
12. Định tuyến RIP (Slide 106-109)

RIP (Routing Information Protocol)

- Công bố vào năm 1982 trong BSD-UNIX
- Thuật toán distance vector
 - Đơn vị đo khoảng cách: số lượng hop (max = 15 hops), mỗi link có giá trị là 1
 - Các DV được trao đổi giữa các neighbors mỗi 30 giây trong thông điệp phản hồi (còn gọi là **advertisement**)
 - Mỗi advertisement: liệt kê lên đến 25 subnet đích



RIP: ví dụ



RIP: lỗi đường kết nối và phục hồi

Nếu không có quảng cáo nào sau 180 giây --> neighbor/kết nối được xem như đã chết

- Những đường đi qua neighbor này bị vô hiệu
- Các quảng cáo mới được gửi tới các neighbor còn lại
- Các neighbor đó tiếp tục gửi ra những quảng cáo mới đó (nếu các bảng bị thay đổi)
- Thông tin về lỗi đường kết nối nhanh chóng (?) lan truyền trên toàn mạng
- **poison reverse** được dùng để ngăn chặn vòng lặp ping-pong (khoảng cách vô hạn = 16 hops)

13. OSPF (Slide 111-113)

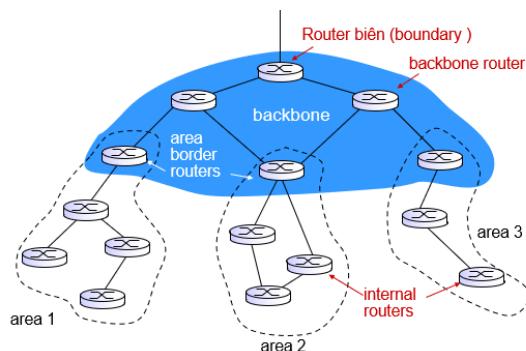
OSPF (Open Shortest Path First)

- “open”: công khai cho mọi đối tượng sử dụng
- Dùng thuật toán Link State
 - Quảng bá gói tin LS
 - Bản đồ cấu trúc mạng tại mỗi node
 - Tính toán đường đi dùng thuật toán Dijkstra
- Thông điệp quảng bá OSPF chứa 1 mục thông tin cho mỗi router lân cận
- Các thông điệp này được phát tán đến **toàn bộ AS**
 - thông điệp OSPF được mang trực tiếp trên IP (chứ không phải là TCP hoặc UDP)
- Giao thức **định tuyến IS-IS**: gần giống với OSPF

Các đặc tính “vượt trội” của OSPF (không có trong RIP)

- **Bảo mật**: tất cả các thông điệp OSPF đều được chứng thực (để chống lại sự xâm nhập có hại)
- Cho phép có **nhiều đường đi** có chi phí như nhau (RIP chỉ cho 1)
- Với mỗi đường kết nối, có nhiều đơn vị tính chi phí (cost metrics) cho dịch vụ (**TOS**) khác nhau (ví dụ: chi phí đường kết nối vệ tinh được thiết lập “thấp” cho ToS nỗ lực tốt nhất; cao cho ToS thời gian thực)
- Hỗ trợ uni- và **multicast** tích hợp:
 - Multicast OSPF (MOSPF) dùng cùng cơ sở dữ liệu cấu trúc mạng như OSPF
- OSPF **phân cấp** được dùng trong các miền lớn (large domains).

OSPF phân cấp



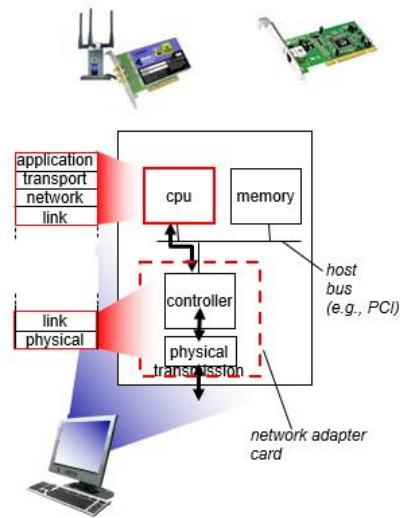
Chương 5.

(20% - 8 câu)

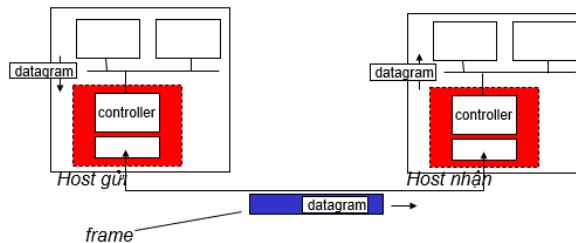
1. Tầng link thực hiện tại (Slide 8-9)

Tầng Liên kết dữ liệu được triển khai ở đâu?

- Trong mọi host
- Tầng Liên kết dữ liệu được triển khai trong “adaptor” (còn gọi là *network interface card* NIC)
 - Ethernet card, 802.11 card; Ethernet chipset
 - Triển khai cả tầng Vật lý và tầng Liên kết dữ liệu
- Nối vào trong các bus hệ thống của host
- Sự kết hợp của phần cứng, phần mềm và firmware



Các Adaptor liên lạc



- ❖ Bên gửi:
 - Đóng gói datagram vào trong frame
 - Thêm các bit kiểm tra lỗi, rdt và điều khiển luồng...
- ❖ Bên nhận
 - Tìm lỗi, rdt và điều khiển luồng...
 - Tách các datagram ra, chuyển lên tầng trên tại nơi nhận

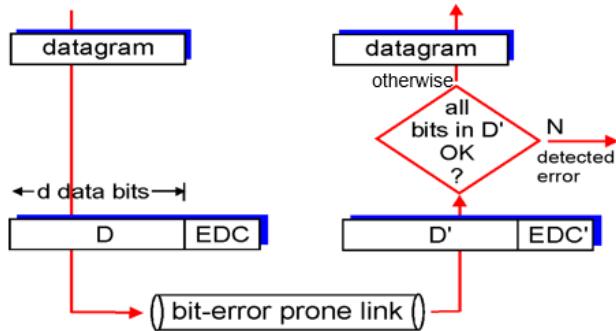
2. Phát hiện lỗi (Slide 11-18): Phát hiện lỗi chẵn lẻ, CRC,...

Phát hiện lỗi

EDC= Error Detection and Correction bits (redundancy)

D = phần dữ liệu được bảo vệ bởi trường EDC, có thể chứa các trường header

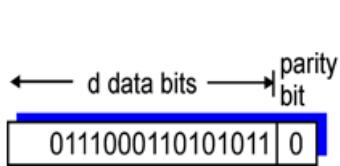
- Việc phát hiện lỗi không đảm 100%!
 - giao thức có thể bỏ qua một số lỗi
 - trường EDC càng lớn thì việc phát hiện và sửa lỗi càng tốt hơn



Kiểm tra chẵn lẻ (Parity checking)

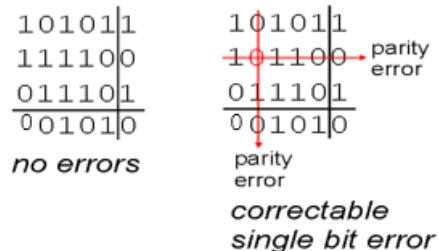
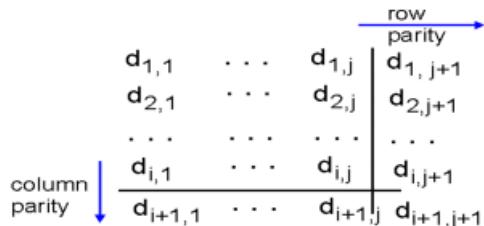
bit parity đơn:

- ❖ Phát hiện các lỗi bit đơn



bit parity 2 chiều:

- ❖ Phát hiện và sửa các lỗi bit đơn



Internet checksum

Mục tiêu: phát hiện “các lỗi” (ví dụ, các bit bị đảo) trong packet được truyền (chú ý: chỉ được dùng tại tầng Vận chuyển)

Bên gửi:

- Xử lý các nội dung của segment như một chuỗi các số nguyên 16-bit
- checksum: là (tổng bù 1) của các nội dung của segment
- Bên gửi đặt giá trị checksum vào trong trường checksum của UDP

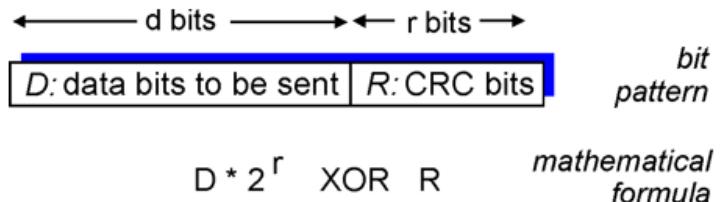
Bên nhận:

- Tính toán checksum của segment vừa nhận
- Kiểm tra xem giá trị của checksum vừa được tính có bằng với giá trị trong trường checksum không:
 - không – phát hiện lỗi
 - có – không phát hiện lỗi. Nhưng có thể còn có lỗi khác không?

Cyclic redundancy check

- Thuật toán phát hiện lỗi tốt hơn

- Xem đoạn bit dữ liệu, D , như một số nhị phân
- Chọn mẫu chiều dài $r+1$ bit (bộ khởi tạo), G
- Mục tiêu: chọn r bit CRC, R , sao cho
 - $\langle D, R \rangle$ chia hết cho G (theo cơ số 2)
 - Bên nhận biết G , chia $\langle D, R \rangle$ cho G . Nếu phần dư khác không: phát hiện lỗi!
 - Có thể phát hiện tất cả các lỗi nhỏ hơn $r+1$ bits
- Được sử dụng rộng rãi trong thực tế (Ethernet, 802.11 WiFi, ATM)



CRC ví dụ

Muốn:

$$D \cdot 2^r \text{ XOR } R = nG$$

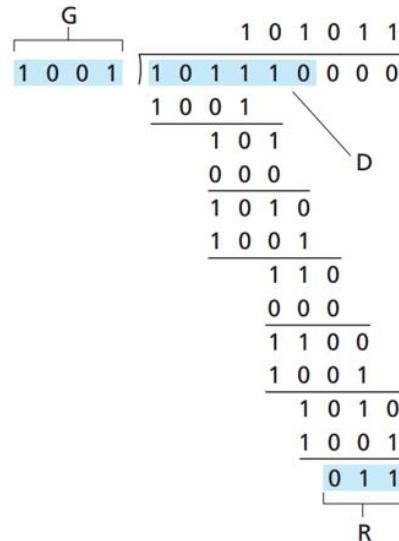
Tương đương:

$$D \cdot 2^r = nG \text{ XOR } R$$

Tương đương:

nếu chúng ta chia $D \cdot 2^r$ cho G , có được phần dư R
thỏa:

$$R = \text{remainder} \left[\frac{D \cdot 2^r}{G} \right]$$

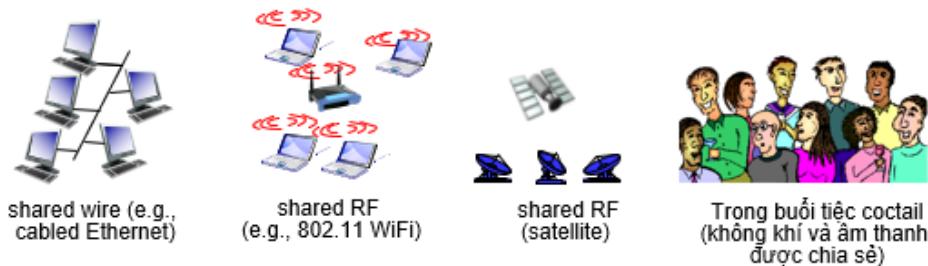


3. Giao thức đa truy cập (Slide 21, 24-27, 29, 33, 35, 37, 39, 44)

Các giao thức và kết nối đa truy cập

2 kiểu “kết nối”:

- Điểm-điểm (point-to-point)
 - PPP cho truy cập dial-up
 - Kết nối point-to-point giữa Ethernet switch và host
- *broadcast (chia sẻ đường truyền dùng chung)*
 - Ethernet mô hình cũ
 - upstream HFC
 - 802.11 wireless LAN



Các giao thức MAC: phân loại

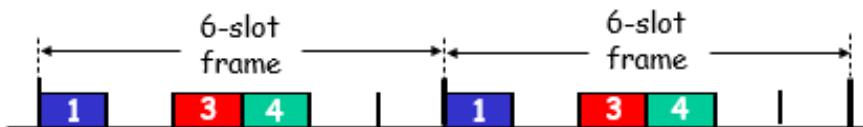
3 loại chính:

- **Phân hoạch kênh (channel partitioning)**
 - Chia kênh truyền thành “các mảnh” nhỏ hơn (các slot thời gian - TDM, tần số - FDM, mã - CDM)
 - Cấp phát mảnh này cho node để sử dụng độc quyền
- **Truy cập ngẫu nhiên (random access)**
 - Kênh truyền không được chia, cho phép đụng độ
 - “giải quyết” đụng độ
- **“Xoay vòng”**
 - Các node thay phiên nhau, nhưng nút có nhiều dữ liệu hơn được giữ thời gian truyền lâu hơn

Các giao thức MAC phân hoạch kênh: TDMA

TDMA: time division multiple access

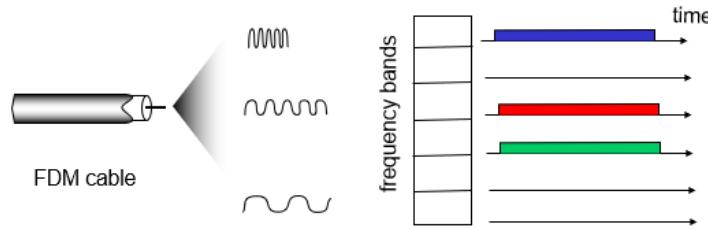
- Truy cập kênh truyền theo hình thức “xoay vòng”
- Mỗi trạm (station) có slot với độ dài cố định (độ dài = thời gian truyền packet) trong mỗi vòng (round)
- Các slot không sử dụng sẽ nhàn rỗi
- Ví dụ: LAN có 6 trạm, 1,3,4 có gói được gửi, các slot 2,5,6 sẽ nhàn rỗi



Các giao thức MAC phân hoạch kênh: FDMA

FDMA: frequency division multiple access

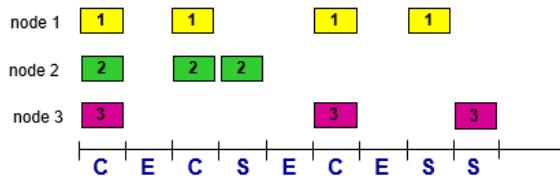
- Phổ kênh truyền được chia thành các dải tần số
- Mỗi trạm được gán một dải tần số cố định
- Trong thời gian không truyền, các dải tần rảnh
- Ví dụ: LAN có 6 station, 1,3,4 có packet truyền, các dải tần số 2,5,6 nhàn rỗi



Các giao thức truy cập ngẫu nhiên

- Khi 1 node có packet cần gửi
 - Truyền dữ liệu với trộn tốc độ của kênh dữ liệu R.
 - Không có sự ưu tiên giữa các node
- 2 hoặc nhiều node truyền → “đụng độ”;
- **Giao thức truy cập ngẫu nhiên MAC** xác định:
 - Cách để phát hiện đụng độ
 - Cách để giải quyết đụng độ (ví dụ: truyền lại sau đó)
- Ví dụ các giao thức MAC truy cập ngẫu nhiên:
 - slotted ALOHA
 - ALOHA
 - CSMA, CSMA/CD, CSMA/CA

Slotted ALOHA



Ưu điểm:

- ❖ Node hoạt động có thể truyền liên tục với tốc độ tối đa của kênh
- ❖ Phân tán cao: chỉ có các slot trong các node cần được đồng bộ
- ❖ Đơn giản

Nhược điểm:

- ❖ Đụng độ, lãng phí slot
- ❖ Các slot trống
- ❖ Các node có thể phát hiện tranh chấp nhanh hơn thời gian để truyền gói
- ❖ Đồng bộ hóa

CSMA (carrier sense multiple access)

CSMA: lắng nghe trước khi truyền:

- Nếu kênh nhàn rỗi: truyền toàn bộ frame
- Nếu kênh truyền bận, trì hoãn truyền
- So sánh với con người: dừng ngắt lời người khác!

CSMA/CD (collision detection)

CSMA/CD: trì hoãn như trong CSMA

- Đụng độ được phát hiện trong thời gian ngắn
- Thông tin đang truyền bị hủy bỏ, giảm sự lãng phí kênh.

- Phát hiện đụng độ:
 - Dễ dàng trong các mạng LAN hữu tuyến: đo cường độ tín hiệu, so sánh với các tín hiệu đã được truyền và nhận
 - Khó thực hiện trong mạng LAN vô tuyến: cường độ truyền cục bộ lấn át cường độ tín hiệu nhận
- Tương tự như hành vi của con người: đàm thoại lịch sự

Thuật toán Ethernet CSMA/CD

1. NIC nhận datagram từ tầng network, tạo frame
2. Nếu NIC dò được kênh rỗng, nó sẽ bắt đầu việc truyền frame. Nếu NIC dò được kênh bận, đợi cho đến khi kênh rỗng, sau đó mới truyền.
3. Nếu NIC truyền toàn bộ frame mà không phát hiện việc truyền khác, NIC được truyền toàn bộ frame đó!
4. Nếu NIC phát hiện có phiên truyền khác trong khi đang truyền, thì nó sẽ hủy bỏ việc truyền và phát tín hiệu tắc nghẽn
5. Sau khi hủy bỏ truyền, NIC thực hiện *binary (exponential) backoff*:
 - Sau lần đụng độ thứ m , NIC chọn ngẫu nhiên số K trong khoảng $\{0, 1, 2, \dots, 2^m - 1\}$. NIC sẽ đợi $K \cdot 512$ thời gian truyề bit (bit time), sau đó trở lại bước 2
 - Càng nhiều đụng độ thì sẽ có khoảng thời gian chờ dài hơn

Các giao thức MAC “Xoay vòng”

Các giao thức phân hoạch kênh MAC (channel partitioning MAC protocols):

- Chia sẻ kênh hiệu quả và công bằng với tải lớn
- Không hiệu quả ở tải thấp: trễ khi truy cập kênh, 1 node chỉ được cấp phát $1/N$ bandwidth ngay cả khi chỉ có 1 node hoạt động!

Các giao thức MAC truy cập nhẫu nhiên (random access MAC protocols)

- Hiệu quả tại tải thấp: node đơn có thể dùng hết khả năng của kênh
- Tải cao: đụng độ cao

Các giao thức “Xoay vòng” (“taking turns” protocols)

Tìm kiếm giải pháp dung hòa tốt nhất!

Tổng kết các giao thức MAC

- *Phân hoạch kênh*, theo thời gian, tần số hoặc mã
 - Phân chia theo thời gian (Time Division), phân chia theo tần số (Frequency Division)
- *Truy cập ngẫu nhiên* (động),
 - ALOHA, S-ALOHA, CSMA, CSMA/CD
 - Cảm nhận sóng mang (carrier sensing): dễ dàng trong một số kỹ thuật (có dây), khó thực hiện trong các công nghệ khác (không dây)
 - CSMA/CD được dùng trong Ethernet
 - CSMA/CA được dùng trong 802.11
- *Xoay vòng*
 - Điều phối từ đơn vị trung tâm, truyền token
 - bluetooth, FDDI, token ring

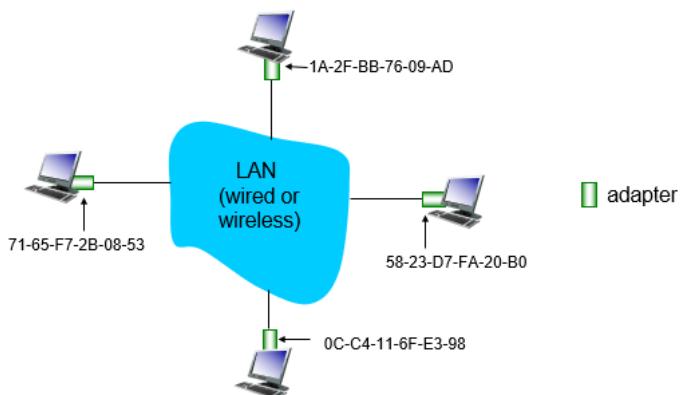
4. Giao thức ARP (Slide 46-56)

Địa chỉ MAC và ARP

- Địa chỉ IP 32-bit:
 - Địa chỉ tầng network cho interface
 - Được sử dụng trong chức năng chuyển dữ liệu tầng 3 (tầng network)
- Địa chỉ MAC (hoặc địa chỉ LAN hoặc physical hoặc Ethernet) :
 - Chức năng: được sử dụng “cục bộ” để chuyển frame từ 1 interface này đến 1 interface được kết nối vật lý trực tiếp với nhau (cùng mạng, trong ngữ cảnh vùng địa chỉ IP)
 - Địa chỉ MAC 48 bit (cho hầu hết các mạng LAN) được ghi vào trong NIC ROM, hoặc thiết lập trong phần mềm
 - Ví dụ: 1A-2F-BB-76-09-AD

Ghi dưới dạng số thập lục phân (hệ 16)
(mỗi “số” đại diện 4 bit)

Mỗi adapter trên mạng LAN có địa chỉ *LAN* duy nhất

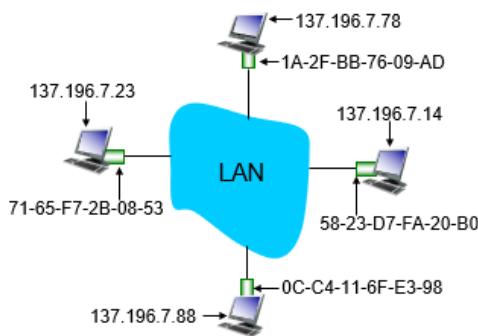


Địa chỉ LAN(tt)

- Sự phân bổ địa chỉ MAC được quản lý bởi IEEE
- Nhà sản xuất mua phần khống gian địa chỉ MAC (bảo đảm duy nhất)
- So sánh:
 - Địa chỉ MAC: như là số chứng minh nhân dân
 - Địa chỉ IP: như là địa chỉ bưu điện
- Địa chỉ MAC không phân cấp, có tính di chuyển
 - Có thể di chuyển card LAN từ 1 mạng LAN này tới mạng LAN khác
- Địa chỉ IP phân cấp, không di chuyển được
 - Địa chỉ phụ thuộc vào subnet IP mà node đó gắn vào

ARP: address resolution protocol

Hỏi: làm cách nào để xác định địa chỉ MAC của interface khi biết được địa chỉ IP của nó?



Bảng ARP: mỗi node (host, router) trên mạng LAN có bảng ARP

- Địa chỉ IP/MAC ánh xạ cho các node trong mạng LAN:

<địa chỉ IP; địa chỉ MAC; TTL>

- TTL (Time To Live): thời gian sau đó địa chỉ ánh xạ sẽ bị lãng quên (thông thường là 20 phút)

Giao thức ARP: cùng mạng LAN

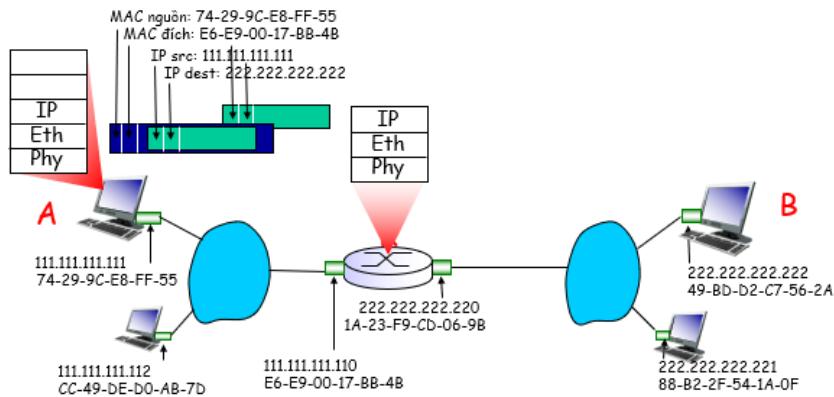
- A muốn gửi datagram tới B
 - Địa chỉ MAC của B không có trong bảng ARP của A.
- A sẽ gửi quảng bá (broadcasts) gói tin ARP query có chứa địa chỉ IP của B
 - Địa chỉ MAC đích = FF-FF-FF-FF-FF-FF
 - Tất cả các node trên mạng LAN sẽ nhận ARP query này
- B nhận gói tin ARP, trả lời tới A với địa chỉ MAC của B
 - Frame được gửi tới địa chỉ MAC của A (unicast)
- A sẽ lưu lại cặp địa chỉ IP-MAC trong bảng ARP của nó cho tới khi thông tin này hết hạn sử dụng
 - soft state: thông tin hết hạn (bỏ đi) trừ khi được làm mới
- ARP là giao thức “plug-and-play”:
 - Các nodes tạo bảng ARP của nó không cần sự can thiệp của người quản trị mạng

Addressing: định tuyến tới mạng LAN khác

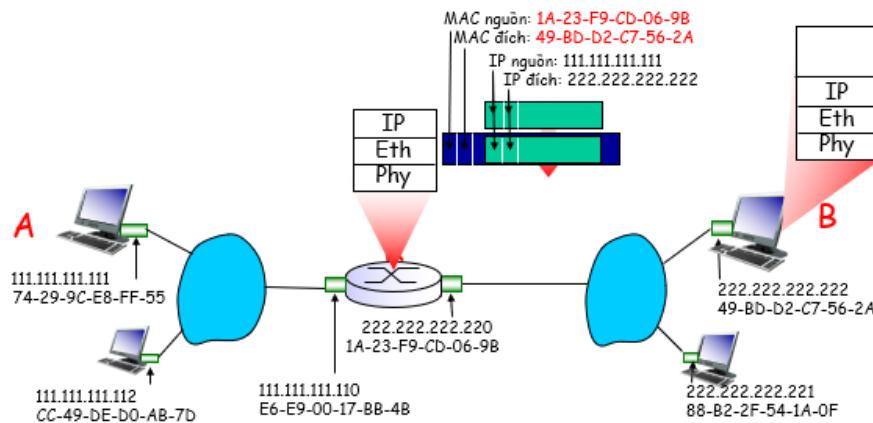
Từng bước: gửi datagram từ A tới B thông qua R

- tập trung vào gán địa chỉ – tại tầng IP (datagram) và MAC (frame)
- giả sử A biết địa chỉ IP của B
- giả sử A biết địa chỉ IP của router gateway R (cách nào?)
- giả sử A biết địa chỉ MAC của R (cách nào?)
- A tạo IP datagram với IP nguồn A, đích B

- A tạo frame tầng Liên kết dữ liệu với địa chỉ MAC của R là địa chỉ đích, frame này chứa IP datagram từ A tới B
- frame được gửi từ A tới R
- frame được R nhận, datagram được gỡ bỏ, được chuyển tới IP



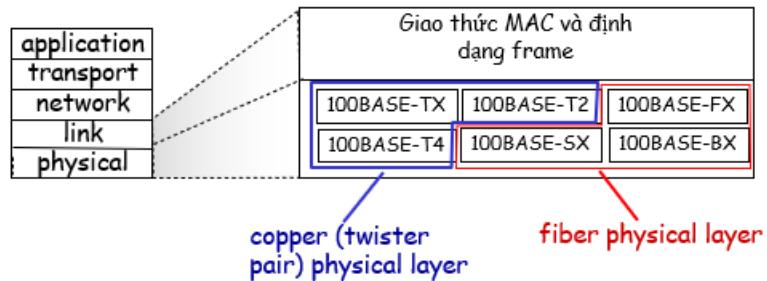
- R chuyển tiếp datagram với IP nguồn A, đích B
- R tạo frame tầng Liên kết dữ liệu với địa chỉ MAC của B là địa chỉ đích, frame này chứa IP datagram từ A-tới-B



5. Chuẩn Ethernet (Slide 63)

Chuẩn Ethernet 802.3: Tầng Liên kết dữ liệu & Tầง Vật lý

- **Nhiều** chuẩn Ethernet khác nhau
 - Cùng giao thức MAC và định dạng frame
 - Tốc độ khác nhau: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10G bps
 - Môi trường truyền tầng Vật lý khác nhau: fiber, cable

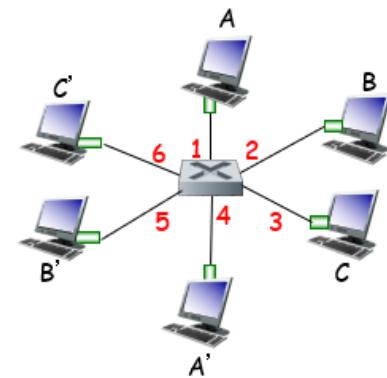


6. Bộ chuyển mạch (Switch) (Slide 66-74)

Switch: nhiều phiên truyền đồng thời

- Các host kết nối trực tiếp tới switch
- Switch lưu tạm (buffer) các packet
- Giao thức Ethernet được sử dụng trên *mỗi* đường kết nối vào, nhưng không có dung độ; full duplex
 - Mỗi đường kết nối là 1 miền dung độ (collision domain) của riêng nó
- switching:** A-tới-A' và B-tới-B' có thể truyền đồng thời mà không có dung độ xảy ra

Hỏi: làm thế nào để switch biết tới A' thì sẽ thông qua interface 4 và tới B' thì thông interface 5?



switch với 6 interface
(1,2,3,4,5,6)

Đáp: mỗi switch có **một bảng switch**, mỗi dòng gồm:

(địa chỉ MAC của host, interface để tới được host đó, time stamp)

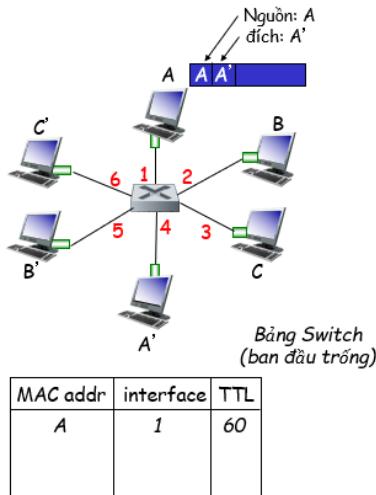
Giống như bảng định tuyến!

Q: những dòng được tạo và được duy trì như thế nào trong bảng switch?

Có giống như giao thức định tuyến hay không?

Switch: tự học

- Switch *học* về vị trí các host có thể truyền tới được thông qua các interface kết nối với các host đó
 - Khi switch nhận được frame, switch “học” vị trí của bên gửi: cổng vào - incoming LAN segment
 - Ghi lại cặp thông tin (bên gửi-vị trí) vào trong bảng switch



Switch: lọc/chuyển tiếp frame

Khi switch nhận được frame:

1. Ghi lại cổng kết nối vào, địa chỉ MAC của host gửi
2. Tạo chỉ mục bảng switch bằng địa chỉ MAC đích
3. Nếu tìm thấy thông tin đích đến

thì {

nếu đích đến nằm trên phân đoạn mạng gửi frame đến

thì bỏ frame

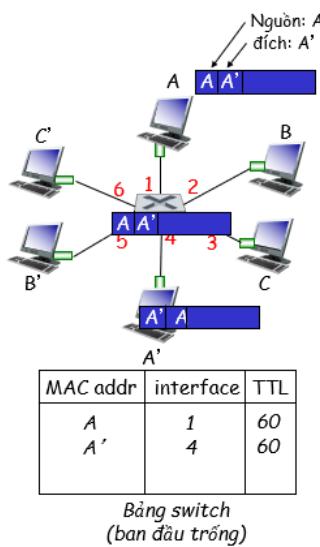
ngược lại chuyển tiếp frame trên interface được chỉ định bởi thông tin trong bảng switch

}

ngược lại flood /* chuyển tiếp trên tất cả interface ngoại trừ interface nhận frame đó*/

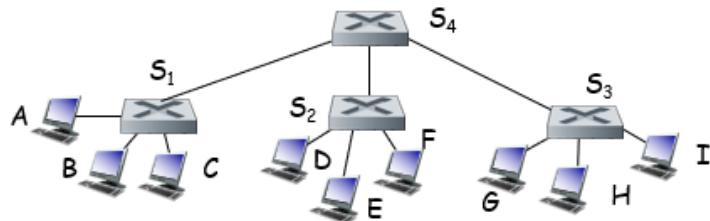
Tự học, chuyển tiếp: ví dụ

- frame có đích đến là A', vị trí của A' không biết: **flood**
- Đích A có vị trí đã được biết trước: **gửi chọn lọc chỉ trên 1 đường kết nối duy nhất**



Kết nối các switch với nhau (Interconnecting switches)

- Các switch có thể được kết nối với nhau

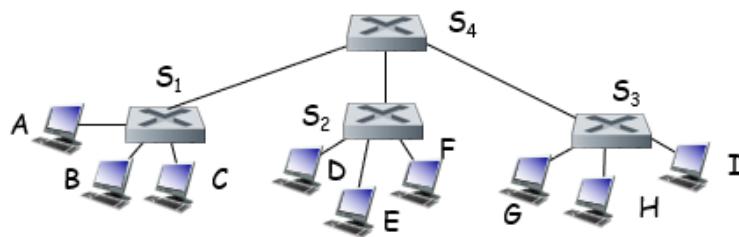


Hỏi: gửi từ A tới G – làm cách nào S₁ biết để chuyển tiếp frame tới F thông qua S₄ và S₃?

Trả: tự học! (làm việc giống y chang như trong trường hợp chỉ có 1 switch!)

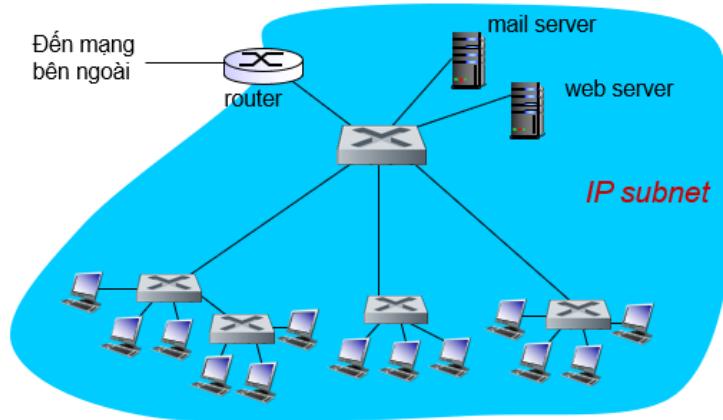
Ví dụ nhiều switch tự học

Giả sử C gửi frame tới I, I trả lời cho C



Hỏi: trình bày các bảng của các switch và cách các gói tin được chuyển đi tại các switch S₁, S₂, S₃, S₄

Mạng của tổ chức



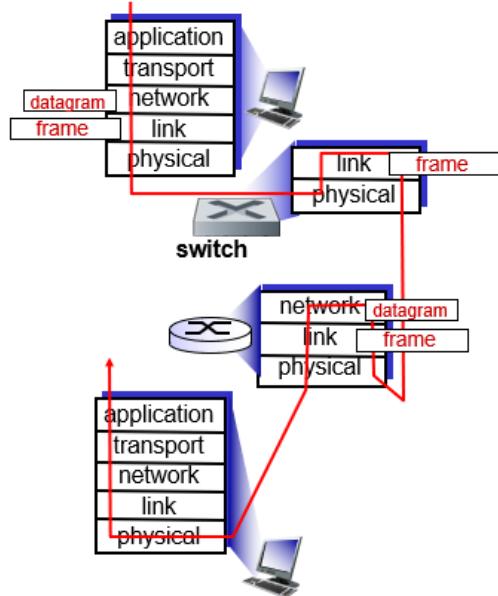
So sánh Switch và Router

Cả 2 đều lưu và chuyển tiếp (store-and-forward):

- router:** thiết bị tầng Mạng (kiểm tra header của tầng Mạng)
- switch:** thiết bị tầng Liên kết dữ liệu (kiểm tra header của tầng Liên kết dữ liệu)

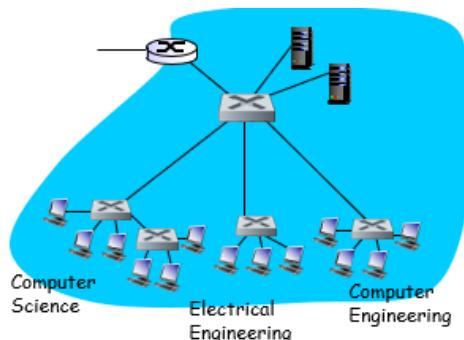
Cả 2 đều có bảng forwarding:

- **router:** tính toán bảng dùng các thuật toán định tuyến, địa chỉ IP
- **switch:** học bảng forwarding dùng cơ chế flooding, tự học, địa chỉ MAC



7. VLan (Slide 75-78)

VLANs: trình bày

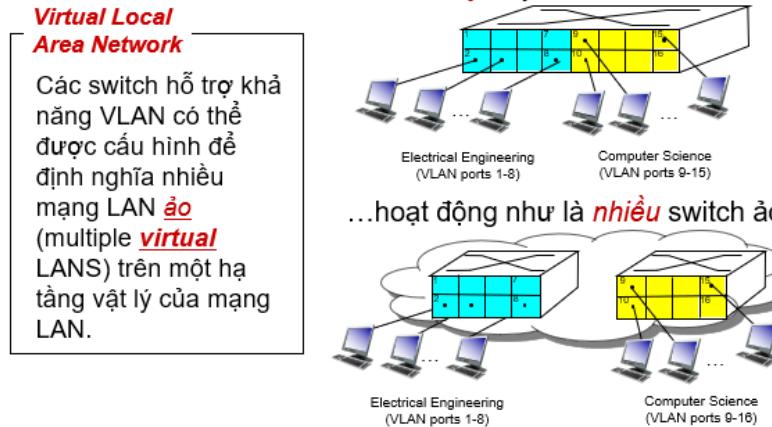


Xem xét:

- Người dùng bên CS di chuyển văn phòng sang EE, nhưng vẫn muốn kết nối CS switch?
- Miền broadcast đơn:
 - Tất cả lưu lượng broadcast tầng 2 (ARP, DHCP, địa chỉ MAC không biết vị trí đích đến ở đâu) phải đi qua toàn mạng LAN
 - An ninh/riêng tư, các vấn đề về hiệu suất

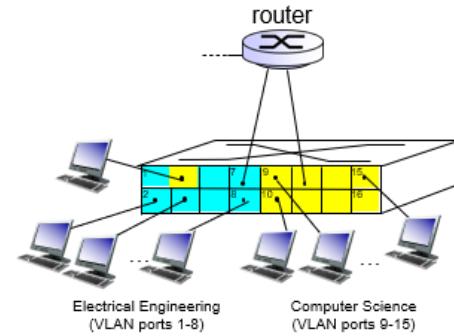
VLANs

port-based VLAN: các port của switch được nhóm lại (bởi phần mềm quản lý switch) để trở thành một switch **vật lý** duy nhất...

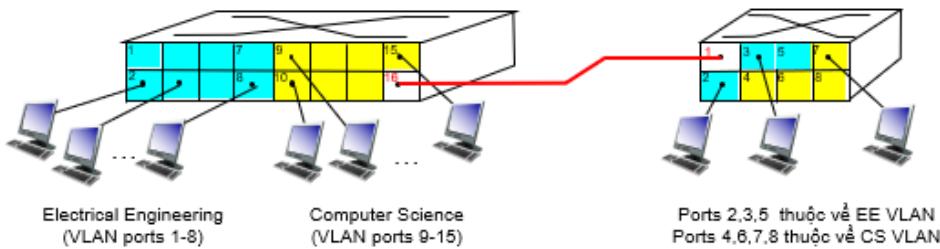


Port-based VLAN

- **Traffic isolation (cô lập traffic):** các frame đến từ các port 1-8 chỉ có thể tới được các port 1-8
 - Cũng có thể định nghĩa VLAN dựa trên địa chỉ MAC của thiết bị đầu cuối, hơn là dựa trên port của switch
- **Thay đổi linh động (dynamic membership):** các port có thể được gán động giữa các VLAN
- **Chuyển tiếp giữa các VLAN:** được thực hiện thông qua định tuyến (cũng giống như nối các switch riêng biệt)
 - Trên thực tế, các nhà cung cấp bán các thiết bị switch đã kết hợp thêm một vài tính năng của router



VLANs nối nhiều switch



- **Trunk port:** mang các frame giữa các VLAN được định nghĩa trên nhiều switch vật lý
 - Các frame được chuyển tiếp bên trong VLAN giữa các switch không thể là các frame 802.1 (phải mang thông tin VLAN ID)
 - Giao thức 802.1q thêm/gỡ bỏ các trường header được thêm vào các frame được chuyển tiếp giữa các trunk port

Mục lục

| | |
|--|----------|
| Chương 1..... | 1 |
| 1. Độ trễ, phân loại và tính toán (Slide 25, 26, 43, 44, 45) | 1 |
| Chuyển mạch gói: lưu và chuyển tiếp (store-and-forward) | 1 |
| Sự mất mát và độ trễ xảy ra như thế nào? | 1 |
| Bốn nguồn gây ra chậm trễ gói tin..... | 2 |
| 2. Mô hình mạng (TCP/IP, OSI) (Slide 60-62): thứ tự các tầng lớp, đơn vị dữ liệu của các tầng và thiết bị tại các tầng..... | 3 |
| Chồng giao thức Internet..... | 3 |
| Mô hình tham chiếu ISO/OSI..... | 3 |
| Đóng gói & Port của một số giao thức..... | 3 |
| Chương 2..... | 4 |
| 1. Các kiến trúc (Client-server và Peer to Peer) (Slide 7,8)..... | 4 |
| Kiến trúc máy khách-máy chủ | 4 |
| Kiến trúc P2P (ngang hàng) | 4 |
| 2. Socket (Slide 10-16) | 4 |
| Sockets | 4 |
| Xác định tiến trình | 5 |
| Giao thức lớp Ứng dụng: định nghĩa | 5 |
| Dịch vụ vận chuyển nào mà ứng dụng cần?..... | 5 |
| Các yêu cầu dịch vụ vận chuyển: các ứng dụng phổ biến | 6 |
| Các dịch vụ thuộc giao thức vận chuyển trên Internet..... | 6 |
| Ứng dụng Internet: Các giao thức lớp ứng dụng..... | 7 |
| 3. HTTP (Slide 20-27, 30-32, 34-38)..... | 7 |
| Tổng quan HTTP..... | 7 |
| Các kết nối HTTP | 7 |
| HTTP không bền vững..... | 8 |
| HTTP bền vững | 9 |
| Thông điệp yêu cầu HTTP..... | 9 |
| Các phương thức | 9 |
| Thông điệp phản hồi HTTP | 10 |
| Các mã trạng thái phản hồi HTTP | 10 |
| Trạng thái người dùng-máy chủ: cookies..... | 10 |

| | |
|--|-----------|
| Cookies: lưu trữ “trạng thái” | 11 |
| Cookies..... | 11 |
| Web caches (web đệm) (proxy server) | 11 |
| Thông tin thêm về Web caching | 11 |
| 4. FTP các khái niệm (<i>Slide 45, 46</i>)..... | 12 |
| FTP: giao thức truyền tập tin..... | 12 |
| FTP: kết nối điều khiển và kết nối dữ liệu riêng biệt | 12 |
| 5. Giao thức về thư điện tử SMTP, POP (<i>Slide 49-52, 57-59</i>)..... | 12 |
| Thư điện tử..... | 12 |
| Tình huống: Alice gửi thông điệp đến Bob | 13 |
| Các giao thức truy cập Mail..... | 14 |
| Giao thức POP3 | 14 |
| POP3 và IMAP..... | 14 |
| 6. DNS (<i>Slide 61-70</i>)..... | 15 |
| Hệ thống tên miền (DNS: domain name system) | 15 |
| DNS: các dịch vụ, cấu trúc | 15 |
| DNS: cơ sở dữ liệu phân cấp, phân tán | 15 |
| DNS: các máy chủ tên miền gốc | 16 |
| TLD, máy chủ có thẩm quyền | 16 |
| Máy chủ tên miền cục bộ | 16 |
| Ví dụ phân giải tên miền DNS..... | 17 |
| DNS: caching, cập nhật các bản ghi | 17 |
| Các bản ghi DNS..... | 18 |
| 7 .Lập trình Socket (<i>Slide 96-105</i>): Phân biệt được đâu là chương trình viết cho server/client, TCP/UDP)..... | 18 |
| Lập trình Socket | 18 |
| Lập trình Socket với <i>UDP</i> | 19 |
| Sự tương tác socket máy khách/máy chủ: UDP | 19 |
| Ứng dụng ví dụ: UDP máy khách | 19 |
| Ứng dụng ví dụ : UDP server | 20 |
| Lập trình Socket với <i>TCP</i> | 20 |
| Tương tác socket máy khách/máy chủ: TCP | 21 |
| Ứng dụng ví dụ : TCP client | 21 |

| | |
|---|-----------|
| Ví dụ ứng dụng: TCP máy chủ..... | 22 |
| Chương 3..... | 22 |
| 1. Multiplexing và Demultiplexing (Slide 8): Xem ảnh mấy cái port và biết nó đi đâu về đâu..... | 22 |
| Multiplexing/demultiplexing | 22 |
| 2. UDP header và tính checksum (Slide 17-19)..... | 23 |
| UDP: segment header | 23 |
| UDP checksum..... | 23 |
| Internet checksum: ví dụ..... | 23 |
| 3. Các nguyên lý truyền dữ liệu tin cậy : Có khoảng 1-2 câu trong đề thi. Phân biệt được giống và khác nhau giữa các version. Mấy rtd. VD nhìn vô cái hình biết version mấy. Tính toán hiệu suất k cần quan tâm. (Slide 24-43, 50, 54). Bài tập bao nhiêu trạng thái, .. | 24 |
| Truyền dữ liệu tin cậy: bắt đầu | 24 |
| Truyền dữ liệu tin cậy: bắt đầu | 24 |
| rdt1.0: truyền tin cậy trên 1 kênh tin cậy | 24 |
| rdt2.0: kênh với các lỗi..... | 25 |
| rdt2.0: đặc điểm kỹ thuật FSM | 25 |
| rdt2.0: hoạt động khi không lỗi | 26 |
| rdt2.0 có lỗi hỏng nghiêm trọng! | 26 |
| rdt2.1: bên gửi, xử lý các ACK/NAK bị hỏng | 26 |
| rdt2.1: bên nhận, xử lý các ACK/NAK bị hỏng | 27 |
| rdt2.1: thảo luận..... | 27 |
| rdt2.2: một giao thức không cần NAK | 27 |
| rdt2.2: các phần bên nhận và gửi | 28 |
| rdt3.0: các kênh với lỗi và mất mát | 28 |
| rdt3.0 bên gửi..... | 28 |
| Hành động của rdt3.0 | 29 |
| Hiệu suất của rdt3.0..... | 29 |
| rdt3.0: hoạt động “stop-and-wait” | 30 |
| Hoạt động GBN..... | 30 |
| Hành động của lặp lại có lựa chọn..... | 30 |
| 4. TCP header (Slide 58-60) | 31 |
| Cấu trúc segment TCP | 31 |
| Số thứ tự TCP và ACK | 31 |
| Số thứ tự TCP và ACK | 31 |

| | |
|--|-----------|
| 5. Tính toán RTT và Time out. Biết cách tính với công thức (Slide 61-63)..... | 32 |
| TCP round trip time và timeout..... | 32 |
| 6. Tình huống truyền lại (Slide 68-69) | 33 |
| TCP: tình huống truyền lại..... | 33 |
| 7. TCP truyền lại nhanh (Slide 71-72) | 33 |
| TCP truyền lại nhanh..... | 33 |
| 8. TCP điều khiển luồng (Slide 75) | 33 |
| TCP điều khiển luồng | 33 |
| 9. Quản lý kết nối (Slide 80, 83) | 34 |
| TCP bắt tay 3 lần (3-way handshake)..... | 34 |
| TCP: đóng kết nối..... | 34 |
| 10. Điều khiển tắc nghẽn (Slide 99-105) | 34 |
| TCP điều khiển tắc nghẽn: tăng theo cấp số cộng, giảm theo cấp số nhân | 34 |
| TCP điều khiển tắc nghẽn: chi tiết | 35 |
| TCP Slow Start | 35 |
| TCP: phát hiện, phản ứng khi mất gói..... | 35 |
| TCP: chuyển từ slow start qua CA..... | 35 |
| Tóm tắt: TCP điều khiển tắc nghẽn..... | 36 |
| TCP thông lượng (throughput)..... | 36 |
| Chương 4..... | 36 |
| 1. Phân biệt được mạng mạch ảo và mạng mạch gói, mục 4.2 (Slide 11-16)..... | 37 |
| Dịch vụ connection-oriented (hướng kết nối) và connection-less (phi kết nối)..... | 37 |
| Các mạch ảo (Virtual circuits) | 37 |
| Triển khai kết nối ảo (VC) | 37 |
| Bảng forwarding của kết nối ảo..... | 37 |
| Các mạch ảo: các giao thức gửi tín hiệu | 38 |
| Mạng chuyển gói (Datagram network) | 38 |
| 2. So trùng prefix dài nhất (Slide 19) | 39 |
| So trùng phần đầu dài nhất (Longest prefix matching) | 39 |
| 3. IP header (Slide 34) | 39 |
| Định dạng IP datagram | 39 |
| 4. Phân mảnh và tổng hợp (Slide 35-36) | 39 |
| Phân mảnh và tổng hợp IP | 39 |

| | |
|--|----|
| 5. Địa chỉ IP : Các lớp địa chỉ, địa chỉ dành riêng, địa chỉ mạng, địa chỉ host, địa chỉ broadcast, địa chỉ default gateway, chia mạng con. Số câu hỏi cho phần này nhiều. Chiếm 6-8 câu. | 40 |
| 6. Giao thức DHCP, cấp IP động (Slide 45-48) | 40 |
| DHCP: Dynamic Host Configuration Protocol | 40 |
| Ngữ cảnh DHCP client-server | 40 |
| DHCP: cung cấp nhiều thông tin | 41 |
| 7. NAT (Slide 56-59) | 41 |
| NAT: network address translation | 41 |
| 8. Giao thức ICMP (Slide 65-66) | 42 |
| ICMP: Internet Control Message Protocol | 42 |
| Traceroute và ICMP | 42 |
| 9. Thuật toán định tuyến Link-state (Slide 79-83) Mấy độ phức tạp hong cần | 43 |
| Thuật toán routing Link-State | 43 |
| Thuật toán Dijkstra | 43 |
| Thuật toán Dijkstra : ví dụ | 44 |
| 10. Thuận toán Distance vector (Slide 86-92) | 45 |
| Thuật toán Distance vector | 45 |
| Bellman-Ford ví dụ | 45 |
| Thuật toán Distance vector | 45 |
| 11. Định tuyến có cấu trúc (Slide 98-100) | 47 |
| Định tuyến có cấu trúc | 47 |
| Kết nối các AS | 47 |
| Tác vụ Inter-AS | 47 |
| 12. Định tuyến RIP (Slide 106-109) | 48 |
| RIP (Routing Information Protocol) | 48 |
| RIP: ví dụ | 48 |
| RIP: lỗi đường kết nối và phục hồi | 49 |
| 13. OSPF (Slide 111-113) | 49 |
| OSPF (Open Shortest Path First) | 49 |
| Các đặc tính “vượt trội” của OSPF (không có trong RIP) | 49 |
| OSPF phân cấp | 49 |
| Chương 5 | 50 |
| 1. Tầng link thực hiện tại (Slide 8-9) | 50 |

| | |
|---|-----------|
| Tầng Liên kết dữ liệu được triển khai ở đâu? | 50 |
| Các Adaptor liên lạc | 50 |
| 2. Phát hiện lỗi (Slide 11-18): Phát hiện lỗi chẵn lẻ, CRC,... | 50 |
| Phát hiện lỗi..... | 50 |
| Kiểm tra chẵn lẻ (Parity checking) | 51 |
| Internet checksum | 51 |
| Cyclic redundancy check | 51 |
| CRC ví dụ | 52 |
| 3. Giao thức đa truy cập (Slide 21, 24-27, 29, 33, 35, 37, 39, 44) | 52 |
| Các giao thức và kết nối đa truy cập..... | 52 |
| Các giao thức MAC: phân loại | 53 |
| Các giao thức MAC phân hoạch kênh:TDMA | 53 |
| Các giao thức MAC phân hoạch kênh: FDMA..... | 53 |
| Các giao thức truy cập ngẫu nhiên | 54 |
| Slotted ALOHA..... | 54 |
| CSMA (carrier sense multiple access) | 54 |
| CSMA/CD (collision detection) | 54 |
| Thuật toán Ethernet CSMA/CD | 55 |
| Các giao thức MAC “Xoay vòng” | 55 |
| Tổng kết các giao thức MAC..... | 55 |
| 4. Giao thức ARP (Slide 46-56) | 56 |
| Địa chỉ MAC và ARP..... | 56 |
| Địa chỉ LAN(tt) | 56 |
| ARP: address resolution protocol | 57 |
| Giao thức ARP: cùng mạng LAN..... | 57 |
| Addressing: định tuyến tới mạng LAN khác | 57 |
| 5. Chuẩn Ethernet (Slide 63) | 58 |
| Chuẩn Ethernet 802.3: Tầng Liên kết dữ liệu & Tầng Vật lý | 58 |
| 6. Bộ chuyển mạch (Switch) (Slide 66-74) | 59 |
| Switch: nhiều phiên truyền đồng thời | 59 |
| Switch: tự học..... | 59 |
| Switch: lọc/chuyển tiếp frame | 60 |
| Tự học, chuyển tiếp: ví dụ | 60 |

| | |
|--|-----------|
| Kết nối các switch với nhau (Interconnecting switches) | 61 |
| Ví dụ nhiều switch tự học | 61 |
| Mạng của tổ chức | 61 |
| So sánh Switch và Router | 61 |
| 7. VLan (Slide 75-78) | 62 |
| VLANS: trình bày | 62 |
| VLANS | 63 |
| Port-based VLAN..... | 63 |
| VLANS nối nhiều switch | 63 |