

Trường Đại Học Bách Khoa Tp.HCM
Hệ Đào Tạo Từ Xa
Khoa Khoa Học và Kỹ Thuật Máy Tính

Mạng máy tính căn bản

Bài giảng 6: Tầng truyền tải

Tham khảo:

Chương 3: “Computer Networking – A top-down approach”
Kurose & Ross, 5th ed., Addison Wesley, 2010.

Chương 3: Tầng truyền tải

Mục tiêu:

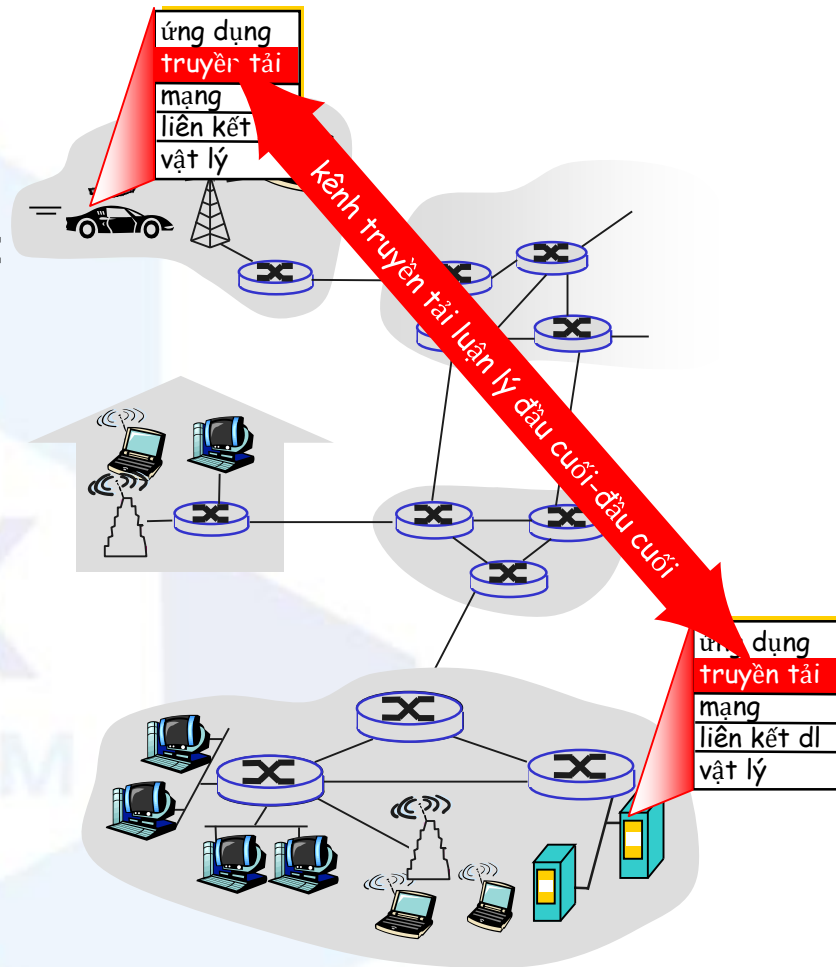
- hiểu rõ các nguyên lý đằng sau các dịch vụ của tầng truyền tải:
 - dồn/tách
 - truyền tải dữ liệu tin cậy
 - kiểm soát lưu lượng
 - kiểm soát tắc nghẽn
- tìm hiểu về các giao thức tầng truyền tải trong Internet:
 - UDP: truyền tải không kết nối
 - TCP: truyền tải hướng kết nối
 - Kiểm soát tắc nghẽn trong TCP

Chương 3: Mục lục

- 3.1 Các dịch vụ tăng-truyền tải
- 3.2 Sự dồn và tách
- 3.3 Sự truyền tải không kết nối: UDP
- 3.4 Sự truyền tải hướng kết nối : TCP
 - cấu trúc đoạn tin
 - truyền tải dữ liệu tin cậy
 - kiểm soát lưu lượng
 - quản lý kết nối
- 3.5 Các nguyên lý của kiểm soát tắc nghẽn
- 3.6 Kiểm soát tắc nghẽn trong TCP

Giao thức và dịch vụ truyền tải

- cung cấp một *kênh liên lạc luận lý* giữa các tiến trình ứng dụng trên những máy khác nhau
- các giao thức truyền tải chạy trên các máy đầu cuối
 - phía gửi: chia thông điệp của ứng dụng thành *những đoạn (segment)*, đẩy xuống tầng mạng
 - phía nhận: ráp các đoạn lại thành thông điệp hoàn chỉnh, đẩy lên tầng ứng dụng
- các giao thức truyền tải
 - Internet: TCP và UDP



Tầng truyền tải so với Tầng mạng

- *tầng mạng*: kênh liên lạc luận lý giữa các **máy**
- *tầng truyền tải*: kênh liên lạc luận lý giữa các **tiến trình**
 - phụ thuộc vào các dịch vụ tầng mạng đồng thời củng cố chúng

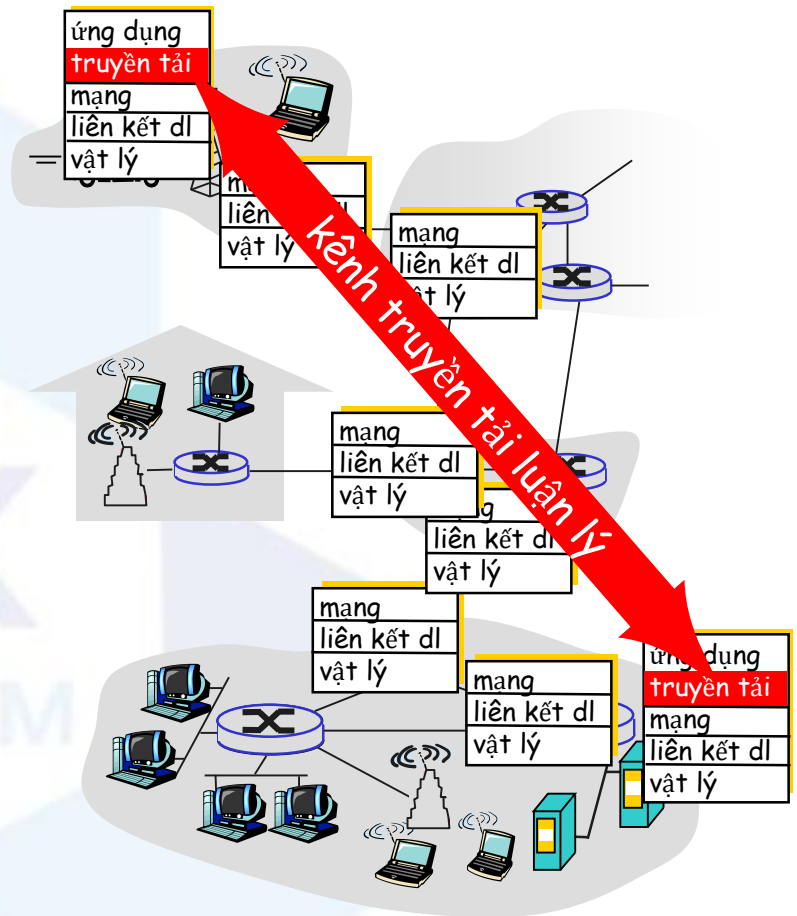
Ví dụ tương đồng – chủ gia đình:

12 đứa trẻ gửi thư cho 12 đứa

- tiến trình = đứa trẻ
- thông điệp ú/d = thư trong phong bì
- máy tính = nhà
- giao thức truyền tải = Ann và Bill
- giao thức mạng = dịch vụ bưu điện

Các giao thức tầng truyền tải trong Internet

- truyền tải tin cậy, có trật tự (TCP)
 - kiểm soát tắc nghẽn
 - kiểm soát lưu lượng
 - thiết lập kết nối
- truyền tải không tin cậy, không có trật tự (UDP)
 - một sự mở rộng của giao thức IP "nỗ lực hết sức" trên tầng truyền tải
- những dịch vụ chưa sẵn sàng:
 - đảm bảo độ trễ
 - đảm bảo băng thông



Chương 3: Mục lục

- 3.1 Các dịch vụ tăng-truyền tải
- 3.2 Sự dồn và tách
- 3.3 Sự truyền tải không kết nối: UDP
- 3.4 Sự truyền tải hướng kết nối : TCP
 - cấu trúc đoạn tin
 - truyền tải dữ liệu tin cậy
 - kiểm soát lưu lượng
 - quản lý kết nối
- 3.5 Các nguyên lý của kiểm soát tắc nghẽn
- 3.6 Kiểm soát tắc nghẽn trong TCP


Dồn/Tách (Multiplexing/Demultiplexing)

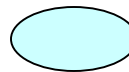
Tách ở máy nhận:

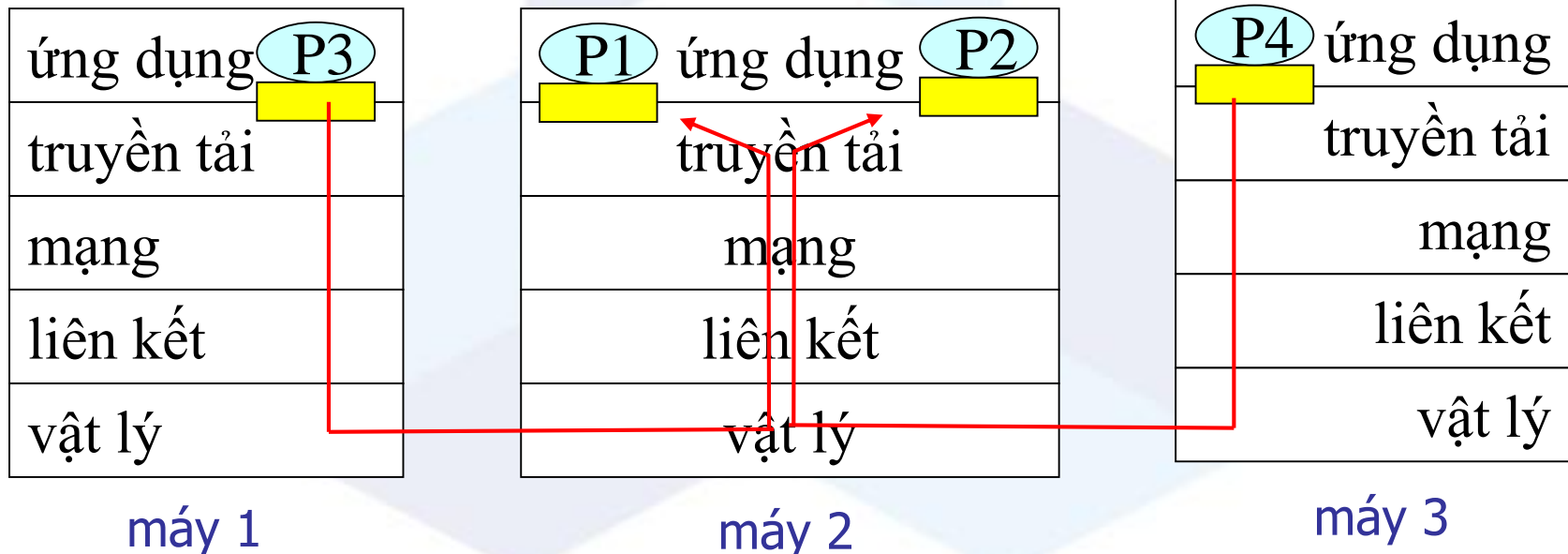
chuyển các đoạn nhận được tới đúng các socket

Dồn ở máy gửi:

thu thập dữ liệu từ nhiều socket, đóng gói dữ liệu với phần mào đầu (được sử dụng sau này để tách)

 = socket

 = tiến trình

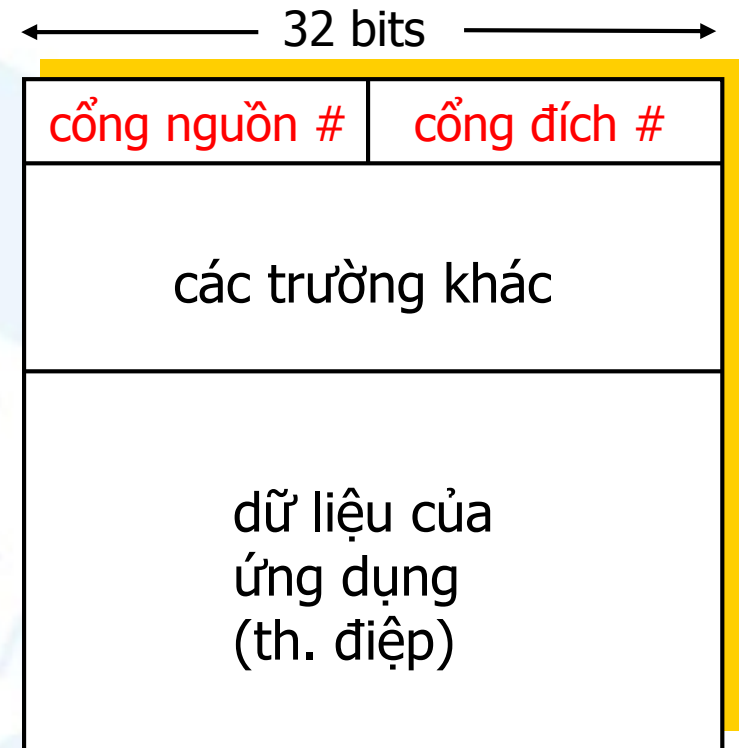


Phương pháp tách làm việc ntn?

- máy nhận gói tin IP

- mỗi gói tin có địa chỉ IP nguồn, địa chỉ IP đích
- mỗi gói tin mang một đoạn dữ liệu tầng truyền tải
- mỗi đoạn có cổng của máy nhận (đích) và máy gửi (nguồn)

- máy dùng địa chỉ IP và số cổng để chuyển hướng đoạn dữ liệu sang socket thích hợp



định dạng đoạn dữ liệu TCP/UDP

Sự tách không kết nối (connectionless)

- Tạo các socket với các cổng:

```
DatagramSocket mySocket1 =  
    new DatagramSocket(12534);  
DatagramSocket mySocket2 =  
    new DatagramSocket(12535);
```

- Socket UDP được xác định bởi cặp:

(địa chỉ IP đích, số cổng đích)

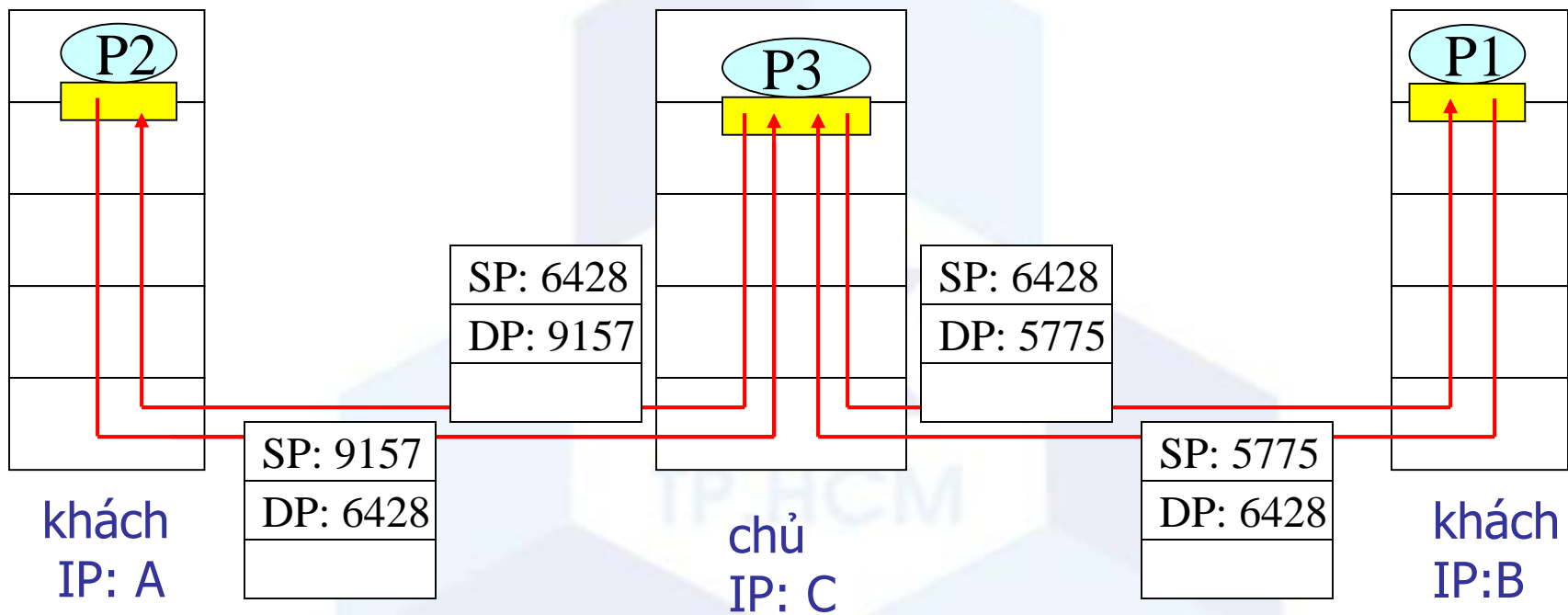
- Khi máy nhận được một đoạn UDP:

- kiểm tra số cổng đích trong đoạn dữ liệu
- chuyển hướng đoạn UDP tới socket với số cổng đó

- Gói tin IP với địa chỉ IP nguồn khác nhau và/hoặc số cổng nguồn khác nhau có thể được chuyển tới cùng một socket

Sự tách không kết nối (tt)

```
DatagramSocket serverSocket = new DatagramSocket(6428);
```

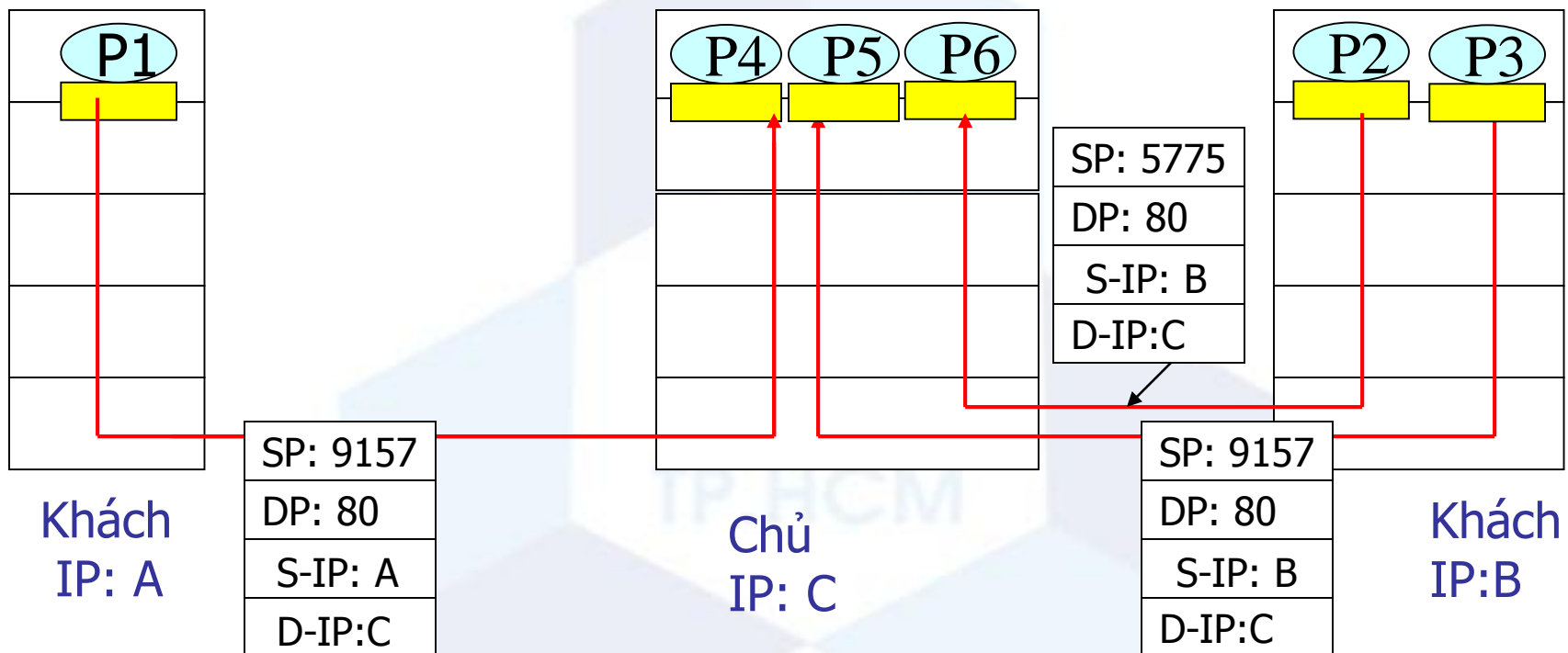


SP (Source Port) cung cấp "địa chỉ phản hồi"

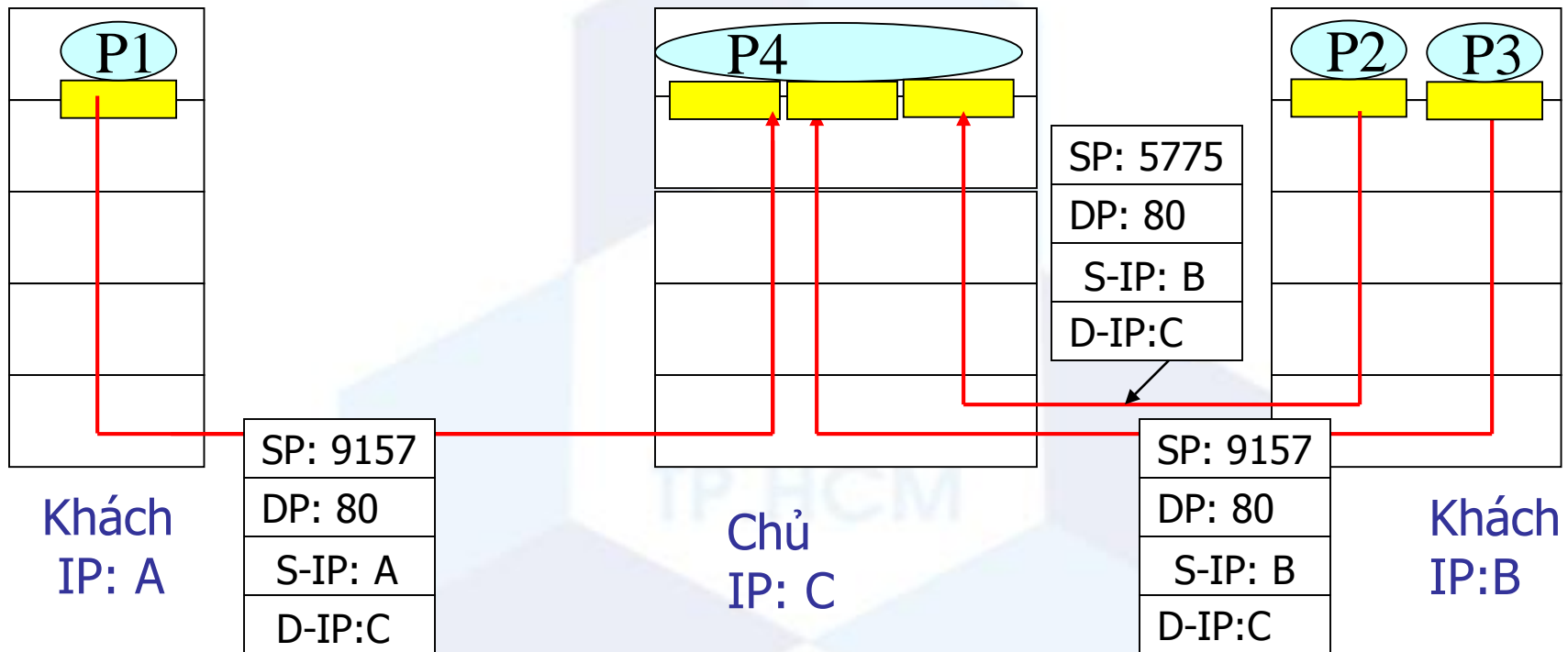
Sự tách hướng kết nối (connection oriented)

- Socket TCP được xác định bởi 4 nhân tố:
 - địa chỉ IP nguồn
 - số cổng nguồn
 - địa chỉ IP đích
 - số cổng đích
- máy nhận dùng cả 4 giá trị trên để định hướng đoạn dữ liệu tới đúng Socket phù hợp
- Máy chủ có thể hỗ trợ đồng thời nhiều socket TCP:
 - mỗi socket được định danh bởi 4 nhân tố của nó
- Máy chủ Web có nhiều socket khác nhau cho mỗi khách kết nối tới
 - HTTP không ổn định sẽ có một socket riêng biệt cho mỗi người dùng

Sự tách hướng kết nối (tt)



Sự tách hướng kết nối: Máy chủ Web chia luồng



Chương 3: Mục lục

- 3.1 Các dịch vụ tầng-truyền tải
- 3.2 Sự dồn và tách
- 3.3 Sự truyền tải không kết nối: UDP
- 3.4 Sự truyền tải hướng kết nối : TCP
 - cấu trúc đoạn tin
 - truyền tải dữ liệu tin cậy
 - kiểm soát lưu lượng
 - quản lý kết nối
- 3.5 Các nguyên lý của kiểm soát tắc nghẽn
- 3.6 Kiểm soát tắc nghẽn trong TCP

UDP: User Datagram Protocol [RFC 768]

- là giao thức truyền tải Internet protocol “không phức tạp”, “giản thiểu”
- dịch vụ “nỗ lực tối đa”, đoạn UDP có thể:
 - bị mất
 - được giao không đúng trật tự cho ứng dụng
- *không-kết-nối:*
 - không có bắt tay giữa người gửi và người nhận UDP
 - mỗi đoạn UDP được xử lý độc lập với những đoạn khác

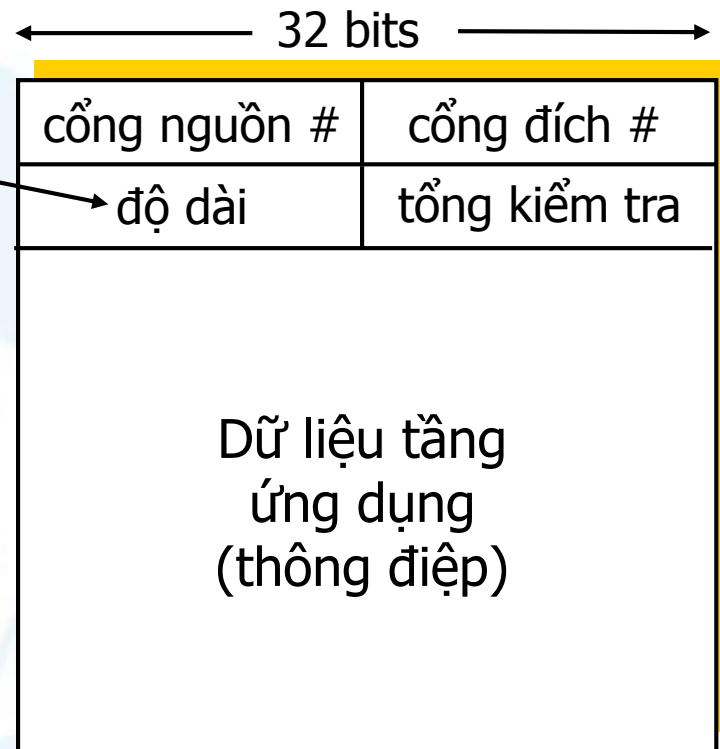
Tại sao cần có UDP?

- không thiết lập kết nối (giảm độ trễ)
- đơn giản: không có các trạng thái kết nối ở người gửi và người nhận
- đoạn mào đầu ngắn (tiết kiệm dung lượng)
- không có kiểm soát tắc nghẽn: UDP có thể truyền đi với tốc độ tối đa

UDP (tt)

- thường được dùng cho các ứng dụng đa phương tiện trực tuyến
 - khả năng chịu mất gói
 - dễ thay đổi tốc độ
- những cách dùng UDP khác
 - DNS
 - SNMP
- truyền tải tin cậy qua UDP: bổ sung tính tin cậy ở tầng ứng dụng
 - cơ chế kiểm soát lỗi thuộc tầng ứng dụng!

độ dài, của
đoạn UDP,
bao gồm cả
mào đầu



định dạng đoạn UDP

Tổng kiểm tra UDP (checksum)

Mục đích: phát hiện lỗi (vd: nhảy bit) trong đoạn dữ liệu được truyền tải

Người gửi:

- xem đoạn nội dung dữ liệu như là một chuỗi gồm những số nguyên 16-bit
- tổng kiểm tra (TKT): cộng (tổng bù 1) toàn bộ nội dung của đoạn từng 16-bit
- người gửi đặt giá trị tổng kiểm tra vào trường "tổng kiểm tra - checksum" UDP

Người nhận:

- tính tổng kiểm tra của đoạn nhận được
- kiểm tra xem TKT tính được có bằng giá trị trong trường TKT ko:
 - Ko – phát hiện ra lỗi.
 - Có – ko phát hiện ra lỗi.
nhưng vẫn có thể có lỗi

Ví dụ TKT Internet

■ Ghi chú

- Khi cộng các số, số nhớ (nếu có) của bit có thứ hạng cao nhất cần phải được cộng dồn vào kết quả

■ Ví dụ: cộng hai số nguyên 16-bit

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

cộng dồn **1** 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1

tổng 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 0

tổng kiểm tra 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1

Chương 3: Mục lục

- 3.1 Các dịch vụ tăng-truyền tải
- 3.2 Sự dồn và tách
- 3.3 Sự truyền tải không kết nối: UDP
- 3.4 Sự truyền tải hướng kết nối : TCP
 - cấu trúc đoạn tin
 - truyền tải dữ liệu tin cậy
 - kiểm soát lưu lượng
 - quản lý kết nối
- 3.5 Các nguyên lý của kiểm soát tắc nghẽn
- 3.6 Kiểm soát tắc nghẽn trong TCP

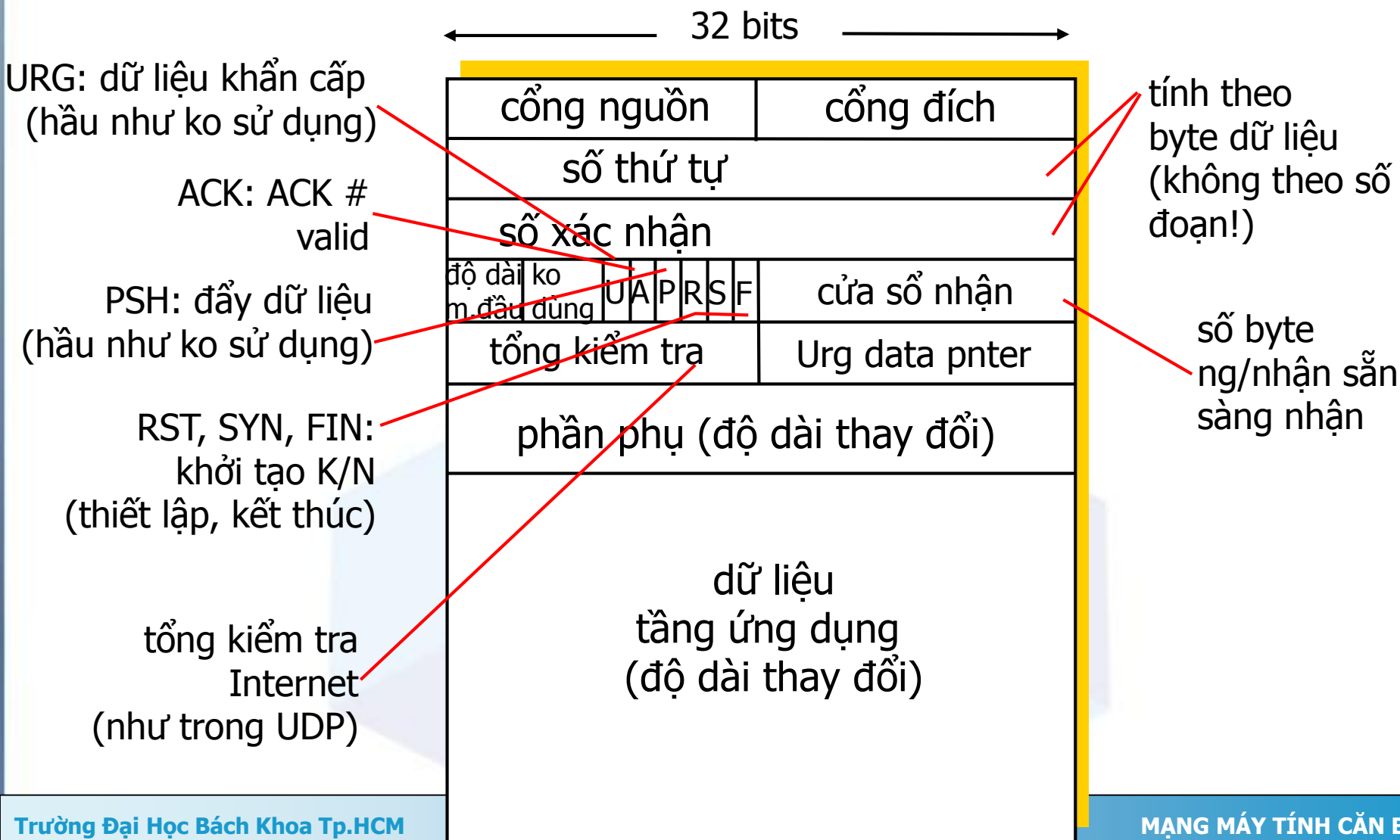
TCP: Tổng quát

RFCs: 793, 1122, 1323, 2018, 2581

- **điểm-tới-điểm:**
 - 1 n/gửi, 1 n/nhận
- **luồng byte tin cậy, theo thứ tự:**
 - ko "biên giới giữa th/điệp"
- **được tạo đường ống:**
 - kiểm tra tắc nghẽn TCP và lưu lượng q/đ kích thước cửa sổ
- **bộ nhớ tạm gửi & nhận**
- **dữ liệu song công (full-duplex):**
 - dữ liệu di chuyển theo 2 hướng trong cùng một kết nối
 - MSS: kích thước đoạn tối đa
- **định hướng kết nối:**
 - bắt tay (trao đổi các th/đ điều khiển) khởi tạo trạng thái của ng/gửi, ng/nhận trước khi trao đổi dữ liệu
- **lưu lượng đc kiểm tra:**
 - n/ gửi sẽ không làm tràn người nhận



Cấu trúc đoạn TCP



TCP số thứ tự (STT) và số ACK

STT (sequence number):

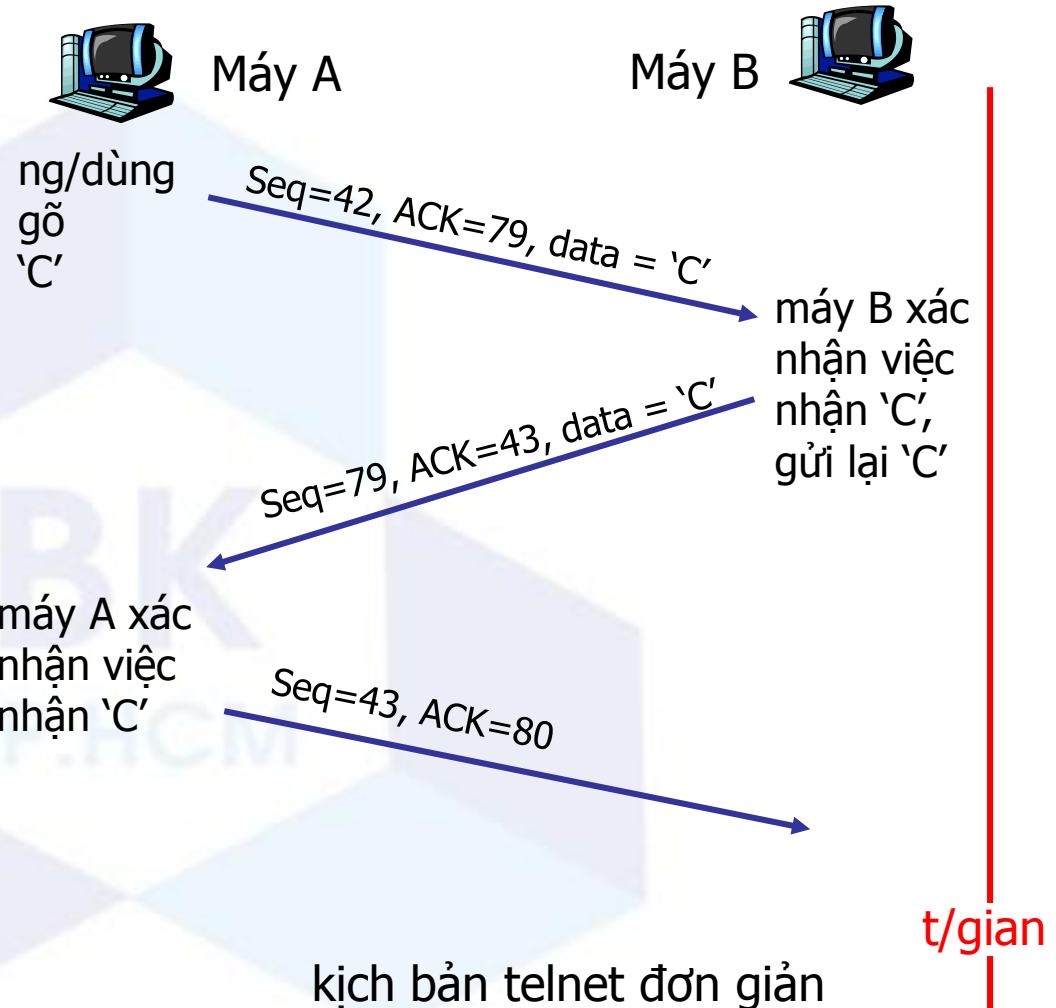
- số thứ tự trong luồng byte của byte đầu tiên trong đoạn

ACKs:

- là STT của byte tiếp theo mà sẽ nhận được từ máy bên kia
- ACK cộng dồn

Hỏi: làm sao xử lý những đoạn không đúng thứ tự

- đáp: TCP ko chỉ rõ – công việc của nhà hiện thực



Thời gian xoay vòng và thời gian chờ TCP

Hỏi: thiết lập giá trị thời gian chờ TCP như thế nào?

- dài hơn RTT
 - nhưng RTT thay đổi
- quá ngắn: thời gian chờ non, gói tin phản hồi chưa kịp quay về
 - không cần thiết phải truyền lại
- quá dài: phản ứng chậm cho việc mất đoạn

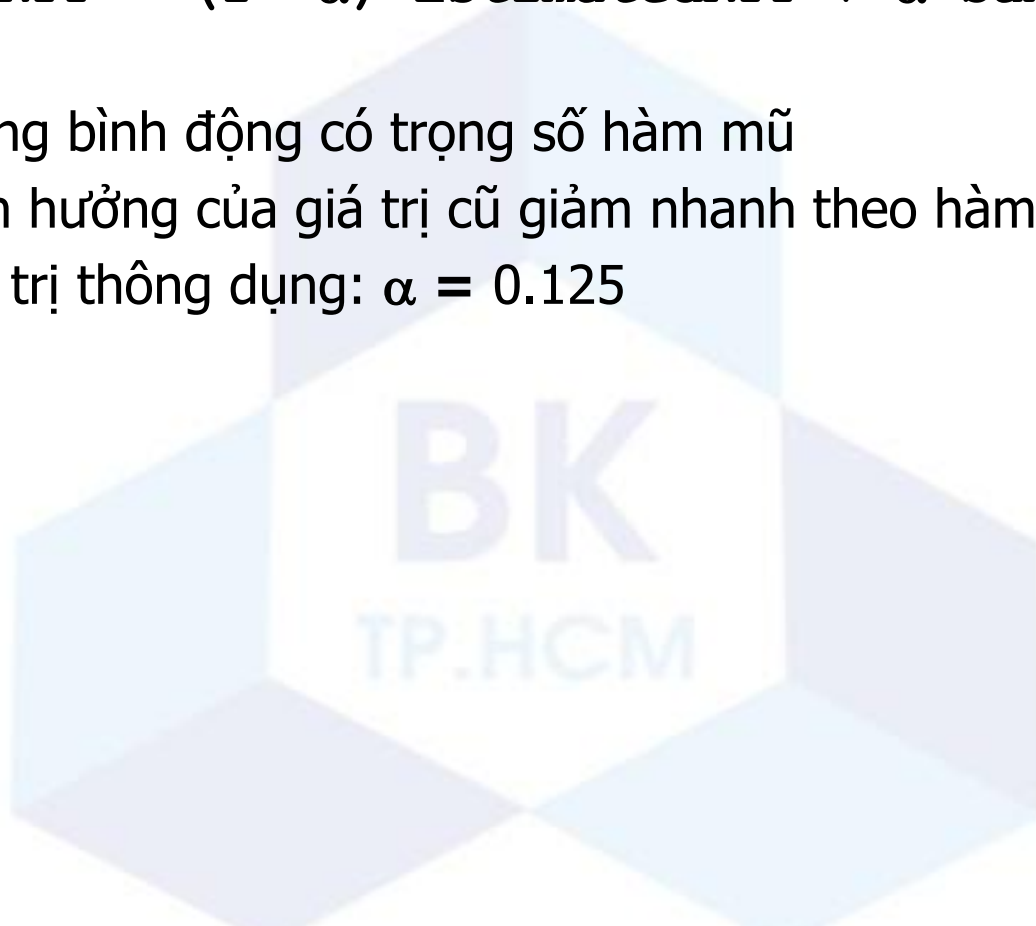
Hỏi: làm sao để đo RTT?

- **SampleRTT**: đo thời gian từ khi truyền gói tin đi và nhận được ACK
 - bỏ qua truyền tải lại
- **SampleRTT** sẽ thay đổi, muốn RTT đo được “mượt hơn”
 - lấy giá trị trung bình của những lần đo gần nhất, không chỉ giá trị hiện thời của **SampleRTT**

Thời gian xoay vòng và thời gian chờ

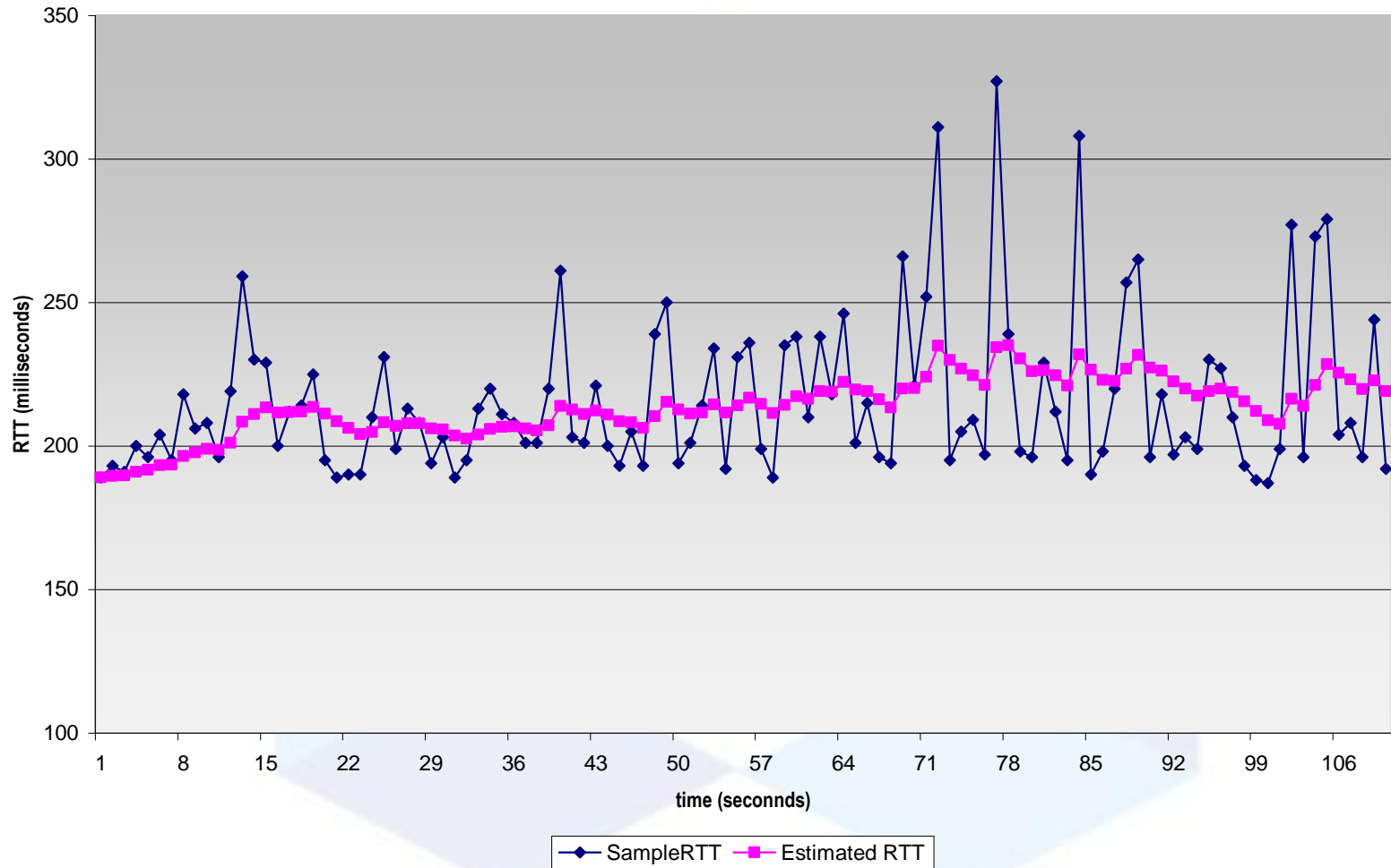
$$\text{EstimatedRTT} = (1 - \alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$$

- trung bình động có trọng số hàm mũ
- ảnh hưởng của giá trị cũ giảm nhanh theo hàm mũ
- giá trị thông dụng: $\alpha = 0.125$



Ví dụ đo RTT:

RTT: gaia.cs.umass.edu to fantasia.eurecom.fr



Thời gian xoay vòng và thời gian chờ

Thiết lập t/g chờ

- **EstimatedRTT** + “biên an toàn”
 - **EstimatedRTT** thay đổi với biên độ lớn -> biên an toàn lớn
- đầu tiên đo độ biến thiên của **EstimatedRTT** so với **SampleRTT** :

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

(thông thường, $\beta = 0.25$)

sau đó thiết lập khoảng t/g chờ:

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

Chương 3: Mục lục

- 3.1 Các dịch vụ tăng-truyền tải
- 3.2 Sự dồn và tách
- 3.3 Sự truyền tải không kết nối: UDP
- 3.4 Sự truyền tải hướng kết nối : TCP
 - cấu trúc đoạn tin
 - truyền tải dữ liệu tin cậy
 - kiểm soát lưu lượng
 - quản lý kết nối
- 3.5 Các nguyên lý của kiểm soát tắc nghẽn
- 3.6 Kiểm soát tắc nghẽn trong TCP

Truyền tải dữ liệu tin cậy TCP

- TCP tạo dịch vụ ttdltc trên nền dịch vụ không tin cậy IP
- Các đoạn dữ liệu được tạo đường ống
- cơ chế ACK cộng dồn
- TCP chỉ sử dụng một bộ đếm thời gian cho truyền tải lại
- Truyền tải lại được kích hoạt bởi:
 - sự kiện hết thời gian chờ
 - trùng lặp ACK
- Đầu tiên xem xét ng/gửi TCP đơn giản:
 - bỏ qua các ack trùng lặp
 - bỏ qua kiểm tra lưu lượng, kiểm tra tắc nghẽn

Các sự kiện phía người gửi TCP:

nhận dữ liệu từ ứ/d:

- Tạo ra đoạn với STT
- STT là số thứ tự trên luồng-byte của byte dữ liệu đầu tiên trong đoạn
- khởi động bộ đếm t/g nếu nó chưa chạy (bộ đếm t/g cho đoạn dữ liệu chưa ACK lâu nhất)
- khoảng t/g hết hạn:
TimeoutInterval

hết giờ:

- gửi lại đoạn dữ liệu mà gây hết t/g chờ
- khởi động lại bđtg

Nhận được ACK:

- Nếu đó là ACK cho các đoạn trước đó chưa được ACK
 - cập nhật danh sách các gói đã được ACK
 - chạy lại bđtg nếu như còn có các đoạn chưa ACK

```
NextSeqNum = InitialSeqNum  
SendBase = InitialSeqNum
```

```
loop (forever) {  
    switch(event)  
    event: nhận được dữ liệu từ ứng dụng tầng trên  
           tạo ra đoạn TCP với STT NextSeqNum  
           if (bđtg không chạy)  
               khởi chạy bđtg  
           đẩy đoạn xuống IP  
           NextSeqNum = NextSeqNum + length(data)  
  
    event: bđtg hết giờ  
           gửi lại đoạn chưa ACK với STT nhỏ nhất  
           khởi chạy bđtg  
  
    event: nhận được ACK, với giá trị trường ACK là y  
           if (y > SendBase) {  
               SendBase = y  
               if (còn đoạn chưa ACK)  
                   khởi chạy bđtg  
           }  
} /* end of loop forever */
```

người gửi TCP (đơn giản hóa)

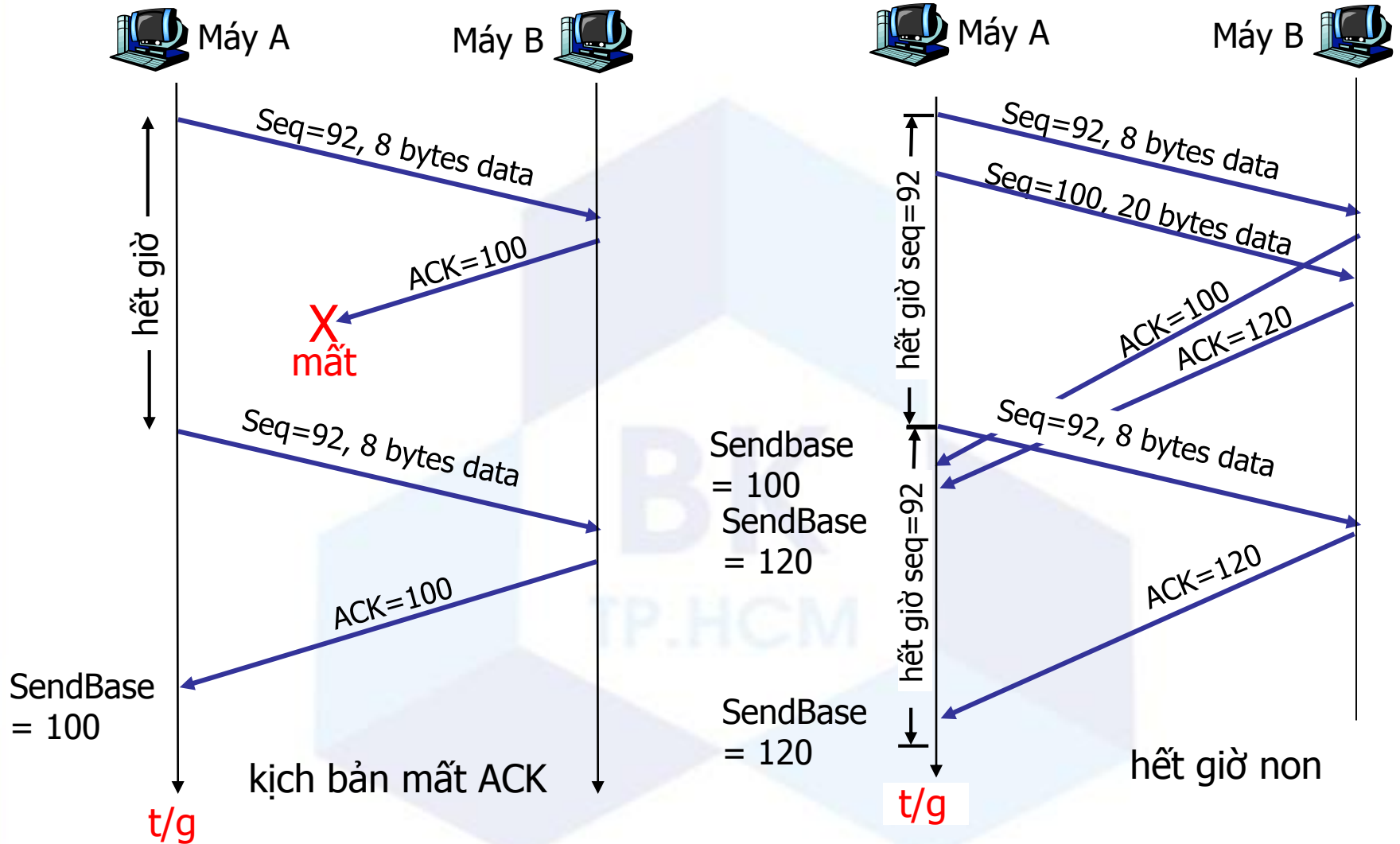
Chú thích:

- SendBase-1: byte được ack cộng dồn cuối cùng

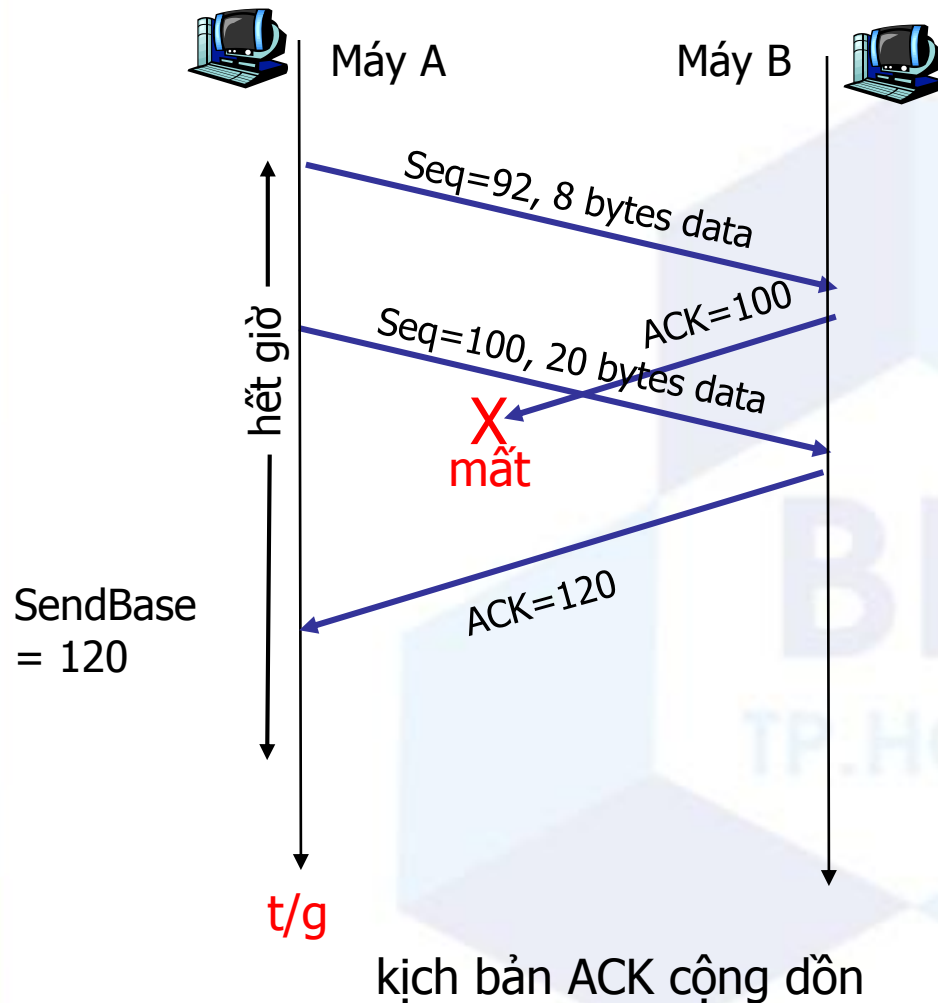
Ví dụ:

- SendBase-1 = 71;
y = 73, vậy người nhận cần 73+ ;
y > SendBase, vì vậy có thêm dữ liệu được ack

TCP: các kịch bản truyền tải lại



TCP: các kịch bản truyền tải lại (tt)

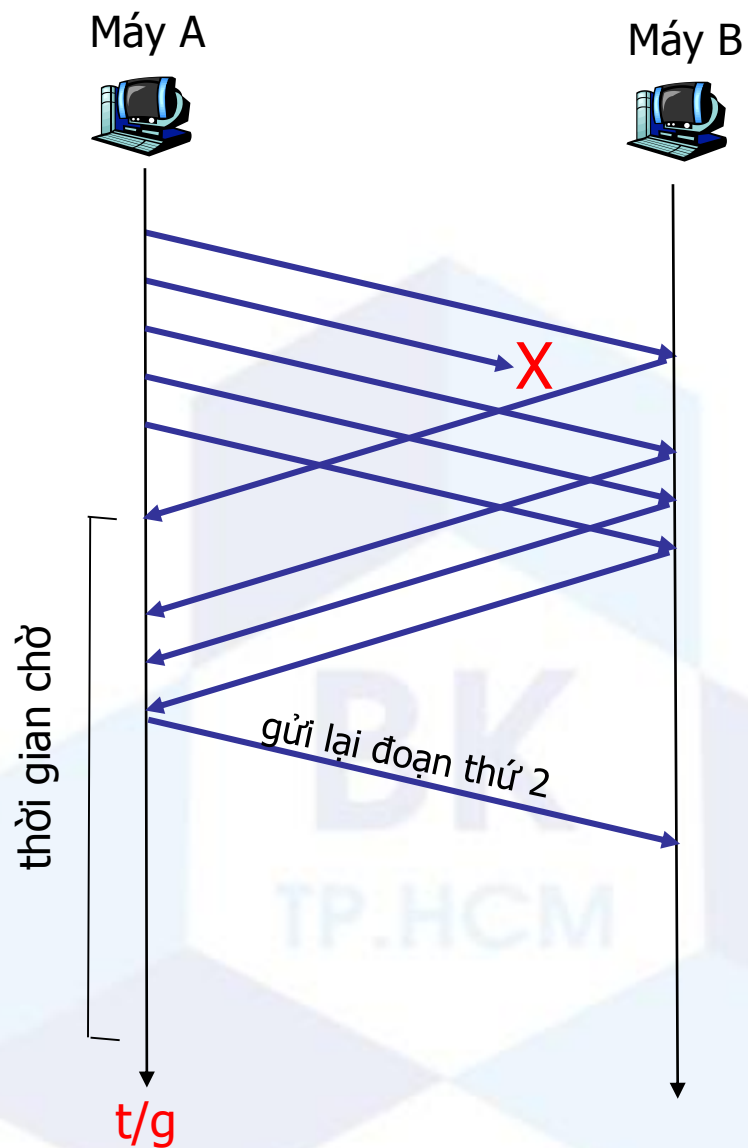


Tạo ACK trong TCP [RFC 1122, RFC 2581]

Sự kiện tại ng/nhận	Hành vi của ng/ nhận TCP
Sự đến của đoạn đúng thứ tự với STT hợp lí. Tất cả dữ liệu từ STT về trước đã được ACK	Trì hoãn việc ACK. Chờ đoạn tiếp theo trong 500ms. Nếu không có đoạn nào tiếp theo, gửi ACK
Sự đến của đoạn đúng thứ tự với STT hợp lí. Một đoạn khác đang chờ được ACK	Ngay lập tức gửi một ACK cộng dồn, xác nhận cả hai đoạn dữ liệu đúng thứ tự
Sự đến của đoạn sai-thứ-tự với STT cao hơn STT mong đợi. Phát hiện ra sự thiếu hụt	Ngay lập tức gửi một <i>ACK lặp</i> , chỉ rõ STT của byte mong đợi tiếp theo
Sự đến của đoạn mà khóa lặp sự thiếu hụt một phần hoặc toàn bộ	Ngay lập tức gửi ACK cộng dồn

Truyền lại nhanh

- Thời gian chờ thường tương đối dài:
 - sẽ bị trì hoãn lâu trước khi gửi lại gói bị mất
- Phát hiện mất đoạn thông qua ACK lặp.
 - Ng/gửi thường gửi nhiều đoạn liên tục
 - Nếu một đoạn bị mất thì thường sẽ có nhiều ACK trùng lặp.
- Nếu người nhận nhận được 3 ACK trùng lặp cho cùng một đoạn dữ liệu, nó sẽ suy ra là các đoạn dữ liệu theo sau đã bị mất:
 - truyền lại nhanh: gửi lại đoạn dữ liệu trước khi bộ đếm thời gian hết hạn



Giải thuật truyền tải lại nhanh:

```
sự kiện: nhận được ACK, với trường ACK có giá trị y
    if (y > SendBase) {
        SendBase = y
        if (không có đoạn nào chưa được ACK)
            khởi động bộ đếm thời gian
    }
    else {
        tăng bộ đếm số ACK cho y trùng
        if (số ACK trùng = 3) {
            gửi lại đoạn với STT y
        }
    }
```

một ACK trùng
cho một đoạn đã được ACK

truyền tải lại nhanh