



BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

Chào mừng đến với buổi training của BHT HTTT

NHẬP MÔN MẠNG MÁY TÍNH

TRAINER: Nguyễn Đặng Đức Minh (TMCL2021)
Trần Tịnh Minh Tú (TMCL2021)



NỘI DUNG:

I. Chương 1

1. Internet là gì?
2. Mạng biên
3. Mạng lõi
4. Độ trễ, sự mất mát, thông lượng
5. Phân tầng giao thức và dịch vụ



NỘI DUNG:

II. Chương 2

1. Kiến trúc ứng dụng mạng
2. Web và HTTP (Hypertext Transfer Protocol)
3. FTP (File Transfer Protocol)
4. Thư điện tử: SMTP
5. DNS (Domain Name System)
6. Lập trình socket với UDP và TCP



NỘI DUNG:

III. Chương 3

1. Tổng quan về tầng transport
2. Multiplexing và Demultiplexing
3. Vận chuyển phi kết nối: UDP
4. Các nguyên lý truyền dữ liệu tin cậy
5. Vận chuyển hướng kết nối: TCP
6. Điều khiển tắc nghẽn trong TCP





BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

CHƯƠNG 1: TỔNG QUAN MẠNG MÁY TÍNH



NỘI DUNG:

1. Internet là gì?
2. Mạng biên
3. Mạng lõi
4. Độ trễ, sự mất mát, thông lượng
5. Phân tầng giao thức và dịch vụ





BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

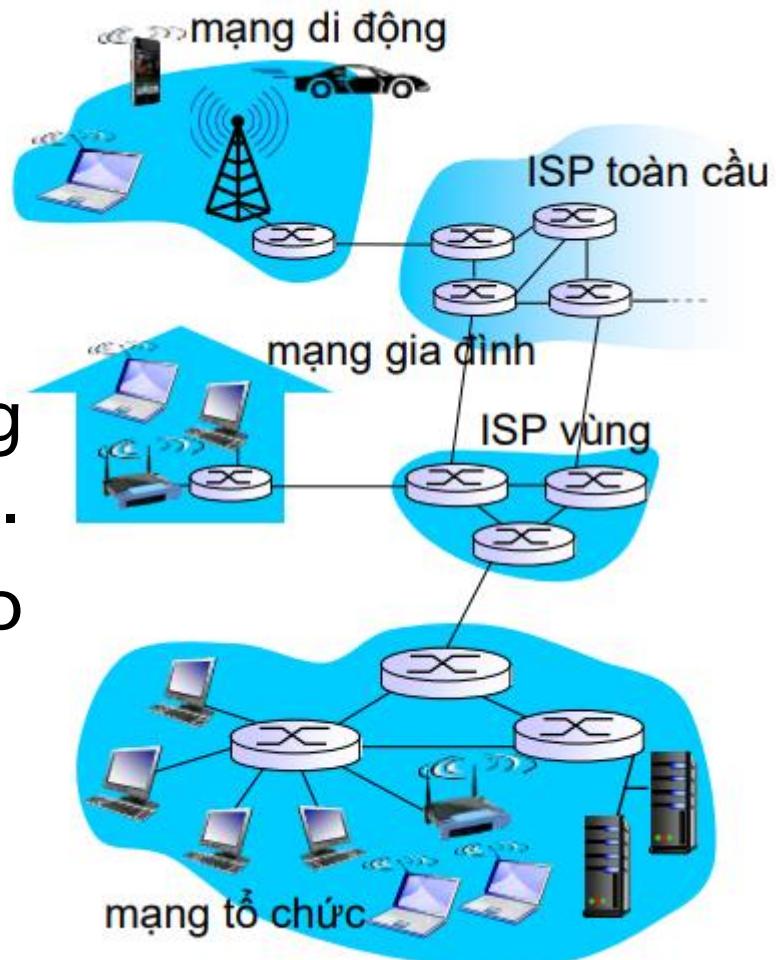
1

INTERNET LÀ GÌ?



Internet là gì?

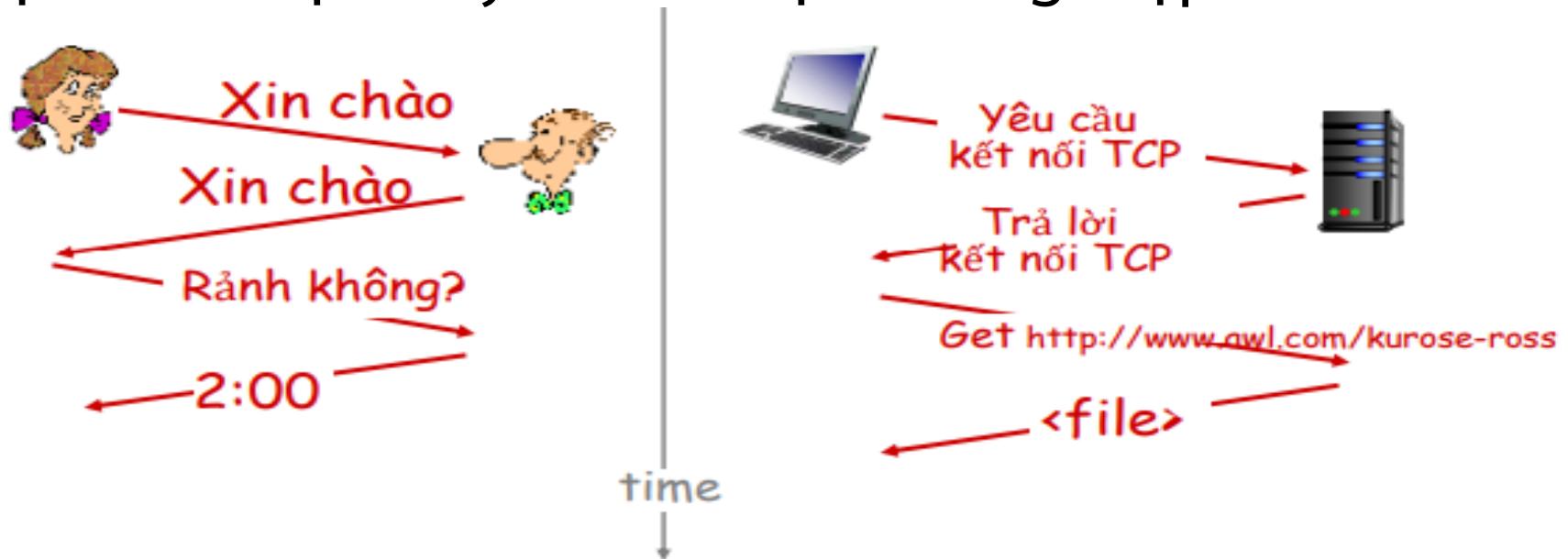
- Là “mạng của các mạng”.
- Ứng dụng:
 - Cung cấp các dịch vụ cho các ứng dụng (Web, VoIP, email, game,...).
 - Cung cấp giao diện lập trình cho các ứng dụng.





Giao thức là gì?

Giao thức định nghĩa định dạng, thứ tự các thông điệp được gửi và nhận giữa các thực thể mạng, và các hành động được thực hiện trên việc truyền và nhận thông điệp.





BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

2

MẠNG BIÊN



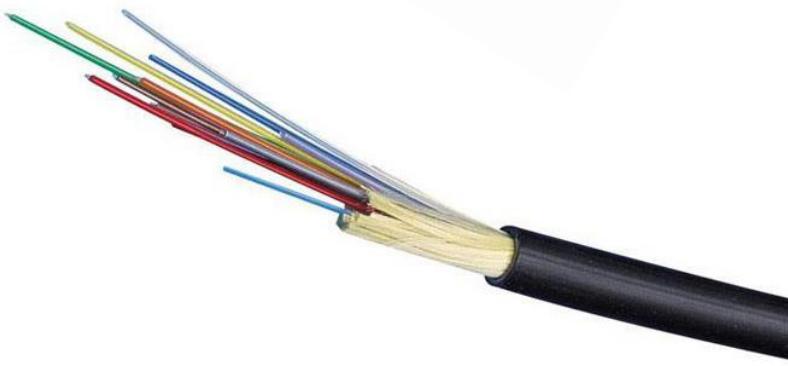
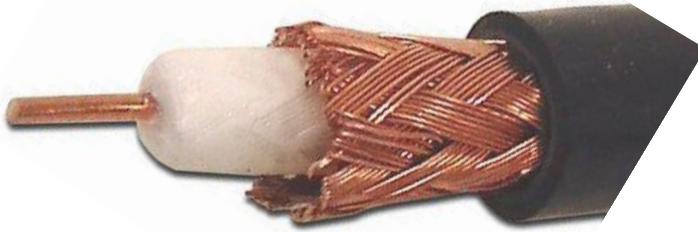
Cấu trúc mạng:

- Mạng biên → hosts (các hệ thống đầu cuối) → server (máy chủ) & client (máy khách).
- Mạng truy nhập → không dây & có dây:
 - Kỹ thuật DSL (đường dây thuê bao kỹ thuật số).
 - Mạng cáp.
 - Kỹ thuật FTTH (mạng gia đình hoặc mạng doanh nghiệp).
 - Ethernet.
- Mạng lõi → router (thiết bị định tuyến).



Đường truyền vật lý:

- Cáp đồng.
- Cáp quang.
- Sóng radio.





BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

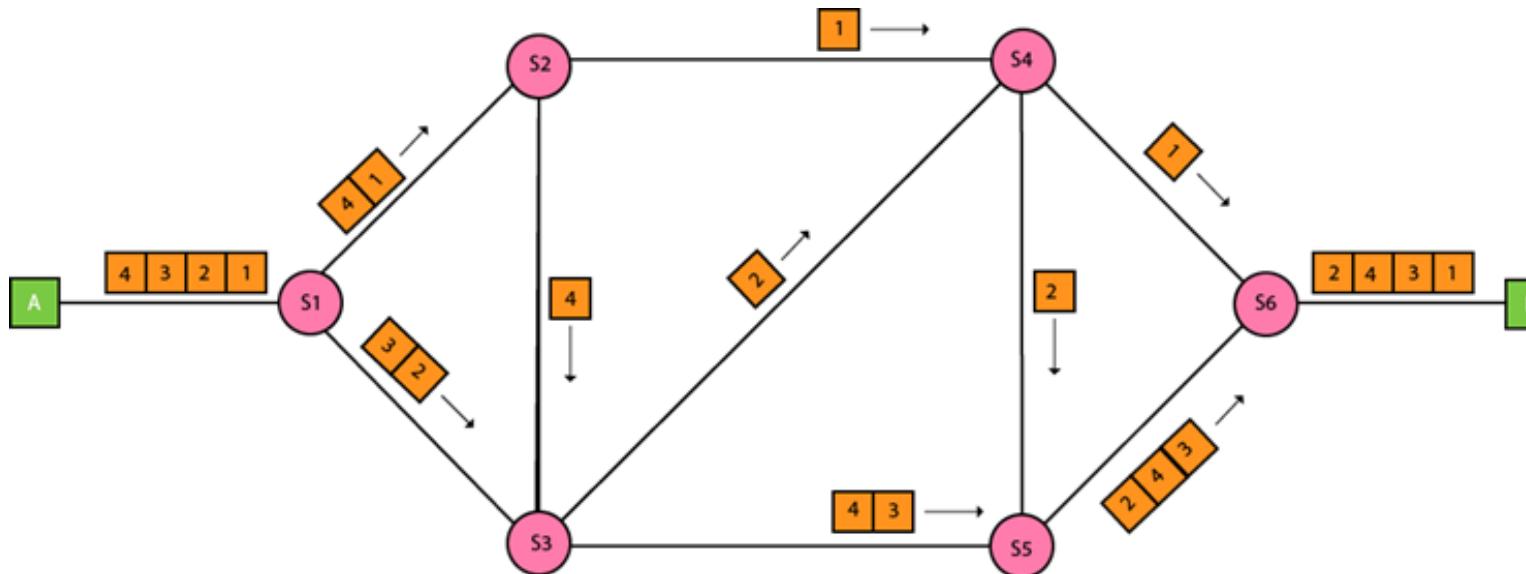
3

MẠNG LÕI



Chuyển mạch gói (packet switching):

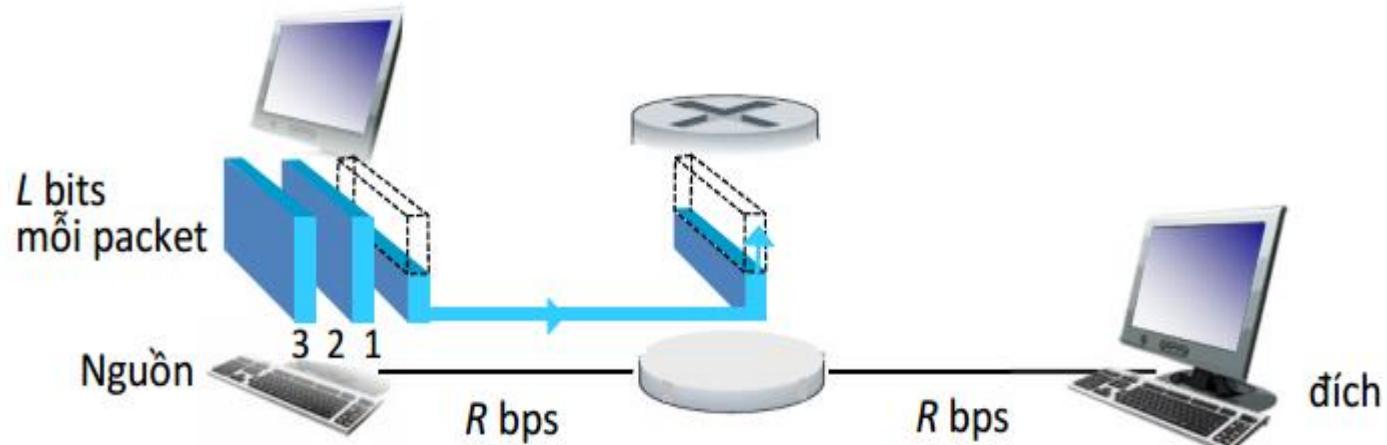
Các hệ thống đầu cuối (host) chia nhỏ dữ liệu tầng ứng dụng thành các packet.





Chuyển mạch gói (packet switching):

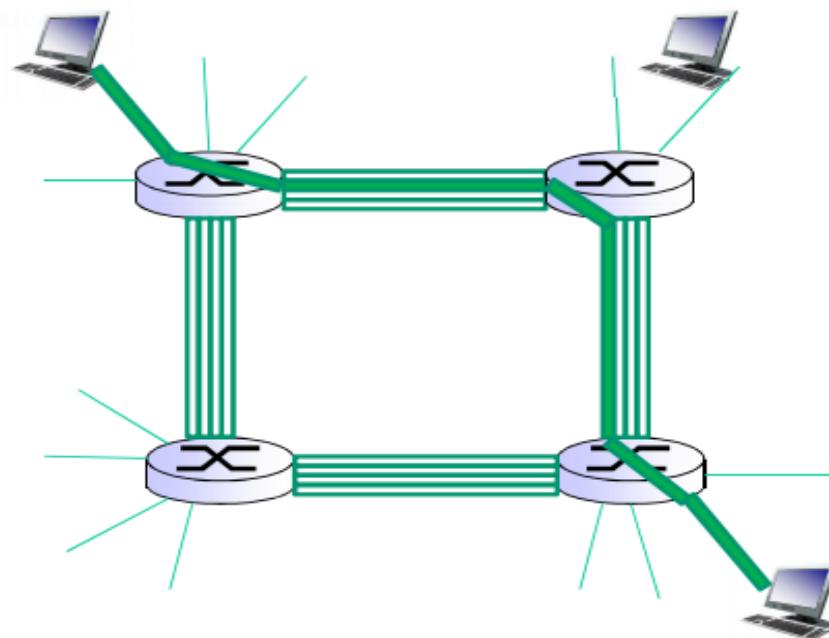
Lưu và chuyển tiếp (store-and-forward): toàn bộ các gói phải đến bộ định tuyến trước khi nó có thể được truyền tải trên đường liên kết tiếp theo.





Chuyển mạch kênh (circuit switching):

Tài nguyên giữa 2 điểm cuối được phân bổ, được dành cho “cuộc gọi” giữa nguồn và đích.





Chuyển mạch kênh (circuit switching):

Gồm 2 cách chia băng thông:

- FDM (frequency division multiplexing):
 - Nhiều người dùng cùng lúc.
 - Tốc độ chậm.
- TDM (time division multiplexing):
 - 1 người dùng.
 - Tốc độ cao.

FDM

Tần số

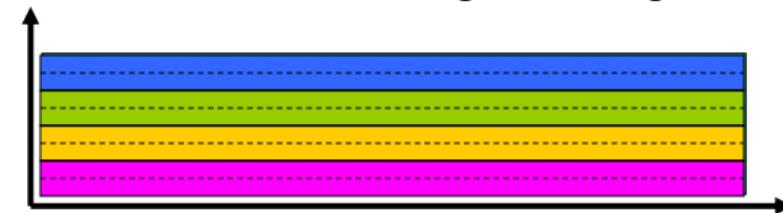
TDM

tần số

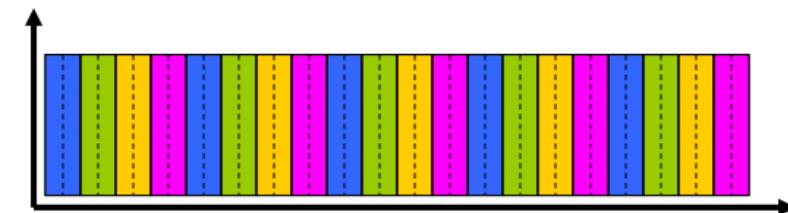
Ví dụ:

4 người dùng

██████████



time



time



BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

4

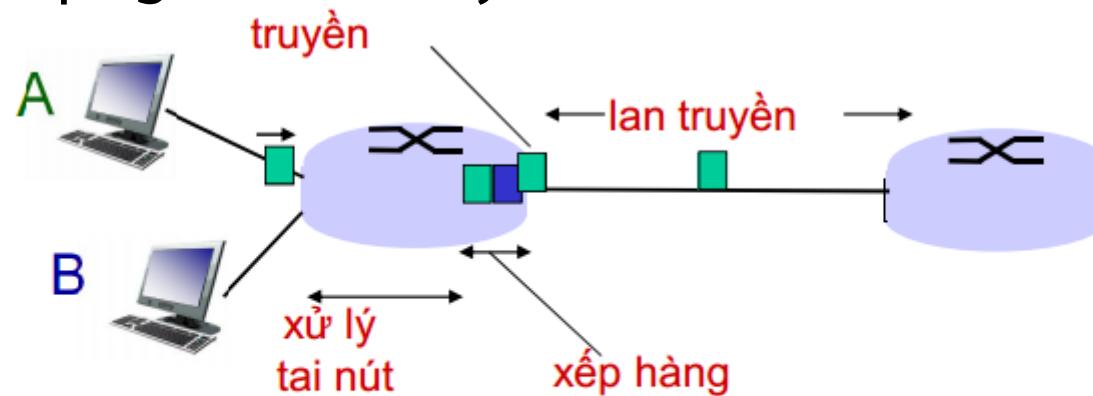
ĐỘ TRỄ, SỰ MẤT MÁT, THÔNG LƯỢNG MẠNG



Độ trễ

Có 4 nguồn gây ra độ trễ:

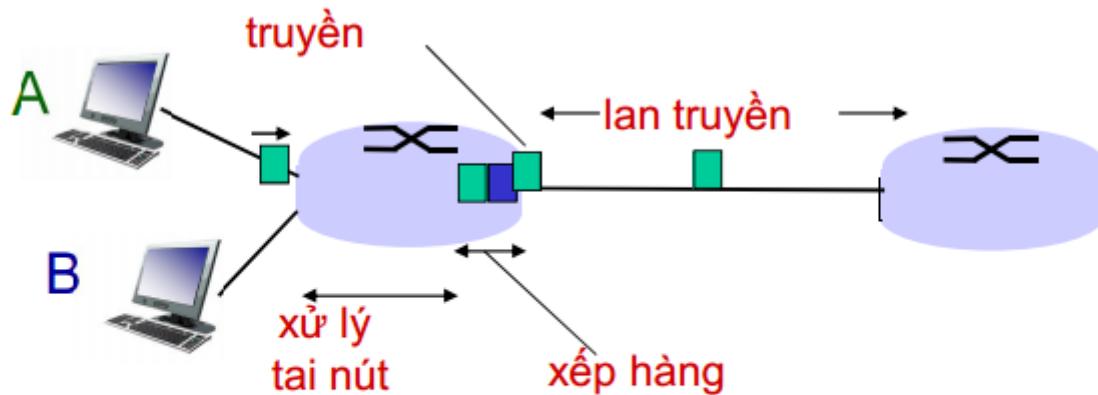
- Xử lý tại nút (nodal processing)
- Xếp hàng (queuing delay)
- Truyền (transmission delay)
- Lan truyền (propagation delay)



$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$



Độ trễ



$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

d_{proc} : xử lý tại nút

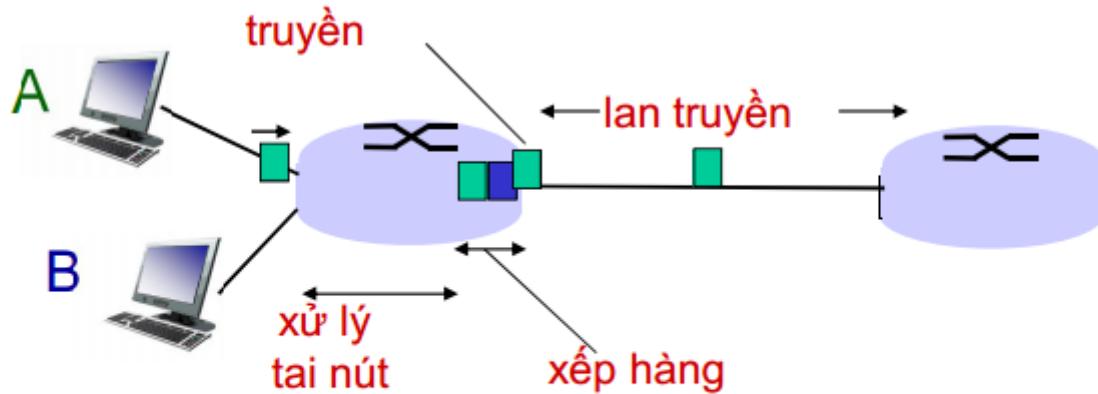
- Kiểm tra các bit lỗi.
- Mặc định $d_{\text{proc}} = 0$

d_{queue} : độ trễ xếp hàng

- Mặc định $d_{\text{queue}} = 0$



Độ trễ



$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

$$d_{\text{trans}} = L / R$$

- L: chiều dài gói (bits)
- R: băng thông đường liên kết (bps)

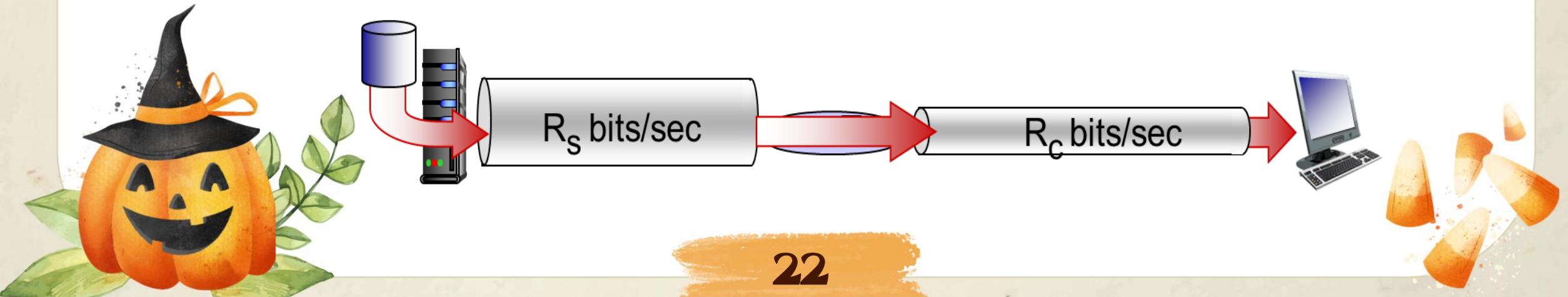
$$d_{\text{prop}} = d / s$$

- d: độ dài của đường liên kết vật lý (m)
- s: tốc độ lan truyền trong môi trường (thiết bị, dây dẫn) (m/s) ($\approx 2 \times 10^8$ m/s)



Thông lượng

- Băng thông (bandwidth):
 - Lượng thông tin tối đa có thể truyền đi trên 1 kết nối mạng trong 1 khoảng thời gian.
 - Đơn vị: bps (bits per second)
- Thông lượng: tốc độ (bits/time unit) mà các bit được truyền giữa người gửi và nhận.
- Nút thắt cổ chai: là điểm tại đó làm giới hạn thông lượng đường truyền.





Trắc nghiệm

Câu 1: Câu nào sau đây đúng khi sắp xếp tăng dần về độ dài thời gian của độ trễ?

- A. $d_{\text{proc}} < d_{\text{trans}} < d_{\text{prop}} < d_{\text{queue}}$
- B. $d_{\text{queue}} < d_{\text{proc}} < d_{\text{trans}} < d_{\text{prop}}$
- C. $d_{\text{proc}} < d_{\text{queue}} < d_{\text{trans}} < d_{\text{prop}}$
- D. $d_{\text{queue}} < d_{\text{trans}} < d_{\text{prop}} < d_{\text{proc}}$



Trắc nghiệm

Câu 2: Các gói tin có kích thước $L = 1000$ bytes được truyền trên một kết nối có tốc độ truyền là $R = 1000$ Kbps. Hỏi tối đa có bao nhiêu gói tin được truyền trong 1s?

- A. 125 gói tin
- B. 150 gói tin
- C. 250 gói tin
- D. 100 gói tin



BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

5

PHÂN TẦNG GIAO THỨC VÀ DỊCH VỤ



Mô hình TCP/IP

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

Ứng dụng

Vận chuyển

Mạng

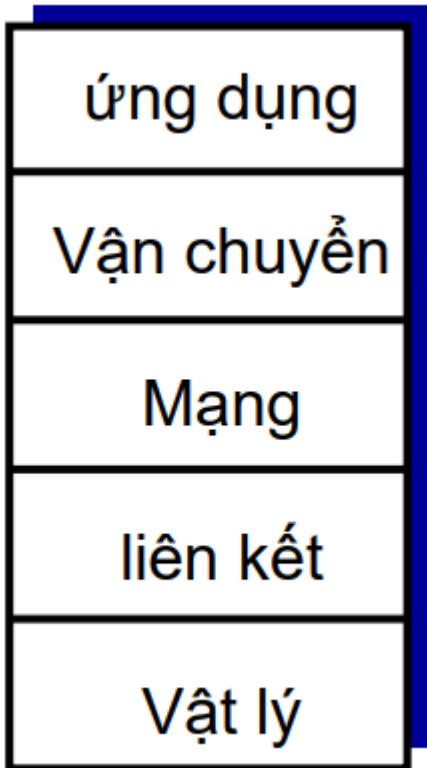
liên kết

Vật lý



Mô hình TCP/IP

Đơn vị dữ liệu





Mô hình ISO/OSI

Ứng dụng
Trình diễn
phiên
Vận chuyển
Mạng
liên kết
Vật lý

- Trình diễn (presentation): cho phép các ứng dụng giải thích ý nghĩa của dữ liệu.
 - VD: mã hóa, nén, những quy ước chuyên biệt.
- Phiên (session): sự đồng bộ hóa, khả năng chịu lỗi, phục hồi sự trao đổi dữ liệu.



Trắc nghiệm

Câu 1: Mô hình TCP/IP gồm mấy tầng?

- A. 4
- B. 5
- C. 6
- D. 7



Trắc nghiệm

Câu 2: Tầng nào sau đây có đơn vị dữ liệu là segment?

- A. Tầng Application
- B. Tầng Transport
- C. Tầng Network
- D. Tầng Data Link



BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

CHƯƠNG 2: TẦNG ỨNG DỤNG (APPLICATION LAYER)



NỘI DUNG:

1. Kiến trúc ứng dụng mạng
2. Web và HTTP (Hypertext Transfer Protocol)
3. FTP (File Transfer Protocol)
4. Thư điện tử: SMTP
5. DNS (Domain Name System)
6. Lập trình socket với UDP và TCP



BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

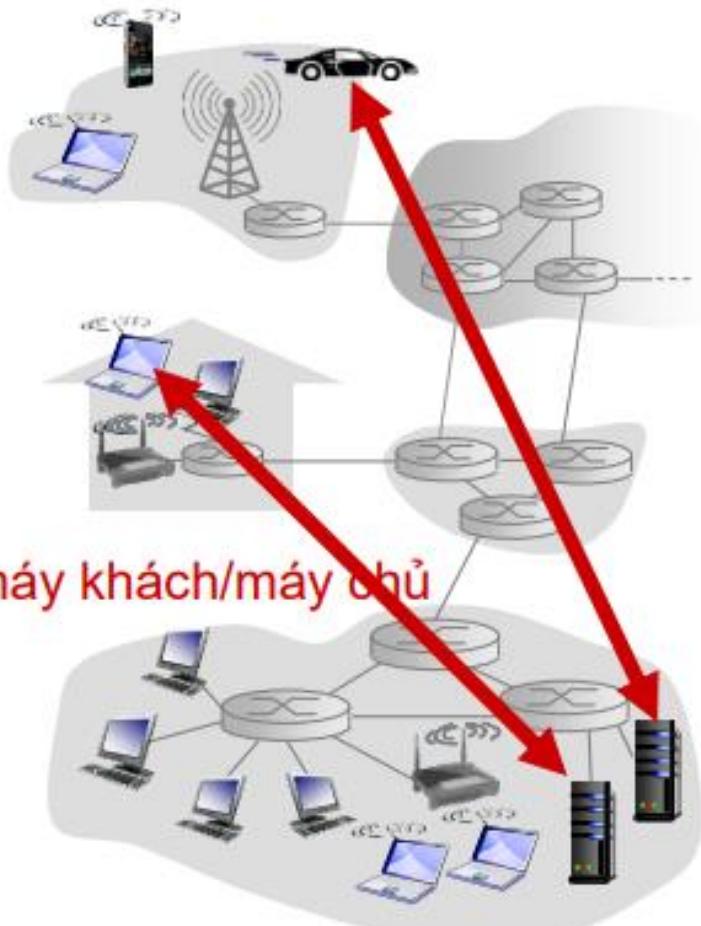
1

KIẾN TRÚC ỨNG DỤNG MẠNG



Kiến trúc client-server

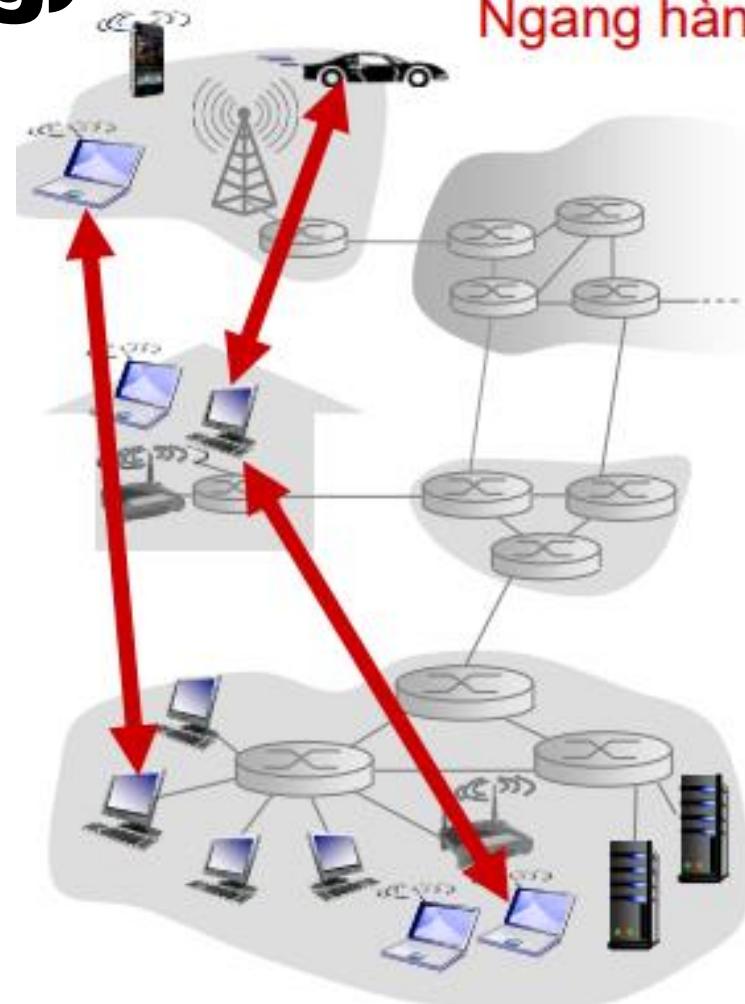
- Máy chủ (server):
 - Luôn luôn hoạt động.
 - IP tĩnh.
 - Tổ chức thành các trung tâm dữ liệu để mở rộng quy mô.
- Máy khách (client):
 - Giao tiếp với server.
 - Không giao tiếp trực tiếp với cái client khác.
 - Hoạt động không liên tục.
 - IP động.





Kiến trúc P2P (ngang hàng)

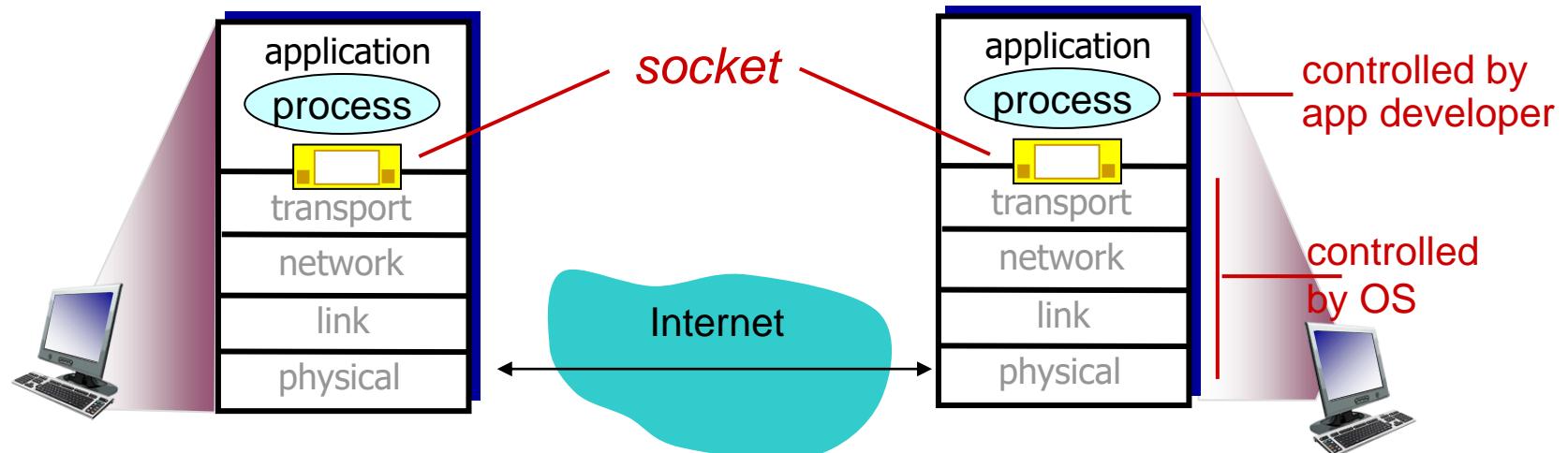
- Không có server.
- Các host truyền thông trực tiếp với nhau.
- Các peer (bên):
 - Cung cấp dịch vụ cho nhau.
 - Kết nối không liên tục.
 - IP động.





Tiến trình mạng

- Tiến trình (process): là chương trình đang chạy trên một máy tính.
- Tiến trình gửi/nhận thông điệp đến/từ socket của nó.
- Để nhận thông điệp, tiến trình phải có định danh (địa chỉ IP và số cổng).





TCP (Transmission Control Protocol) & UDP (User Datagram Protocol)

TCP	UDP
Tin cậy, theo thứ tự	Không tin cậy, không theo thứ tự
Điều khiển luồng (flow control)	
Điều khiển tắc nghẽn (congestion control)	
Hướng kết nối (connection-oriented)	Phi kết nối (connectionless)



BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

2

WEB VÀ HTTP (HYPERTEXT TRANSFER PROTOCOL)



HTTP là gì ?

- Là giao thức thuộc tầng Application trong TCP/IP
- Port number: 80 – TCP
- HTTP “không lưu trang thái” (không duy trì thông tin về các yêu cầu trước)
- Trong mô hình client-server, http được áp dụng như sau:
 - Client: trình duyệt (dùng http) gửi yêu cầu, nhận phản hồi và hiển thị các đối tượng web
 - Server: Web server (dùng http) gửi các đối tượng để trả lời yêu cầu





Các kết nối HTTP

HTTP không bền vững	HTTP bền vững
HTTP 1.0	HTTP 1.1
Chỉ tối đa 1 đối tượng được gửi qua kết nối TCP (sau đó kết nối bị đóng)	Nhiều đối tượng có thể được gửi qua 1 kết nối TCP
2 RTT cho mỗi đối tượng	Không có pipelining: 1 RTT cho việc mở kết nối Có pipelining: 1 RTT cho mỗi đối tượng (không cần kết nối lại)

(RTT – round trip time: thời gian để 1 gói tin nhỏ đi từ máy khách đến server và quay ngược lại)



Thông điệp HTTP

➤ Có 2 loại thông điệp HTTP:

- Request: gồm dòng yêu cầu, header, thân.
- Response: gồm dòng trạng thái, header, data được yêu cầu.

Dòng yêu cầu
(các lệnh GET, POST,
HEAD)

Các dòng
header

ký tự xuống dòng,
về đầu dòng mới chỉ
điểm cuối cùng
của thông điệp

GET /index.html HTTP/1.1\r\nHost: www-net.cs.umass.edu\r\nUser-Agent: Firefox/3.6.10\r\nAccept: text/html,application/xhtml+xml\r\nAccept-Language: en-us,en;q=0.5\r\nAccept-Encoding: gzip,deflate\r\nAccept-Charset: ISO-8859-1,utf-8;q=0.7\r\nKeep-Alive: 115\r\nConnection: keep-alive\r\n\r\n

Ký tự xuống dòng

Ký tự về đầu dòng



Thông điệp yêu cầu HTTP

- GET: trình duyệt yêu cầu đối tượng, và đối tượng được chỉ rõ trong trường URL.
- POST: dữ liệu được nhập vào mẫu (form) chứa trong phần than thông điệp.
- HEAD: yêu cầu server loại bỏ object được yêu cầu ra khỏi thông điệp phản hồi.
- PUT (HTTP 1.1): tải tập tin trong than thực thể đến đường dẫn được xác định trong trường URL.
- DELETE (HTTP 1.1): xóa tập tin được chỉ định trong trường URL.



Thông điệp phản hồi HTTP

- 200 OK: yêu cầu thành công, đối tượng được yêu cầu sau ở trong thông điệp này.
- 301 Moved Permanently: object được yêu cầu đã được di chuyển, vị trí mới được xác định sau trong trường Location.
- 400 Bad Request: server không hiểu thông điệp yêu cầu.
- 404 Not Found: thông tin được yêu cầu không tìm thấy trên server này.
- 505 HTTP Version Not Supported.



BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

3

FTP

(FILE TRANSFER PROTOCOL)



Giới thiệu FTP

- Là một giao thức dung để truyền tập tin (file) từ máy tính này sang máy tính khác.
- FTP dung hai kết nối TCP song song để truyền tập tin:
 - FTP data: TCP – port: 20.
 - FTP control: TCP – port: 21.
- FTP là giao thức không an toàn.





Một số lệnh FTP

- USER username
- PASS password
- LIST: trả về danh sách tập tin trên thư mục hiện tại.
- RETR filename: lấy tập tin.
- STOR filename: lưu tập tin vào trong máy ở xa.





BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

4

THƯ ĐIỆN TỬ: SMTP



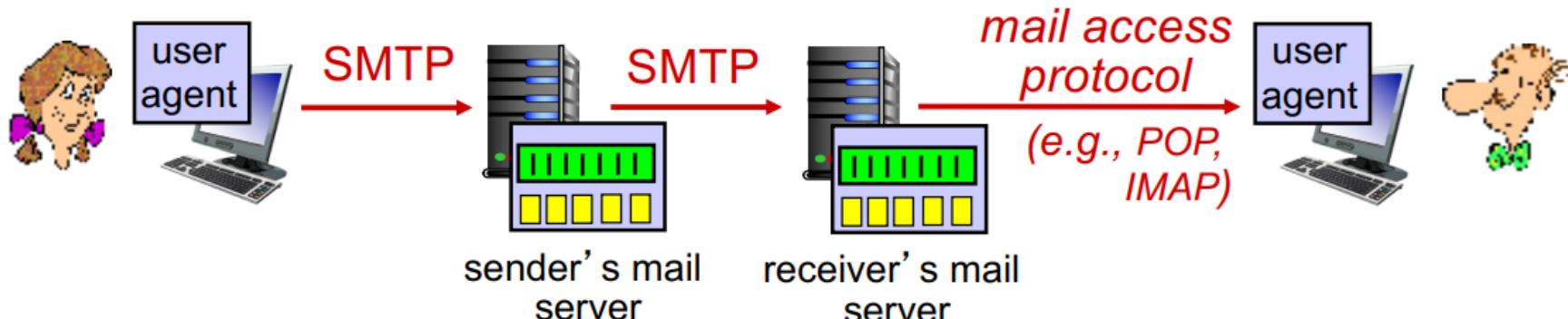
Các thành phần chính

- User agents: soạn thảo, sửa đổi, đọc các thông điệp thư điện tử.
 - VD: Outlook, iphone mail client, ...
- Mail servers:
 - Hộp thư (mailbox).
 - Hàng thông điệp (message queue).
- SMTP: Simple Mail Transfer Protocol.



SMTP: giao thức gửi tin

- Sử dụng TCP để gửi thông điệp mail từ client đến server (port: 25).
- Truyền trực tiếp: server gửi đến server nhận.
- Gồm 3 giai đoạn truyền: bắt tay (chào hỏi), truyền thông điệp, đóng.
- Tương tác lệnh/phản hồi (như HTTP, FTP).
- Thông điệp phải ở dạng mã ASCII 7 bit.
- Một số lệnh SMTP thông dụng: HELO, MAIL FROM, RCPT TO, DATA, QUIT.





BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

5

DNS (DOMAIN NAME SYSTEM)

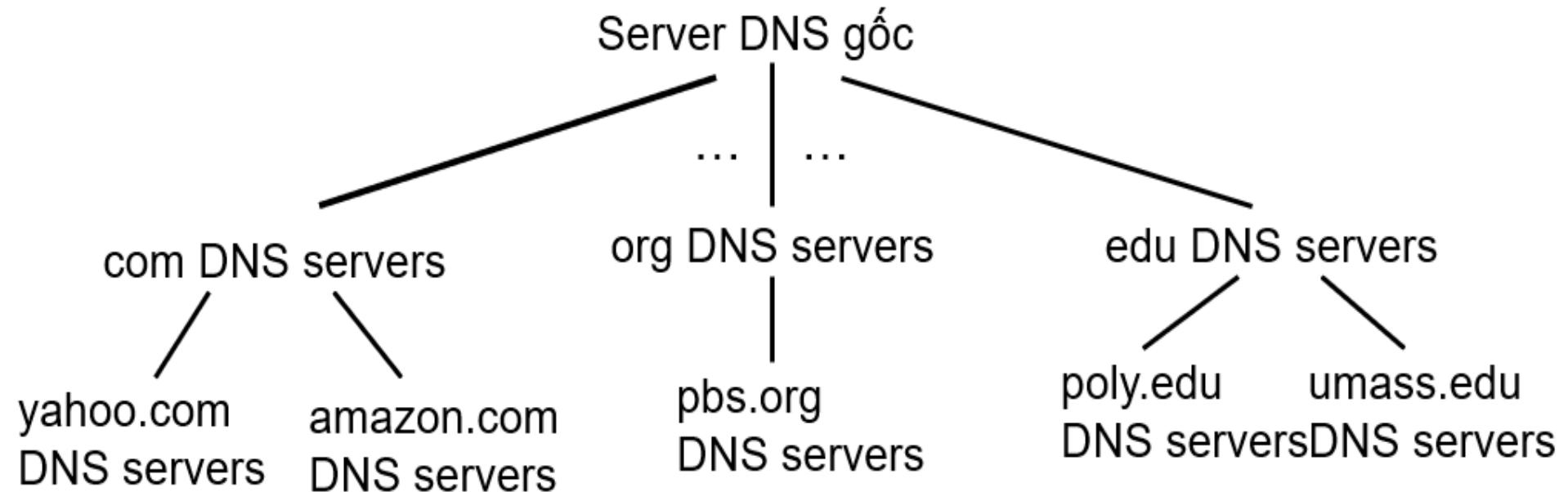


DNS là gì?

- DNS là hệ thống phân giải tên miền.
- Tên miền (Domain): địa chỉ trang web.
- Gõ tên miền → DNS: tự động ánh xạ sang địa chỉ IP.
- Dịch vụ DNS cung cấp:
 - Dịch tên máy ra địa chỉ IP.
 - Bí danh máy.
 - Bí danh mail servers.
 - Cân bằng tải: nhiều địa chỉ IP tương ứng cho 1 tên.



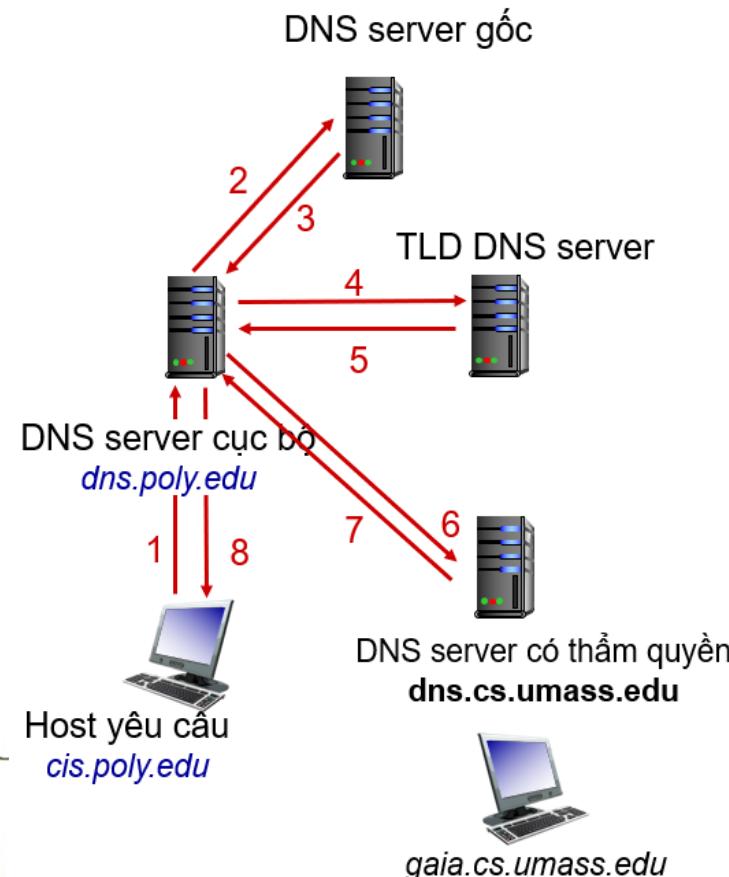
Cơ sở dữ liệu phân cấp, phân tán



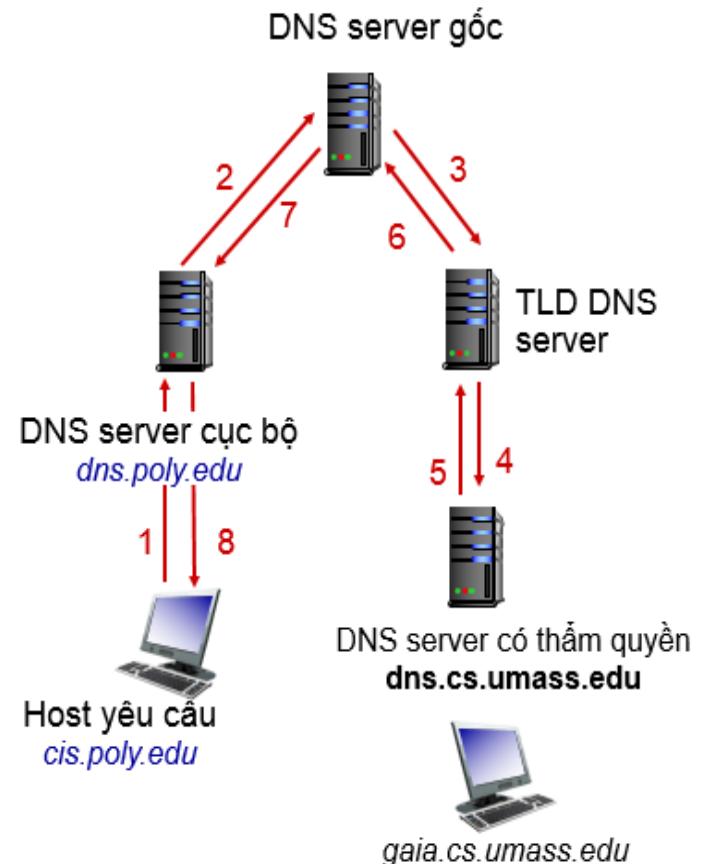


Phân giải tên miền

Truy vấn tuần tự (iterated query):



Truy vấn đệ quy (recursive query):





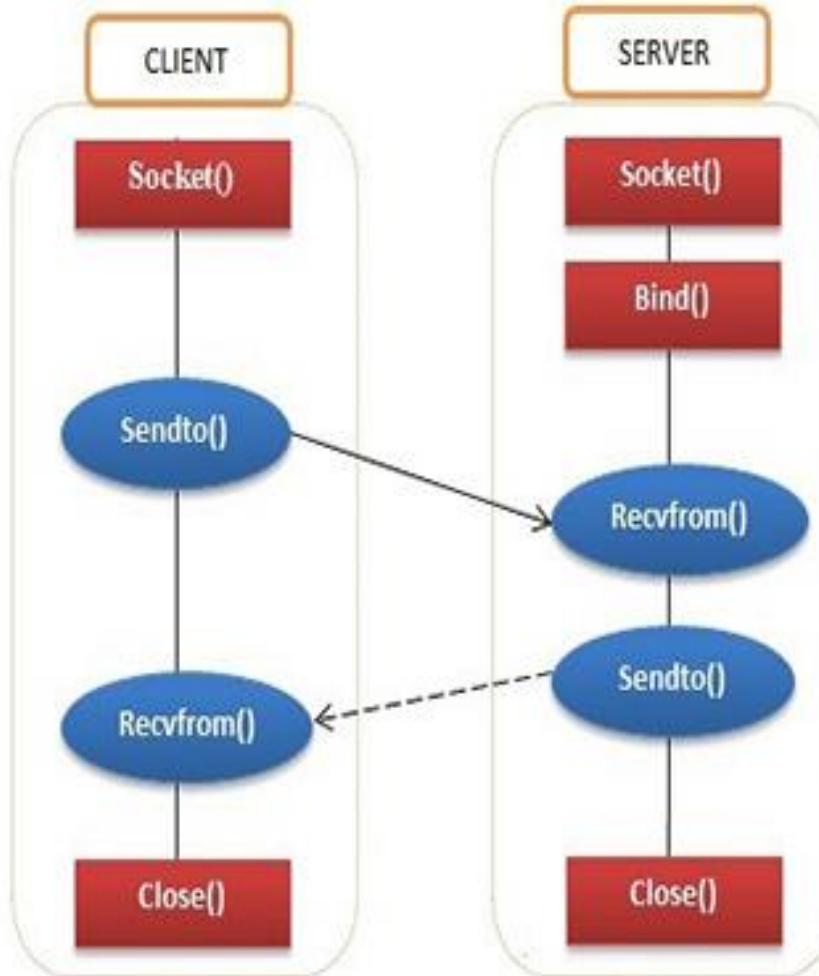
BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

6

LẬP TRÌNH SOCKET VỚI UDP VÀ TCP

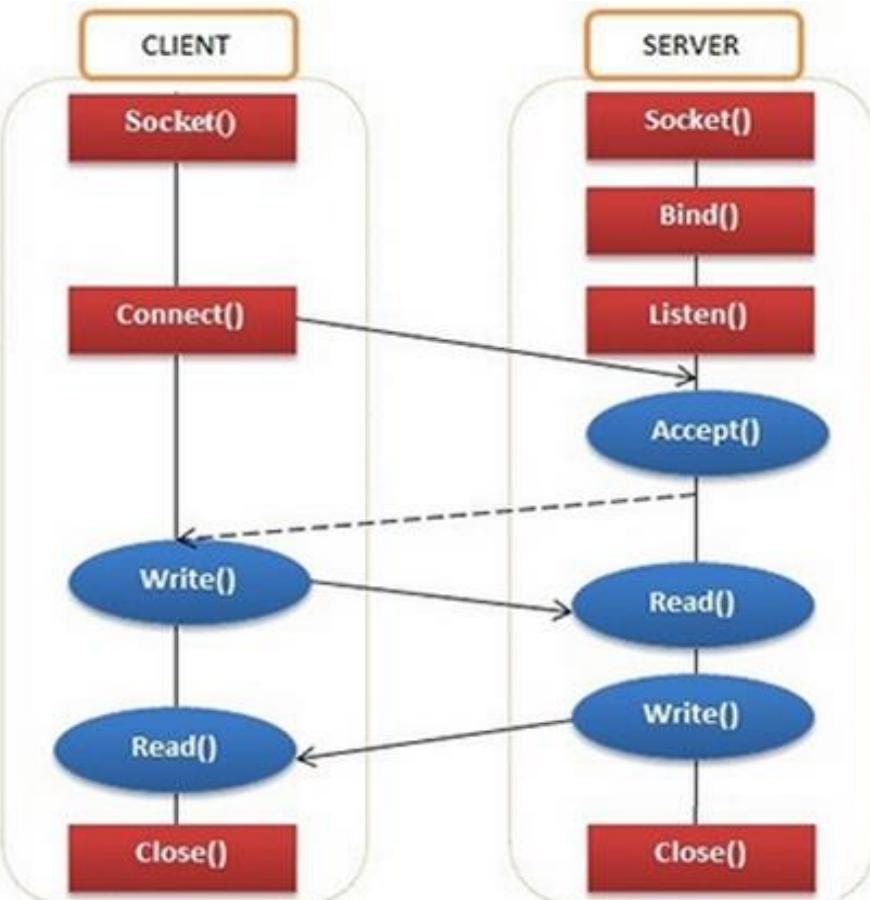


Datagram Socket (UDP)





Stream Socket (TCP)





Trắc nghiệm

Câu 1: Hãy xác định xem đoạn mã sau đây được viết cho ứng dụng nào?

- A. UDP Server
- B. UDP Client
- C. TCP Server
- D. TCP Client

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



Trắc nghiệm

Câu 2: Hãy xác định xem đoạn mã sau đây được viết cho ứng dụng nào?

- A. UDP Server
- B. UDP Client
- C. TCP Server
- D. TCP Client

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



Trắc nghiệm

Câu 3: Ý nào sau đây là đúng khi nói về TCP?

- A. Truyền không tin cậy, không theo thứ tự.
- B. Phi kết nối (connectionless).
- C. Không hỗ trợ điều khiển luồng.
- D. Hướng kết nối (connection-oriented).



BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

CHƯƠNG 3: TẦNG VẬN CHUYỂN (TRANSPORT LAYER)



NỘI DUNG:

1. Tổng quan về tầng transport
2. Multiplexing và Demultiplexing
3. Vận chuyển phi kết nối: UDP
4. Các nguyên lý truyền dữ liệu tin cậy
5. Vận chuyển hướng kết nối: TCP
6. Điều khiển tắc nghẽn trong TCP





BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

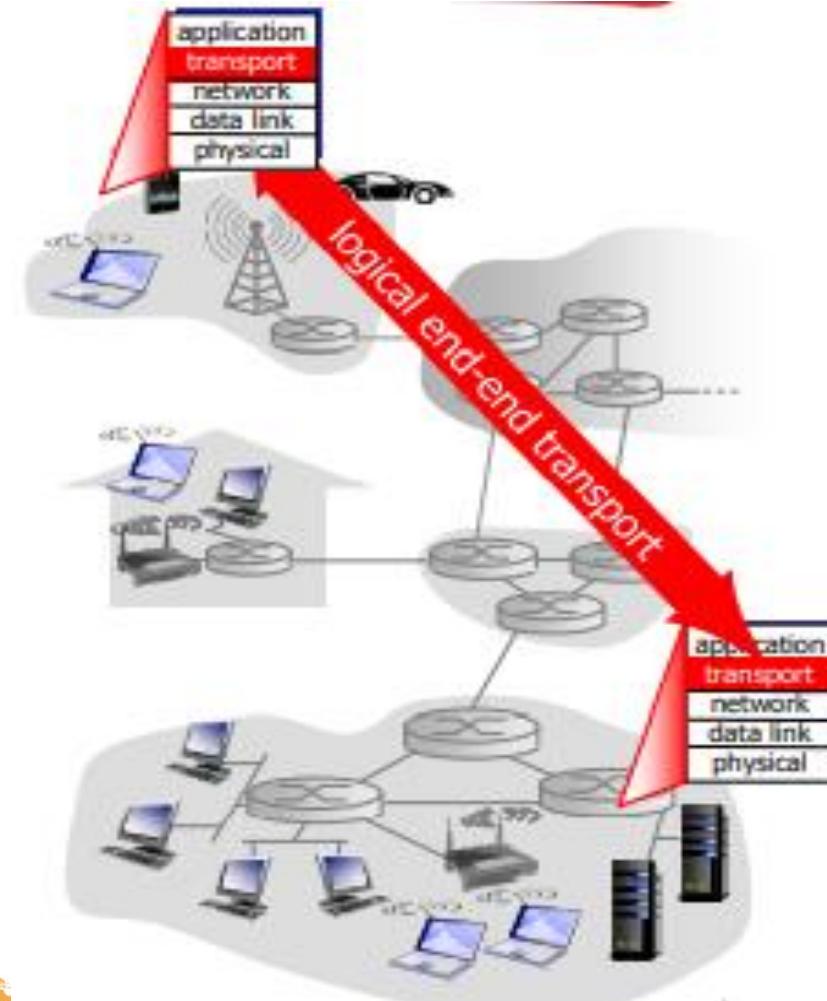
1

TỔNG QUAN VỀ TẦNG TRANSPORT



Dịch vụ tầng Transport

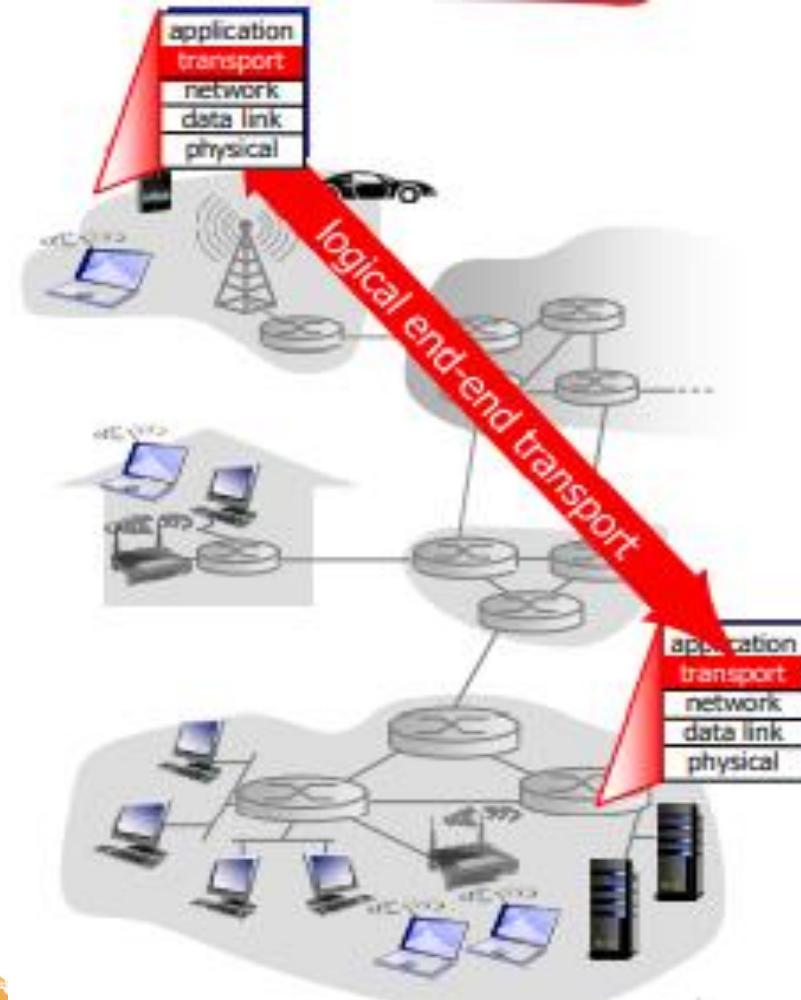
- Giao thức tầng transport cung cấp truyền thông logic giữa các process (process-to-process).
- Giao thức tầng transport chạy trên các hệ thống đầu cuối
- Đơn vị dữ liệu: segment





Dịch vụ tầng Transport

- Bên gửi: nhận data từ tầng Application → chia nhỏ và đóng gói thành segments → chuyển cho tầng Network.
- Bên nhận: nhận segments từ tầng Network gửi lên → tái kết hợp thành các thông điệp → chuyển lên tầng Application.





Giao thức tầng Transport

- UDP (User Datagram Protocol):
 - Truyền không tin cậy, không theo thứ tự
 - Truyền tổng lực (best-effort)
 - Phi kết nối (connectionless)
- TCP (Transmission Control Protocol):
 - Truyền tin cậy, theo thứ tự
 - Hướng kết nối (connection-oriented)
 - Điều khiển luồng (flow control)
 - Điều khiển tắc nghẽn (congestion control)



BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

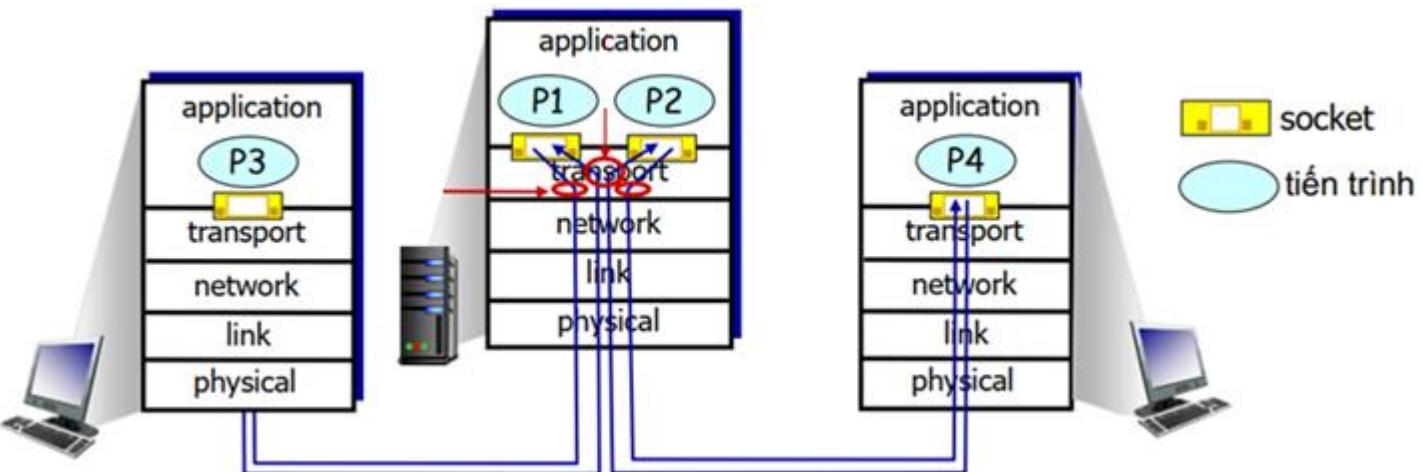
2

MULTIPLEXING VÀ DEMULTIPLEXING



Multiplexing & Demultiplexing

- Multiplexing tại bên gửi: xử lý dữ liệu từ nhiều socket, thêm thông tin header về tầng transport vào segment (được sử dụng sau cho demultiplexing).
- Demultiplexing tại bên nhận: sử dụng thông tin trong header để chuyển segment vừa nhận vào đúng socket.





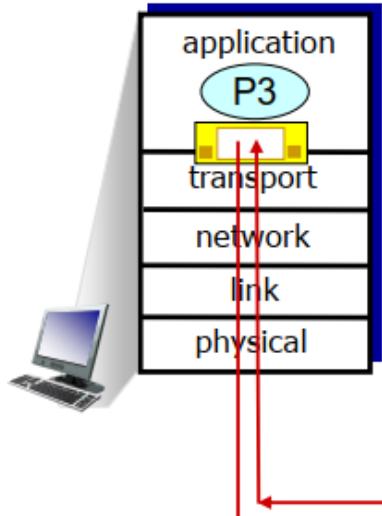
Demultiplexing: phi kết nối (UDP) và hướng kết nối (TCP)

UDP (connectionless)	TCP (connection-oriented)
UDP socket được xác định bởi 2 yếu tố: port đích, IP đích.	TCP socket được xác định bởi 4 yếu tố: port nguồn, port đích, IP nguồn, IP đích.

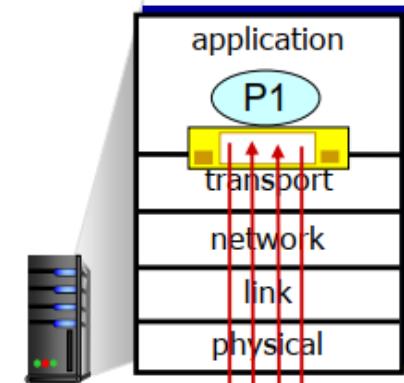


Demultiplexing: Connectionless

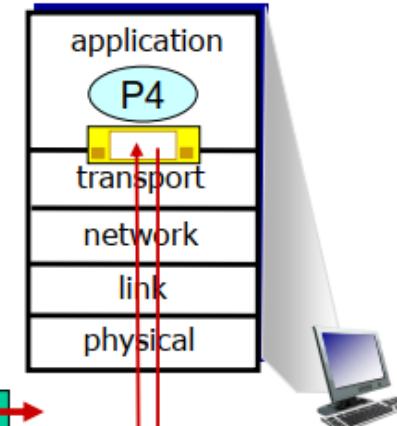
```
DatagramSocket  
mySocket2 = new  
DatagramSocket  
(9157);
```



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

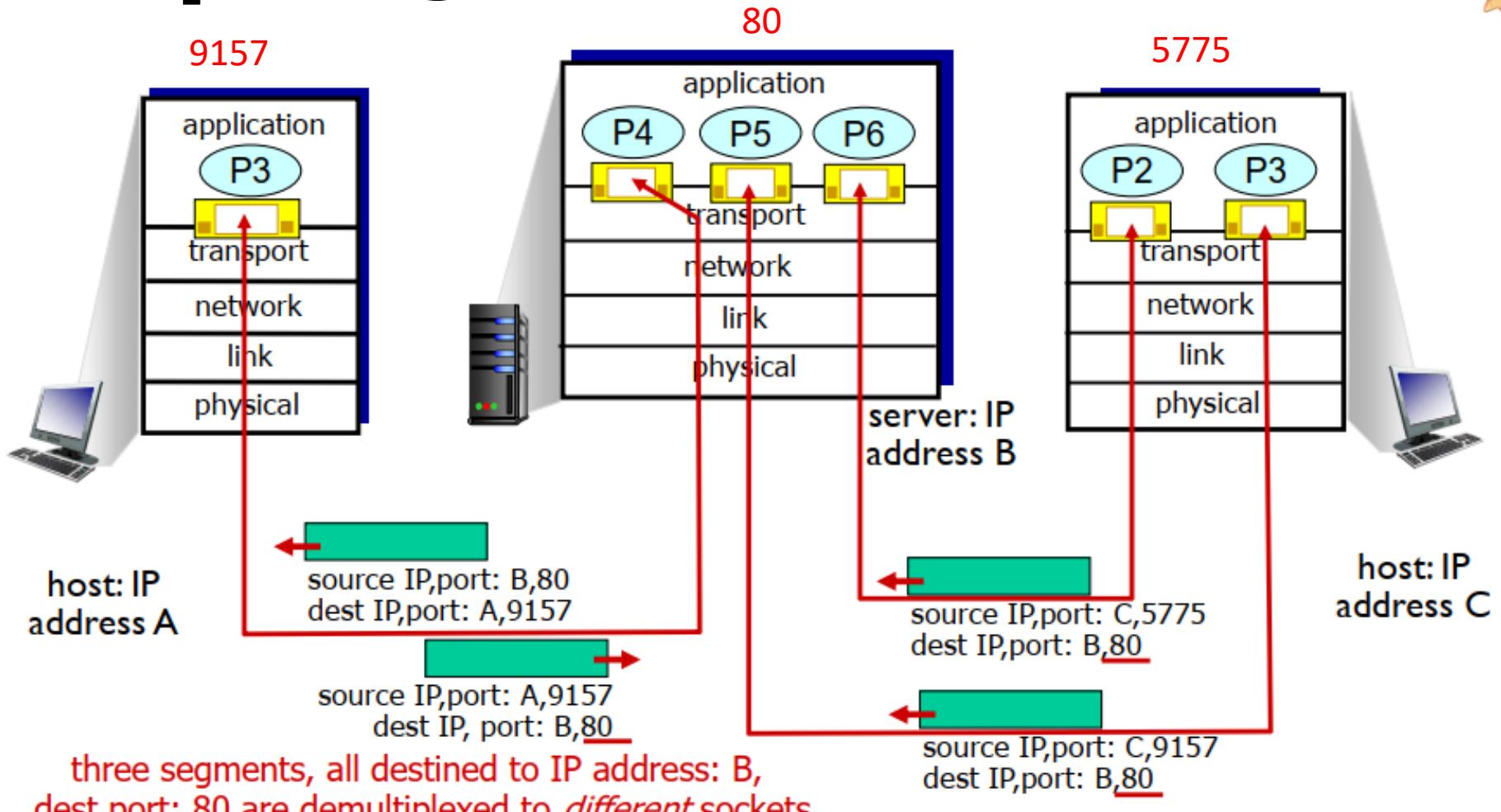


```
DatagramSocket  
mySocket1 = new  
DatagramSocket  
(5775);
```





Demultiplexing: Connection-oriented

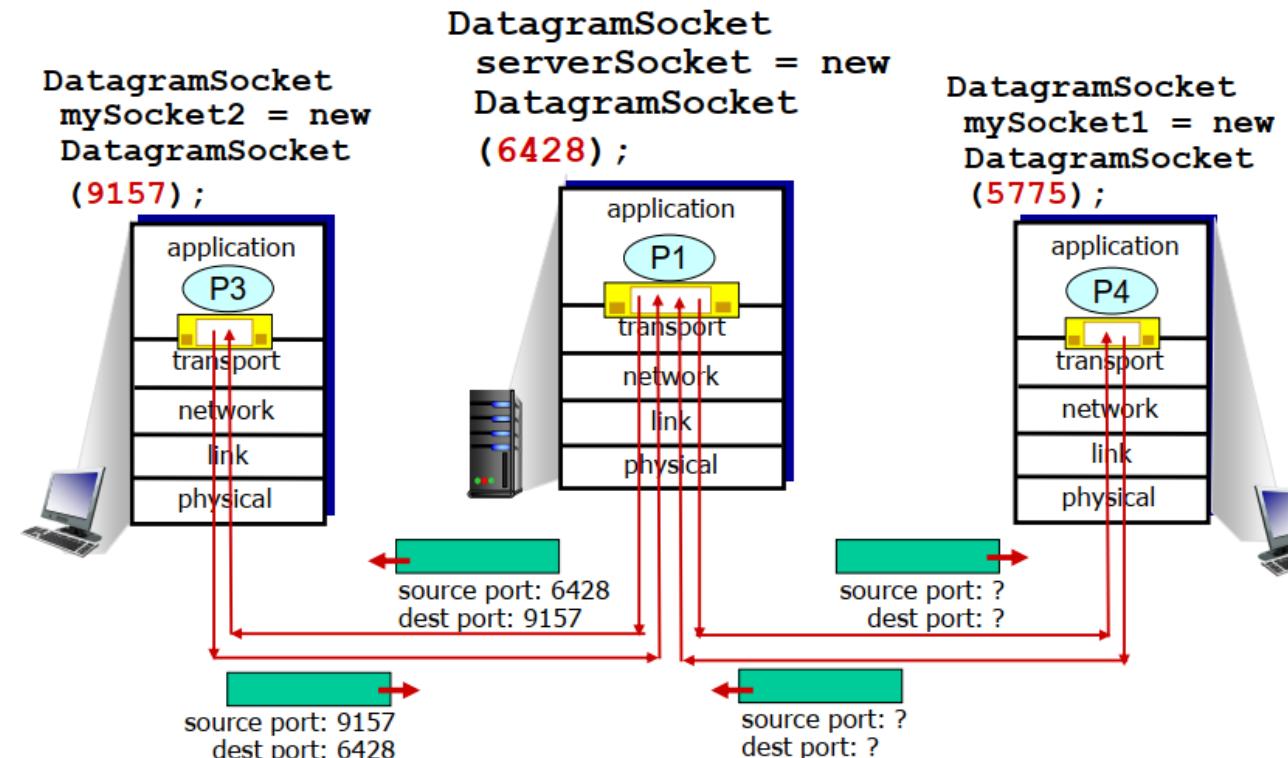




Trắc nghiệm

Câu 1: Cho sơ đồ multiplexing/demultiplexing như hình bên. Số port nguồn và số port đích khi truyền từ P1 sang P4 theo thứ tự là?

- A. 6428, 5775
- B. 5775, 6428
- C. 6428, 6428
- D. 5775, 5775





BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

3

VẬN CHUYỂN PHI KẾT NỐI: UDP



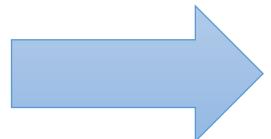
Tổng quan UDP

- Truyền tổng lực best-effort: UDP segments có thể bị mất mát và truyền không theo thứ tự.
- Phi kết nối (Connectionless):
 - Không bắt tay giữa bên gửi và nhận.
 - Không thiết lập và duy trì kết nối giữa bên gửi và nhận.
 - Mỗi segment UDP được xử lý độc lập.
- Ứng dụng dùng UDP:
 - DNS (Domain Name System).
 - SNMP (Simple Network Management Protocol).
 - Ứng dụng đa phương tiện trực tuyến chịu mất mát, cần tốc độ.

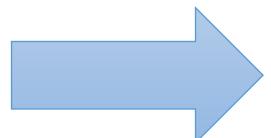


Tổng quan UDP

Giải pháp để truyền tin
cây trên UDP?



Thêm độ tin cậy tại tầng Application

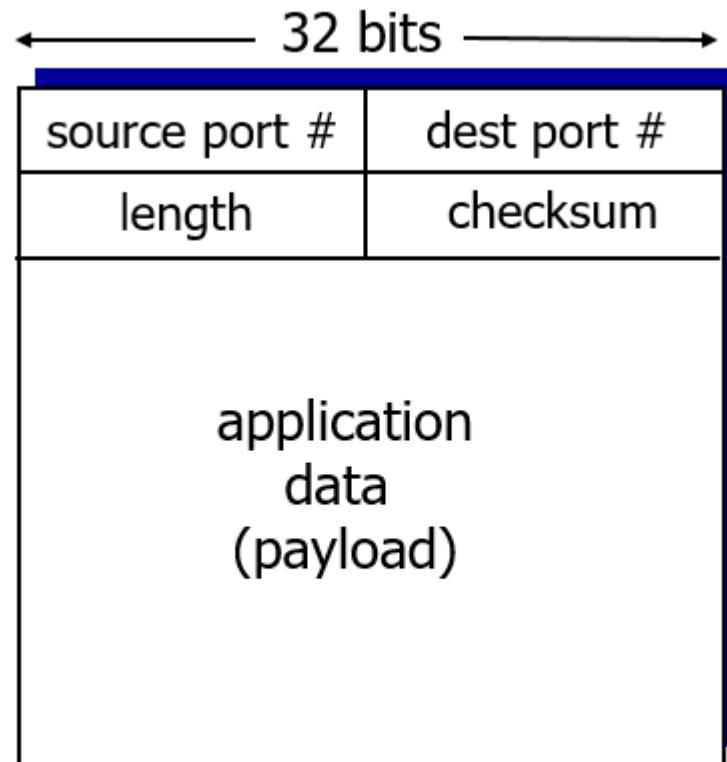


Phục hồi lỗi tại các ứng dụng cụ thể



Cấu trúc UDP Segment

- Header: kích thước 8 byte, gồm 4 trường, mỗi trường 2 byte:
 - Port nguồn.
 - Port đích.
 - Length: độ dài (header + payload) (byte).
 - Checksum: dò tìm lỗi.

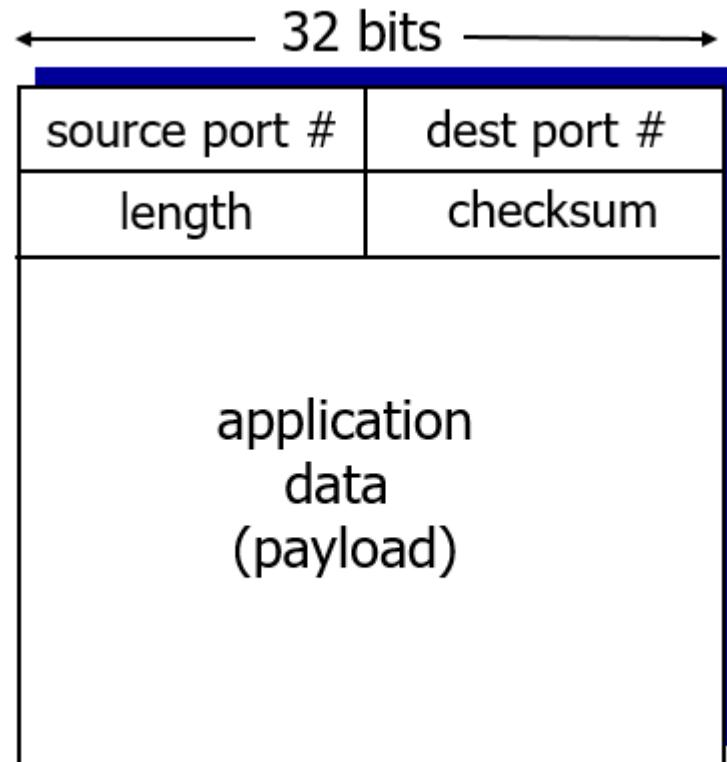


UDP segment format



UDP Checksum

- Bên gửi: xem nội dung segment như chuỗi 16 bit → tính tổng bù 1 của các chuỗi → lưu vào trường checksum.
- Bên nhận: tính lại checksum, so sánh với giá trị lưu trong checksum → Nếu khác, có lỗi.



UDP segment format



Tính Checksum

VD: tính checksum của 2 chuỗi số sau: 1110110001100111, 1010011101111101.

bit dư:

$$\begin{array}{r} 1110110001100111 \\ 1010011101111101 \\ \hline 11001001111100100 \end{array}$$

A red bracket underlines the last four bits of the result, and a red arrow points to the right from the bracket.

Tổng
Checksum

$$\begin{array}{r} 100100111100101 \\ 0110110000011010 \\ \hline \end{array}$$



Trắc nghiệm

Câu 1: UDP checksum có tác dụng gì?

- A. Kiểm tra lỗi trong gói dữ liệu tại bên nhận.
- B. Lưu kích thước của gói dữ liệu.
- C. Kiểm tra thứ tự của các gói dữ liệu tại bên nhận.
- D. Lưu địa chỉ IP nguồn và IP đích.



Trắc nghiệm

Câu 2: Độ dài header trong UDP segment là?

- A. 20
- B. 4
- C. 6
- D. 8



Trắc nghiệm

Câu 3: Tính checksum của 2 chuỗi số sau:
1011101101111001, 1100011001010111

- A. 1000000101010000
- B. 1000000111010001
- C. 0111111000101110
- D. 0111111000001110



BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

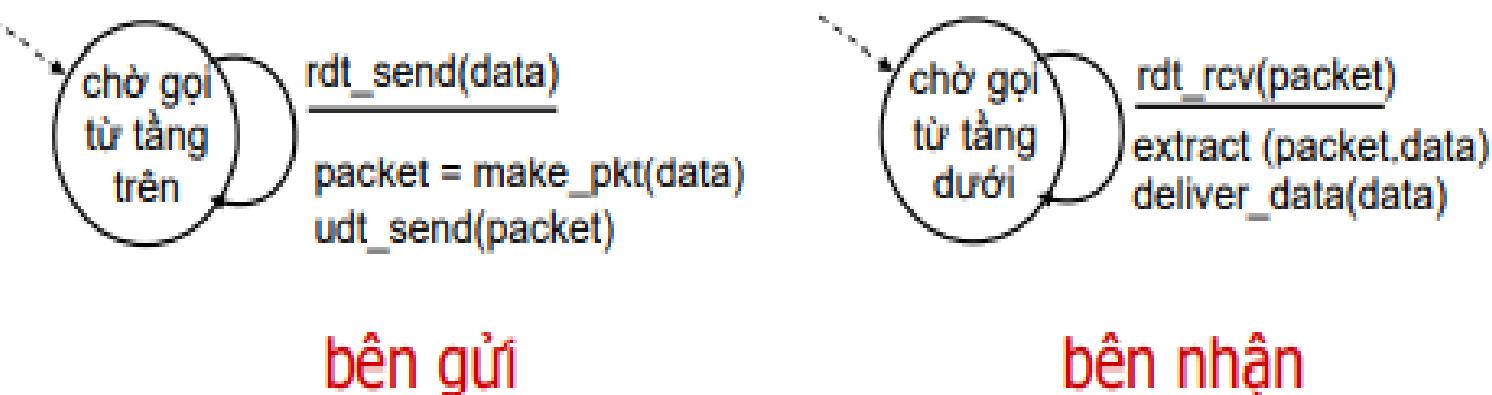
4

CÁC NGUYÊN LÝ TRUYỀN DỮ LIỆU TIN CẬY



RDT 1.0: Kênh tin cậy hoàn toàn

- Kênh truyền: tin cậy hoàn toàn (không lỗi bit, không mất gói).
- Bên gửi và nhận chỉ có 1 trạng thái duy nhất.
- Bên nhận không cần phản hồi.
- Sơ đồ FSM (finite-state machine):





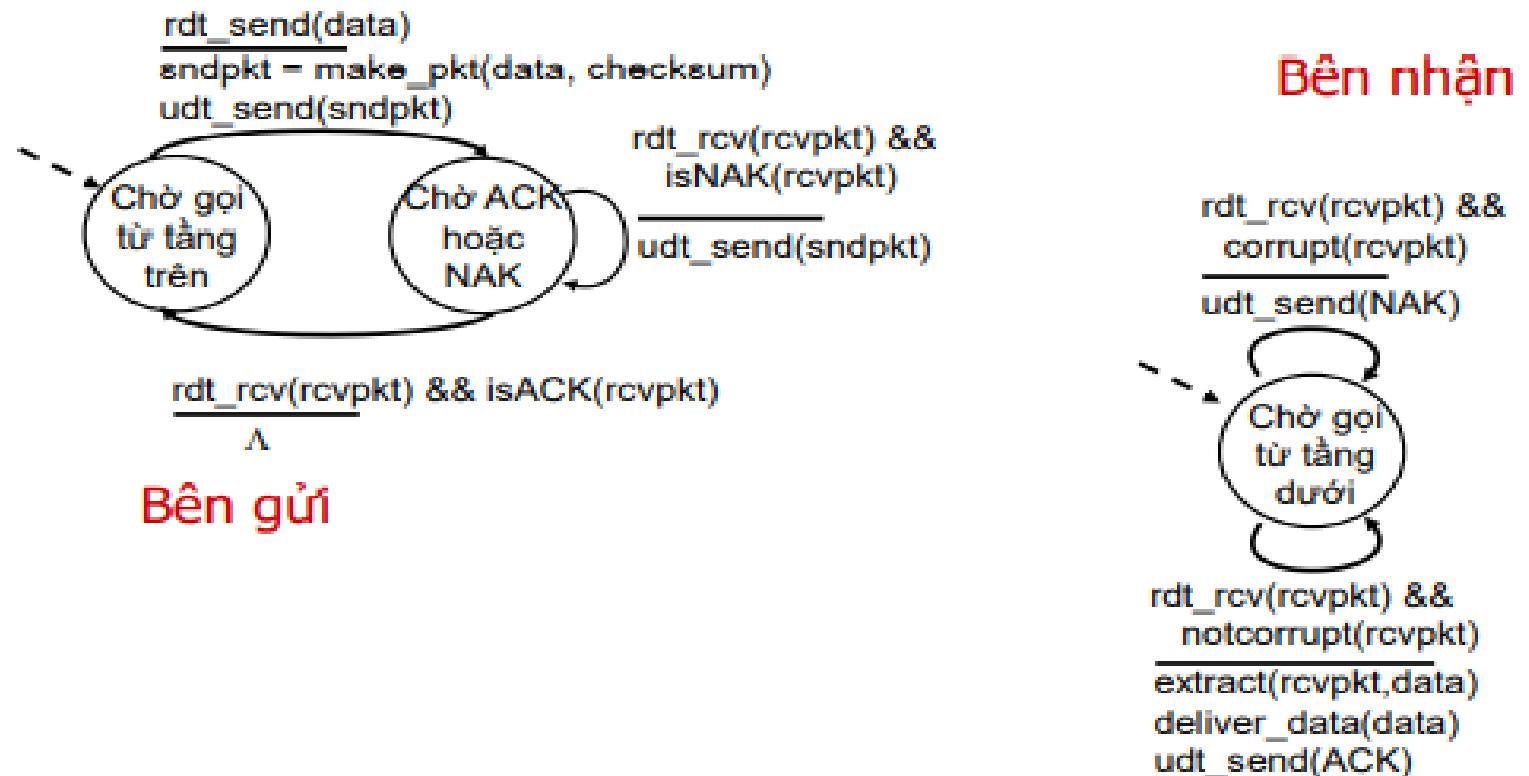
RDT 2.0: Kênh với các lỗi

- Kênh truyền: có thể lỗi bit, không mất gói.
- Các cơ chế bổ sung:
- Phát hiện lỗi: checksum
- Bên nhận phản hồi: các thông điệp điều khiển (ACK: packet được nhận thành công, NAK: packet bị lỗi).
- Bên gửi truyền lại: khi nhận được NAK (báo lỗi).



RDT 2.0: Kênh với các lỗi

- Sơ đồ FSM (finite-state machine):





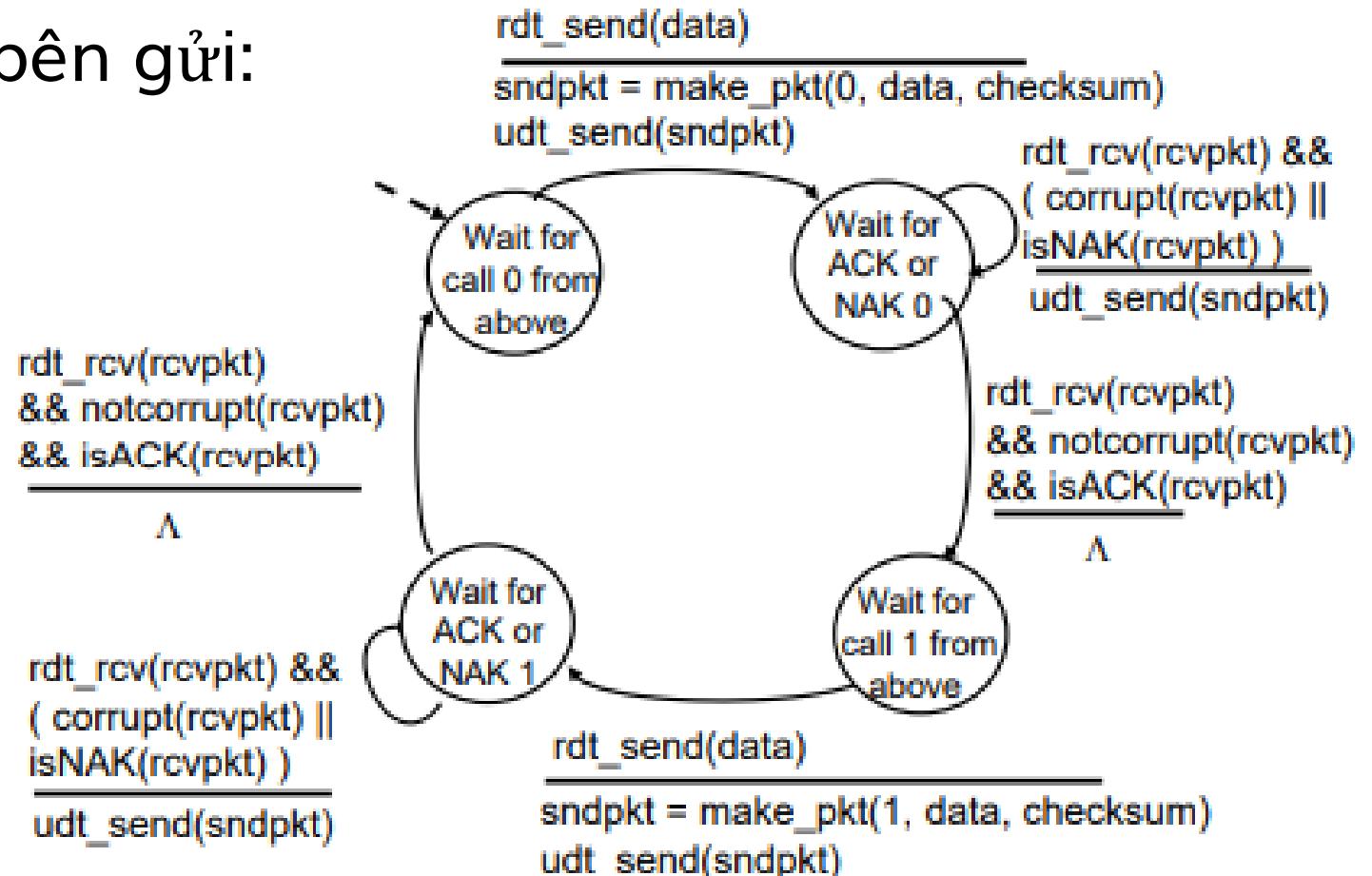
RDT 2.1: ACK/NAK bị hỏng

- RDT 2.0 chưa tính đến tình huống ACK/NAK bị hỏng.
- Giải quyết: bên gửi truyền lại packet nếu ACK/NAK bị hỏng.
- Vấn đề: trùng lặp packet tại bên nhận?
 - Bên gửi: thêm số thứ tự (sequence number (1 bit): 0 và 1) cho mỗi packet.
 - Bên nhận: loại bỏ các packet trùng lặp.



RDT 2.1: ACK/NAK bị hỏng

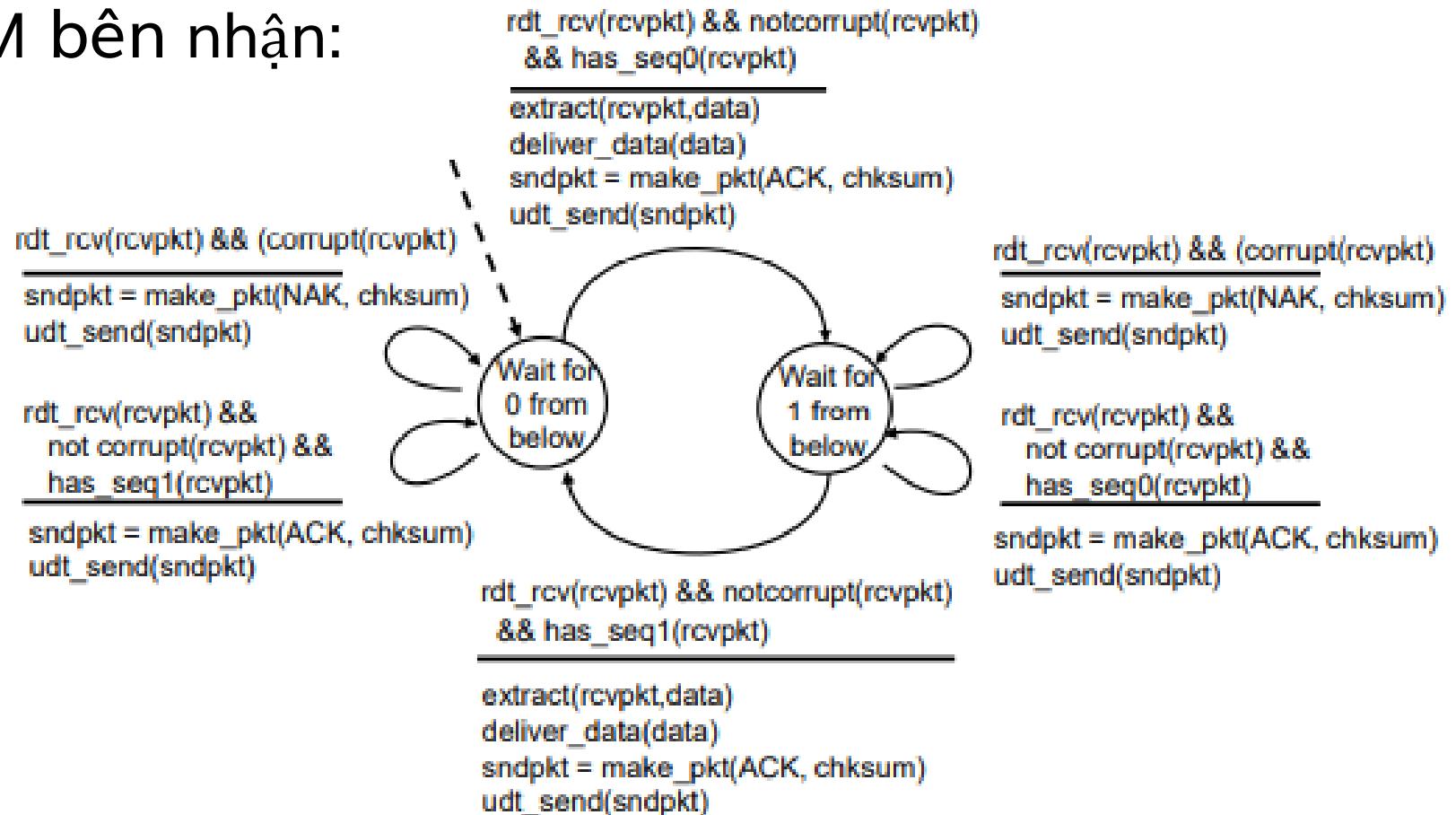
➤ FSM bên gửi:





RDT 2.1: ACK/NAK bị hỏng

➤ FSM bên nhận:





RDT 2.2: Không dùng NAK

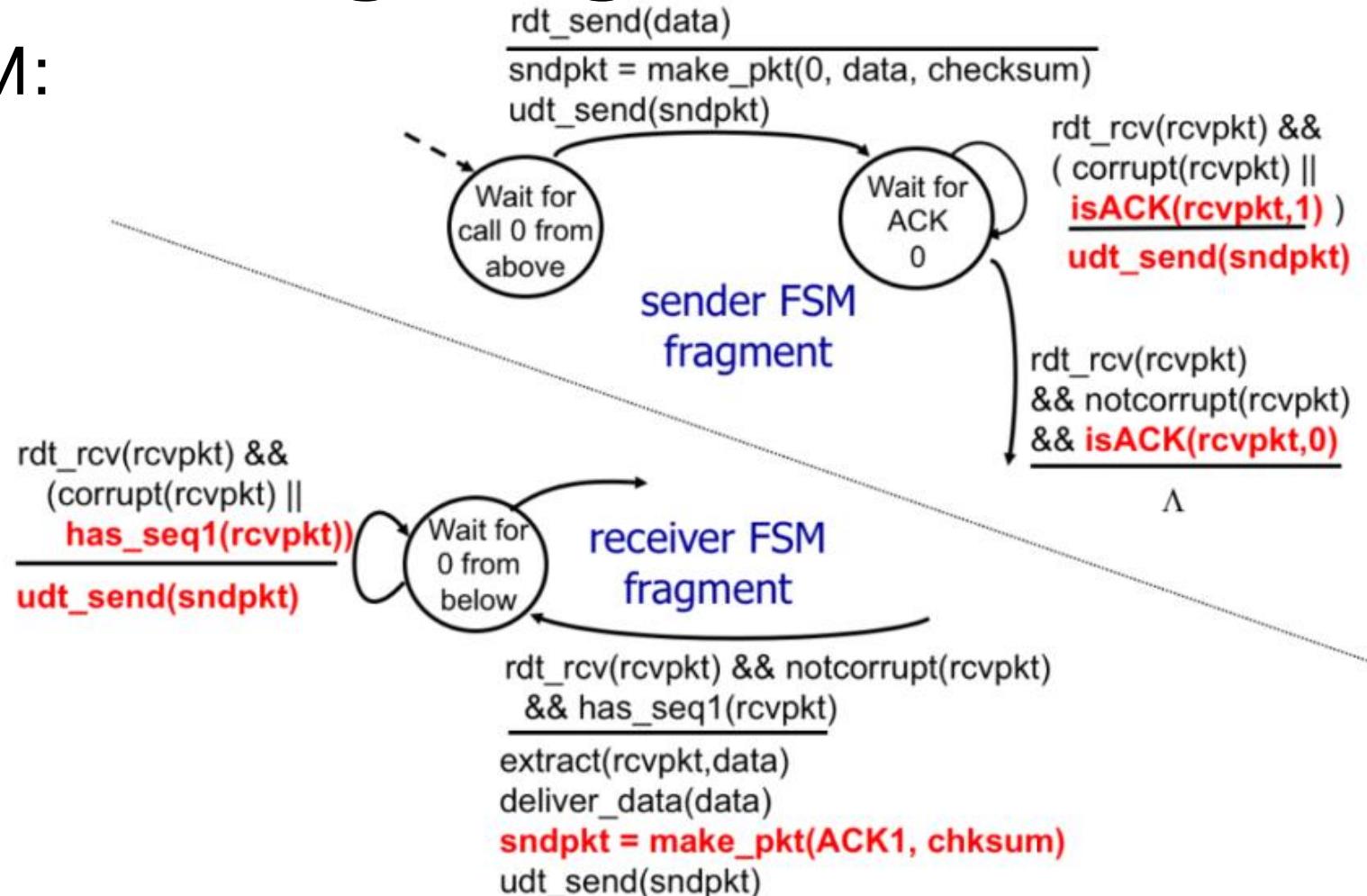
- Giống rdt 2.1 nhưng không sử dụng NAK
- Bên nhận: gửi ACK kèm sequence number cho gói cuối cùng nhận được thành công
- Bên gửi: nhận được 2 ACK trùng sequence number → gửi lại





RDT 2.2: Không dùng NAK

➤ FSM:





RDT 3.0: Kênh với lỗi, mất mát

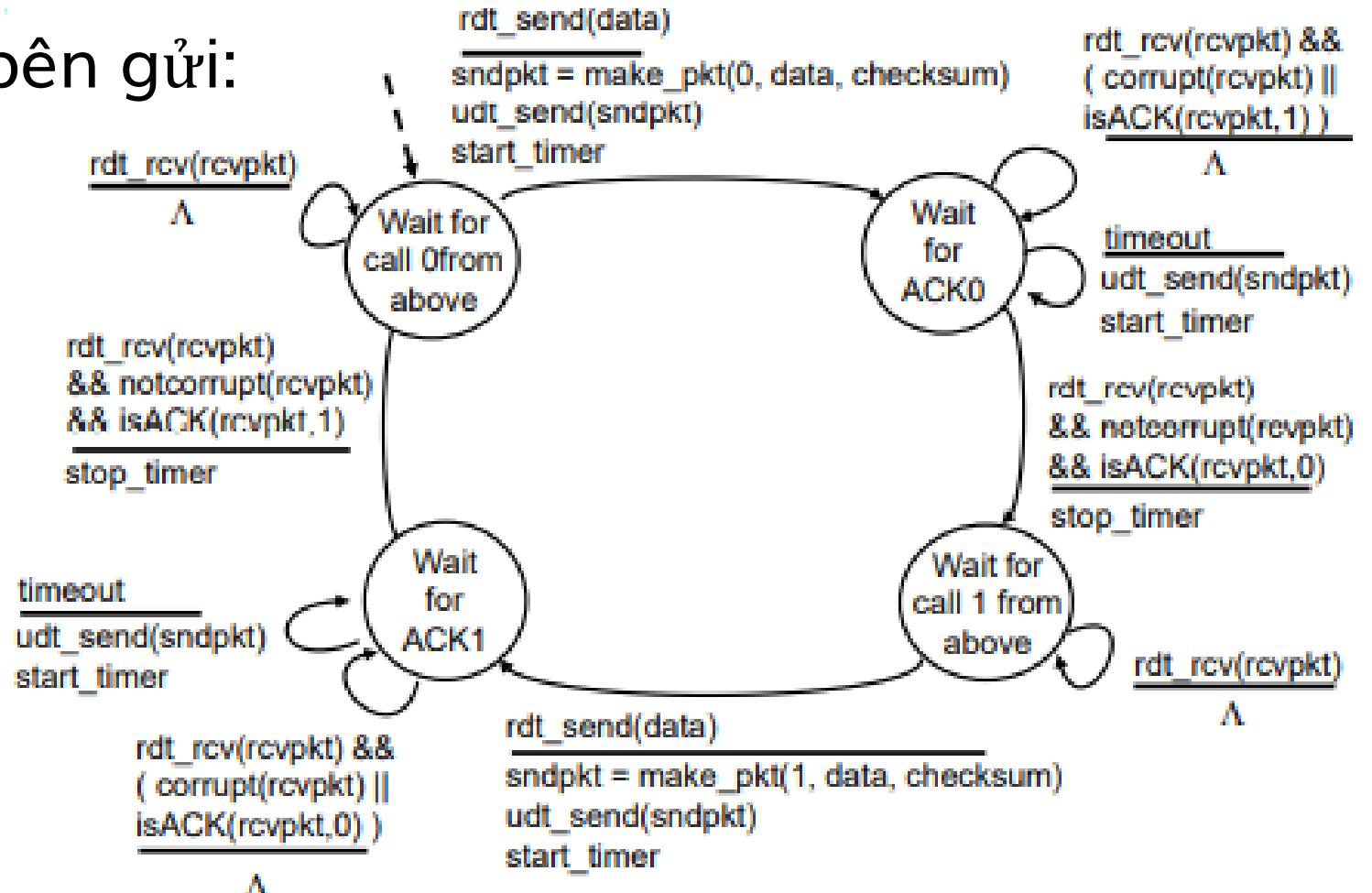
- Kênh truyền: có thể lỗi bit, có thể mất gói.
- Cơ chế bổ sung:
 - Truyền lại nếu không có ACK được nhận trong khoảng thời gian hợp lý.
 - Bộ định giờ đếm lùi (countdown timer).





RDT 3.0: Kênh với lỗi, mất mát

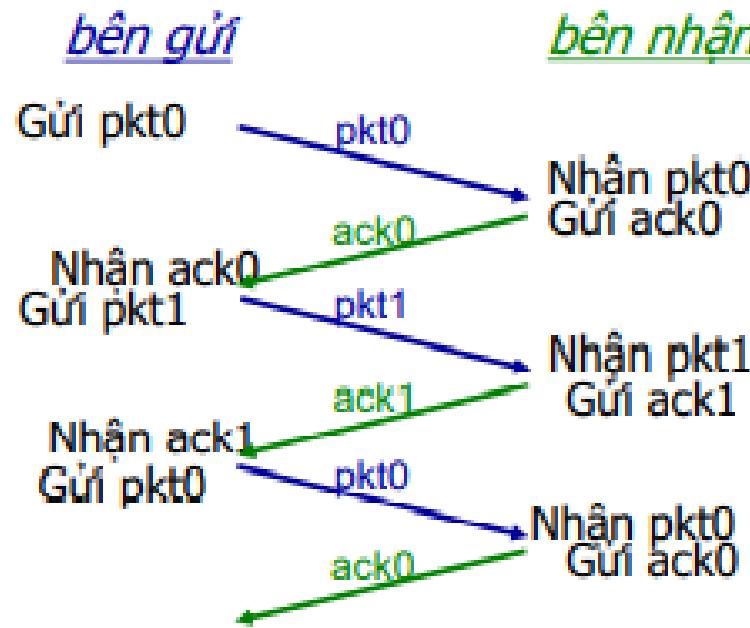
➤ FSM bên gửi:



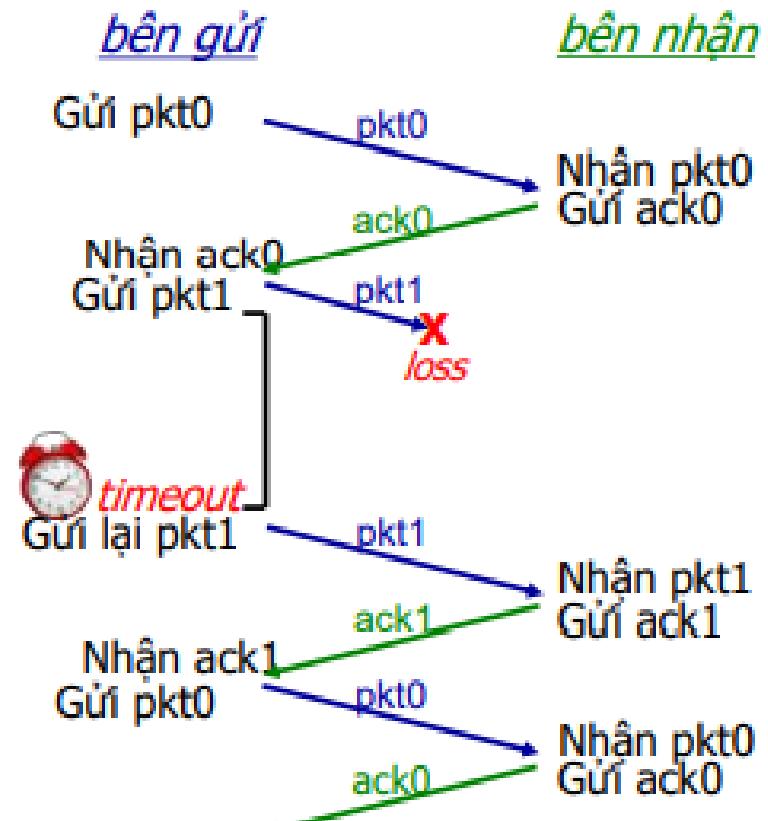


RDT 3.0: Kênh với lỗi, mất mát

➤ Hành động của rdt 3.0:



(a) Không mất mát

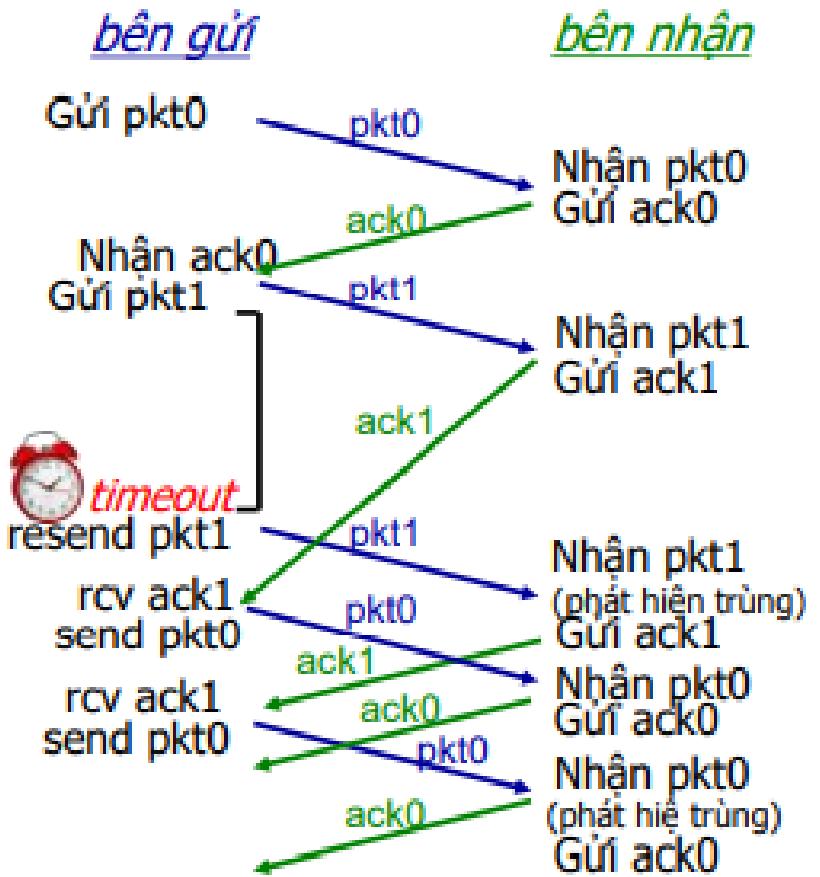
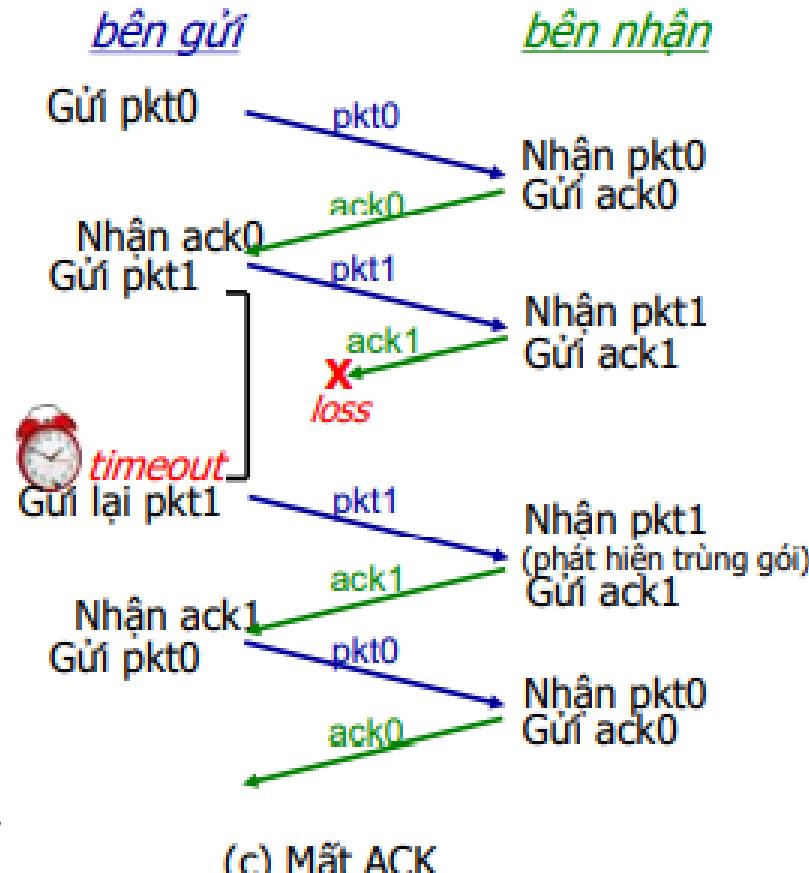


(b) Mất gói



RDT 3.0: Kênh với lỗi, mất mát

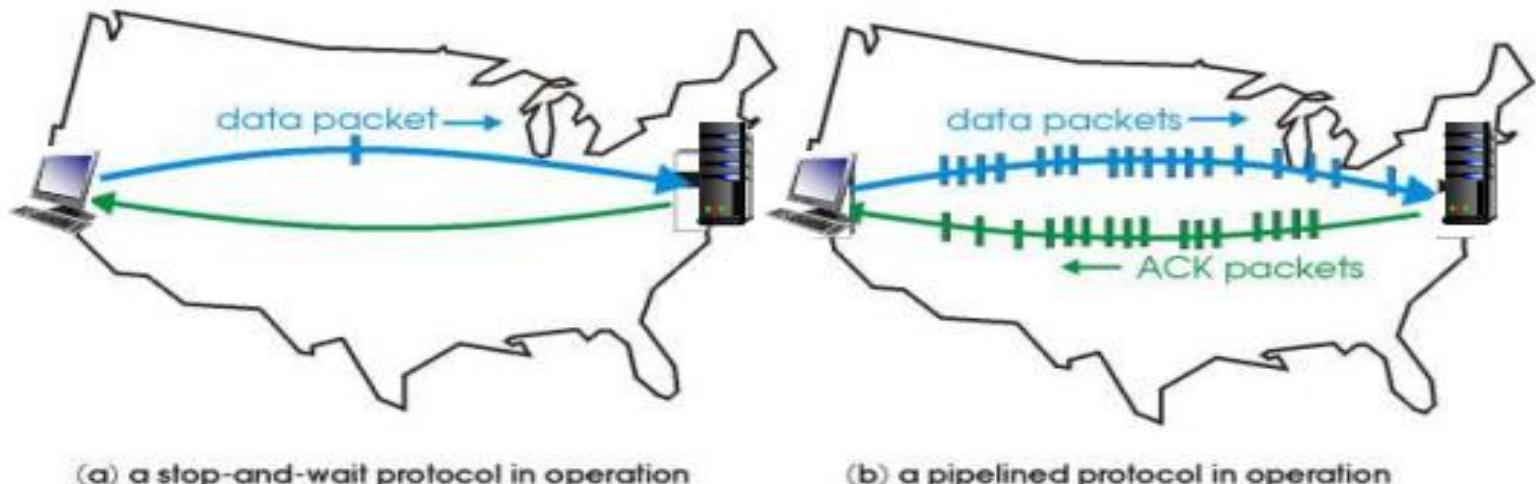
➤ Hành động của rdt 3.0:





Giới thiệu Pipelined Protocol

- Rdt cho hiệu suất thấp hơn do cơ chế stop-and-wait.
- Pipelining: bên gửi cho phép gửi nhiều gói đồng thời, không cần chờ báo nhận được.
- Có 2 dạng phổ biến: Go-Back-N và Lặp có lựa chọn (Selective Repeat).





Hoạt động Go-Back-N

sender window (N=4)

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

bên gửi

send pkt0
send pkt1
send pkt2
send pkt3
(wait)

rcv ack0, send pkt4
rcv ack1, send pkt5

ignore duplicate ACK

pkt 2 timeout

send pkt2
send pkt3
send pkt4
send pkt5

bên nhận

receive pkt0, send ack0
receive pkt1, send ack1
receive pkt3, discard,
(re)send ack1

receive pkt4, discard,
(re)send ack1
receive pkt5, discard,
(re)send ack1

rcv pkt2, deliver, send ack2
rcv pkt3, deliver, send ack3
rcv pkt4, deliver, send ack4
rcv pkt5, deliver, send ack5

- Gửi N packet liên tiếp.
- Các gói nhận không đúng thứ tự → hủy đi mà không đếm.
- Timeout: truyền lại tất cả các packet mà không được ACK.



Hoạt động Selective Repeat

sender window (N=4)

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

Bên gửi

gửi pkt0
gửi pkt1
gửi pkt2
gửi pkt3
(đợi)

nhận ack0, gửi pkt4
nhận ack1, gửi pkt5

Ghi nhận ack3 đã đến

pkt 2 timeout
gửi pkt2

Ghi nhận ack4 đã đến

Ghi nhận ack5 đã đến

Q: việc gì xảy ra khi ack2 đến?

Bên nhận

nhận pkt0, gửi ack0
nhận pkt1, gửi ack1

nhận pkt3, buffer,
gửi ack3

nhận pkt4, buffer,
gửi ack4

nhận pkt5, buffer,
gửi ack5

nhận pkt2; chuyển pkt2,
pkt3, pkt4, pkt5; gửi ack2

- Gửi N packet liên tiếp
- Các gói nhận không đúng thứ tự → vẫn được ACK và lưu vào buffer
- Timeout: chỉ truyền lại packet không được ACK



Trắc nghiệm

Câu 1: Quá trình truyền nhận 10 segment được thực hiện theo Go-Back-N protocol. Bên gửi có window size = 4 và gửi đi 4 segment theo thứ tự là 0, 1, 2, 3. Sau đó bên gửi nhận về 3 ACK theo thứ tự là 0, 2 và 3. Hành động tiếp theo của bên gửi là:

- A. Gửi lại segment 1
- B. Gửi lại segment 1, 2, 3 và gửi tiếp segment 4
- C. Gửi tiếp segment 4
- D. Gửi tiếp segment 4, 5, 6



Trắc nghiệm

Câu 2: Quá trình truyền nhận 10 segment được thực hiện theo Selective Repeat protocol. Bên gửi có window size = 4 và gửi đi 4 segment theo thứ tự là 0, 1, 2, 3. Sau đó bên gửi nhận về 3 ACK theo thứ tự là 0, 2 và 3. Giả sử rằng quá trình timeout chưa xảy ra. Hành động tiếp theo của bên gửi là:

- A. Gửi lại segment 1
- B. Gửi lại segment 1, 2, 3 và gửi tiếp segment 4
- C. Gửi tiếp segment 4
- D. Gửi tiếp segment 4, 5, 6



BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

5

VẬN CHUYỂN HƯỚNG KẾT NỐI: TCP

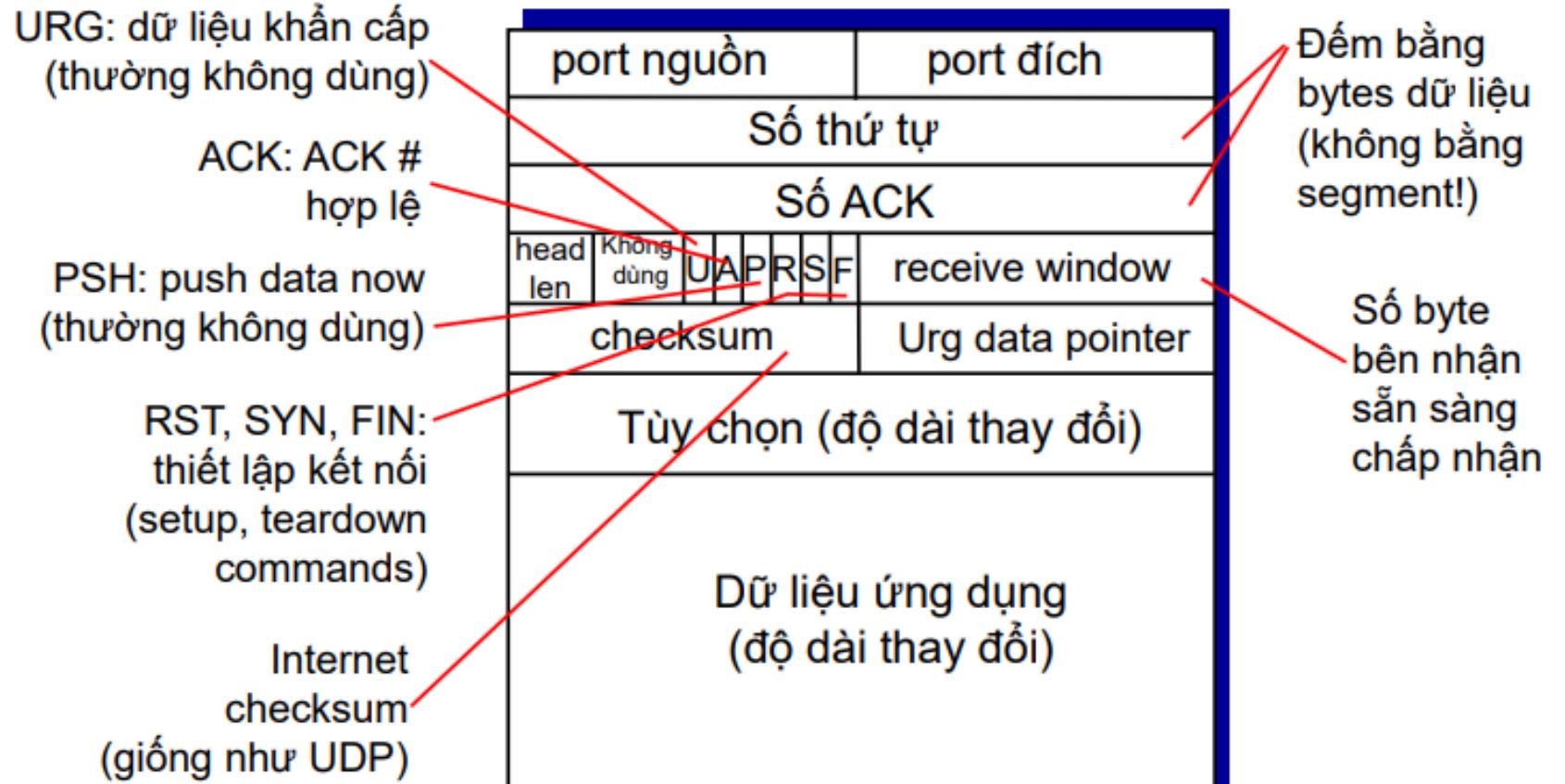


Tổng quan TCP

- Point-to-Point: 1 bên gửi, 1 bên nhận.
- Truyền tin cây (rdt), dòng byte theo thứ tự.
- Hướng kết nối (connection-oriented): bắt tay 3 bước (trao đổi các thông điệp điều khiển) khởi tạo trạng thái bên gửi và nhận trước khi trao đổi dữ liệu.
- Pipelined: điều khiển luồng và tắt nghẽn của TCP thiết lập kích thước cửa sổ (window size).
- Dữ liệu full duplex: luồng dữ liệu đi 2 chiều trong cùng 1 kết nối



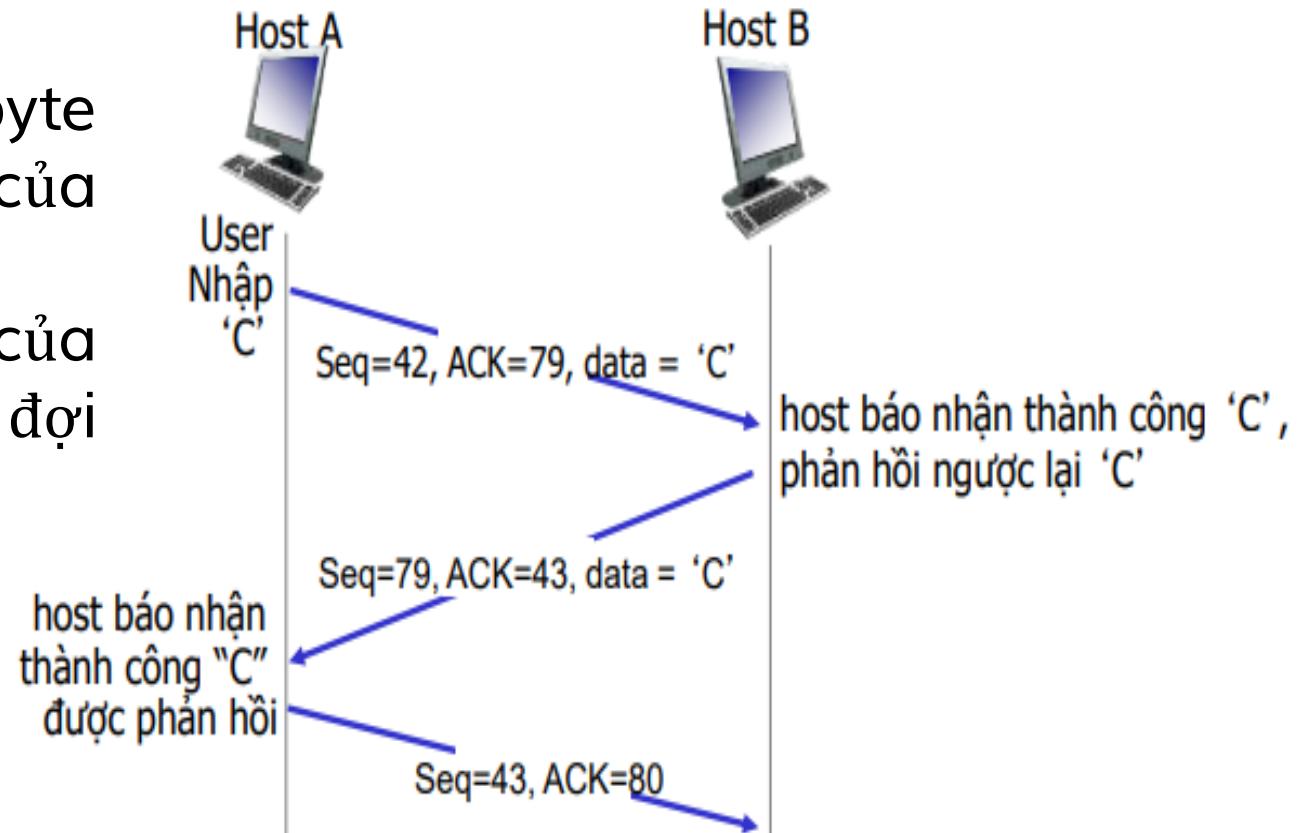
Cấu trúc TCP segment





Số thứ tự và số ACK

- Số thứ tự: là số của “byte đầu tiên” trong dữ liệu của segment.
- Số ACK: là số thứ tự của byte kế tiếp được mong đợi từ phía bên kia.



Tình huống telnet đơn giản

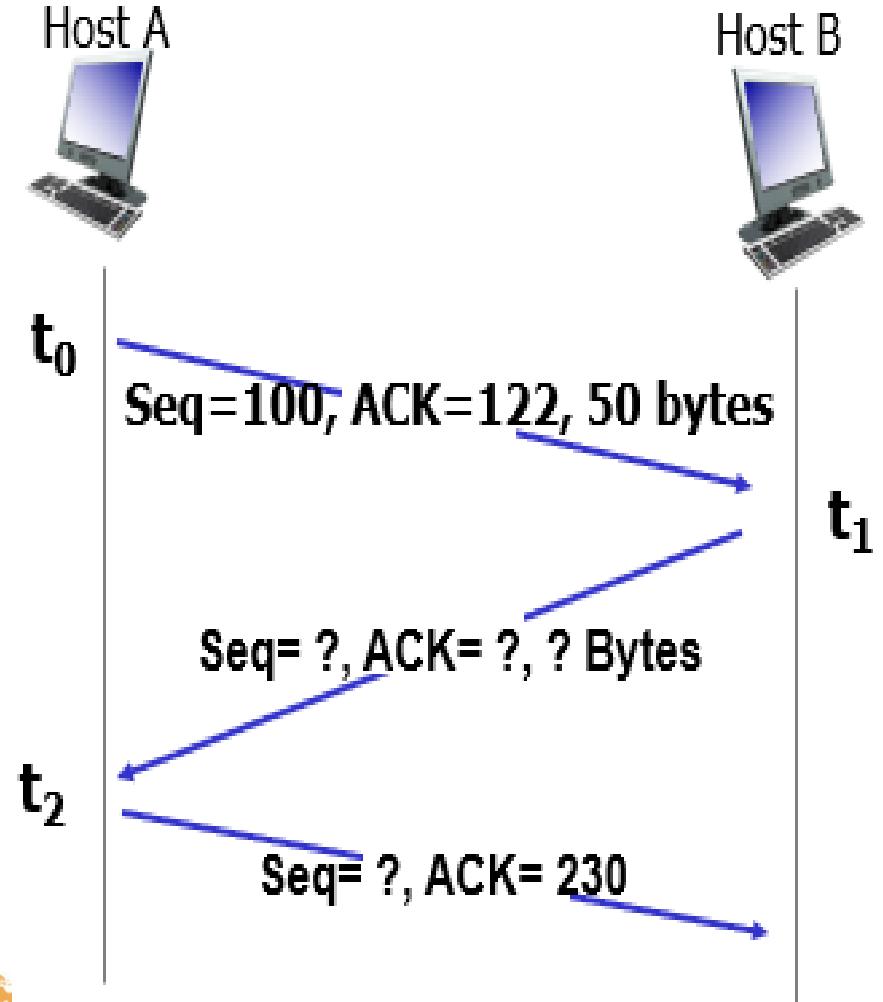


Trắc nghiệm

Cho sơ đồ truyền nhận TCP segment giữa 2 host như hình bên:

Câu 1: Giá trị Seq, ACK của B gửi về A tại thời điểm t_1 lần lượt là?

- A. 122, 172
- B. 100, 150
- C. 122, 150
- D. 150, 122





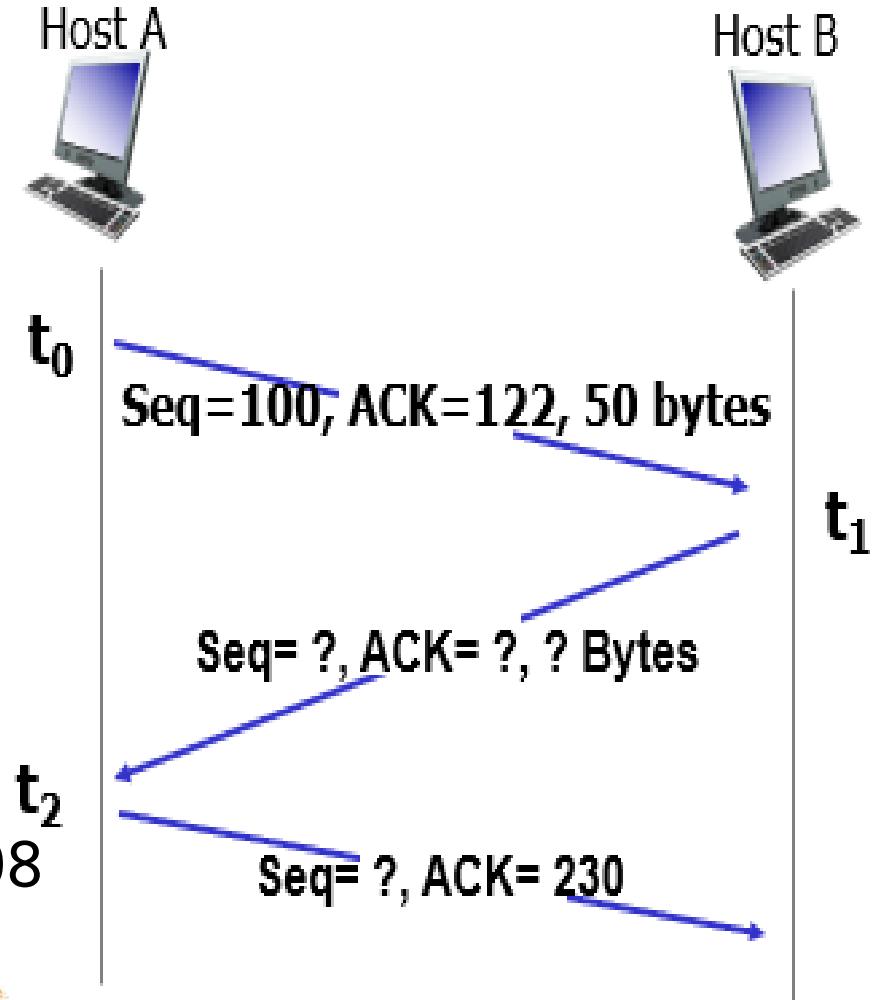
Trắc nghiệm

Câu 2: Lượng data (bytes) của B gửi về A tại thời điểm t_1 là?

- A. 180
- B. 80
- C. 108
- D. 166

$$\text{Số ACK} = \text{Seq} + \text{Data}$$

$$\text{Data} = \text{ACK} - \text{Seq} = 230 - 122 = 108$$

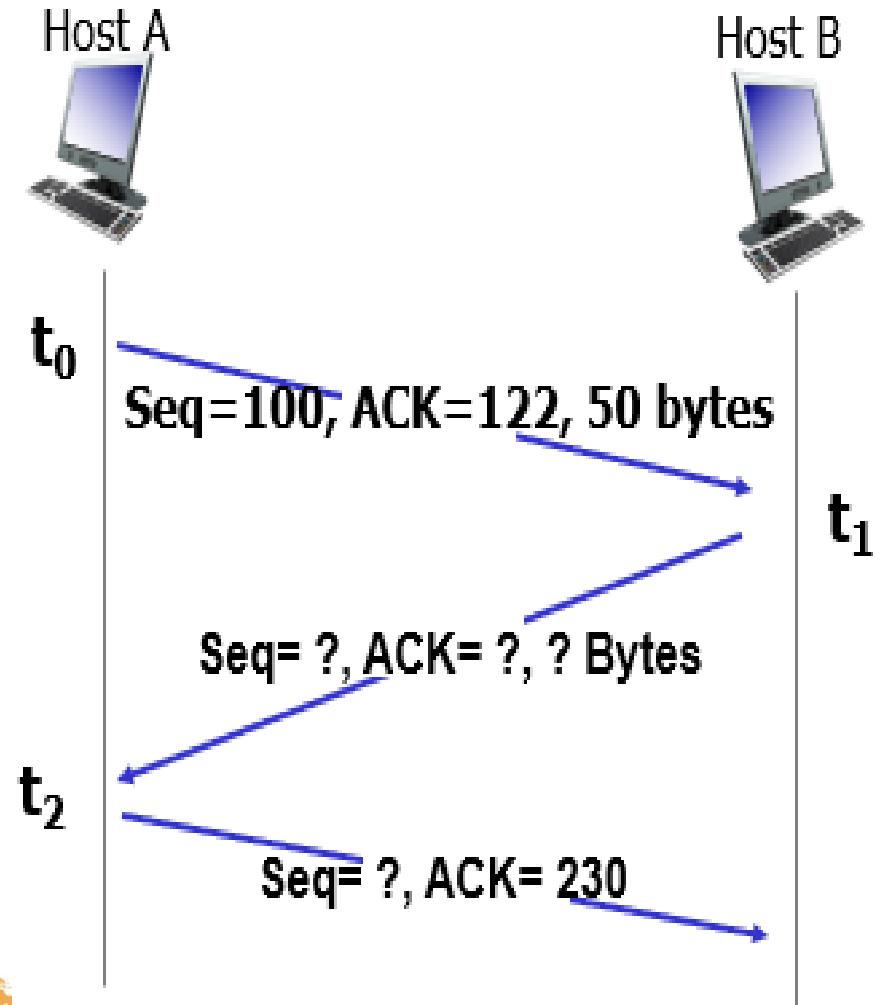




Trắc nghiệm

Câu 3: Seq của A truyền qua B tại thời điểm t_2 là?

- A. 150
- B. 122
- C. 230
- D. 108





TCP Round trip time (RTT) và timeout

➤ Công thức:

- EstimatedRTT = $(1 - \alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$
- DevRTT = $(1 - \beta) * \text{DevRTT} + \beta * |\text{SampleRTT} - \text{EstimatedRTT}|$
- TimeoutInterval = EstimatedRTT + 4 * DevRTT

➤ Chú thích:

- SampleRTT: RTT thực tế, thời gian tính từ khi truyền segment đến khi báo nhận ACK
- DevRTT: RTT ước tính
- TimeoutInterval: khoảng thời gian định thì

➤ Thông thường: chọn $\alpha = 0.125$, $\beta = 0.25$



TCP Round trip time (RTT) và timeout

Giá trị đầu tiên của EstimatedRTT và DevRTT được tính như thế nào ?

- A. EstimatedRTT = SampleRTT; DevRTT = SampleRTT / 2
- B. EstimatedRTT = SampleRTT / 2; DevRTT = SampleRTT
- C. EstimatedRTT = SampleRTT * 2, DevRTT = SampleRTT / 2
- D. EstimatedRTT = SampleRTT, DevRTT = SampleRTT * 2



Trắc nghiệm

Hiện tại, thông số EstimatedRTT là 150ms và DevRTT là 10ms. TCP sender nhận về thêm 3 ACK và đo được SampleRTT lần lượt là 80 ms, 110 ms và 140 ms. Lấy $\alpha = 0.125$ và $\beta = 0.25$.

Câu 1: Tính EstimatedRTT sau khi nhận về ACK thứ hai:

- A. 150 ms
- B. 137.34 ms
- C. 141.25 ms
- D. 145 ms

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

$$\text{EstimatedRTT}_1 = (1 - 0.125) * 150 + 0.125 * 80 = 141.25 \text{ ms}$$

$$\text{EstimatedRTT}_2 = (1 - 0.125) * 141.25 + 0.125 * 110 = 137.34 \text{ ms}$$



Trắc nghiệm

Hiện tại, thông số EstimatedRTT là 150ms và DevRTT là 10ms. TCP sender nhận về thêm 3 ACK và đo được SampleRTT lần lượt là 80 ms, 110 ms và 140 ms. Lấy $\alpha = 0.125$ và $\beta = 0.25$.

Câu 2: Tính DevRTT sau khi nhận về ACK thứ hai:

- A. 25 ms
- B. 53.125 ms
- C. 26.5625 ms**
- D. 22.8125 ms

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

Lần 1: $\text{DevRTT}_1 = (1 - 0.25) * 10 + 0.25 * |80 - 150| = 25 \text{ ms}$

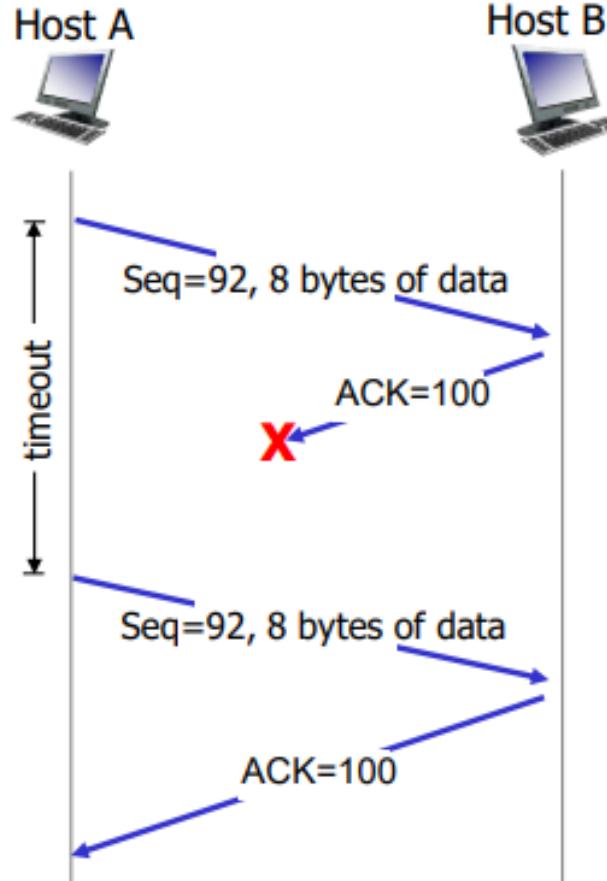
Lần 2:

$\text{EstimatedRTT}_1 = (1 - 0.125) * 150 + 0.125 * 80 = 141.25 \text{ ms}$

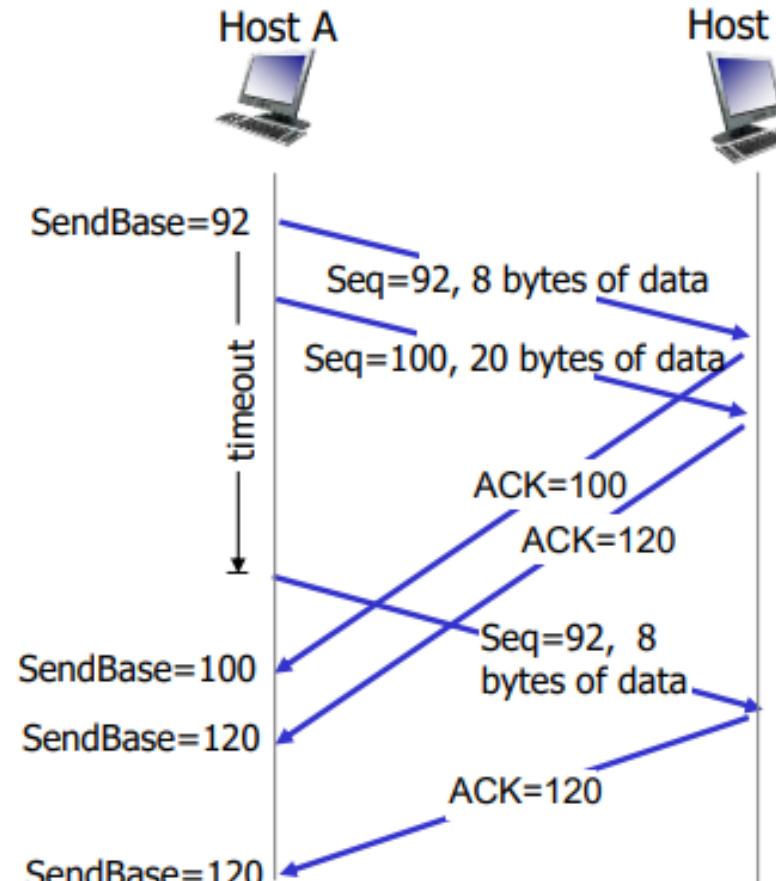
$\text{DevRTT}_2 = (1 - 0.25) * 25 + 0.25 * |110 - 141.25| = 26.5625 \text{ ms}$



TCP: tình huống truyềnlại



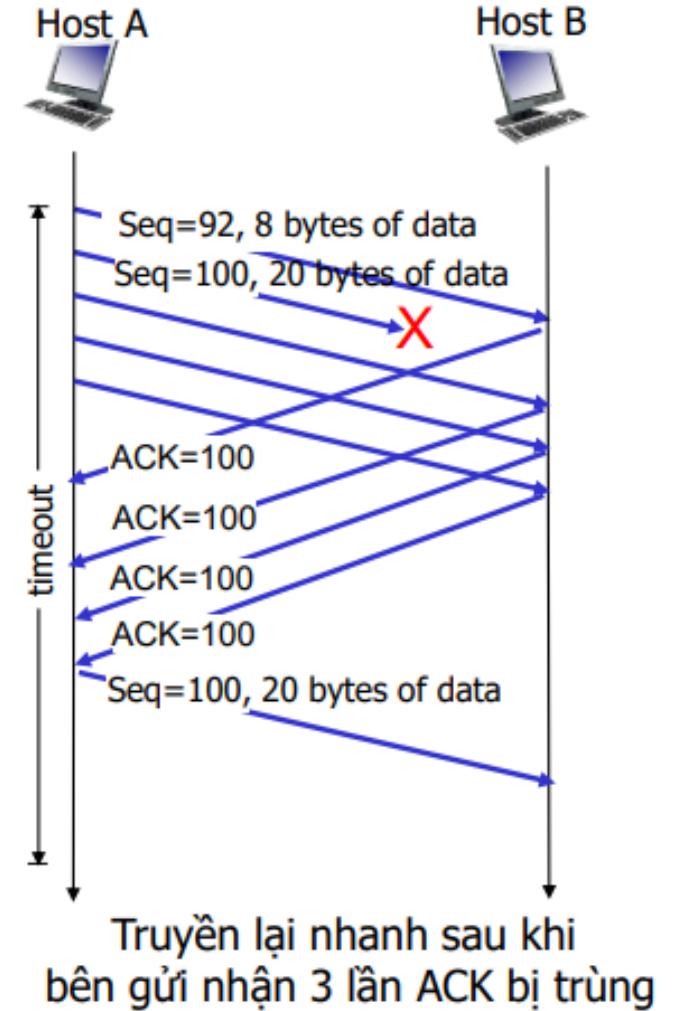
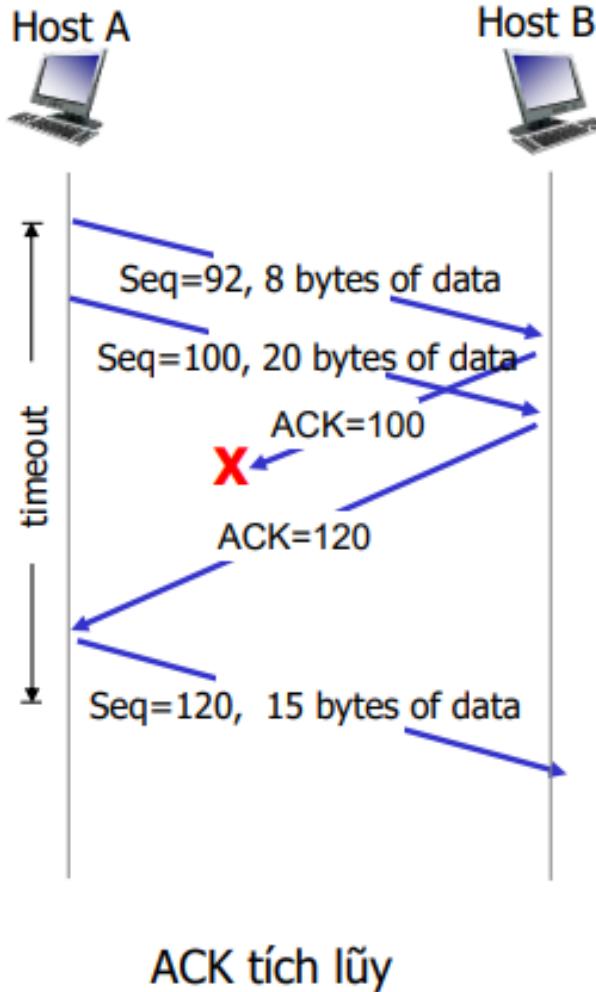
Tình huống mất ACK



Timeout sớm



TCP: tình huống truyền lại





TCP bắt tay 3-way

Trạng thái client

LISTEN

SYNSENT

ESTAB
SYNACK(x) vừa được nhận
cho hay server vẫn còn sống;
send ACK for SYNACK;
this segment may contain
client-to-server data

Chọn số thứ tự ban đầu, x
Gửi TCP SYN msg



SYNbit=1, Seq= x

SYNbit=1, Seq= y
ACKbit=1; ACKnum= $x+1$

ACKbit=1, ACKnum= $y+1$

Trạng thái server

LISTEN

SYN RCVD

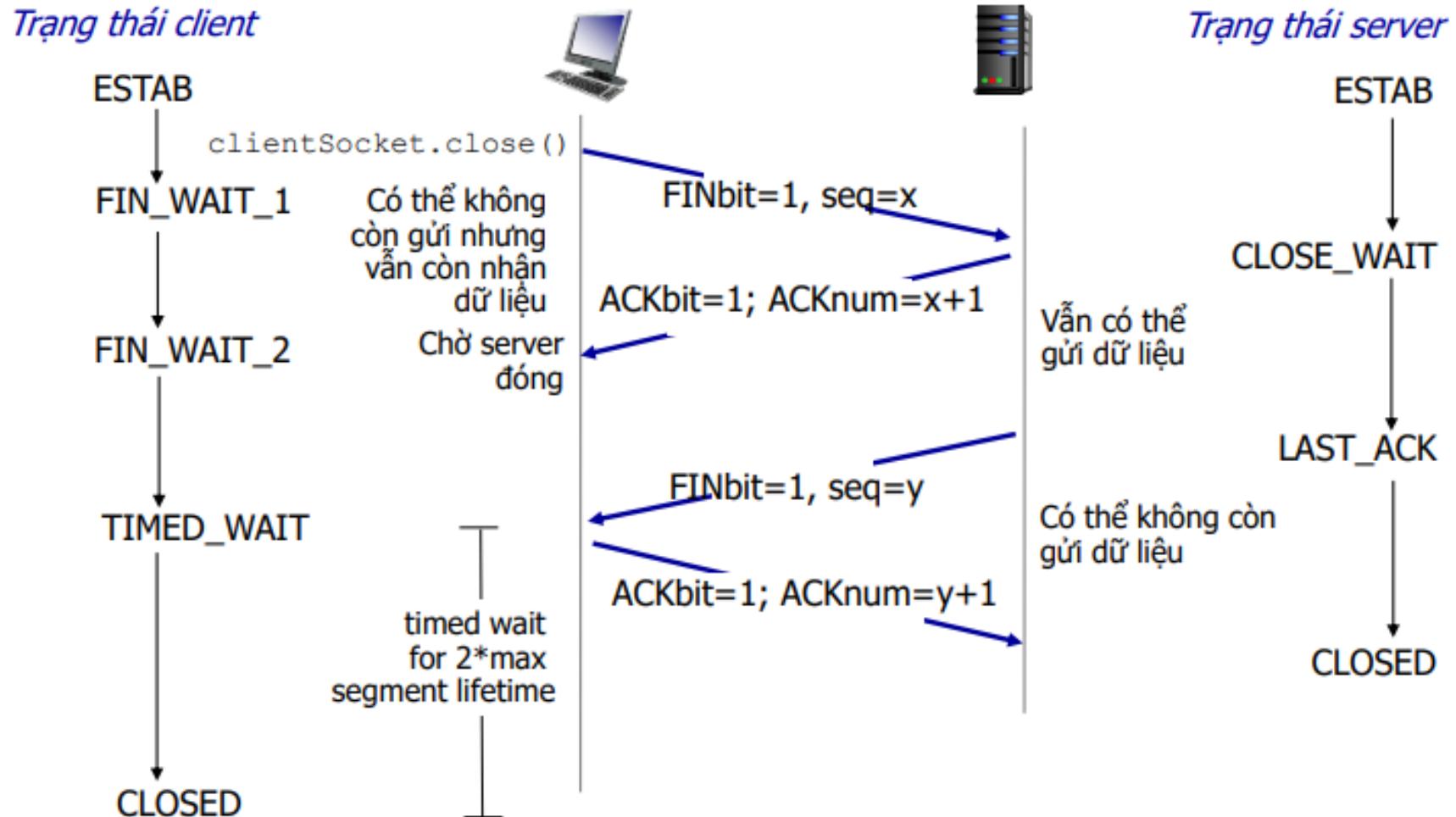
Chọn số thứ tự ban đầu, y
gửi TCP SYNACK
msg, acking SYN

ACK(y) vừa được nhận
cho hay client vẫn sống

ESTAB



TCP đóng kết nối





Trắc nghiệm

Câu 1: Gói tin TCP yêu cầu kết nối có giá trị của các cờ là:

- A. SYN = 1, ACK = 1
- B. SYN = 1, ACK = 0
- C. SYN = 0, ACK = 1
- D. SYN = 0, ACK = 0



Trắc nghiệm

Câu 2: Khi đóng kết nối thì bên gửi và bên nhận sẽ làm gì?

- A. Gửi TCP segment với cờ FIN = 1.
- B. Gửi TCP segment với cờ FIN = 0.
- C. Đợi timeout và tự đóng kết nối.
- D. Tất cả đều sai.



BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

6

ĐIỀU KHIỂN TẮT NGHẼN

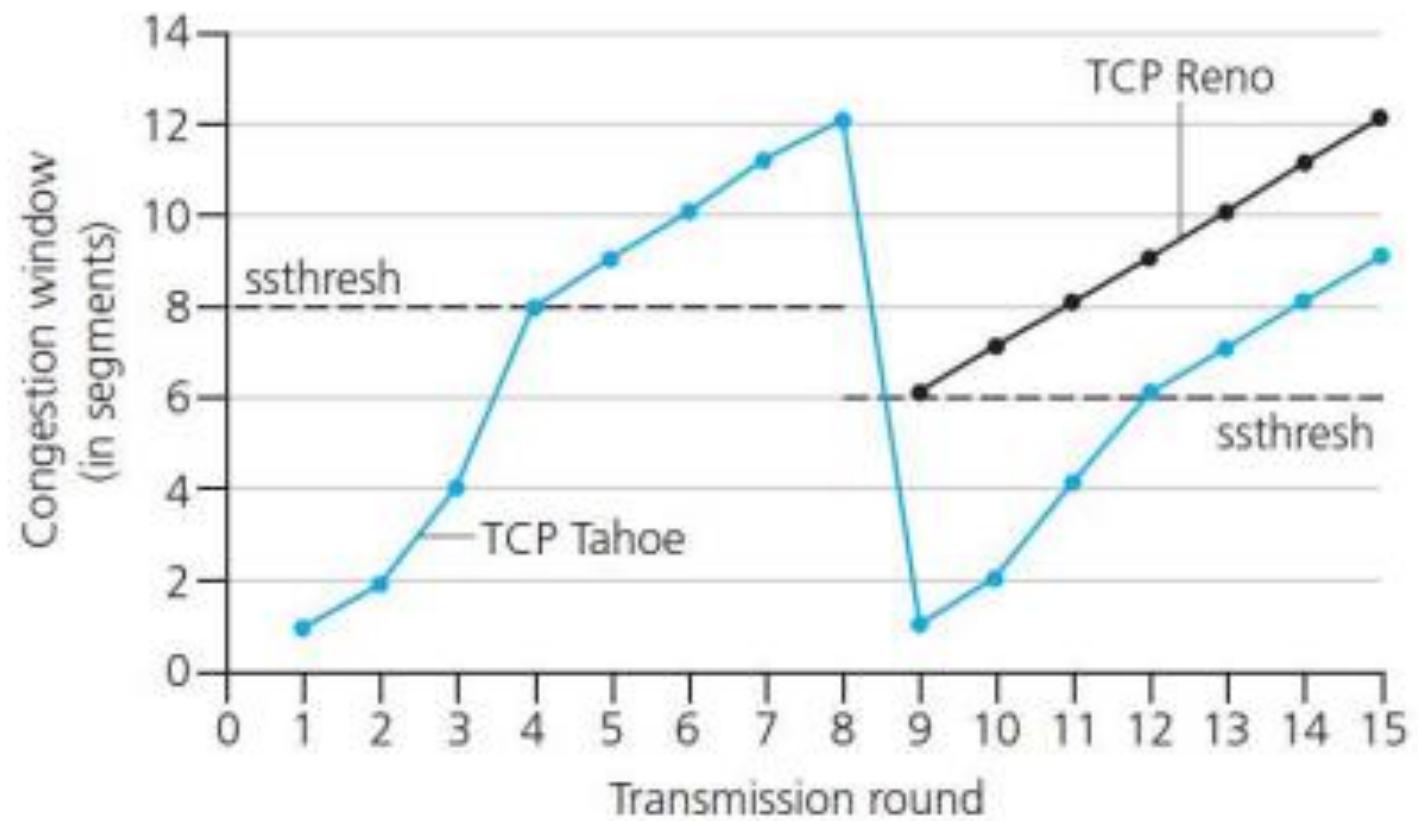


TCP Điều khiển tắt nghẽn

- Nguyên lý AIMD (additive increase, multiplicative decrease): bên gửi tăng tốc độ truyền (kích thước cửa sổ), thăm dò băng thông có thể sử dụng, cho đến mất mát gói xảy ra.
 - Additive increase: tăng cwnd bởi 1 MSS (Maximum Segment Size) mỗi RTT cho đến khi mất gói xảy ra.
 - Multiplicative decrease: giảm một nửa cwnd sau khi mất gói xảy ra.
- (cwnd: congestion window)



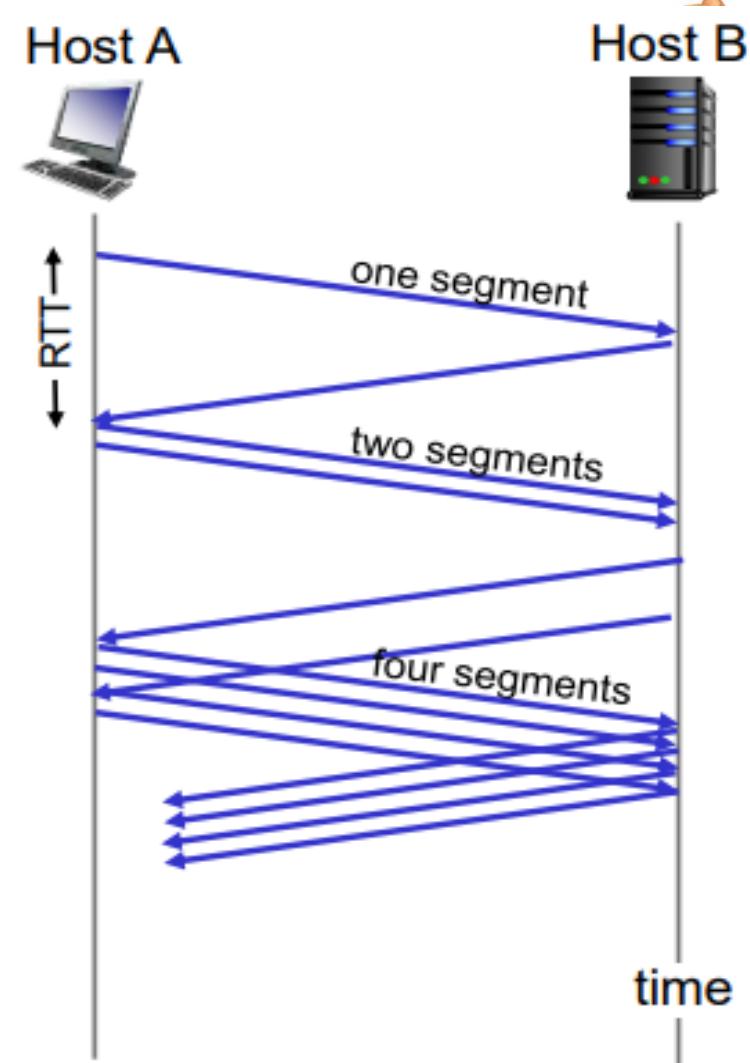
TCP Điều khiển tắt nghẽn





TCP Slow Start

- Khi kết nối bắt đầu, tăng tốc độ theo cấp số nhân cho đến khi sự kiện mất gói đầu tiên xảy ra.





TCP: phát hiện, phản ứng khi mất gói

➤ TCP RENO:

- Mất gói được chỉ ra bởi timeout:
 - ✓ Thiết lập lại cwnd = 1 MSS.
 - ✓ Sau đó kích thước cửa sổ sẽ tăng theo cấp số nhân (slow start) đến ssthresh (ngưỡng), sau đó sẽ tăng tuyến tính.

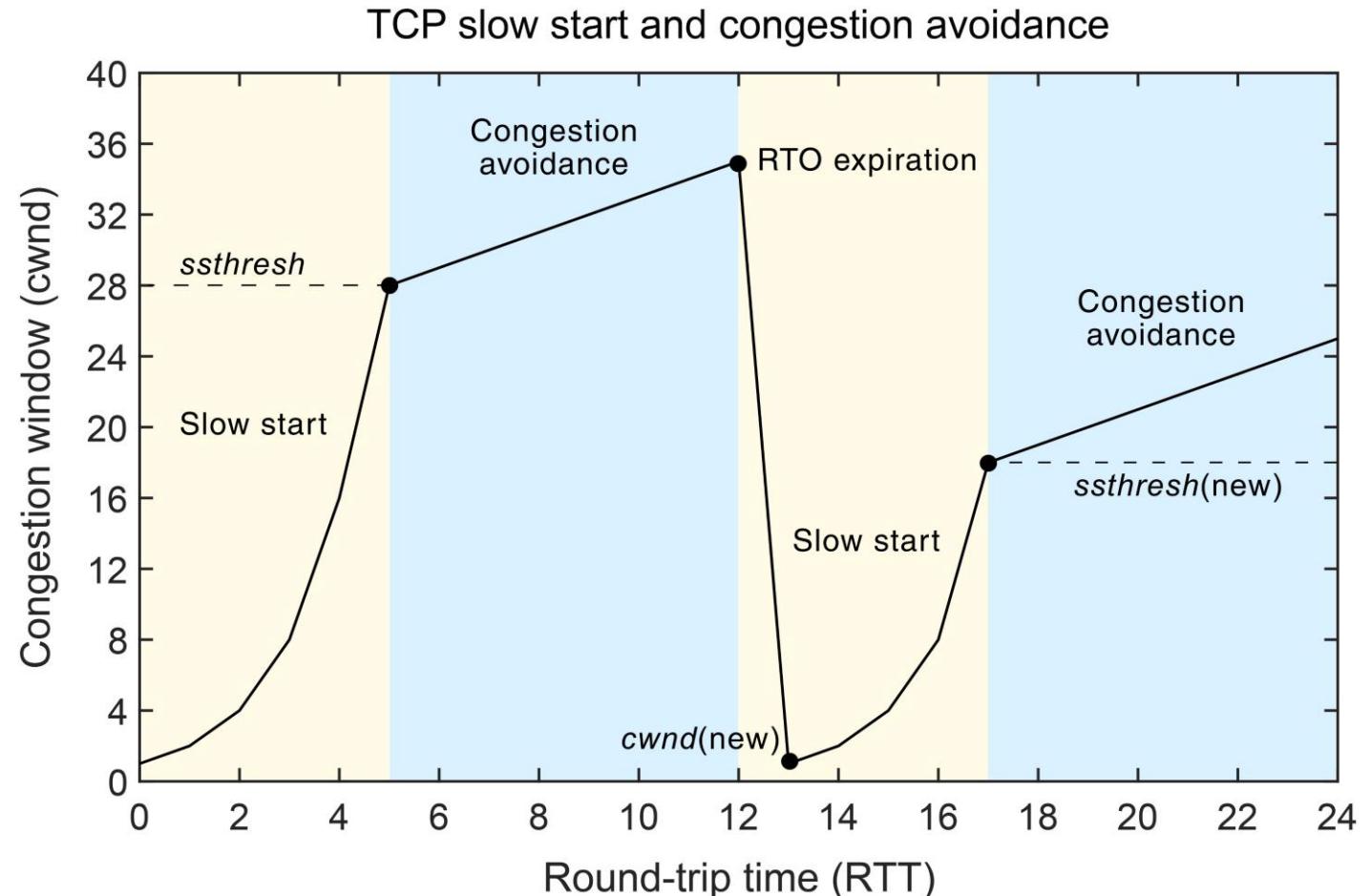
- Mất gói được xác định bởi 3 ACK trùng nhau:

- ✓ cwnd giảm một nửa sau đó tăng theo tuyến tính.

➤ TCP Tahoe: luôn luôn thiết lập cwnd = 1 (timeout hoặc 3 ACK trùng nhau).

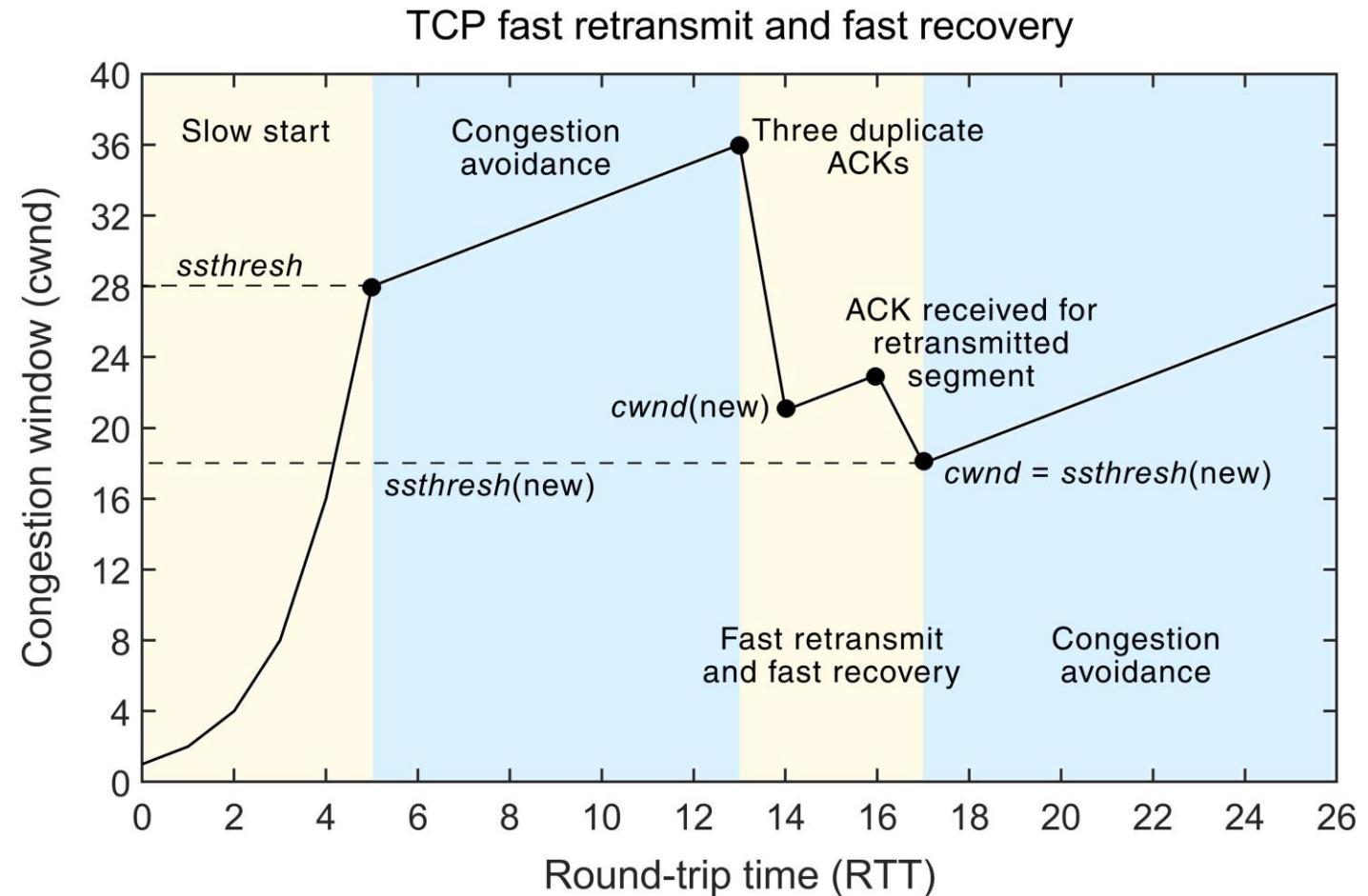


TCP: phát hiện, phản ứng khi mất gói





TCP: phát hiện, phản ứng khi mất gói



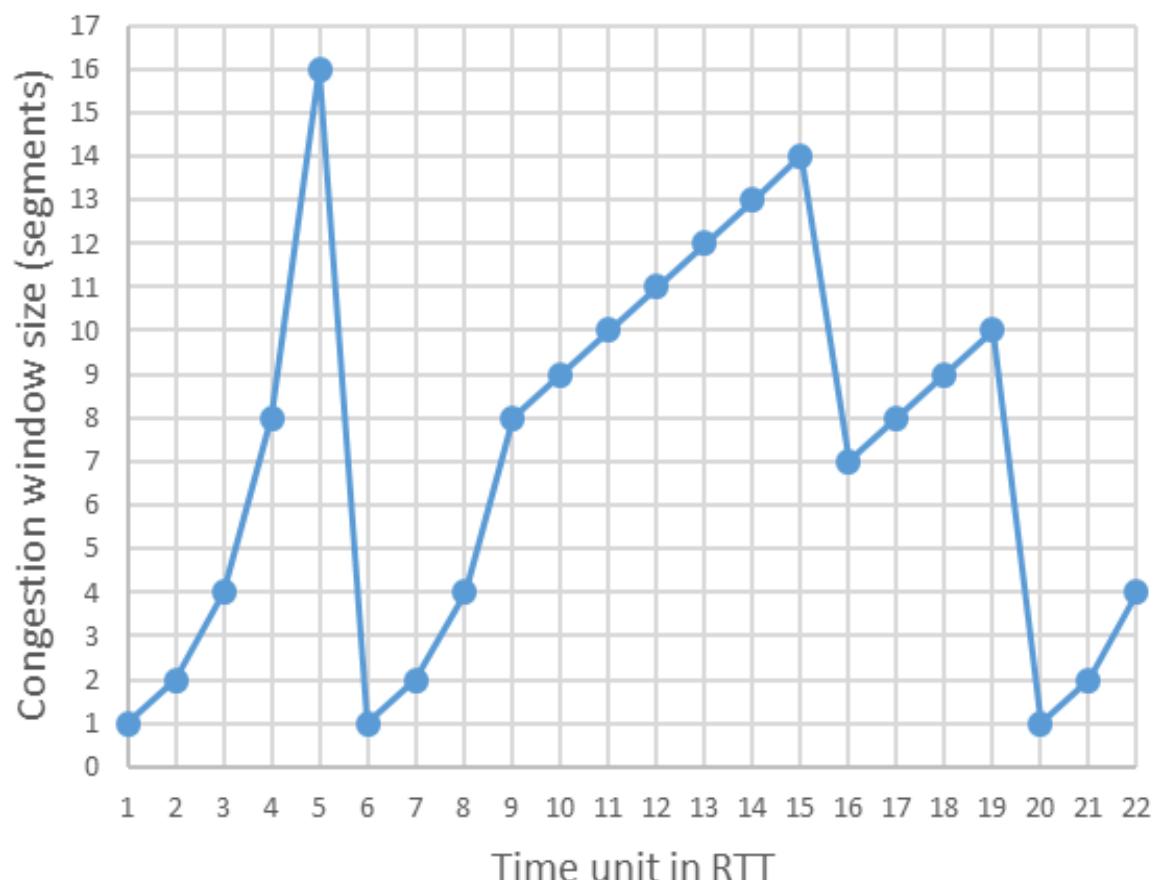


Trắc nghiệm

Cho biểu đồ TCP congestion như bên dưới. Trả lời các câu hỏi sau:

Câu 1: Giai đoạn slow start diễn ra ở:

- A. 2 - 4
- B. 5 - 6
- C. 9 - 15
- D. 16 - 19



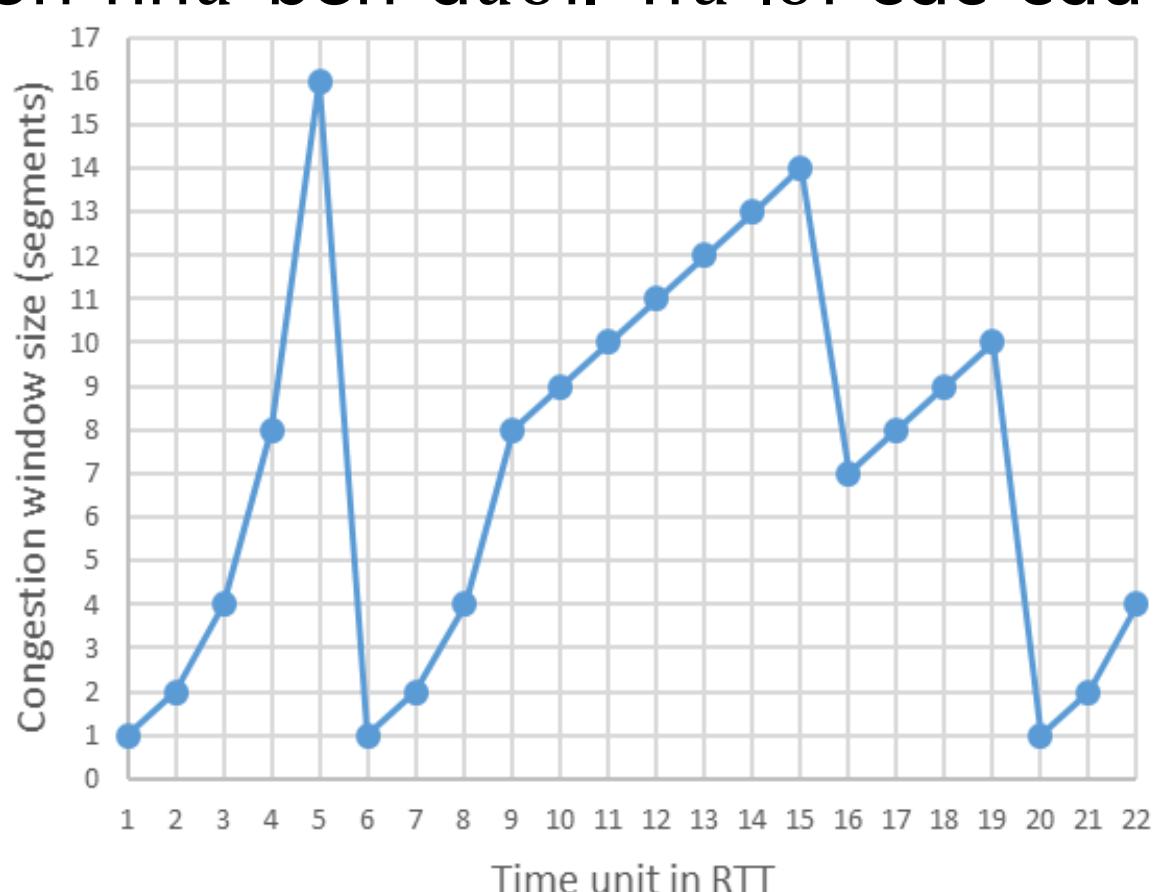


Trắc nghiệm

Cho biểu đồ TCP congestion như bên dưới. Trả lời các câu hỏi sau:

Câu 2: ssthresh tại RTT thứ 13 là

- A. 8
- B. 16
- C. 5
- D. 10



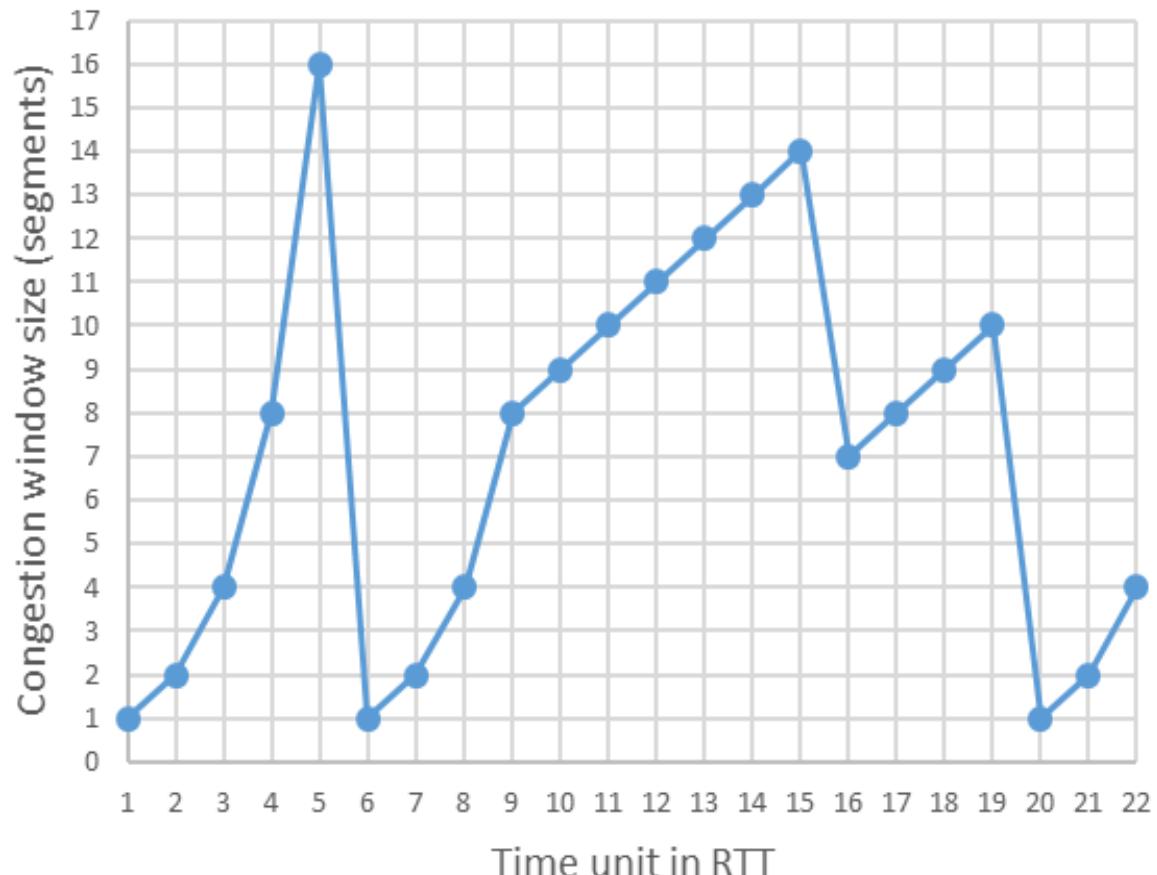


Trắc nghiệm

Cho biểu đồ TCP congestion như bên dưới. Trả lời các câu hỏi sau:

Câu 3: Segment thứ 95
được gửi tại thời điểm
RTT thứ mấy?

- A. 14
- B. 13
- C. 95
- D. 96





BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

Chào mừng đến với buổi training của BHT HTTT

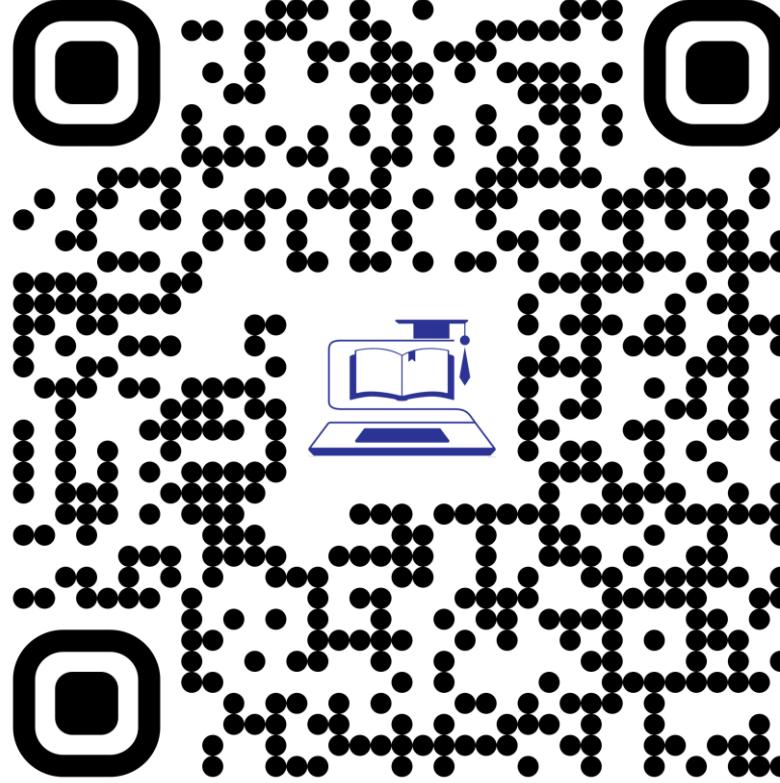
XIN CẢM ƠN CÁC BẠN ĐÃ LẮNG NGHE

CHÚC CÁC BẠN THI TỐT <3
MÃI IU <3



BAN HỌC TẬP KHOA HỆ THỐNG THÔNG TIN

<https://bit.ly/3sLPosw>





Một số tài liệu tham khảo thêm:

- ❑ Mã trạng thái: <https://topdev.vn/blog/http-status-code-la-gi/>
- ❑ port và protocol: <https://viblo.asia/p/ly-giai-chi-trong-3-phut-port-number-la-gi-tong-hop-ve-port-number-tieu-bieu-L4x5xGrYlBM>