 **TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐHQG HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**MÔN: MẠNG MÁY TÍNH**

**CÁI BÌA**

**Sinh viên thực hiện:**

**Trương Nguyễn Anh Hoàng – 1712039**

# Mục lục

[Mục lục 2](#_Toc11887323)

[1. TỔNG QUAN VỀ MẠNG MÁY TÍNH VÀ INTERNET 4](#_Toc11887324)

[1.1. Mạng máy tính/internet và các thành phần 4](#_Toc11887325)

[1.2. Phần lõi của mạng Internet 5](#_Toc11887326)

[1.3. Phần lõi của mạng Internet (tt) 6](#_Toc11887327)

[1.4. So sánh mạng chuyển mạch gói và mạng chuyển mạch mạch 7](#_Toc11887328)

[1.5. Đỗ trễ gói tin, thông lượng mạng – các loại độ trễ 8](#_Toc11887329)

[2. KIẾN TRÚC PHÂN TẦNG – MÔ HÌNH OSI 9](#_Toc11887330)

[2.1 Kiến trúc phân tầng 9](#_Toc11887331)

[2.2 Mô hình OSI + mô hình Internet 10](#_Toc11887332)

[2.3 Mô hình OSI + Mô hình Internet 11](#_Toc11887333)

[3. GIẢI BT CHƯƠNG 1 12](#_Toc11887334)

[4. TẦNG ỨNG DỤNG 15](#_Toc11887335)

[4.1 Kiến trúc ứng dụng 15](#_Toc11887336)

[4.2 Truyền thông giữa các tiến trình (ứng dụng) 16](#_Toc11887337)

[4.3 Một số dịch vụ mà tầng Vận chuyển cung cấp cho Tầng ứng dụng 17](#_Toc11887338)

[4.4 Một số khái niệm khác 18](#_Toc11887339)

[5. WEB VÀ GIAO THỨC HTTP 19](#_Toc11887340)

[5.1 Định nghĩa cơ bản 19](#_Toc11887341)

[5.2 Chi tiết về HTTP – HTTP request 20](#_Toc11887342)

[5.3 Chi tiết về HTTP – HTTP response 21](#_Toc11887343)

[5.4 Cookie 22](#_Toc11887344)

[5.5 Web Caching 23](#_Toc11887345)

[6. GIAO THỨC TRUYỀN TIN FTP 24](#_Toc11887346)

[7. EMAIL VÀ CÁC GIAO THỨC CHO EMAIL 25](#_Toc11887347)

[7.1 Cách thức vận hành của Email hiện đại và giao thức SMTP 25](#_Toc11887348)

[8. DỊCH VỤ DNS 26](#_Toc11887349)

[8.1 Hệ thống tên miền DNS (Domain System Name) 26](#_Toc11887350)

[26](#_Toc11887351)

[9. GIẢI BÀI TẬP TẦNG ỨNG DỤNG (SÁCH trang 182) 27](#_Toc11887352)

[1. Cơ chế DHCP: mô tả ở tầng sau 27](#_Toc11887353)

[2. Quá trình hoạt động của giao thức DNS: 27](#_Toc11887354)

[3. Tại sao nói FTP gửi thông tin điều khiển “out-of-band”: 27](#_Toc11887355)

[4. Phân biệt HTTP thường trực và HTTP không thường trực: 27](#_Toc11887356)

[5. Ứng dụng yêu cầu độ chính xác cao nên dùng TCP hay UDP: 28](#_Toc11887357)

[6. Ứng dụng yêu cầu tốc độ cao nên dùng TCP hay UDP: 28](#_Toc11887358)

[7. Tại sao HTTP, POP3, FTP đều dùng TCP mà không dùng UDP: 28](#_Toc11887359)

[8. HTTP Client cần lấy document từ 1 URL chưa biết IP, cần dùng giao thức nào: 28](#_Toc11887360)

[9. Một site thương mại điện tử muốn lưu trữ thông tin mua hàng bằng các sử dụng cookies, cơ chế và minh hoạ bằng site đó: 29](#_Toc11887361)

[10. Trình bày giao thức bắt tay: Tầng Vận chuyển 29](#_Toc11887362)

[11. Bốn ứng dụng phù hợp P2P: 29](#_Toc11887363)

[+ Phân phối tập tin ngang hàng (BitTorrent) 29](#_Toc11887364)

[12. Các câu đúng sai: 29](#_Toc11887365)

[13. Bảo vệ hệ thống mạng của công ty bị DHCP-Spoofing (giả mạo DHCP): 30](#_Toc11887366)

[+ Cấu hình các cổng trên switch thành 2 kiểu: cổng trust và cổng untrust. 30](#_Toc11887367)

[14. Phân tích gói tin HTTP Request: 30](#_Toc11887368)

[15. Phân tích gói tin HTTP Response: 30](#_Toc11887369)

[10. TẦNG VẬN CHUYỂN 31](#_Toc11887370)

[10.1 Tổng quan về tầng vận chuyển 31](#_Toc11887371)

[10.2 Dồn kênh và phân kênh 32](#_Toc11887372)

[10.3 Dồn kênh và phân kênh phi kết nối/hướng kết nối 33](#_Toc11887373)

[10.4 Dồn kênh và phân kênh phi kết nối/hướng kết nối 34](#_Toc11887374)

[10.5 Giao thức tầng Vận chuyển – giao thức UDP 35](#_Toc11887375)

[10.6 Truyền tin đáng tin cậy và giải pháp – ACK/NAK và thời gian chờ 36](#_Toc11887376)

[10.7 Truyền tin đáng tin cậy và giải pháp – Seq và pipelined 37](#_Toc11887377)

[10.8 Truyền tin đáng tin cậy và giải pháp – Quay lui (Go-back-N) 38](#_Toc11887378)

[10.9 Truyền tin đáng tin cậy và giải pháp – Gửi lại có chọn (Selective Repeat) 39](#_Toc11887379)

[10.10 Minh hoạ về Select Repeat 40](#_Toc11887380)

[10.11 Giao thức vận chuyển hướng kết nối – giao thức TCP 41](#_Toc11887381)

[10.12 Cấu trúc gói tin TCP chi tiết 42](#_Toc11887382)

[10.13 Cơ chế vận hành của truyền dữ liệu bằng giao thức TCP 43](#_Toc11887383)

[10.14 Thiết lập kết nối TCP, cơ chế bắt tay handshaking 3 bước: 44](#_Toc11887384)

[10.15 Kiểm soát tắc nghẽn trong TCP 45](#_Toc11887385)

[11. GIẢI BÀI TẬP TẦNG ỨNG DỤNG (SÁCH trang 273) 46](#_Toc11887386)

## TỔNG QUAN VỀ MẠNG MÁY TÍNH VÀ INTERNET

### Mạng máy tính/internet và các thành phần

+ Mạng internet là một hệ thống thiết bị kết nối với nhau

+ Các thiết bị này gọi là các hệ thống cuối (end system)

+ Liên kết giữa các thiết bị gọi là link

+ Bộ chuyển gói tin là nơi trung chuyển các gói tin giữa các liên kết

+ Mạng internet là mạng giữa các mạng với nhau

+ Các hệ cuối truy cập/gia nhập Internet bằng cách kết nối tới ISP

+ ISP là nhà cung cấp dịch vụ internet, tự bản thân ISP đã là một bộ chuyển/bộ định tuyến/liên kết truyền thông

+ Giao thức truyền là tập hợp các quy tắc chuẩn dùng cho việc biễu diễn dữ liệu, phát tín hiệu, … -> các việc cần thiết để gửi thông tin qua các liên kết truyền thông. Nhờ giao thức mà các thiết bị có thể giao tiếp với nhau

+ Giao thức chính là cách thức truyền dẫn thông tin giữa các hệ cuối

+ Giao thức của yếu của internet hiện đang sử dụng là giao thức TCP và giao thức IP

+ Một số kỹ thuật truy cập internet hiện nay: quay số/dial-up (các thiết bị usb 3G, D-com); DSL (ADSL, v.v); cáp (cáp đồng trục, cáp quang, v.v); Ethernet, Wifi (2 kỹ thuật này giúp kết nối tới bộ định tuyến biên trong mạng cục bộ)

Mạng internet

**Summary:** Mạng internet là một hệ thống rất lớn gồm nhiều thiết bị cuối (máy tính, điện thoại, ..) kết nối với nhau bằng mạng lưới liên kết (link). Chúng giao tiếp với nhau bằng giao thức (protocol). Để giao tiếp chuẩn xác thì cần xác định đúng lộ trình cần đi (route) giữa nơi nhận và gửi (route bao gồm rất nhiều link) vì mạng Internet quá khổng lồ. Giữa các link này nối với nhau bằng bộ chuyển gói tin (router/switch/v.v) để vận chuyển gói tin đi chuẩn xác.

### Phần lõi của mạng Internet

Thành phần mạng lõi

Mạng chuyển mạch mạch

+ Phần lõi của mạng Internet chính là lược bỏ đi các thành phần biên/thiết bị cuối, chỉ giữ lại link và các bộ chuyển gói tin. Đây chính là phần lõi của mạch.

+ Phần biên của mạng (các hệ cuối) chỉ mang trách nhiệm lưu trữ và giao tiếp, xử lý và vận hành mạng Internet chính là phần lõi

+Nếu chỉ chuyển chân phương thì với số lượng gói tin khổng lồ sẽ gây ra tắc nghẽn thông tin -> đề xuất mô hình giải quyết

+ Mạng chuyển mạch mạch là một mô hình được đề xuất để chuyển dữ liệu qua mạng Internet.

+ **Cách vận hành:** tạo một **“mạch”** **(đi qua nhiều link)** từ host gửi đến host nhận. Chiếm giữ mạch đó dành riêng cho 2 host

+ Mạch này tồn tại đến khi kết thúc quá trình giao tiếp, băng thông cố định = 1/n băng thông đường truyền. Với n là số mạch của link

+ Lưu ý: “mạch” là một phần của link; **mạch < link, kênh ⬄ mạch**

+ Có 2 kỹ thuật thực hiện tạo “mạch” trong link đó là TDM và FDM.

+ TDM dành riêng thời gian cho từng kênh trong link, với thời gian rất nhỏ để tạo ra sự “liên tục” trong việc truyền dẫn

+ FDM chia thành nhiều băng tần con với tần số khác nhau tương ứng với từng kênh và truyền đồng thời

-> Sẽ xuất hiện các quãng nghỉ/thời gian chết trong thời gian kết nối nên 2 host **tạm ngưng giao tiếp**

**Summary:** Mạng chuyển mạch mạch: host A giao tiếp với host B -> định tuyến: đi lần lượt qua các link A1,B1,C1,D1,E1 -> khởi tạo mạch/kết nối chiếm giữ toàn bộ các link trên (chỉ dành riêng cho A,B nói chuyện) -> giao tiếp (truyền/nhận tin) -> kết thúc giao tiếp -> giải phóng kết nối

Cùng lúc sẽ có nhiều host giao tiếp với nhau, link sẽ chia nhỏ ra thành nhiều phần thuộc nhiều mạch, các mạch này sẽ khác đường đi.

### Phần lõi của mạng Internet (tt)

Mạng chuyển mạch gói

+ Mạng chuyển mạch mạch được cho là phung phí băng thông, vì sẽ chiếm giữ băng thông trong **toàn bộ thời gian** khi 2 host còn kết nối với nhau, mặc dù trong thời gian đó, sẽ có quãng 2 host ngưng giao tiếp

+ Với hệ thống mạng càng lớn, việc thiết lập “mạch” giữa 2 host sẽ càng phức tạp và khó khăn hơn, vì có quá nhiều request cần xử lý

+ Đề xuất ra mạng chuyển mạch gói.

+ **Cách vận hành:** Phân cắt dữ liệu thành nhiều gói tin

+ Gói tin có thông số gửi nhận, v.v

+ Gửi gói tin đến hàng đợi -> chờ được gửi đi

+ Ở hàng đợi có vô số gói tin thuộc các kết nối khác cũng chờ gửi đi

+ Bộ định tuyến xác định gói tin cần đi đâu và vận chuyển

+ Gói tin di chuyển với tốc độ tối đa của băng thông

+ Nhận gói tin tại host nhận -> ghép gói tin

-> Băng thông lúc nào cũng trong trạng thái làm việc tối đa

-> Không có quãng nghỉ xuất hiện trong băng thông

-> Tối ưu về việc tận dụng gói tin

+ Yếu điểm: Dễ xảy ra tắc nghẽn tại hàng đợi nếu tốc độ truyền từ **host gửi -> bộ chuyển** quá nhanh so với **bộ chuyển -> bộ khác.** Khi đó sẽ có rất nhiều gói tin chờ tại hàng đợi -> hao tốn thời gian chờ

**Summary:** Mạng chuyển mạch gói: host A giao tiếp host B -> chia nhỏ dữ liệu thành gói tin với thông tin nhận gửi đầy đủ -> gửi lên bộ chuyển gần nhất, không quan tâm route, không **kết nối trực tiếp** với host B -> gói tin xếp vào hàng đợi tại bộ chuyển với nhiều gói tin thuộc các kết nối khác, chờ gửi đi -> Khi gửi đi, các bộ chuyển sẽ định tuyến liên tục để gửi tới host B.

Chuyển mạch mạch -> định tuyến trước -> chiếm đường đi -> gửi/nhận -> free đường đi

Chuyển mạch gói -> không định tuyến trước -> xếp hàng chờ -> định tuyến tại các link -> nhận

### So sánh mạng chuyển mạch gói và mạng chuyển mạch mạch

Mạng chuyển mạch mạch

Mạng chuyển mạch gói

**- Ưu:** Ổn định với tốc độ được đảm bảo, khó xảy ra vấn đề khi truyền tải/giao tiếp

**- Khuyết:** Cấp phát bất chấp nhu cầu, không tối ưu băng thông – có thời gian nghỉ/thừa -> phung phí băng thông, phức tạp khi hệ thống quá lớn

**- Ưu:** Tận dụng tối đa băng thông vì sử dụng băng thông theo nhu cầu sử dụng, đơn giản khi triển khai với qua mô lớn.

- **Khuyết:** Độ biến động cao, thiếu sự ổn định hơn so với mạng chuyển mạch mạch, dễ bị tắc nghẽn băng thông, **độ trễ hàng chờ không dự đoán trước được** (sẽ đề cập ở phần sau).

-> Mạch chuyển mạch gói hiệu quả hơn ở đa số tình huống trong thời điểm hiện tại

-> Internet vận hành dựa trên **mạng chuyển mạch gói**

Dễ thấy, các cuộc gọi VoIP (Mess, Zalo, v.v) kém ổn định hơn hẳn so với cuộc gọi điện thoại. Bù lại chi phí thì lại rẻ hơn nhiều vì sử dụng tài nguyên hợp lý hơn.

**Summary:** Tuỳ thuộc vào tình huống cụ thể -> lựa chọn mạng chuyển mạch cho phù hợp. Cả 2 mạng đều vẫn còn phổ biến trong các hệ thống mạng viễn thông. Mạng internet dựa trên mạng chuyển mạch gói.

### Đỗ trễ gói tin, thông lượng mạng – các loại độ trễ

Độ trễ gói tin

Độ trễ tại nút (dproc)

Độ trễ hàng chờ (dqueue)

Độ trễ đường truyền (dtrans)

Độ trễ lan truyền (dprop)

Tổng độ trễ tại nút

Độ trễ giữa 2 nút mạng

Thông lượng mạng

- Là thời gian gửi từ host gửi đến host nhận trong mạch gói

- Gồm rất nhiều loại độ trễ khác nhau tổng hợp

- Tại mỗi nút mạng, gói tin sẽ được xử lý (kiểm tra xem có bị lỗi không, xác định header, xác định port, v.v) -> thời gian xử lý tại nút

+ Mỗi gói tin tại hàng chờ phải chờ đến lượt mới được gửi đi, thời gian chờ này là độ trễ hàng chờ, không cố định, rất nhỏ

+ Thời gian đẩy gói tin L bits lên đường truyền có tốc độ R bits thì có dtrans = L/R

+ Thời gian lan truyền từ sau khi đi ra khỏi bộ định tuyến A đến bộ định tuyến B với **tốc độ truyền s** và **khoảng cách M** là dproc = M/s

+ dnodal = dproc + dqueue + dtrans + dprop

+ dend-end = N\*(dproc + dtrans + dprop) với N-1 bộ định tuyến giống nhau nằm ở giữa

Thông lượng mạng là lượng dữ liệu thực tế, throughput < bandwidth

**Summary:** Sử dụng mạng chuyển mạch gói/Internet sẽ có hiện tượng trễ gói tin, thậm chí mất gói tin nếu hàng chờ đầy.

## KIẾN TRÚC PHÂN TẦNG – MÔ HÌNH OSI

### Kiến trúc phân tầng

+ Xem mạng như là một chồng “sách”, mỗi “quyển sách” là 1 tầng

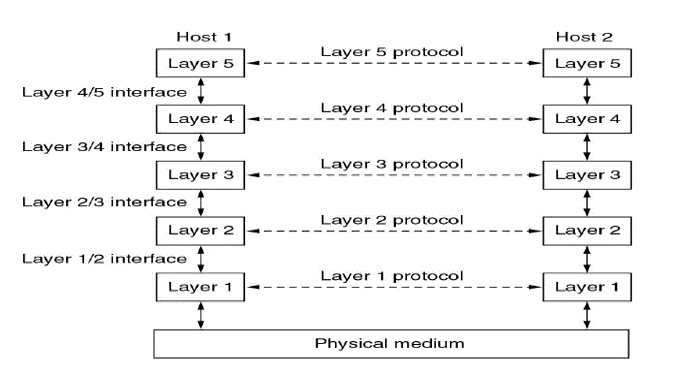
+ Lớp thứ N sẽ cung cấp dịch vụ cho lớp thứ N+1

+ Mỗi lớp sẽ có giao thức riêng để giao tiếp với nhau

+ Dữ liệu sẽ được xử lý từ tầng cao xuống tầng thấp rồi đi lên ngược trở lại, từ tầng thấp đế tầng cao

+ Ưu điểm: Đơn giản, giảm độ phức tạp khi xử lý yêu cầu (chia nhỏ công việc cho từng tầng). Dễ quản lý và phát triển

+ Minh hoạ giao tiếp giữa 2 Host 1 và Host 2



Kiến trúc phân tầng

**Summary:**

### Mô hình OSI + mô hình Internet

+ Cung cấp dịch vụ mạng cho end-user sử dụng

+ Giao thức tiêu biểu: HTTP (web), SMTP ( gửi mail), FTP (tập tin), POP3 (nhận mail), IMAP (nhận header-mail)

+ Gói tin ở tầng ứng dụng nhằm trao đổi thông tin giữa các ứng dụng với nhau. Gói tin ở tầng này được gọi là **“message”**

+ Quy mô giao thức: giữa các ứng dụng với nhau

+ Hỗ trợ các dịch vụ mạng thông dịch dữ liệu **tức** nén dữ liệu, mã hoá dữ liệu, mô tả dữ liệu, … (Tầng Trình bày(N) **cung cấp dịch vụ** cho tầng Ứng dụng (N+1))

+ Tầng này giúp đồng nhất các dạng biễu diễn dữ liệu để giúp ứng dụng dễ dàng trong việc trao đổi dữ liệu

+ Hỗ trợ phân định ranh giới các phiên của ứng dụng, đồng bộ hoá việc trao đổi dữ liệu giữa các bên tham gia

+ 2 tầng trên không nằm trong mô hình Internet hiện tại

+ Tuỳ thuộc vào người lập trình ứng dụng có thể cài đặt thêm 2 chức năng tương ứng 2 tầng trên vào máy.

+ Quan trọng hay không tuỳ nhu cầu và tình huống cụ thể

Tầng Ứng dụng

Tầng Trình bày

Tầng Phiên

Về tầng Trình bày và tầng Phiên

**Summary:** Tầng Ứng dụng là tầng gần gũi nhất với người dùng hiện tại, các giao thức ở tầng ứng dụng nhằm cung cấp dịch vụ mạng.

### Mô hình OSI + Mô hình Internet

+ Giao tiếp giữa các nút gửi/nhận (end system) nhưng không tương tác giữa các nút trung gian. (end-to-end).

+ Gồm 2 giao thức chính là TCP và UDP.

**+ Xác minh địa chỉ cần gửi và cách thức truyền gói tin đi**

+ Các gói tin ở tầng này gọi là **“segment”**

+ Quy mô giao thức: Giữa 2 end-user ở biên (chỉ gửi/nhận)

**+ Định tuyến cho các gói tin**

+ Giao thức: IP và các giao thức định tuyến

+ Gói tin ở tầng này gọi là **“datagram”**

+ Quy mô giao thức: Toàn bộ đường đi của gói tin (route)

**+ Chuyển các “datagram” đi qua nút kế tiếp trên đường đi**

+ Giao thức: Ethernet, Wifi. PPP. v/v

+ Gói tin ở tầng này gọi là **“frame”**

+ Quy mô giao thức: Nội bộ đường truyền (link)

**+ Cung cấp phương thức truy cập vào đường truyền vật lý**

+ Cung cấp các chuẩn về điện, dây cáp, đầu nối, v,v

Tầng Giao vận

Tầng Network

Tầng Data-link

Tầng Vật lý

**Summary:** Quy mô của các giao thức tương ứng sẽ nhỏ dần và càng ngày càng chi tiết hơn: app giao tiếp với app -> các máy trong cùng app giao tiếp với nhau -> các nút đầu – nút cuối -> các nút trung gian giao tiếp với nhau -> dữ liệu trong link được truyền đi.

## GIẢI BT CHƯƠNG 1

1. **Ưu khuyết mạng chuyển mạch mạch so với mạng chuyển mạch gói:**

+ Ưu: Ổn định, tốc độ được đảm bảo, khó xảy ra sự cố truyền dẫn

+ Khuyết: chưa tối ưu cho việc cấp băng thông, cấp băng thông không theo nhu cầu, phung phí băng thông

1. **Sự khác nhau giữa OSI và TCP/IP**

+ OSI là mô hình mang tính lý thuyết, chưa có hạ tầng cụ thể, còn TCP/IP với 2 giao thức chủ chốt là mô hình thực sự của Internet

+ OSI có 7 tầng, TCP/IP có 5 tầng

1. **Bộ định tuyến (router) trên mạng:**

+ Bộ định tuyến trên mạng thực hiện chức năng của tầng Network (Mạng) trong mô hình TCP

+ Tầng này có chức năng định tuyến đường đi (route) cho gói tin, sau khi gói tin đã nhận được địa chỉ gửi/nhận từ tầng Transport

1. **Sự khác nhau của virus và worm:**

+ Virus và worm đều có khả năng tự nhân bản

+ Worm có thể tự nhân bản trên chính nó mà không cần vật chủ (file cụ thể)

+ Virus phải cấy vào một file nào đó và phát tán

+ Worm lây lan qua lỗ hổng bảo mật, phụ thuộc vào lỗ hổng, không quan trọng người dùng (nếu lỗ hổng chưa bị fix)

+ Virus thường lây lan thông qua các tập tin, đi theo sự bất cẩn của người dùng

1. **Cách mạng botnet được tạo ra và cách sử dụng để tấn công DDoS:**

+ Tin tặc phát tán virus và worm, các máy tính bị nhiễm tiếp tục lây lan

+ Các máy tính sau khi bị nhiễm, sẽ có bot tự động chiếm quyền control máy

+ Cách chiếm quyền có thể log vào 1 server điều khiển từ xa

+ Các máy cùng log vào server tạo thành một mạng và hoạt động dựa theo lệnh của server điều khiển

+ DdoS là dùng hệ thống máy tính này tấn công vào máy nạn nhân bằng cách gửi rất nhiều gói tin hoặc thiết lập rất nhiều kết nối, làm máy tính bị tổn thương

1. **5 giao thức thuộc tầng Ứng dụng:** HTTP, FTP, SMTP, IMAP, POP3, ..
2. **Phân biệt TDM và FDM:**

+ TDM là kỹ thuật ghép kênh bằng cách phân chia thời gian ra cho các kênh với mỗi kênh là một thời gian nhất định, kết thúc thời gian sẽ tới phiên kênh khác truyền.

+ FDM là kỹ thuật ghép kênh bằng cách chia băng tần đường truyền thành nhiều dải con, mỗi kênh được truyền trên dải con đó

+ TDM truyền ngắt quãng, cách mỗi khe thời gian lại tiếp tục truyền

+ FDM chia nhỏ băng tần và truyền song song các kênh cùng 1 lúc

1. **Gửi từ host nguồn đến host đích theo 1 đường đi cố định sẽ có:**

+ Độ trễ xử lý tại nút

+ Độ trễ đẩy lên đường truyền

+ Độ trễ lan truyền

+ Độ trễ biến thiên duy nhất là độ trễ hàng chờ

1. **Thời gian lan truyền 1 gói tin 1000 byte :**

+ Độ trễ lan truyền: 2 500 000/2.5 .10^8 = 0.1 s

+ Độ trễ đẩy lên đường truyền: 8000/ 2000000 = 0.04s

+ Tổng thời gian = 0.14s

1. **Nhiều user cùng chia sẻ đường truyền tốc độ 2Mbps:**
   1. Nếu dùng chuyển mạch mạch: nếu có n mạch thì mỗi mạch có băng thông là 1/n băng thông đường truyền. Mà tối thiểu cần 1 Mbps => chỉ có 2 user
   2. Nếu < 2 user thì băng thông tối đa một lúc là 2 Mbps, vừa đủ, không phải xếp hàng đợi

Nếu > 3 user thì băng thông tối đa một lúc phải chịu tả là >2 Mbps -> phải xếp hàng đợi. Vì một lúc băng thông 2 Mbps không thể truyền hết gói tin đi

1. **Host A gửi tập tin có kích thước lớn cho host B, đi qua 3 liên kết khác nhau:**
   1. Tính thông lượng truyền file = tốc độ nhỏ nhất = 500 kbps. Giải thích: muốn truyền file đi qua bao nhiêu liên kết thì **toàn bộ bytes (dữ liệu của file) vẫn phải đi qua link có tốc độ chậm nhất**.
   2. 4tr byte = 32 000 Kbit = 32000/500 = 64s
   3. Thông lượng còn 100 kbps, thời gian -> 320s
2. **Một ứng dụng truyền đi trên mạng với tốc độ ổn định , chạy trong thời gian dài:**
   1. Mạng chuyển mạch gói thích hợp hơn, vì ứng dụng chạy trong thời gian dài, nên sẽ **chiếm băng thông trong thời gian rất dài** nếu dùng mạng chuyển mạch mạch, gây phung phí băng thông với các quãng nghỉ không truyền dữ liệu. Sử dụng mạng chuyển mạch gói có khuyết điểm là tốc độ biến động gây tắc nghẽn nhưng **ứng dụng này có tốc độ ổn định** nên không ảnh hưởng nhiều.
   2. Không cần điều khiển tắc nghẽn vì tổng tốc độ truyền ứng dụng < tốc độ truyền của tất cả kết nối, nên sẽ không có hàng đợi xuất hiện -> không tắc nghẽn
3. **Dạng bài tập tính toán chính của chương một:**

Độ trễ gói tin:

- Cần quan tâm các giá trị sau: dnodal = dproc + dqueue + dtrans + dprop

- **dproc** là độ trễ xử lý tại nút

- **dqueue** là thời gian xếp hàng chờ

- **dtrans** là thời gian đẩy gói tin từ hàng chờ lên đường truyền

- **dprop** là độ trễ lan truyền

- Thứ tự: **dproc -> dqueue -> dtrans -> dqueue**

- Khác ở phần thông lượng, thời gian truyền = max(thời gian truyền qua các liên kết), ở đây các bước phải lần lượt hoàn thành mới thực hiện bước tiếp theo. Vì vậy thời gian bằng tổng các thời gian

- **Mps** là tốc độ megabit/s, để dùng đơn vị này, ta đổi bytes thành bits

- Tại thời điểm t = x, bit đầu tiên ở đâu thì lần lượt so với các độ trễ để biết. Khi so sánh x và **dtrans,** cẩn thận trường hợp bit đầu tiên đã đến đích (**dtrans > dprop**)

- Thường thì **dqueue** sẽ không được xét tới

- Nếu n gói tin vào hàng đợi thì không xét thời gian xử lý tại nút nữa

- Nếu gửi n gói tin thì phải xử lý tại nút cho cả 5 gói tin

- Nếu ở giữa 2 host có nhiều liên kết thì phải xét từng đường liên kết

- Nếu các đường liên kết giống nhau thì vận dụng công thức:

dend-end = N\*(dproc + dtrans + dprop) với N-1 nút ở giữa

## TẦNG ỨNG DỤNG

### Kiến trúc ứng dụng

+ Là mô tả cấu trúc vận hành ứng dụng

+ Kiến trúc ứng dụng không phần tầng, chia làm 2 loại chính là : server-client và peer-to-peer

+ Chia toàn bộ end-system của ứng dụng thành: server và client

+ Server là các máy cố định, luôn luôn hoạt động trên ứng dụng

+ Server chịu trách nhiệm cung cấp dịch vụ cho các máy khác, các máy này gọi là client, cũng là máy của user

+ Giữa các máy client không liên lạc được với nhau, chỉ có thể giao tiếp trung gian thông qua máy chủ

+ Một số dịch vụ điển hình: Web, Email (vẫn có dịch vụ mail P2P nhưng không đáng kể), FTP, DHCP, DNS, v.v

+ Client-server hao tốn nhiều chi phí vận hành và bảo trì hơn P2P

+ Mô hình peer-to-peer không chia làm server và client, tất cả các máy đều ngang hàng với nhau

+ Truyền thông trực tiếp giữa các cặp máy kết nối với nhau

+ Mạng ngang hàng mang tính chất **phân tán,** dễ dàng mở rộng.

+ Kiến trúc ngang hành có độ bảo mật kém hơn client-server

+ Một số dịch vụ P2P điển hình: torrent, Skype, IPTV, v.v

Kiến trúc ứng dụng

Kiến trúc khách chủ

Kiến trúc ngang hàng

**Summary:** Tuỳ vào nhu cầu sử dụng mà người lập trình ứng dụng đề xuất ra kiến trúc cho ứng dụng. Như chúng ta có thể thấy, kiến trúc ứng dụng chỉ quan tâm đến **mô hình giao tiếp (đề cao về cấu trúc, về quan hệ giao tiếp)** giữa các ứng dụng, giữa các end-system với nhau, chứ không quan tâm đến việc chúng **giao tiếp cụ thể với nhau như thế nào**, giao tiếp như thế nào là nhiệm vụ của các tầng phía dưới.

### Truyền thông giữa các tiến trình (ứng dụng)

+ Tiến trình là một phần của ứng dụng đang chạy trên máy

+ VD: browser đang mở nhiều tab, mỗi tab truy cập 1 site khác nhau Có n tab -> tổng cộng có n tiến trình Web-client đang chạy

+ Tiến trình thực ra là “chương trình” đang chạy trên máy tính

+ Vì ứng dụng ngày ngay tích hợp và xử lý quá nhiều chương trình cùng lúc nên thuật ngữ tiến trình ra đời

+ Phục vụ giao tiếp giữa 2 tiến trình trên 2 máy khác nhau

+ Bất kỳ tiến trình nào giao tiếp với nhau cũng cần **thông qua tầng mạng**, tiến trình **gửi/nhận các thông điệp từ mạng** gọi là **socket .**

+ Socket là một **tiến trình riêng biệt** không liên quan đến các tiến trình cung cấp dịch vụ cho end-user của ứng dụng

+ Socket là interface hỗ trợ giữa tầng Ứng dụng và tầng Vận chuyển

+ Socket có thể giúp người lập trình ở tầng Ứng dụng quản lý một số thuộc tính ở tầng Vận chuyển như: lựa chọn giao thức vận chuyển (**UDP/TCP**), cài đặt buffer\_size, max\_segment\_size, v.v (**như 1 filter**)

+ Mô hình hoạt động: **+** Tức là **tiến trình end-user <-> socket nhận data <-> giao tiếp ở tầng mạng <-> socket gửi data <-> tiến trình end-user**

Tiến trình

Socket

**Summary:** Vậy ta hiểu socket như một tiến trình trung gian giúp kết nối giữa tầng Ứng dụng và tầng Giao vận. Tiến trình này giúp người ở Tầng Ứng dụng thiết đặt một số thông số kết nối cho tầng Giao vận

### Một số dịch vụ mà tầng Vận chuyển cung cấp cho Tầng ứng dụng

Yêu cầu dịch vụ từ tầng Ứng dụng

Dịch vụ TCP (Cụ thể ở phần sau)

Dịch vụ UDP (Cụ thể ở phần sau)

+ **Truyền dữ liệu đáng tin cậy** (không mất gói tin, không lỗi dữ liệu, đảm bảo đúng thứ tự, truyền đúng địa chỉ)

+ Đảm bảo **thông lượng(bandwidth)** tối thiểu nào đó

+ Thời gian ( đảm bảo gửi tin trong một giới hạn thời gian nhất định, đặc biệt cần thiết cho các ứng dụng real-time)

+ Mỗi ứng dụng khác nhau sẽ có các yêu cầu/tiêu chuẩn khác nhau, dựa vào 4 tiêu chí trên để lựa chọn dịch vụ

+ Cung cấp dịch vụ hướng kết nối: (sẽ nói rõ hơn ở phần sau)

+ Trao đổi thông tin điều khiển cho nhau (qua giao thức handshaking) -> trao đổi message đồng thời (2 chiều) -> huỷ kết nối

+ Vì có thủ tục bắt tay (handshaking), nên đảm bảo thông tin sẽ được gửi đúng địa chỉ. Thủ tục bắt tay được trình bày ở **tầng Vận chuyển**

+ TCP còn cung cấp dịch vụ truyền dữ liệu đáng tin cậy, dựa vào TCP, các socket có thể truyền gói dữ liệu chính xác, đúng thứ tự

+ Hỗ trợ kiểm soát tắc nghẽn

+ Dịch vụ UDP chỉ có nhiệm vụ duy nhất là truyền dẫn thông tin

+ Không có thủ tục bắt tay trước khi trao đổi thông tin

+ Không cung cấp dịch vụ đảm bảo gói tin sẽ gửi đúng địa chỉ

+ Không đảm bảo các gói tin sẽ được gửi đúng thứ tự lúc gửi

+ Không hỗ trợ kiểm soát tắc nghẽn, gửi tin với tốc độ tuỳ ý

-> Vì không phải kiểm soát và điều khiển nhiều nên nhanh hơn TCP

**Summary:** Dựa vào nhu cầu cần thiết mà lựa chọn dịch vụ TCP hay UDP, ngày nay, phần lớn các ứng dụng đều sử dụng dịch vụ TCP

### Một số khái niệm khác

+ 2 giao thức UDP và TCP không có chức năng mã hoá dữ liệu

+ Phát triển thêm tính năng SSL: mã hoá, toàn vẹn dữ liệu, chứng thực đầu cuối (gửi/nhận)

+ Tiến trình ứng dụng ->(gửi dữ liệu) SSL Socket để mã hoá dữ liệu -> gửi đến TCP Socket -> Internet -> TCP Socket nhận -> SSL Socket nhận (giải mã) -> bên nhận

+ Định danh tiến trình

+ Giúp máy tính phân biệt các tiến trình với nhau

+ Ngoài port mặc định, có thể định danh cho tiến trình bằng các port free

+ Ví dụ: port 80 -> HTTP; port 67,68 -> DHCP; port 53 -> DNS

Bảo mật cho UDP/TCP

Port

**Summary:**

## WEB VÀ GIAO THỨC HTTP

### Định nghĩa cơ bản

+ Cơ bản

+ Web là một dịch vụ mạng có cấu trúc client-server

+ Là giao thức tầng Ứng dụng dành riêng cho dịch vụ Web

+ Port sử dụng: 80 cho server; tuỳ ý cho client

**- Giao thức HTTP có nhiệm vụ:**

+ Định nghĩa cấu trúc **“message”** (nôm na: define message\_struct)

+ Định nghĩa cách thức mà client và server giao tiếp

**- Cách vận hành:**

+ Sử dụng TCP làm giao thức truyền thông ở tầng Vận chuyển

+ Client truy cập web: tạo Socket -> tạo kết nối TCP -> gửi request -> server nhận request -> trả response -> gửi về bằng TCP -> nhận response -> kết thúc

+ Server không quan tâm đến trạng thái của client, chỉ nhận request và trả về response -> HTTP là **giao thức phi trạng thái**

+ Bài tập tầng ứng dụng, câu 5

Web

Giao thức HTTP

+ Kết nối thường trực và không thường trực

**Summary:** Giao thức HTTP là một giao thức sử dụng trong dịch vụ Web, giao thức này hoạt động nhằm giao tiếp giữa server và client. HTTP là một giao thức phi trạng thái -> có nguy cơ bị DdoS, load lại nhiều lần 1 tài nguyên, v.v

### Chi tiết về HTTP – HTTP request

Cấu trúc message **request**

**Minh hoạ**

+ HTTP request được gửi từ client -> webserver

+ Gồm 2 phần: **header request** và **body request**

**- Header request gồm request line và header line**

+ Request line = Method + URL + ver

+ **Method**: GET (yêu cầu nhận dữ liệu), POST(yêu cầu gửi dữ liệu), HEAD (chỉ lấy header), PUT/DELETE (thêm/xoá đối tượng ở server),…

+ Ver là phiên bản HTTP đang sử dụng (1.0, 1.1, v.v)

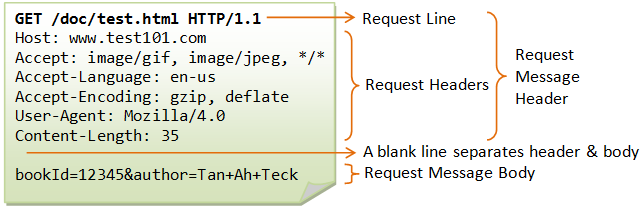
+ Các dòng tham số (header line) cho biết thông tin chi tiết về client, yêu cầu của client đối với web server. Phục vụ cho một số dịch vụ cần thiết khác như proxy hay caching.

+ Về HTTP header: thông tin thêm về chức năng của các dòng tham số tham khảo tại

**- Body request:**

+ Đối với GET sẽ là trống

+ Đối với POST sẽ chứa các tham số người dùng nhập vào để gửi lên dữ liệu. POST tương tự GET nhưng nội dung trả về phụ thuộc vào tham số trong body request, còn GET là cố định



**Summary:** Gói tin request được client gửi đi có nhiệm vụ thông báo nhu cầu của client **(method)** và tiêu chuẩn cụ thể của client (**Accept-Language, Accept-Charset**..); thông tin về client (**From, User-Agent, Range**, v.v); một số điều kiện kiểm soát khác phục vụ cho các tính năng như proxy, cache/browser-cache/proxy-cache, v.v như **(If-None-Match, If-Modified-Since,…)**

### Chi tiết về HTTP – HTTP response

Cấu trúc mess **response**

Minh hoạ

+ HTTP response được gửi từ webserver – client sau khi nhận request

+ Gồm 3 phần: **status line, header line**  và entity body

**- Status line =** ver + status code + message code

+ (Status code; message code) là 1 cặp thể hiện tình trạng phản hồi của máy chủ.

+ Có nhiều tình trạng phản hồi khác nhau như:

- 200 OK: yêu cầu hợp lệ và thông tin yêu cầu được trả về

- 400 Bad Request: Không hiểu được yêu cầu

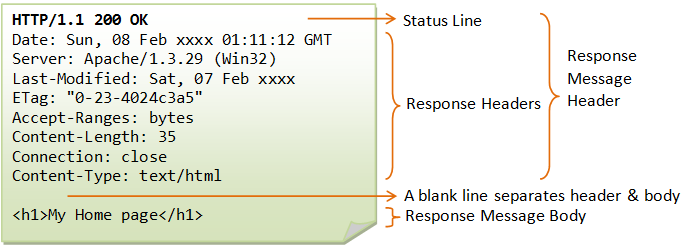
- 404 Not Found: Không tìm thấy đối tượng được yêu cầu

…

+ Tương tự như header line của message request, header line của response cũng cung cấp nhiều loại thông tin khác nhau (thông tin về server, thông tin cung cấp cho các dịch vụ cache, proxy, v.v)

**- Entity body:**

+ Chứa nội dung đối tượng được yêu cầu



**Summary:** Gói tin response trả lời gói tin request, cung cấp các thông tin gói tin request yêu cầu. Ngoài các thông số về server **(Server; Connection; Date, v.v)**, gói tin response còn chứa các thông tin điều khiển cho quá trình cache và một số dịch vụ khác tương ứng như header line ở message request (**Last-Modified; Etag; Content-Length;..).**

### Cookie

Cookie

+ Cookie là một kỹ thuật định danh và xác định trạng thái client

+ HTTP là một giao thức phi trạng thái, tuy nhiên nhu cầu thực tế phát sinh server muốn quản lý trạng thái hiện tại của client (on/off, tương tác sản phẩm nào,..) -> kỹ thuật cookie ra đời

**- Cách vận hành:**

+ Trình duyệt của end-user (client) có một tập cookies

+ Tập cookies này có cặp giá trị đặc trưng là ( ID ; site tương ứng)

+ Server có một CSDL quản lý cookies -> người dùng

+ Khi truy cập lần đầu/đăng nhập, server tạo cookies ID tương ứng cho người dùng -> lưu vào CSDL (ID này là unique trong CSDL)

+ Server gửi thêm dòng **“Set-cookies: ID”** báo cho trình duyệt web

+ Trình duyệt thêm vào tập cookies: **(ID; site)**

+ Lần sau trình duyệt truy cập/đăng nhập vào thì sẽ báo cho server biết ID của mình. Server từ đó có thể tìm kiếm dữ liệu của người dùng mang ID đó mà trả về tham số chỉ thuộc về người dùng

**- Ứng dụng:**

+ Quản lý các phiên đăng nhập, thông tin thuộc về tài khoản

+ Quản lý thông tin giỏ hàng (web mua sắm)

+ Nâng cao trải nghiệm người dùng (gợi ý mua sắm, auto-fill thông tin,..

**Summary:** Cookies là một **kỹ thuật** được cài đặt thêm vào để **định danh người dùng** và xử lý các thông tin đặc thù thuộc về người dùng đó. Mỗi người dùng sẽ được tạo 1 mã ID và khai báo mã ID đó lên server để định danh bản thân.

### Web Caching

Web cache

+ Web cache là một dạng proxy server

+ Web cache là trung gian giữa client và server

+ Web cache vừa có thể là client vừa có thể là server

**-** **Tác dụng của web cache:**

+ Giảm thời gian đáp ứng yêu cầu (load web) đối với client

+ Giảm áp lực tải/lưu lượng truy cập đối với server

**- Cách vận hành:**

+ Web cache lưu trữ entity body tương ứng của các url vào bộ nhớ cache. (Tạo ra bản sao khi nhận response từ server)

+ Web cache nhận request từ client

+ Web cache gửi response lên server

+ Web cache kiểm tra các thông tin điều khiển (kiểm tra xem bản sao còn sử dụng được không, hay đã bị thay đổi)

+ Nếu cache còn sử dụng được, không load body response từ server mà dùng bộ nhớ cache có sẵn để gửi về client.

+ Nếu cache hết hạn hoặc chưa có cache -> cập nhật/tạo lại cache

**- Các tham số điều khiển việc cache:**

+ Last-Modified và If-modified-since:

+ Etag và If-none-match

+ Cache-control

+ Status code cho phép cache: 304 Not Modified

**Summary:** Web cache như một bộ nhớ tạm để lưu trữ lại các trang web cần thiết để tiết kiệm thời gian load cho sau này. Việc này giúp tăng tốc load site cho client và giảm tải cho server. Nhưng phải kiểm soát và điều khiển bộ nhớ cache chuẩn xác. Tránh thời hợp load ra site sai lệch với site mà server cung cấp (do đối tượng đã bị chỉnh sửa)

## GIAO THỨC TRUYỀN TIN FTP

+ Tác dụng là truyền tập tin giữa 2 user

+ Port sử dụng: 21 và 20

+ Tương đồng với giao thức HTTP

+ Quản lý và xác minh người dùng bằng user/pass

+ Có ít nhất 2 kênh TCP song song: 1 để điều khiển (control, dùng port 21); 1 để truyền tập tin (dùng port 20)

+ Mỗi tập tin truyền sẽ có 1 kênh truyền tập tin riêng biệt

+ Chỉ có một kênh điều khiển duy nhất trong suốt phiên sử dụng

+ Kênh điều khiển liên tục duy trì và kiểm tra trạng thái của client

-> Giao thức FTP quan tâm tới trạng thái của client

+ FTP client định danh và tạo control-connection đến FTP server

+ FTP server trả về response + status code

+ FTP client có thể gửi request tương ứng để lấy tập tin/chỉ mục/v.v từ FTP server

Giao thức FTP

Điểm khác biệt so với giao thức HTTP

Cách thức vận hành

**Summary:** Giao thức FTP sử dụng để truyền tin cũng có cách thức vận hành/ message/ kết nối tương đồng với giao thức HTTP. Điểm khác biệt là HTTP phi trạng thái còn FTP quan tâm trạng thái. Ngoài ra FTP sử dụng nhiều kênh TCP song song, trong đó có 1 kênh để control. Còn HTTP gói tất cả vào 1 kênh TCP duy nhất

## EMAIL VÀ CÁC GIAO THỨC CHO EMAIL

### Cách thức vận hành của Email hiện đại và giao thức SMTP

+ Có 4 thực thể, minh hoạ bằng giao tiếp giữa A và B qua email

[Trình xem thư A – máy chủ A – máy chủ B – trình xem thư B]

+ Máy chủ là hộp thư của user

+ Trình xem thư có thể là ứng dụng mail/trình duyệt web/v.v

+ Để gửi mail dùng giao thức SMTP hoặc SMTP + HTTP

+ Để đọc mail từ hộp thư dùng giao thức POP3/IMAP/HTTP

+ Đọc thư có thể truy cập trực tiếp vào máy chủ A hoặc B để đọc thư, nhưng với lượng user lớn và sử dụng máy chủ chung như hiện nay, truy cập trực tiếp sẽ xảy ra nhiều vấn đề -> giao thức truy cập mail

+ POP3 là giao thức đọc mail tải toàn bộ mail xuống và là giao thức 1 chiều, trong khi đó IMAP chỉ get header và là giao thức 2 chiều

+ Giao thức SMTP là giao thức để truyền thư từ user -> máy chủ người gửi và từ đó tới -> máy chủ người nhận

+ Port sử dụng : 25

+ SMTP khách tạo kết nối TCP tới SMTP chủ qua port 25

+ Thiết lập kết nối thành công, 2 máy trao đổi thông tin điều khiển

+ Sau khi hoàn tất quá trình giới thiệu và bắt tay (handshaking), SMTP khách có thể bắt đầu gửi thư cho SMTP chủ

+ Không như FTP, muốn gửi thư khác, có thể dùng lại kết nối TCP, không cần tạo lại

+ Sau khi không còn thư để gửi – đóng kết nối.

Email hiện đại

Giao thức SMTP

Cách thức vận hành

**Summary:** Giao thức SMTP dùng để gửi thư từ máy chủ này sang máy chủ khác hoặc gửi từ user -> máy chủ của user đó. Phần sau sẽ xem xét về giao thức truy cập để xem thư (từ máy chủ của user -> user)

## DỊCH VỤ DNS

### Hệ thống tên miền DNS (Domain System Name)

Tên miền (domain)

Dịch vụ DNS

Cách hoạt động của dịch vụ DNS

+ Có 2 cách để định danh 1 máy tính trên Internet: domain và IP

+ Domain gần gũi và dễ sử dụng với người dùng

+ IP cần thiết cho các router và thiết bị

+ Dịch vụ DNS có tác dụng: dịch domain thành địa chỉ IP

+ Dịch vụ này cung cấp để phục vụ các giao thức tầng Ứng dụng như HTTP, SMTP, FTP

+ Port sử dụng: 53

+ Loại kết nối truyền tải: UDP

+ Bắt buộc phải có ánh xạ từ (domain -> IP) trong CSDL

+ Tránh trường hợp Internet bị ngưng trệ khi máy chủ DNS có vấn đề -> thiết kế máy chủ DNS theo mô hình phân tán

+ DNS còn được thiết kế theo mô hình phân cấp

+ Máy chủ root -> các máy chủ đuôi (.com, .org,. edu, .vn,..) -> máy chủ chính (google.com, wapvip.org, v.v) -> máy chủ con (trans.google.com…)

+ Ứng dụng cần dịch domain -> IP, kích hoạt DNS, dịch vụ DNS gửi truy vấn lên mạng, tới máy chủ DNS -> phản hồi IP.

+ Cách thức hoạt động chi tiết -> BT tầng ứng dụng, câu 2

+ Máy chủ DNS thường là máy UNIX chạy phần mềm BIND

+ Thông tin thêm về DNS -> trang 149

### 

**Summary:** Dịch vụ DNS là dịch vụ tối cần thiết cho các giao thức và dịch vụ mạng khác ở tầng Ứng dụng. DNS được thiết kế theo mô hình phân tán và phân cấp. Khi một máy chủ DNS bị lỗi chỉ xảy ra lỗi cục bộ tại vị trí đó là các địa chỉ phụ thuộc

## GIẢI BÀI TẬP TẦNG ỨNG DỤNG (SÁCH trang 182)

### Cơ chế DHCP: mô tả ở tầng sau

### Quá trình hoạt động của giao thức DNS:

+ Yêu cầu truy vấn, liên lạc tới DNS local

+ Kiểm trong trong dns-cache của client

+ Nếu không thấy, DNS local gửi request tới DNS root gần nhất hoặc được set

+ DNS root trả lời lại IP của Top-level-DNS (máy chủ đuôi .com, .org,..) quản lý

+ DNS local lại gửi request tới Top-level-DNS

+ Top-level-DNS chứa IP quản lý các domain cấp 2 (abc.com; abc.org; cde.com)

+ Top-level-DNS gửi về IP của DNS quản lý domain cấp 2

+ Tương tự v cho đến khi đọc hết domain

+ DNS local sẽ nhận được IP với tên miền đầy đủ (is.abc.com chẳng hạn)

### Tại sao nói FTP gửi thông tin điều khiển “out-of-band”:

+ “Out-of-band” data (dữ liệu ngoài băng) là dữ liệu được truyền qua luồng độc lập với luồng dữ liệu chính trong “band”

+ Gọi là FTP gửi thông tin điều khiển “out-of-band” vì FTP dùng ít nhất 2 kết nối TCP song song để truyền dữ liệu, trong đó 1 kết nối chỉ truyền thông tin điều khiển, không truyền dữ liệu chính/truyền tập tin

### Phân biệt HTTP thường trực và HTTP không thường trực:

+ Kết nối HTTP không thường trực 1 lần chỉ mở 1 kết nối TCP cho 1 đối tượng đi qua, còn HTTP thường trực cho phép nhiều đối tượng cùng đi qua.

+ HTTP không thường trực xuất hiên ở **HTTP 1.0** còn HTTP thường trực xuất hiện ở **HTTP 1.1**

+ HTTP không thường trực sau khi phân tích, gửi/nhận 1 yêu cầu hoàn tất sẽ đóng kết nối. Còn HTTP thường trực sau khi hoàn thành xong 1 yêu cầu sẽ tiếp tục với yêu cầu kế tiếp **trên cùng 1 kết nối TCP**

+ HTTP không thường trực chỉ dùng kết nối được 1 lần, HTTP thường trực có thể tái sử dụng.

### Ứng dụng yêu cầu độ chính xác cao nên dùng TCP hay UDP:

- Nên dùng TCP ở tầng Vận chuyển. Vì:

+ UDP không hỗ trợ xác minh gửi/nhận (bắt tay)

+ UDP không hỗ trợ truyền tin đáng tin cậy

+ UDP không đảm bảo gửi đúng thứ tự

+ TCP hỗ trợ truyền tin đáng tin cậy ( đúng địa chỉ, không lỗi dữ liệu, đúng thứ tự gói tin, không mất dữ liệu

+ TCP có xác minh địa chỉ với quá trình trao đổi điều khiển

### Ứng dụng yêu cầu tốc độ cao nên dùng TCP hay UDP:

- Nên dùng UDP ở tầng Vận chuyển. Vì:

+ UDP không kiểm soát điều khiển, xác minh địa chỉ

+ TCP phải tạo kết nối xác minh, rồi mới tạo kết nối dữ liệu, quá trình này tốn 2 Round-trip-time (là thời gian tính từ lúc client gửi request tới lúc nhận response)

+ Trong khi đó UDP chỉ tốn 1 RTT -> nhanh hơn

### Tại sao HTTP, POP3, FTP đều dùng TCP mà không dùng UDP:

**+ Vì yêu cầu của giao thức:** cả 3 giao thức trên đều yêu cầu độ chính xác cao, truyền tin, load web hay gửi mail không thể xảy ra dữ liệu bị lỗi/mất/… Vì nếu dữ liệu nhận về bị lỗi thì người dùng k thể xem web/xem mail/đúng tập tin cần chuyển. Dữ liệu càng không thể gửi sai địa chỉ, vì mail và tập tin là dữ liệu riêng tư. Mà UDP không hỗ trợ **truyền tin đáng tin cậy.**

**+ Vì có các dịch vụ khác hỗ trợ về mặt tốc độ truyền tải:** Bản thân các giao thức trên đều có thể sử dụng các cơ chế tăng tốc độ truyền tin như caching để tăng tốc độ sử dụng giao thức. Nên để vừa đảm bảo yêu cầu giao thức (đáng tin cậy) vừa nâng cao trải nghiệm người dùng, người lập trình dịch vụ có thể tích hợp thêm các dịch vụ tăng tốc này, thay vì chấp nhận đánh đổi bằng cách dùng UDP

### HTTP Client cần lấy document từ 1 URL chưa biết IP, cần dùng giao thức nào:

- Tầng Ứng dụng sử dụng các giao thức: DNS, HTTP

- Tầng Vận chuyển sử dụng các giao thức: UDP (dùng cho DNS), TCP (dùng HTTP)

### Một site thương mại điện tử muốn lưu trữ thông tin mua hàng bằng các sử dụng cookies, cơ chế và minh hoạ bằng site đó:

- Site có một CSDL cookies để định danh khách hàng bằng cookies

- Khi khách hàng đăng nhập lần đầu (chưa có cookies) sẽ được site khởi tạo cho 1 cookies ID tương ứng. Và gửi “Set cookies ID:” này về cho browser

- Những lần sau, khi truy cập site, browser sử dụng ID đã được cấp để thông báo với site về trạng thái của trình duyệt, từ đó site có thể thông qua ID và định danh.

**\* Minh hoạ**: Giả sử tên site e-commerce là **abc.com**

+ User đăng nhập vào site **abc.com** lần đầu tiên

+ Web server của site tại một cookie ID tương ứng cho user này là ID = **712**.

+ Web server thêm ID này vào CSDL cookie của webserver

+ Web server gửi ID này về cho browser, browser thêm ID và site vào tập cookies (thêm dòng {712; abc.com})

+ User truy cập, mua hàng bình thường. Server sẽ lưu các thông tin này tương ứng với cookie ID đã cấp cho user

+ Lần sau truy cập **abc.com,** browser của user gửi cookie ID đã được cấp từ trước cho site, site dự vào ID này tìm kiếm thông tin của user tương ứng

+ Site trả về thông tin mua hàng trước đây đã lưu trữ cho người dùng xem

### Trình bày giao thức bắt tay: Tầng Vận chuyển

### Bốn ứng dụng phù hợp P2P:

### + Phân phối tập tin ngang hàng (BitTorrent)

+ Dịch vụ điện thoại ngang hàng (Skype)

+ Cơ sở dữ liệu phân tán ngang hàng

+ Tiền ảo (Bitcoin)

### Các câu đúng sai:

1. Sai, người dùng phải gửi 4 yêu cầu tương ứng với 4 đối tượng
2. Đúng, miễn là chúng nằm cùng một kết nối TCP
3. Sai, vì với kết nối không thường trực, mỗi kết nối TCP chỉ có 1 gói tin request gửi đi và 1 gói tin response trả về

### Bảo vệ hệ thống mạng của công ty bị DHCP-Spoofing (giả mạo DHCP):

### + Cấu hình các cổng trên switch thành 2 kiểu: cổng trust và cổng untrust.

+ Cổng trust là cổng có thể cho phép gửi đi tất cả các loại các bản tin DHCP.

+ Cổng untrust chỉ có thể gửi đi các bản tin xin cấp phát địa chỉ IP

+ Thiết lập cổng của máy chủ DHCP chính thức là trust, còn lại untrust

### Phân tích gói tin HTTP Request:

1. URL đầy đủ: gaia.cs.umass.edu/cs453/index.html
2. HTTP ver: HTTP/1.1
3. Yêu cầu kết nối thường trực (Connection: keep-alive)
4. IP của máy tính chạy browser: không có thông tin
5. Loại trình duyệt: có thể là Firefox (Mozilla 5.0/Gecko hay NetScape đều nói tới Firefox). Loại trình duyệt cần quan tâm bởi vì các trình duyệt các nhau có các quy chuẩn khác nhau về trình bày website, cache, tính năng. Nên phải biết loại trình duyệt để tối ưu và trả về phiên bản web cho phù hợp

### Phân tích gói tin HTTP Response:

1. Server tìm được đối tượng được request (200 OK). Thời điểm trả về: Tue, 07...
2. Lần cuối cùng thay đổi trang web là Sat, 10 ….
3. Kích thước trang gửi về: 3874 byte (Content-length)
4. 5 byte đầu tiên trang gửi về là <lf> <title>... tới dòng áp chót. Server chấp nhận kết nối thường trực (Keep-alive: timeout = max = 100)

## TẦNG VẬN CHUYỂN

### Tổng quan về tầng vận chuyển

+ Tầng Ứng dụng cung cấp giao thức cho các ứng dụng/dịch vụ

+ Tầng Vận chuyển giải quyết nhu cầu **giao tiếp giữa các ứng dụng trên các máy khác nhau**

+ Tầng Vận chuyển cung cấp một **kết nối logic** để các **ứng dụng trên 2 hệ cuối (end system)** giao tiếp với nhau

+ Tầng Vận chuyển **không chịu trách nhiệm** cung cấp kết nối trên các **bộ định tuyến,** nó chỉ giải quyết nhu cầu cho **đầu gửi và nhận**

+ Giải quyết nhu cầu cho các bộ định tuyến là **tầng Mạng (Network**)

+ Gói tin thông điệp (message) từ tầng Ứng dụng được chuyển đến tầng Vận chuyển.

+ Các gói tin này có thể bị phân thành nhiều mảnh nhỏ và thêm header riêng của tầng Vận chuyển, tạo thành các phân đoạn **(segment)**

+ Các phân đoạn này được truyền tải xuống tầng Mạng và chuyển đến bên nhận

+ Tầng Vận chuyển nhận gói tin và thực hiện ngược lại quy trình

Vai trò của tầng Vận chuyển

Cách thức vận hành (mức khái niệm) của tầng Vận chuyển

**Summary:** Tầng vận chuyển quản lý và giải quyết nhu cầu giao tiếp giữa các ứng dụng trên các máy tính khác nhau. Tức nó phụ trách quản lý cho máy gửi đầu tiên và máy nhận cuối cùng, không quan tâm đến đường đi (route). Vì vậy giao thức truyền tin trong tầng này được gọi là end-to-end (end system to end system)

### Dồn kênh và phân kênh

Nhắc lại về giao tiếp giữa 2 ứng dụng/tiến trình trên 2 máy khác nhau

Dồn kênh

Phân kênh

+ Process A – Socket 1 – Internet -Socket 2 – Process B

+ Mỗi process có một socket riêng để truyền/nhận dữ liệu

+ Vấn đề đặt ra, giả sử có nhiều process trên máy của 1 user truyền nhận thông tin đồng thời, làm sao tầng Vận chuyển biết gói tin nào của process nào để gửi vào đúng socket ?

+ Việc gom nhiều gói tin truyền đi từ nhiều ứng dụng trên cùng 1 máy để gửi gọi là **dồn kênh.**

+ Để xác định socket cần gửi đến sau khi nhận gói tin, segment bắt buộc phải có dữ liệu phục vụ tương ứng (chi tiết sẽ trình bày sau)

+ Việc nhận gói tin xong phân gói tin về đúng socket của process chính chủ gọi là **phân kênh.**

**Summary:** Một máy, đồng thời sẽ có nhiều process hoạt động cùng lúc. Các process này cũng sẽ có nhu cầu gửi/nhận tin đồng thời. Để gửi và nhận chuẩn xác các gói tin cho từng process t dùng kỹ thuật dồn kênh và phân kênh.

VD: Máy 1 có 2 ứng dụng A đang load web, ứng dụng B đang nhận dữ liệu. Khi đó máy 1 sẽ đồng thời nhận 2 loại gói tin: response từ web server cho ứng dụng A và dữ liệu tập tin cho ứng dụng B. Phân kênh là cần thiết để chuyển dữ liệu về đúng ứng dụng

### Dồn kênh và phân kênh phi kết nối/hướng kết nối

Cơ chế dồn kênh và phân kênh phi kết nối

Cơ chế dồn kênh và phân kênh hướng kết nối

+ Header sẽ được ghi thêm vào Port nguồn – Port đích

+ Port là số dùng để định danh process (xem chương trước)

+ Dựa vào port nguồn và port đích này, tầng vận chuyển sẽ biết gửi gói tin về đúng process cần gửi đến

+ Dồn kênh và phân kênh phi kết nối được dùng cho **UDP**

**Minh hoạ 1:** Máy 1 có 2 process B, C cùng kết nối với webserver 3 yêu cầu 2 đối tượng khác nhau. Webserver gửi về 2 response tương ứng, tầng VC ghi thêm port đích, phân kênh về cho 2 process -> **OK**

+ Khuyết điểm: Nếu có nhiều kết nối sẽ dẫn đến sai lệch.

**Minh hoạ 2:** Máy 1 có process B với port 123456 kết nối tới webserver 3 yêu cầu đối tượng a. Máy 2 có process C cũng có port 123456 kết nối với webserver 3 yêu cầu đối tượng b -> Không thể phân biệt được process B và process C vì cùng port.

-> Cơ chế dồn kênh và phân kênh phi kết nối mới chỉ giải quyết được bài toán **nhiều process trên cùng 1 máy.**

+ Header sẽ được ghi thêm vào bộ 4 giá trị (IP nguồn; port nguồn) và (IP đích; port đích). Giải quyết vấn đề minh hoạ 2 đưa ra:

**Minh hoạ 3:** Máy 1 có process B với port 123456 với IP 172.x.x.x kết nối tới webserver 3 yêu cầu đối tượng a. Máy 2 có process C cũng có port 123456 với IP 172.x.x.y kết nối với webserver 3 yêu cầu đối tượng b -> Dùng các giá trị IP nguồn/đích để phân biệt 2 máy và dùng port nguồn/đích để phân biệt process trên cùng 1 máy

+ Dồn kênh và phân kênh hướng kết nối giải quyết được bài toán dồn kênh và phân kênh cho **nhiều máy, mỗi máy nhiều process**

### 

**Summary:**

### Dồn kênh và phân kênh phi kết nối/hướng kết nối

Cơ chế dồn kênh và phân kênh phi kết nối

Cơ chế dồn kênh và phân kênh hướng kết nối

+ Header sẽ được ghi thêm vào Port nguồn – Port đích

+ Port là số dùng để định danh process (xem chương trước)

+ Dựa vào port nguồn và port đích này, tầng vận chuyển sẽ biết gửi gói tin về đúng process cần gửi đến

+ Dồn kênh và phân kênh phi kết nối được dùng cho **UDP**

**Minh hoạ 1:** Máy 1 có 2 process B, C cùng kết nối với webserver 3 yêu cầu 2 đối tượng khác nhau. Webserver gửi về 2 response tương ứng, tầng VC ghi thêm port đích, phân kênh về cho 2 process -> **OK**

+ Khuyết điểm: Nếu có nhiều kết nối sẽ dẫn đến sai lệch.

**Minh hoạ 2:** Máy 1 có process B với port 123456 kết nối tới webserver 3 yêu cầu đối tượng a. Máy 2 có process C cũng có port 123456 kết nối với webserver 3 yêu cầu đối tượng b -> Không thể phân biệt được process B và process C vì cùng port.

-> Cơ chế dồn kênh và phân kênh phi kết nối mới chỉ giải quyết được bài toán **nhiều process trên cùng 1 máy.**

+ Header sẽ được ghi thêm vào bộ 4 giá trị (IP nguồn; port nguồn) và (IP đích; port đích). Giải quyết vấn đề minh hoạ 2 đưa ra:

**Minh hoạ 3:** Máy 1 có process B với port 123456 với IP 172.x.x.x kết nối tới webserver 3 yêu cầu đối tượng a. Máy 2 có process C cũng có port 123456 với IP 172.x.x.y kết nối với webserver 3 yêu cầu đối tượng b -> Dùng các giá trị IP nguồn/đích để phân biệt 2 máy và dùng port nguồn/đích để phân biệt process trên cùng 1 máy

+ Dồn kênh và phân kênh hướng kết nối giải quyết được bài toán dồn kênh và phân kênh cho **nhiều máy, mỗi máy nhiều process**

### 

**Summary:**

### Giao thức tầng Vận chuyển – giao thức UDP

Giao thức UDP

Khi nào thì sử dụng UDP, vì sao sử dụng UDP

Một số ứng dụng dùng UDP và nguyên nhân

+ Giao thức UDP hỗ trợ dồn kênh, phân kênh và kiểm tra lỗi

+ Giao thức UDP không thực hiện giao thức bắt tay (không thiết lập kết nối) vì vậy giao thức UDP là giao thức phi kết nối.

+ Giao thức UDP truyền dữ liệu không tin cậy (có thể mất gói tin, sai thứ tự, v.v). Khi mất gói, UDP kết thúc request, không có cơ chế khôi phục gói.

+ Kích thước header của UDP nhỏ hơn TCP nhiều -> nhanh hơn

+ Các gói tin truyền tải bằng UDP không bị giới hạn tốc độ

+ Ứng dụng yêu cầu tốc độ cao, độ trễ thấp

+ UDP chuyển ngay segment xuống tầng Mạng sau khi đóng gói, không có các bước xử lý khác

+ UDP không tốn thời gian thiết lập kết nối

+ UDP không quản lý trạng thái kết nối của các hệ thống đầu cuối nên máy chủ có thể hỗ trợ nhiều máy khách hơn (more free space)

+ DNS: 2 máy chủ DNS kết nối có thể ở rất xa nhau, nếu thiết lập kết nối -> tốn thời gian, chậm trễ phân giải -> UDP

+ RIP (giao thức định tuyến): mỗi 5p, RIP sẽ cập nhật thông tin lại từ đầu, nên việc sai lệch và mất thông tin không quan trọng.

+ SNMP ( giao thức quản trị mạng), các ứng dụng quản trị mạng thường xuyên phải vận hành trong điều kiện mạng quá tải. Nếu dùng TCP, 2 lợi thế là kiểm soát nghẽn và truyền tin tin cậy không thực hiện được -> không thực tế -> UDP

**Summary:** UDP là một giao thức truyền dữ liệu do tầng Vận chuyển cung cấp, nó là một giao thức phi kết nối (không thiết lập kết nối/bắt tay), phi trạng thái (không quản lý trạng thái). UDP chỉ cung cấp dịch vụ tối thiểu cần có là truyền dữ liệu (không tin cậy) và kiểm soát lỗi. Vì không kiểm soát nhiều thông tin nên UDP có tốc độ cao hơn nhiều so với TCP.

### Truyền tin đáng tin cậy và giải pháp – ACK/NAK và thời gian chờ

Truyền tin đáng tin cậy

Toàn bộ quá trình sẽ minh hoạ bằng **bên A – bên B**

Cách thức kiểm soát lỗi gói tin (lỗi bit) và phục hồi gói tin

Cách thức kiểm soát việc bị mất gói tin.

**Một dịch vụ truyền tin đáng tin cậy phải đáp ứng được:**

+ Dữ liệu/gói tin truyền đi không bị lỗi (sai lệch bit)

+ Dữ liệu gửi đi không bị mất gói tin

+ Gói tin gửi đi không bị sai thứ tự

**Tình huống:** Bên A gửi gói tin cho bên B, **có thể** bị lỗi bit, cần biết **lỗi hay không**? và nếu lỗi thì **phục hồi bằng cách gửi lại -> solve:**

+ Thêm phương pháp phát hiện lỗi tại bên nhận: checksum, hash, v.v. So sánh **mã** **ghi trong gói tin** với **mã** **check thực tế bên B**)

+ Báo nhận được, lỗi hay không lỗi về bên gửi bằng cách gửi ngược về bên A gói tin ACK (ok, không vấn đề) hoặc gói NAK (lỗi, gửi lại)

+ Gửi lại gói tin bị lỗi cho bên B nếu nhận được gói NAK. Gửi gói tiếp theo nếu nhận được gói ACK.

**Tình huống:** Khi bị mất gói, **bên A sẽ không nhận được gói ACK/NAK** từ bên B vì **gói tin chưa tới được bên B.**

**Cách giải quyết:**

+ Thêm bộ đếm thời gian cho bên gửi (bên A)

+ Nếu hết thời gian chờ vẫn chưa thấy bên B gửi ACK/NAK -> gửi lại

+ Nếu trong thời gian chờ nhận được gói tin, ngắt bộ đếm, thực hiện gửi lại nếu nhận NAK hoặc gửi gói tiếp theo nếu nhận ACK

**Summary:** Truyền tin đáng tin cậy là một trong những dịch vụ quan trọng bậc nhất tầng vận chuyển. Để đảm bảo các tiêu chí khác nhau của dịch vụ này, ta cần phối hợp nhiều thuộc tính điều khiển khác nhau. Riêng để kiểm soát việc gửi nhận của **1 gói** **tin,** ta cần: checksum; gói tin báo nhận/lỗi (ACK/NAK); bộ đếm thời gian

### Truyền tin đáng tin cậy và giải pháp – Seq và pipelined

Quá trình minh hoạ sẽ ứng dụng các kỹ thuật kiểm soát ở phần trước

Cách thức kiểm soát việc mất gói và lỗi bit cho nhiều gói tin

Cách thức tăng hiệu suất gửi bằng giao thức pipelined

**Tình huống**: Lỗi bit/mất gói trên **chính gói tin ACK/ NA** khi **gửi 2 gói tin** nếu **2 gói ACK/NAK giống nhau** thì không biết gói nào đã đến

+ Thêm số thứ tự (**Seq**) cho gói tin và cho cả ACK/NAK

+ VD: Gửi 2 gói tin x,y. Gói tin x gửi đi có Seq = 1. Bên A mong chờ nhận ACK/NAK 1. Gói tin y gửi đi có Seq = 2. Bên A lại chờ nhận ACK/NAK 2. Nếu **gói y mất gói**, nhận ACK 1 thì vẫn biết x đã đến.

+ Thêm số thứ tự cũng giải quyết vẫn đề lỗi/mất ACK/NAK.

VD: Tương tự như trên. Nhưng **gói y vẫn đến bên B**. ACK 2 **bị lỗi/mất gói** -> hết thời gian chờ -> gửi lại y đến B với Seq 2 -> gửi ACK 2 về A -> OK, gửi tiếp gói tin tiếp theo

+ Dễ thấy, lỗi/mất ACK sẽ gây trùng gói tin y, cài đặt thêm xử lý trùng

**Tình huống:** Dễ thấy, theo như cách thức trình bày ở trên, bên A sau khi gửi gói x, sẽ ngừng lại chờ gói ACK phản hồi. Rồi lại gửi gói y, chờ phản hồi -> cơ chế stop-wait, xử lý đơn luồng -> chắc chắn đúng thứ tự nhưng tốn rất nhiều thời gian

**Cách giải quyết:**

+ Cho phép gửi nhiều gói tin một lúc mà không chờ báo nhận

-> Tăng tốc độ rất nhiều -> có thể sai thứ tự nếu xảy ra sự cố lỗi gói/mất gói.

-> Cần kiểm soát việc gửi và sử dụng sequence number chuẩn xác

**Summary:** Phần sau sẽ giải quyết vấn đề gửi đúng thứ tự và summary phần dịch vụ truyền tin đáng tin cậy

### Truyền tin đáng tin cậy và giải pháp – Quay lui (Go-back-N)

+ Cho phép gửi nhiều gói tin đi mà không chờ nhận gói ACK/NAK

+ Giới hạn N gói tối đa cho phép gửi đi khi chưa có báo nhận ACK. Các gói tin này có stt liên tục nhau và được gọi là frame (“cửa sổ”)

+ Nếu số gói chưa báo nhận vượt mức cho phép, không cho gửi

**VD: Giới hạn = 3. Cần truyền đi 7 gói, đánh số từ 0-6. Lúc gửi:**

- Gói tin 0, 1 đã gửi xong và nhận ACK 0, ACK 1

- Gói tin 2, 3 đã gửi đi và chưa nhận ACK

- Vậy gói tin 4 có thể tiếp tục được gửi đi mà không cần chờ ACK 2,3

- Max = 3 nên gói tin 5 phải chờ bên A nhận ACK 2 mới được gửi.

- Nếu gói tin 2 bị mất gói, gói tin 5 phải chờ đến khi nhận ACK 2

------------------------------------------------------------------------

+ Bên nhận chỉ chấp nhận gói tin đúng thứ tự (gói N -> gói N+1)

+ Nếu đúng thứ tự, chuyển dữ liệu lên tầng trên

+ Bên nhận chỉ quan tâm một tham số duy nhất:**“gói tin đang chờ”**

+ Nếu không đúng thứ tự, **huỷ gói**, **trả về ACK đã nhận trước đó.**

**VD:** -Vì gói tin 2 bị mất gói, nên gói tin 3 đến bên nhận trước.

- Bên nhận đã nhận và chuyển đi gói tin 1, đang chờ gói tin 2

- Gói tin 3 đến -> sai thứ tự -> huỷ gói (xoá) -> **gửi ACK 1**

- Hết thời gian chờ gói 2 -> gửi lại gói 2 tới bên nhận

- Đúng thứ tự, chuyển đi tầng trên, chờ gói 3, **gửi ACK 2**

- Bên A nhận ACK 2, cho phép gói tin 5 được gửi đi

- Hết thời gian chờ gói 3 -> gửi lại gói 3 ….

-> Liên hệ với bên gửi: Nếu gói đầu tiên (trong N gói gửi đi) chưa có ACK, các gói sau cũng không có ACK-> **ngừng,** **chờ** ACK gói đầu tiên

**Về bên gửi:**

**Về bên nhận:**

**Summary:** Ta dễ thấy, giao thức Quay lui kiểm soát việc gửi đúng thứ tự bằng cách: **bên nhận chỉ nhận gói tin đúng thứ tự**. Tuy nhiên nếu có một gói tin bị lỗi/mất thì nhiều gói tin khác (mặc dù không lỗi) cũng phải gửi lại nhiều lần vì sai thứ tự -> tốn băng thông hơn -> kém hiệu quả.

### Truyền tin đáng tin cậy và giải pháp – Gửi lại có chọn (Selective Repeat)

+ Vẫn sử dụng cơ chế giới hạn số gói tối đa chưa được báo nhận

+ Giống Go-back-N, nếu gói tin đầu tiên trong N gói gửi đi chưa có gói báo nhận ACK, không cho phép “trượt cửa sổ” để gửi đi tập tin mới. (Minh hoạ hình 2)

+ Khi gói tin đầu “cửa sổ” báo nhận ACK, toàn bộ cửa sổ sẽ di chuyển đến gói tin gần nhất chưa báo ACK, gọi là “trượt cửa sổ” (MH hình 3)

**VD: Giới hạn = 3. Cần truyền đi 7 gói, đánh số từ 0-6. Lúc gửi:**

- Gói tin 0, 1 đã gửi xong và nhận ACK 0, ACK 1

- Gói tin 2, 3 đã gửi đi và chưa nhận ACK

- Vậy gói tin 4 có thể tiếp tục được gửi đi mà không cần chờ ACK 2,3

- Giả sử gói 2 bị mất gói khi truyền tải.

------------------------------------------------------------------------

+ Bên nhận nhận tất cả các gói tin được gửi đến (gói không lỗi)

+ Bên nhận cũng quản lý một “cửa số” tương tự như bên gửi

+ Đầu khung cửa sổ là gói tin chưa được nhận có stt nhỏ nhất.

+ Nếu không đúng thứ tự, **không huỷ gói, lưu vào bộ nhớ đệm, gửi trả về ACK đúng với Seq của gói.**

+ Nếu đúng thứ tự, lấy ra các **gói tin có số thứ tự liên tục phía sau nó** có mặt trong bộ đệm rồi chuyển tất cả lên tầng trên

--------------------------------------------------------------------------+ Đồng nghĩa với việc “cửa sổ” ở bên gửi có thể có các gói tin đã báo ACK. (Gói đầu cửa sổ bị lỗi k báo ACK, các gói sau đến và báo ACK, nhưng vì gói đầu không báo ACK nên cửa sổ giữ nguyên)

+ Bên gửi **chỉ gửi lại những gói tin bị lỗi**, không phải gửi lại những gói tin (gửi đúng không lỗi) đến bên nhận sai thứ tự

**Về bên gửi:**

**Về bên nhận:**

**Trở về bên gửi (tt):**

**Summary:** Ta dễ thấy, giao thức này tạo một bộ nhớ đệm ở bên nhận để lưu trữ các gói tin đến sai thứ tự, và dựa vào cơ chế “cửa sổ” để đảm bảo lúc gửi lên tầng trên, thứ tự các gói tin vẫn được đảm bảo -> dùng băng thông hiệu quả hơn, thiết kế phức tạp hơn.

### Minh hoạ về Select Repeat

**Tình huống:** Gửi 7 gói tin từ A -> B, cửa sổ = 3, gói tin 0, 1 đã gửi xong và báo ACK

**-> Lúc này tình trạng 2 bên như sau: (Hình 1)**

**- Bên A:** + Gói tin 2,3,4 đang được gửi đi và chưa có báo nhận

+ Gói tin 5,6 đang chờ được gửi

**- Bên B:** + Gói tin đã nhận được là 2 gói tin 0,1

+ Đầu frame là gói chưa đc nhận nhỏ nhất -> 2,3,4

**-> Gói tin 2 bị mất gói, gói tin 3,4 đến B trước**

**- Bên B:** + Gói tin 0,1 đã được nhận; gói tin 3,4 được nhận nhưng sai thứ tự, trong cache

+ Đầu frame là gói chưa đc nhận nhỏ nhất -> 2,3,4 = frame

**- Bên A:** + Gói tin 3,4 có báo nhận ACK 3, ACK 4

+ Frame vẫn là 2,3,4 vì 2 chưa được ACK

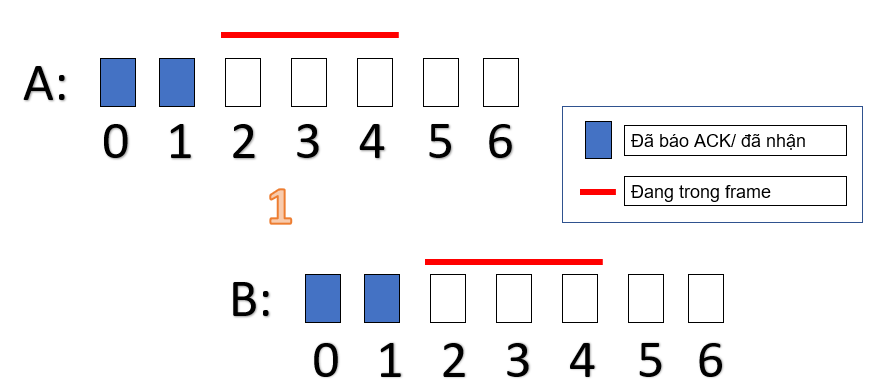
+ Gói tin 5,6 đang chờ được gửi

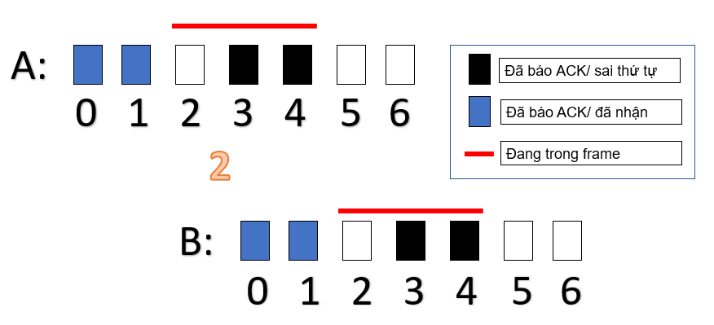
**-> Time out gói 2, gửi lại gói 2, gói 2 đến B**

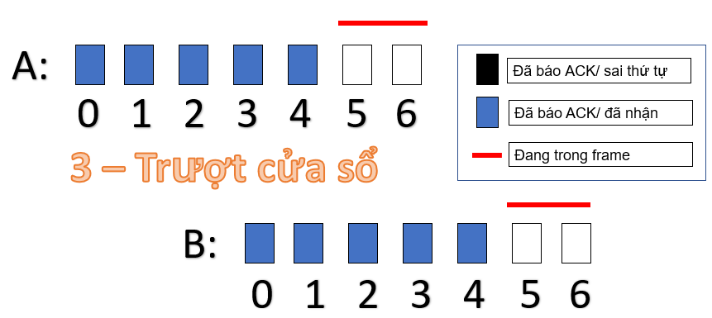
**- Bên B:**  + Gói tin 2 được nhận, lấy ra 3,4 liên tục ngay sau trong cache, chuyển đi

+ Cửa sổ trượt từ (2,3,4) đến (5,6)

**- Bên A:**  + Nhận ACK 2 báo gói tin 2 đến nơi, trượt cửa sổ

+ Cửa sổ trượt từ (2,3,4) đến (5,6). Gửi đi gói tin 5,6



****

### Giao thức vận chuyển hướng kết nối – giao thức TCP

Giao thức TCP

Gói tin TCP

+ Giao thức TCP phục vụ cho việc truyền dữ liệu giữa **2 tiến trình ở tầng Ứng dụng** với nhau.

+ Giao thức TCP là **giao thức hướng kết nối**, tức phải thiết lập kết nối trước bằng thao tác “bắt tay ba bước”.

+ Giao thức TCP tạo ra **kết nối point – to – point 2 chiều** cho phép gửi và nhận giữa 2 tiến trình với nhau.

+ Một kết nối TCP gồm 2 tập hợp gửi/nhận, mỗi tập hợp gồm: bộ nhớ đệm, tham số điều khiển, kết nối socket tới tiến trình tầng Ứng dụng.

+ Gói tin từ tầng Ứng dụng sau khi được chia nhỏ và đóng gói bởi giao thức TCP sẽ trở thành gói tin TCP (hay chính xác hơn là segment)

+ Gói tin TCP gồm header và dữ liệu

+ Header của gói tin TCP gồm:

- Cổng nguồn/đích của tiến trình tầng Ứng dụng (Src/Dst Port)

- Số thứ tự (Sequence number) phục vụ truyền tin đáng tin cậy

- Số báo nhận (ACK) phục vụ truyền tin đáng tin cậy

- Header-length, cờ, checksum, …

… (cụ thể trang 229 SGK)

- **Dữ liệu truyền đi**

**Summary:** Như vậy để truyền tin đáng tin cậy với hiệu suất cao ta cần có các thuộc tính điều khiển sau: **checksum; số thứ tự; bộ đếm thời gian; gói tin báo nhận ACK/NAK**. Ngoài ra, ta còn cần cơ chế kiểm soát thứ tự và bộ nhớ đệm tại bên nhận. Về cụ thể cơ chế truyền tải thực tế của dịch vụ này, sẽ được trình bày ở phần giao thức vận chuyển TCP.

### Cấu trúc gói tin TCP chi tiết

Trường số thứ tự

(Sequence number)

Trường số báo nhận

(ACK#)

+ Seq# = số thứ tự của byte đầu tiên của gói tin trong trường dữ liệu

+ VD: gửi đi 5000 bytes dữ liệu, chia làm 4 gói tin mỗi gói 1250 bytes.

Vậy Seq# của 4 gói lần lượt là 0; 1250; 2500; 3750

+ TCP là kết nối 2 chiều A gửi đến B thì B cũng gửi lại đến A

+ Thay vì gửi gói tin báo nhận thì TCP tạo một tham số là “số báo nhận” và đưa vào trong gói tin gửi dữ liệu từ A->B

+ Trường số báo nhận là số byte mà bên gửi muốn nhận từ bên nhận

+ Vì kết nối TCP là kết nối 2 chiều nên sẽ có 2 trường tương ứng

VD: Bên A và bên B trao đổi dữ liệu, bên A gửi cho bên B 1 đoạn dữ liệu 1000 bytes chia làm 2 gói tin với Seq# = 0;500.

Đồng thời bên B cũng muốn gửi 1 đoạn dữ liệu 1500 bytes chia làm 5 gói cho A nhưng trước đó đã có gửi 2 gói có Seq# = 0 và 300

Theo lý thuyết: + Bên A gửi gói tin 1 và mong chờ nhận về **ACK 1**

+ Bên B nhận gói tin 1 và gửi lại gói ACK1

…..

TCP cài đặt: + Bên A gửi gói tin đầu tiên với Seq# = 0 và ACK# = 600

+ Bên B nhận và gửi lại gói với Seq = 600 và **ACK# = 500**

+ Bên A nhận và gửi lại gói với Seq = 500 và ACK = 900

- Vậy bên B đã nhận được gói tin đầu tiên (500/1000) bytes từ A, bên B chờ bên A gửi gói thứ 2 (từ byte thứ 500-999) với **ACK# = 500**

- Bên A đã nhận được gói tin thứ 3 (900/1500) byte và chờ bên B gửi gói tin thứ 4 (từ byte thứ 900-1199) với **ACK# = 900**

-> **tham số ACK# = 500 chính là biễu diễn của gói ACK 1.**

**Summary:** Như vậy, trường báo nhận va trường thứ tự dùng để kiểm tra thứ tự các gói tin trong dịch vụ truyền dữ liệu đáng tin cậy do TCP cung cấp. Vì TCP là kết nối 2 chiều nên cần 2 trường tương ứng như vậy

### ­­ Cơ chế vận hành của truyền dữ liệu bằng giao thức TCP

**Thông tin chung:**

**Bên gửi:**

**Bên nhận:**

**Khác biệt so với Go-back-N**

+ Sử dụng pipeline để gửi nhiều gói tin cùng lúc

+ Sử dụng số báo nhận ACK# để báo nhận.

+ Thiết lập thời gian timeout cho gói tin ở đầu buffer

+ Gửi lại toàn bộ dữ liệu trong buffer khi timeout

+ Ngoài ra TCP vẫn dùng các kỹ thuật kiểm tra lỗi để kiểm tra gói

(**Gần giống** với giao thức Quay lui Go-back-N, buffer ⬄ frame)

+ Nhận dữ liệu từ tầng ứng dụng, tạo segment (thêm header), thiết lập thời gian timeout, đẩy gói tin vào buffer gửi đi.

+ Nhận gói tin có báo nhận ACK# (gửi thành công), “trượt cửa sổ”

+ Nếu timeout, collapse buffer, gửi lại hết các gói tin còn trong buffer

+ Nếu dữ liệu đúng thứ tự (Seq = ACK đang chờ) -> chấp nhận -> gửi ACK# mới về cho bên gửi

+ Nếu dữ liệu không đúng thứ tự -> Gửi lại ACK# đang chờ (khiến bên gửi timeout)

+ Giả sử gửi 1-> N gói tin, đều đến được bên nhận, nhưng gói ACK# của gói thứ k bị mất. (k < N)

+ Go-back-N sẽ gửi lại các gói từ k->N

+ TCP sẽ không gửi lại gói tin k nếu ACK# của gói k+1 tới trước khi timeout. Vì lúc này bên nhận của TCP không bị thiếu gói tin nên không có vấn đề gì.

**Summary:** Như vậy TCP sử dụng kiểm tra lỗi; pipeline; số báo nhận ACK; bộ đếm thời gian và cơ chế xử lý mất gói riêng để kiểm soát việc truyền tin đáng tin cậy của mình. Cách thức hoạt động đã đề cập rõ ở các phần phía trên. Cơ chế xử lý mất gói của TCP gần giống Go-back-N nhưng tối ưu hơn -> tận dụng băng thông tốt hơn

### Thiết lập kết nối TCP, cơ chế bắt tay handshaking 3 bước:

**Bước 1:**

**Bước 2:**

**Bước 3:**

**Thông tin thêm**

+ TCP máy khách gửi gói TCP đặc biệt tới TCP máy chủ

+ Gói tin này gọi là **gói tin SYN**, không có dữ liệu cờ SYN đc bật (1)

+ Seq# của gói tin này là 1 số thứ tự ngẫu nhiên, tạm gọi là **client**

+ Gói tin SYN đến tầng mạng, gói vào 1 gói tin IP và gửi đi

+ Máy chủ nhận được gói tin SYN, rút trích thông tin

+ Máy chủ đồng ý kết nối, gửi trả về **gói tin SYNACK**

+ Gói tin này gồm số ACK# = **client**, Seq# = **server**(tạo random) và cờ SYN bật. Gói tin này mang ý nghĩa đồng ý tạo kết nối

+ Máy khách nhận được gói tin SYNACK, rút trích thông tin

+ Máy khách gửi về máy chủ 1 gói tin khác để báo nhận (ACK) gói tin của máy chủ. Gói tin trả về cờ SYN = 0 vì đã kết nối xong

+ Gói tin này đã có thể gửi kèm dữ liệu

+ Để đóng kết nối sử dụng cờ FIN, cơ chế tương tự cờ SYN

+ Trong thời gian kết nối TCP sống, giao thức TCP có nhiều trạng thái khác nhau như CLOSED, SYN\_SENT, ESTABLISHED, FIN\_WAIT, TIME\_WAIT, CLOSED đối với máy khách

+ Hoặc CLOSED, LISTEN, SYN\_RCVD (receive), ESTABLISHED, CLOSE\_WAIT, LAST\_ACK, CLOSE đối với máy chủ.

**Summary:** Như vậy, để bắt đầu việc truyền tin giữa 2 tiến trình, TCP sẽ tiến hành thiết lập kết nối trước để đảm bảo gửi đúng địa chỉ, việc thiết lập/đóng kết nối này được sử dụng điều khiển bởi 2 cờ SYN/FIN.

### Kiểm soát luồng (Flow control)

Kiểm soát luồng

Cụ thể cơ chế vận hành

Trường hợp cần xử lý

+ TCP hỗ trợ dịch vụ kiểm soát luồng gửi dữ liệu

+ Mục đích của dịch vụ này là để hạn chế việc bên gửi làm tràn bộ đệm của bên nhận

+ Cơ chế đề suất: so tốc độ gửi dữ liệu của bên gửi và tốc độ đọc dữ liệu của bên nhận để khớp với nhau

+ TCP **duy trì cung cấp** tham số “Receive windows” trong gói tin

+ Tham số này cho biết bộ đệm bên nhận còn trống bao nhiêu

+ Vì TCP là giao thức 2 chiều nên mỗi bên sẽ có một “cửa sổ nhận” khác nhau và riêng biệt

+ Cửa sổ nhận cập nhật lại liên tục trong mỗi gói tin trả về

**Tình huống:**

+ Như vậy bên A muốn biết bên B còn trống hay không thì phải gửi 1 gói tin đến B, chờ tham số “rcv window” nằm trong gói tin ACK gửi về.

+ Nếu tham số gửi về = 0 -> A không gửi gói tin nữa

+ Vì A không gửi gói tin nên không biết khi nào bộ đệm lại trống (các dữ liệu trong bộ đệm được chuyển đi)

-> Ngưng gửi dữ liệu.

+ Cách giải quyết là nếu rcvwindow = 0, vẫn liên tục gửi các gói tin 1 byte đến bên B, đến khi nào nhận đc rcvwindow trống thì tiếp tục gửi.

**Summary:** Như vậy ngoài dịch vụ truyền tin đáng tin cậy, TCP còn hỗ trợ kiểm soát luồng để **bên gửi không làm tràn bộ nhớ đệm bên nhận.** Cụ thể được thực hiện bởi tham số “Receive Window”- cửa sổ nhận. Tham số này được cập nhật liên tục qua mỗi gói tin.

### Kiểm soát tắc nghẽn trong TCP

**Tắc nghẽn**

Cách thức xử lý của giao thức TCP – kiểm soát tắc nghẽn

+ Tắc nghẽn là khi có quá nhiều gói tin tại 1 node -> không xử lý đc

**- Nguyên nhân xảy ra tắc nghẽn:**

+ Tốc độ gửi gói tin đến node nhanh hơn thông lượng của node -> độ trễ hàng chờ tăng -> timeout

+ Tràn bộ đệm của node, không lưu được thêm các gói tin tới sau dẫn đến hiện tượng mất gói -> gửi lại gói tin

+ Cạnh tranh luồng dữ liệu giữa các liên kết trung gian -> tắc nghẽn

**- Khởi động chậm:** Khi kết nối TCP được thiết lập thành công

+ TCP gửi các gói tin đầu tiên xuống tầng mạng với tốc độ khởi điểm rất thấp, và tăng tốc từ từ **theo hàm số mũ** tương ứng với mỗi lần báo nhận thành công trước thời gian chờ

+ Báo nhận thành công tức mạng còn thông thoáng -> tăng tốc

+ Tồn tại 1 ngưỡng cho phép, sau khi vượt ngưỡng thì chuyển pha

+ Nếu chưa đạt ngưỡng mà bị timeout -> bắt đầu lại từ đầu

**- Tránh tắc nghẽn:**

+ Pha khởi động chậm chuyển sang tránh tắc nghẽn khi **cửa sổ tắc nghẽn >= ngưỡng cho phép (cwnd >= ssthresh**

+ Tốc độ tăng của cstn sẽ chậm xuống so với lúc ở pha khởi động chậm, vì mức cstn đã ở rất cao rồi (nếu cao quá -> nghẽn)

+ Lúc này tốc độ sẽ **tăng tuyến tính** (+1 MSS mỗi 1 RTT)

+ Khi timeout hoặc trùng ACK (trùng ACK cũng là 1 dấu hiệu của tắc nghẽn) -> **khởi động chậm**

**Summary:** Như vậy, để đảm bảo hạn chế xảy ra tắc nghẽn -> mất gói, lỗi gói -> TCP sử dụng 2 pha khởi động chậm và tránh tắc nghẽn để kiếm soát tốc độ gửi gói tin đi

### Cụ thể về cơ chế vận hành của kiểm soát tắc nghẽn TCP Reno

**Nhắc lại về TCP cơ bản**

TCP Reno có thêm

Cách thức vận hành của TCP Reno (từ đó => TCP cơ bản bằng cách bỏ pha phục hồi nhanh)

+ Gồm 2 pha: khởi động chậm và tránh tắc nghẽn

+ Khởi động chậm xuất phát điểm thấp, tăng theo hàm số mũ

+ Thông báo tắc nghẽn: có gói tin ACK trùng/timeout – mất gói

**- Phục hồi nhanh:** Pha này xuất hiện khi có 3 gói tin ACK trùng nhau

+ Trong pha phục hồi nhanh, nếu nhận đc gói ACK mới -> về pha tránh tắc nghẽn

+ Nếu bị timeout -> về pha khởi động chậm

+ Nếu khởi động chậm có 3 gói tin trùng nhau -> phục hồi nhanh

+ Nếu tránh tắc nghẽn có 3 gói ACK trùng nhau -> phuc hồi nhanh

**- Chi tiết về các thông số: (cửa số tắc nghẽn và ngưỡng cho phép)**

+ Sau khi thiết lập, tiến hành **khởi động chậm** với cwnd = 1

+ ssthreshold (ngưỡng) được thiết lập trước đó, ssthresh = x

+ Với mỗi gói tin thành công, báo ACK, tăng cwnd lên gấp đôi

+ Nếu xảy ra mất gói, ssthresh = cwnd/2; cwnd = 1 bắt đầu lại pha

+ Nếu không xảy ra mất gói, sẽ đến lúc cwnd >= ssthresh. Khi đó tiến hành chuyển sang pha **tránh tắc nghẽn**

+ Trong pha tránh tắc nghẽn, cwnd tăng tuyến tính, cwnd += 1

+ Nếu xảy ra mất gói, ssthresh = cwnd/2; cwnd = 1 về kđchậm

+ Nếu xảy ra trùng 3 gói ACK, ssthresh = cwnd/2 và cwnd = ssthresh + 3 về giai đoạn phục hồi nhanh, nếu phục hồi được gói ACK -> lại tăng tuyến tính. Nếu vẫn timeout, về lại khởi động chậm

**Summary:**

## GIẢI BÀI TẬP TẦNG ỨNG DỤNG (SÁCH trang 273)

1. **Liệu có ứng dụng nào có thể truyền tin đáng tin cậy mà dùng UDP không:**

Có. Khi đó ứng dụng sẽ **tự cài đặt** các cơ chế kiểm soát và điều khiển việc truyền dữ liệu theo nguyên lý truyền tin đáng tin cậy (như gửi lại request khi timeout, báo nhận, phát hiện lỗi…) ở **các giao thức của tầng ứng dụng**

1. **Các nhà phát triển ứng dụng thích chọn UDP hơn TCP vì:**

+ Đặc thù của ứng dụng: có thể tự phục hồi dữ liệu mà không cần cơ chế truyền dữ liệu tin cậy, yêu cầu tốc độ cao, ứng dụng real-time, yêu cầu chấp nhận mất gói, lỗi, ..

+ Nhu cầu của người lập trình: người lập trình muốn có những cơ chế kiểm soát riêng, muốn tăng tốc ứng dụng mà không kiểm soát tắc nghẽn

1. **Dữ liệu thoại (voice) và video trên mạng thường gửi qua:**

- UDP. Bởi vì:

+ Đây là các ứng dụng realtime, xem đến đâu load đến đó, dùng đến đó

+ Yêu cầu tốc độ truyền tải cao, delay thấp

+ Chấp nhận mất gói để cuộc nói chuyện, video được tự nhiên

+ Nếu dùng TCP với kiểm soát tắc nghẽn và kiểm soát truyền tin sẽ dẫn đến delay khi voice và load video chậm, hay bị dừng.

1. **Thiết lập port cho 2 client A, B tạo Telnet tới server S:**
2. A gửi đến S: port tại A: ngẫu nhiên (port còn trống); port tại S: 23
3. B gửi đến S: port tại B: ngẫu nhiên (port còn trống); port tại S: 23
4. S gửi đến A: 23 – random (ngược lại so với A, B)
5. S gửi đến B: 23 – random. Nếu A, B là 2 máy tính khác nhau thì số hiệu cổng nguồn trong đoạn từ A -> S và từ B -> S có thể giống vì IP chúng khác nhau.
6. Nếu chúng là cùng một máy thì bắt buộc phải dùng 2 port khác nhau vì:

+ Một máy tính k cho phép 2 tiến trình dùng chung 1 port

+ Server S sẽ không phân biệt được 2 tiến trình A, B

1. **Nếu giống checksum thì có thể sure là không lỗi hay không:**

Không. Vì bản thân cơ chế checksum có thể có 2 bộ giá trị khác nhau nhưng cùng giá trị checksum: đó là 2 bộ giá trị là đảo bit của nhau. Cần kiểm tra bằng nhiều cơ chế mạnh hơn như CRC, parity … để chắc chắn

1. **Phát biểu đúng sai:**
2. Sai. Host B vẫn tạo gói tin và thêm trường ACK# tương ứng với phần tiếp theo nó cần
3. Sai. Cửa sổ bên nhận thay đổi dựa trên số byte trống của bộ đệm nhận dữ liệu
4. Sai. Vì có thể hết thời gian chờ, full cửa sổ, hoặc gửi lại gói tin m đó
5. Sai. Xem phần ước lượng RTT
6. Sai. ACK cho segment 38 phải là 38, ACK sau khi nhận segment 38 và 4 byte mới là 42
7. **A->B có port nguồn là x và port đích là y, gói tin theo chiều ngược lại (gói trả lời):**

Có port nguồn là y và port đích là x

1. **Điền thông tin:**

DHCP: port client: 68; port server 67; giao thức sử dụng TCP; ứng dụng; cấp phát IP động

HTTP: port ở server: 80; giao thức sử dụng TCP; giao thức truy cập web

SMTP: port ở server 25; giao thức sử dụng TCP; giao thức gửi thư điện tử

DNS; port ở server: 53; giao thức sử dụng UDP; giao thức phân giải tên miền

1. **Host C chạy socket UDP ở port 6789, hai host A, B cùng gửi đến port này:**

+ Gói tin từ A, B vẫn đến được host C

+ Host C sẽ phân kênh dựa trên port nguồn của A và B khác nhau, vì host C chạy UDP nên nếu A, B cùng port nguồn sẽ không phân biệt được

1. **Webserver chạy ở host C dùng port 80. Webserver nhận yêu cầu từ 2 host A, B. Nếu A và B gửi yêu cầu tới C thì có cùng socket hay không ? Nếu yêu cầu tới các socket khác nhau thì 2 socket đó có được là port 80 không**

+ Dù là kết nối thường trực hay không thường trực, thì 2 máy khác nhau gửi yêu cầu sẽ thì server sẽ tạo từng kết nối socket khác nhau chứ không dùng chung.

+ Cả 2 socket đều dùng port đích là 80: nhưng phân biệt với nhau bởi IP nguồn khác nhau nên vẫn có port 80 giống nhau bình thường.

1. **Tại sao lại cần Seq và Timer ?**

+ Seq để phân biệt các gói tin với nhau trong trường hợp mất gói, gửi lại gói và đặc biệt là kiểm soát thứ tự gửi

+ Timer dùng để kiểm soát việc mất gói, mất ACK để phục hồi gói tin

1. **Host A gửi lần lượt 2 gói có Seq = 90 và Seq = 110 tới host B**
2. Gói tin đầu tiên có dung lượng là 20 bytes (bytes thứ 90 – 109)
3. Host B gửi cho host A khi gói đầu bị mất mà gói 2 đến đích là ACK# = 90
4. **Host A gửi tới host B byte thứ 126….**
5. Gói thứ 2 có Seq# = 177; port nguồn 302; port đích 80
6. ACK của segment đầu tiên có ACK# = 177; port nguồn 80; port đích 302
7. Nếu phân đoạn thứ 2 tới trước ACK# = 127; port nguồn 80; đích 302
8. Báo nhận đầu tiên mất, báo nhận thứ 2 tới sau thời gian time out: thì **gửi lại cả 2 phân đoạn**. Không biết vẽ trên Word, tham khảo trang 241.
9. **Khác nhau giữa kỹ thuật quay lui và kỹ thuật gửi lại có chọn**

- Kỹ thuật quay lui:

+ Gửi lại tất cả các gói tin đã gửi sau gói tin bị lỗi

+ Tiêu tốn băng thông lớn nếu tỉ lệ lỗi cao

+ Ít phức tạp hơn. Bên gửi chỉ quản lý thứ tự, bên nhận chỉ có 1 tham số mong chờ

+ Bên nhận không lưu trữ gói tin đến sai thứ tự

+ Không phải tìm kiếm và phân biệt các gói tin (đã nhận/đã nhận, sai thứ tự/ …)

- Kỹ thuật gửi lại có chọn

+ Chỉ gửi lại những gói tin bị lỗi hoặt mất

+ Quản lý băng thông hiệu quả hơn, ít tiêu tốn vào việc gửi lại vô ích

+ Phức tạp hơn quay lui. Đòi hỏi xử lý logic và quản lý thứ tự tại bên gửi, quản lý thứ tự, tìm kiếm và lưu trữ bộ nhớ đệm tại bên nhận

+ Bên nhận lưu trữ gói tin đến sai thứ tự vào bộ đệm

+ Bên nhận phải tìm kiếm đúng gói tin cần gửi theo thứ tự và các gói tin liên tục theo sau ở trong bộ đệm

1. **So sánh giữa dừng chờ và pipeline**

- Kỹ thuật dừng chờ:

+ Truyền và nhận từng gói tin 1

+ Số gói tin có khả năng gửi đc cùng lúc: 1

+ Hiệu suất sử dụng băng thông thấp

+ Các gói tin gửi đi chắc chắn đúng thứ tự

+ Không cần xử lý bên gửi – bên nhận phức tạp

- Kỹ thuật pipeline:

+ Truyền đồng thời

+ Số gói tin có khả năng gửi được: nhiều gói tin

+ Hiệu suất sử dụng băng thông cao

+ Các gói tin gửi đi có thể sai thứ tự nếu lỗi

+ Phải xử lý cơ chế gửi – nhận nếu muốn truyền tin đáng tin cậy

1. **Khác nhau giữa kiểm soát luồng và kiểm soát tắc nghẽn**

- Kiểm soát luồng:

+ Kiểm soát **dung lượng gói tin** bên gửi, tránh làm tràn bộ đệm bên nhận

+ Kiểm soát bằng thuộc tính “cửa sổ nhận” = số bytes còn trống của bộ nhớ đệm

+ Cơ chế hoạt động 2 chiều, đơn giản bằng cách kiểm soát 2 thuộc tính cửa sổ nhận ở 2 bên khi giao tiếp

+ Các gói tin gửi đi chắc chắn đúng thứ tự

+ Không cần xử lý bên gửi – bên nhận phức tạp

- Kiểm soát tắc nghẽn:

+ Kiểm soát **tốc độ gửi gói tin** bên gửi, tránh làm tắc nghẽn xảy ra

+ Kiểm soát bằng số lượng gói tin ACK trùng, dấu hiệu mất gói, timeout

+ Cơ chế hoạt động phức tạp, gồm 3 pha khác nhau, chuyển đổi qua lại giữa 3 pha này nhờ các dấu hiệu. Các thuộc tính cần kiểm soát là : số gói ACK trùng, cửa sổ tắc nghẽn, isMất\_gói, ngưỡng cho phép (ssthresh)

1. **Thế nào là giai đoạn phục hồi nhanh**

+ Giai đoạn phục hồi nhanh xuất hiện khi có 3 gói tin ACK trùng nhau

+ Trong giai đoạn này, ssthresh= ½ cwnd; cwnd = ssthresh + 3

+ Tức tốc độ gửi giảm đi gần một nửa giá trị, chờ phục hồi gói tin ACK bị mất

+ Nếu nhận lại được gói tin ACK bị mất -> tránh tắc nghẽn

+ Nếu gói tin bị mất, hết thời gian chờ -> khởi động chậm

1. **Phân tích biểu đồ**
2. Khởi động chậm: Từ lần truyền 1-6; Từ lần truyền 23-sau
3. Tránh tắc nghẽn: Từ lần truyền 6-16; Từ lần truyền 17-22
4. Lượt truyền thứ 16, bên nhận nhận được 3 gói tin ACK giống nhau. Vì:

+ Sau lượt truyền thứ 16, lượt thứ 17 không rơi về pha khởi động chậm nên không xảy ra mất gói tin

+ Sau lượt truyền thứ 16, lượt 17-23 lại tăng tuyến tính tức là pha tránh tắc nghẽn, vậy sau lượt 16 là pha phục hồi nhanh, và trong pha này nhận được gói ACK bị mất

1. Sau lượt truyền thứ 22, pha chuyển từ tránh tắc nghẽn thành pha khởi động chậm với cwnd = 1 -> mất gói
2. Giá trị ssthresh ban đầu là 30 < ssthresh < 35. Vì vượt qua mốc này tại lượt thứ 6, pha khởi động chậm chuyển thành pha tránh tắc nghẽn
3. Tại lượt truyền thứ 18, ssthresh = ½ cwnd lúc mất gói. Giá trị này được thiết lập tại lượt thứ 16, khi chuyển từ pha tránh tắc nghẽn sang pha phục hồi nhanh. Giá trị ssthresh = >40/2

## TẦNG MẠNG

### Tổng quan về tầng mạng

+ Tầng Mạng cung cấp dịch vụ host-to-host

+ Dịch vụ của tầng Mạng sẽ cung cấp cho tầng Vận chuyển

+ Tầng Vận chuyển cung cấp kết nối logic giữa các tiến trình

+ Tầng Mạng cung cấp kết nối logic giữa các host

- Tại host bên gửi:

+Nhận segment từ tầng transport, đóng gói và gửi đi

- Tại host bên nhận:

+ Nhận packet từ tầng datalink, chuyển thành segment rồi gửi lên

- Tại router:

+ Dựa vào thông tin đích đến để chuyển packet đến host nhận

+ **Định tuyến** là xác định đường đi của gói tin từ nơi gửi đến nơi nhận

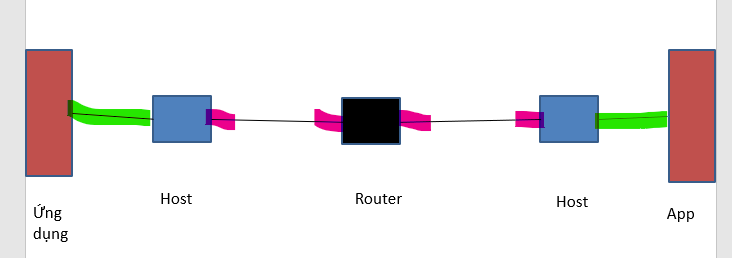
+ **Chuyển tiếp** là chuyển gói tin từ ngõ vào của bộ định tuyến, chuyển tiếp đến ngõ ra tương ứng (đúng đường đi)

\*Một bộ định tuyến cho nhiều ngõ vào/ra

Vai trò của tầng mạng

Công việc của tầng mạng

Định tuyến và chuyển tiếp



**Summary:**

Tô xanh: transport

Tím: network

### Mạng chuyển gói và mạch ảo

Tầng mạng cung cấp 2 loại dịch vụ

Mạch ảo

Mạch chuyển gói

+ Dịch vụ hướng kết nối (mạch ảo), cần thiết lập kết nối

+ Dịch vụ phi kết nối (mạch chuyển gói), không cần thiết lập kết nối

+ Một kiến trúc mạng máy tính chỉ dùng một loại dịch vụ

+ Thiết lập mạch và duy trì kết nối khi truyền dữ liệu

+ Mỗi link sẽ có một VC number

+ Mỗi gói tin sẽ có 1 VC ID

+ Khi gói tin đi qua router: thay thế VC ID của gói tin để đi tới link tiếp theo

+ Mạng tiêu biểu sử dụng: ATM, X.25, Frame-Relay

+ Không thiết lập mạch hay kết nối

+ Mỗi gói tin có thông tin địa chỉ đích

+ Mỗi router có 1 bảng định tuyến, địa chỉ đó -> đường đó

+ Dùng trong mạng internet

**Summary:**

### Giao thức DHCP

Công dụng của giao thức

Cách thức vận hành của DHCP

Cách bước tiến hành thiết lập DHCP cho máy mới

+ Tự động cấp phát IP động cho máy trong mạng

+ Tự động gán kèm mặt nạ mạng con (subnet mask)

+ Tự động gán kèm bộ định tuyến cổng ra

+ Tự động hoá việc cấu hình liên quan tới tầng mạng

+ DHCP là một giao thức khách – chủ. Sẽ có 1 máy chủ DHCP và các máy con trong mạng.

+ DHCP có một pool địa chỉ mạng riêng được phép cấp

+ DHCP quản lý danh sách này xem địa chỉ nào còn dư để cấp

+ Mỗi khi có máy gia nhập/thoát, DHCP sẽ cập nhật lại danh sách này. Mỗi khi có máy gia nhập, DHCP cấp phát 1 địa chỉ valid cho nó

+ Client kết nối vào mạng, phát ra gói tin DHCP Discover.

+ **DHCP Discover:** Tìm kiếm máy chủ DHCP trong mạng bằng cách gửi đi gói tin UDP này đến tất cả các nút mạng nằm trong mạng.

+ Máy chủ sau khi nhận được gói tin Discover, gửi cho máy khách gói tin DHCP Offer.

+ **DHCP Offer:** Gửi lại các thông tin IP dự định cấp, thời gian hiệu lựa, ID của gói tin Discorver. Gửi bằng UDP và tới tất cả các nút mạng

+ Client sau khi nhận được các gói tin DHCP Offer (nếu mạng có nhiều máy chủ DHCP), lựa chọn một trong số đó, thường sẽ là gói tin tới sớm nhất. Và gửi lại gói DHCP Request. (Vẫn là gửi broadcast)

+ **DHCP Request:** Hồi đáp gói DHCP Offer, tb tới server, chấp nhận IP dự định được cấp, yêu cầu DHCP server cấp IP cho mình.

+ **DHCP ACK**: Trả lời gói tin DHCP Request bằng 1 gói báo nhận ACK, xoá địa chỉ IP trong DHCP Offer ra khỏi bảng địa chỉ, xác nhận các tham số được yêu cầu.

### 

**Summary:** DHCP dùng port 68 ở client và 67 ở server

### Giao thức NAT

Vai trò của NAT

Cơ chế chính của NAT

Static NAT:

Dynamic NAT

Overloading NAT

Overlapping NAT nằm ở cuối chương

Cơ chế dồn kênh và phân kênh hướng kết nối

+ Nhiệm vụ chính là chuyển tiép gói tin giữa các lớp mạng khác nhau

+ NAT có nhiệm vụ thông dịch IP cho gói tin

+ Thường NAT sẽ đổi IP Private thành IP Public

+ NAT giúp cho tiết kiệm IP, gom nhiều máy về dùng chung 1 IP

+ NAT cũng đóng vai trò như là 1 firewall, chặn, gói tin gửi đến/gửi từ một địa chỉ nào đó hoặc truy cập vào port cấm

+ NAT có một bảng chuyển đổi riêng để chuyển đổi IP

+ IP đi qua NAT sẽ được chuyển đổi theo đúng chiều của nó

+ Một IP private map với 1 IP public

+ NAT tĩnh dùng khi thiết bị cần **truy cập từ bên ngoài mạng**.

+ VD: máy B cần điều khiển từ máy A ngoài mạng con, mạng con của máy B dùng NAT tĩnh

+ Một địa chỉ IP private map với nhiều địa chỉ Public (1 nhóm)

+ Xử lý khi địa chỉ public bị bận, không truy cập được

+ VD: máy A bị hư, dùng máy C để điều khiển máy B, A, C đều ngoài mạng con của máy B. mạng máy B dùng NAT động

+ Nhiều địa chỉ IP private kết nối tới 1 địa chỉ IP public

+ Phân biệt với nhau bằng port. 1 IP pri map tới IP public qua 1 port nào đó. VD: 192.168.1.10 -> 172.18.123.100:10

\*IP private là IP chỉ sử dụng nội bộ trong mạng con, không có tác dụng liên lạc ra ngoài, muốn liên lạc ra ngoài dùng IP public

\*IP public là IP thực tế để giao tiếp ra bên ngoài

### 

**Summary:** Đi qua NAT thì IP private sẽ đổi thành IP public, mạng Internet thường dùng Overload

### Giao thức ICMP

Vai trò của ICMP

Nguyên tắc vận hành

ICMP gửi thông báo lỗi khi nào

Ứng dụng của ICMP thực tế

- Tầng mạng gồm 3 phần: giao thức IP dùng để định danh;

+ Giao thức định tuyến dùng để định tuyến gói tin

+ Giao thức ICMP dùng để thông báo lỗi và báo hiệu bộ định tuyến

+ Được xem như một giao thức ở phía trên lớp IP

+ Gói tin ICMP được đóng gói ngay trong dữ liệu IP. Mô hình là gói tin tầng mạng (IP header -> ICMP header nằm phía trong)

+ Có nhiều kiểu tin ICMP, bảng 4.17 trang 139

+ ICMP gồm một trường mã và một trường checksum

+ Phối hợp với IP header sẽ biết thiết bị nào báo lỗi

+ Gửi thông báo lỗi khi: datagram không đến đích, timeout, lỗi trong header, router host bị nghẽn

+ Không gửi thông báo lỗi khi: Bản thân gói tin ICMP lỗi, các gói dữ liệu định tuyến (broadcast, multicast), …

+ Điều khiển luồng dữ liệu.: Khi dòng dữ liệu gửi tới quá nhanh gây tắc nghẽn, tràn bộ đệm, trạm đích gửi thông báo ICMP báo tạm ngừng truyền thông tin

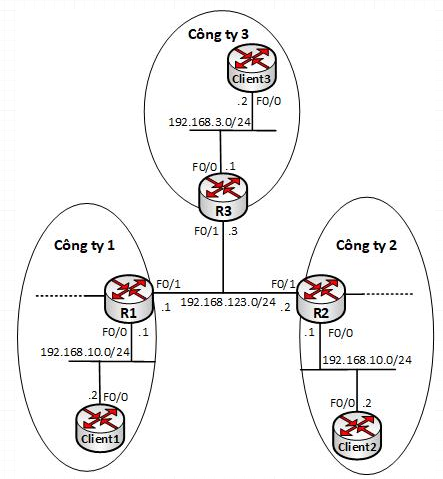
+ Thông báo lỗi: Báo lỗi khi không tìm thấy trạm đích, hoặc không tìm thấy port về lại máy nguồn

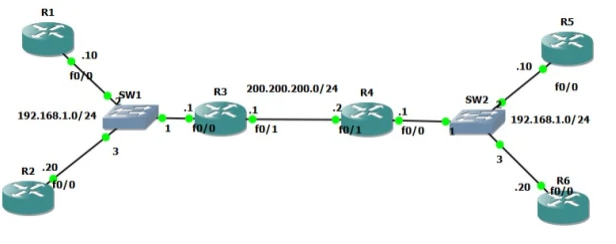
+ Kiểm tra xem máy có còn tồn tại/hoạt động không (lệnh ping)

**Summary:** ICMP là giao thức quan trọng của tầng mạng dùng để kiểm soát việc truyền tin và báo lỗi trong quá trình truyền, gói tin ICMP được đóng gói trong datagram, nằm dưới IP header

## GIẢI BÀI TẬP MẠNG (SÁCH trang 388)

+ **Overlapping NAT**: Được dùng khi **xảy ra trùng lặp subnet IP giữa 2 máy trong cùng hệ thống mạng (do dùng chung 1 IP public khi gộp 2 network hoặc IP nội bộ của mạng này là IP public của mạng khác, như hình dưới R1-R2 trùng lặp IP, hoặc hình 2).**





* Lúc này NAT sẽ **thực hiện overlapping IP**, khi **ở trong** mạng của R1(hình 1)/R3 (hình 2), cụ thể là client1/r1+r2 thì được các subnet khác như client3+client2/r5+r6 nhìn thấy là địa chỉ IP của R1, chứ không phải là địa chỉ IP của subnet.
* Ví dụ:

1. **Hai/ba tính năng quan trọng nhất của tầng mạng trong mạng chuyển gói/mạch ảo:**

+ Mạch chuyển gói: chuyển tiếp và định tuyến

+ Mạch chuyển mạch ảo: chuyển tiếp, định tuyến và thiết lập kết nối

1. **Sự khác nhau giữa routing và forwading**

+ Chuyển tiếp là sử dụng dữ liệu cục bộ, chuyển dữ liệu/gói tin từ cổng vào đến cổng ra tương ứng như đường đi của nó

+ Định tuyến là sử dụng dữ liệu toàn cục, xác định đường đi của dữ liệu.

1. **Bộ định tuyến có địa chỉ IP không**

+ Về lý thuyết, bộ định tuyến có thể có nhiều địa chỉ IP

+ Mỗi địa chỉ IP tương ứng với một cổng giao tiếp đang hoạt động

+ Bao nhiêu cổng giao tiếp đang hoạt động thì bấy nhiêu địa chỉ IP

1. **So sánh thuật toán định tuyến vectơ khoảng cách và trạng thái liên kết**

- Thuật toán vector khoảng cách:

+ Sử dụng thuật toán Bellman Ford

+ Tính toán đường đi ngắn nhất được thực hiện liên tục

+ Một nút chỉ biết nút lân cận mà nó phải chuyển tiếp gói tin đi qua để đến đích theo đường ngắn nhất đã tính toán

+ Tính toán dựa theo số hops ít nhất

+ Cập nhât bằng broadcast

- Thuật toán trạng thái liên kết:

+ Sử dụng thuật toán Dijsktra

+ Sử dụng thông tin trên toàn hệ thống mạng

+ Tính toán dựa trên chi phí ít nhất

+ Cập nhật bằng multicast

1. **Có nhất thiết mỗi vùng tự trị - AS cần dùng chung một thuật toán tìm đường**

Không nhất thiết, mỗi vùng tự trị là một vùng độc lập, vùng tự trị có quyền tự chủ trong việc quản lý định tuyến.

1. **So sánh thuật toán RIP và OSPF**

- Thuật toán RIP:

+ Sử dụng thuật toán định tuyến vectơ khoảng cách

+ Dựa trên tính toán: đi qua ít hop nhất

+ Địa chỉ multicast sử dụng để cập nhật: 224.0.0.9 (v2) hoặc broadcast (v1)

+ Giao thức và port sử dụng: UDP/port 20

+ Cập nhật mỗi 30s

- Thuật toán trạng thái liên kết:

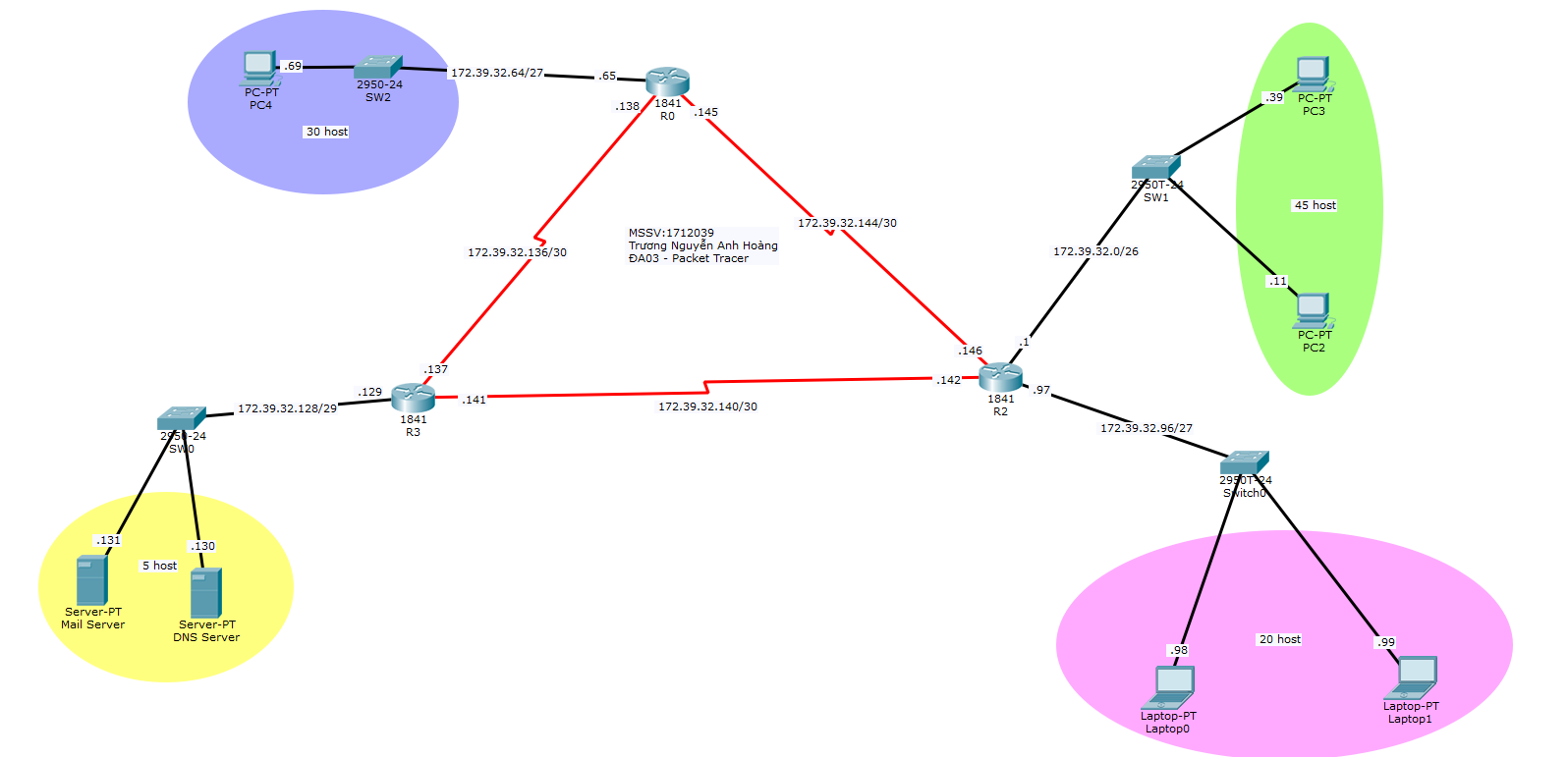
+ Sử dụng thuật toán trạng thái liên kết

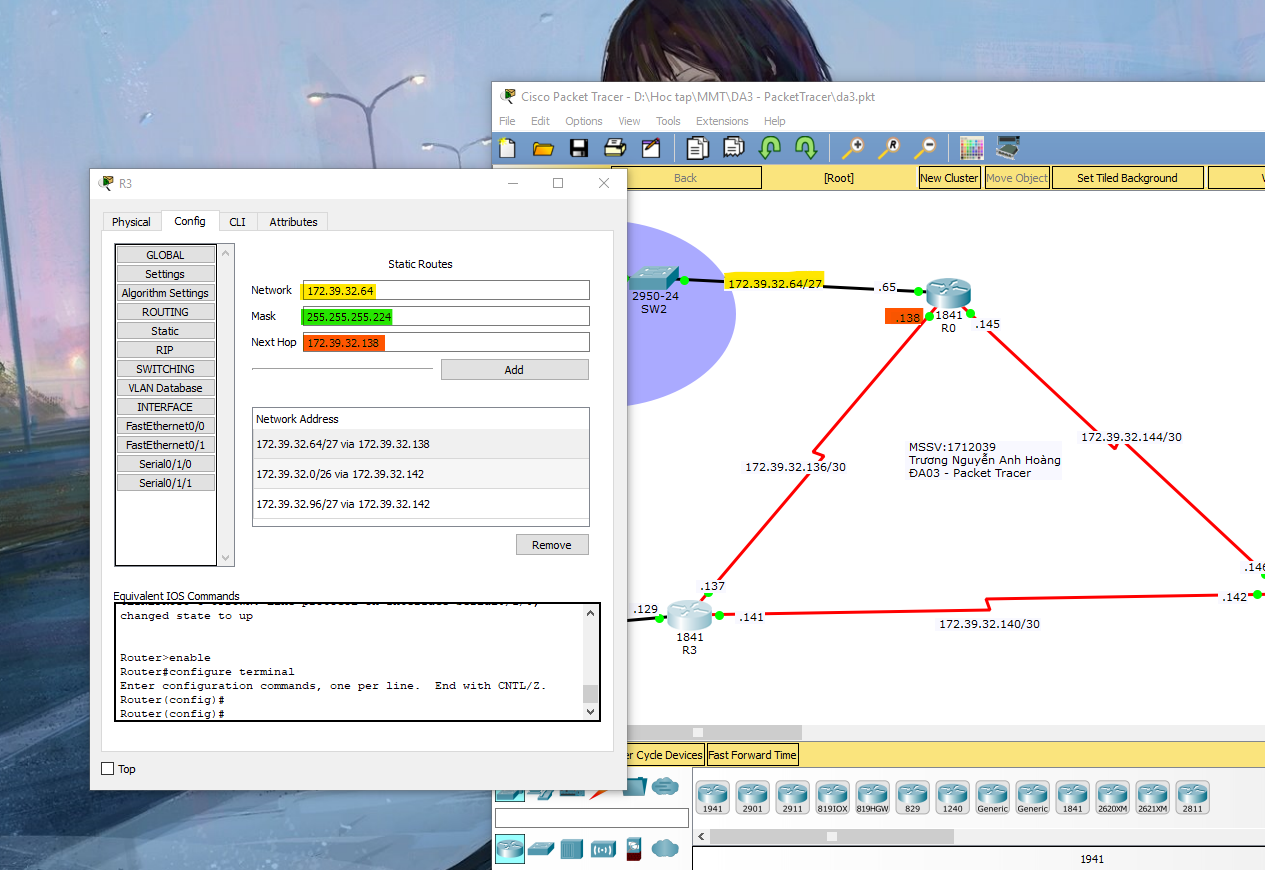
+ Dựa trên băng thông để tính toán

+ Địa chỉ multicast sử dụng 224.0.0.5 or .6

+ Giao thức và port sử dụng: IP và port 89

+ Cập nhật khi có thay đổi xảy ra





|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Cấp cho | Size | Địa chỉ đường mạng | Subnet Mask | Dải IP (Start IP – End IP) |
| 1 | 45 host | 64 | 172.39.32.0/26 | 255.255.255.192 | 172.39.32.1 – 172.39.32.62 |
| 2 | 30 host | 32 | 172.39.32.64/27 | 255.255.255.224 | 172.39.32.65 – 172.39.32.94 |
| 3 | 20 host | 32 | 172.39.32.96/27 | 255.255.255.224 | 172.39.32.97 – 172.39.32.126 |
| 4 | 5 host | 8 | 172.39.32.128/29 | 255.255.255.248 | 172.39.32.129 – 172.39.32.134 |
| 5 | 2 host | 4 | 172.39.32.136/30 | 255.255.255.252 | 172.39.32.137 – 172.39.32.138 |
| 6 | 2 host | 4 | 172.39.32.140/30 | 255.255.255.252 | 172.39.32.141 – 172.39.32.142 |
| 7 | 2 host | 4 | 172.39.32.144/30 | 255.255.255.252 | 172.39.32.145 – 172.39.32.146 |

