



Training giữa kì 1

Nhập môn mạng máy tính

Trainer

Nguyễn Nhật Tân
Châu Thế Vĩ



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

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





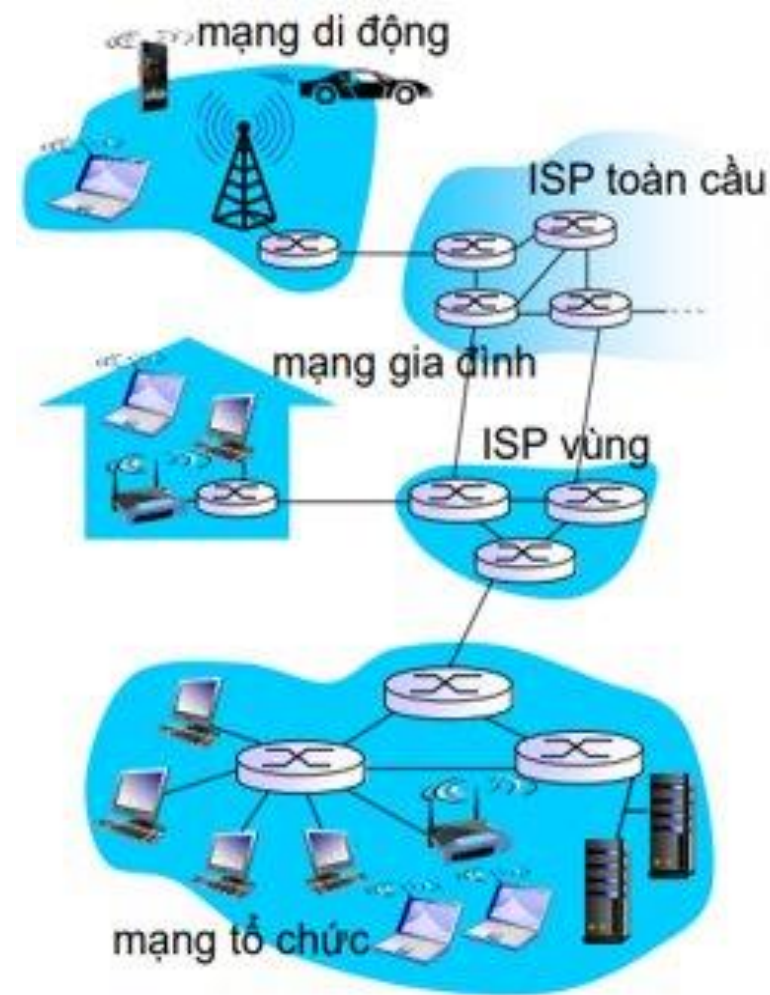
CHƯƠNG 1: TỔNG QUAN MẠNG MÁY TÍNH

1. INTERNET LÀ GÌ

Là “mạng của các mạng”

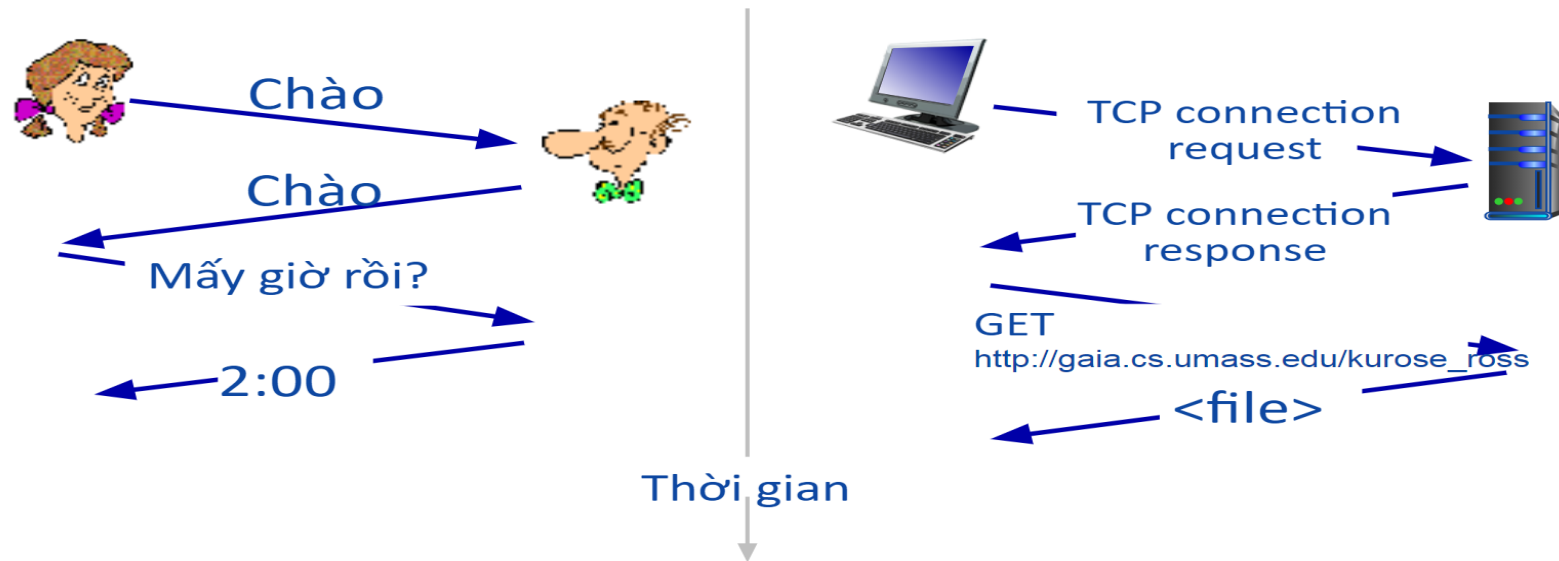
Ứng dụng:

- 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 cấu trúc, 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.





2. MẠNG BIÊN

Cấu trúc mạng:

- Mạng biên:
 - + Hosts (các thiết bị đầu cuối): client (Máy khách) và server (máy chủ).
 - + Máy chủ thường được đặt ở trung tâm dữ liệu.
- Mạng truy cập -> liên kết truyền thông có dây và không dây :
 - + Kỹ thuật DSL (Đường dây thuê bao kỹ thuật số)
 - + Dừng mạng cáp
 - + Kỹ thuật FTTH (mạng gia đình hoặc mạng doanh nghiệp)
 - + Ethernet
 - + Wifi





ĐƯỜNG TRUYỀN VẬT LÝ

- Cáp đồng
- Cáp quang
- Sóng radio

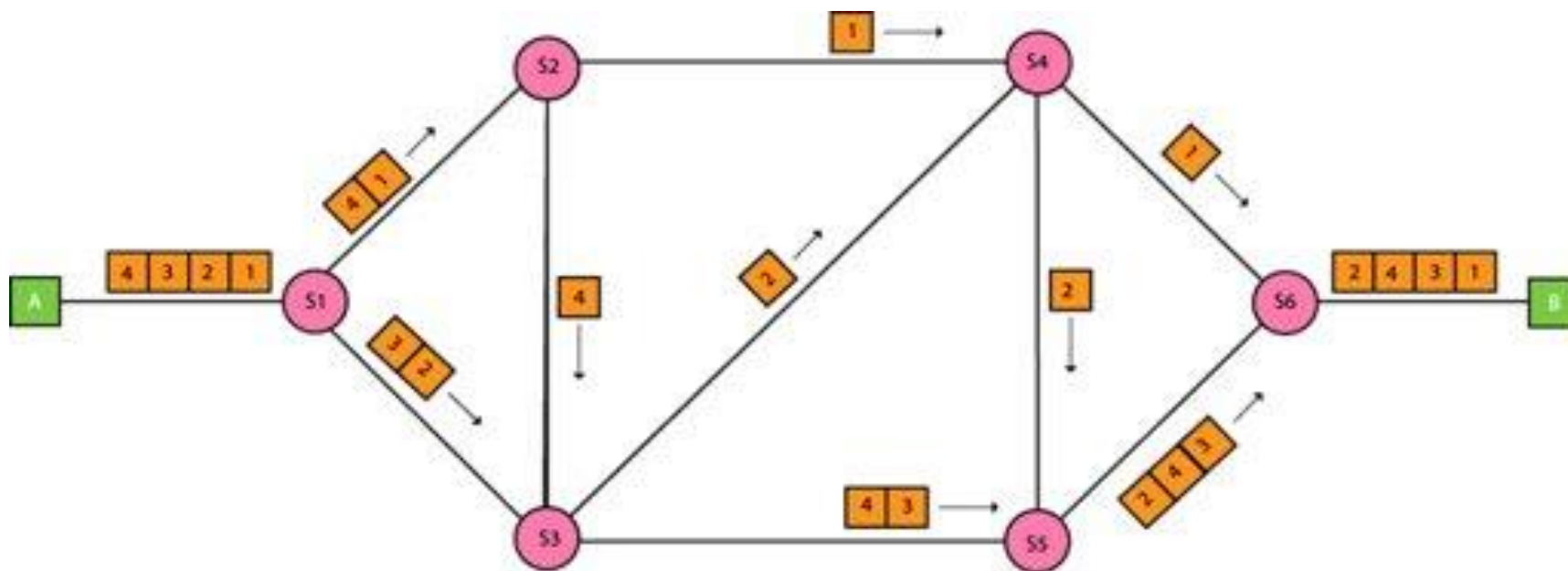




3. MẠNG LỖI:

CHUYỂN MẠCH GÓI (Packet switching)

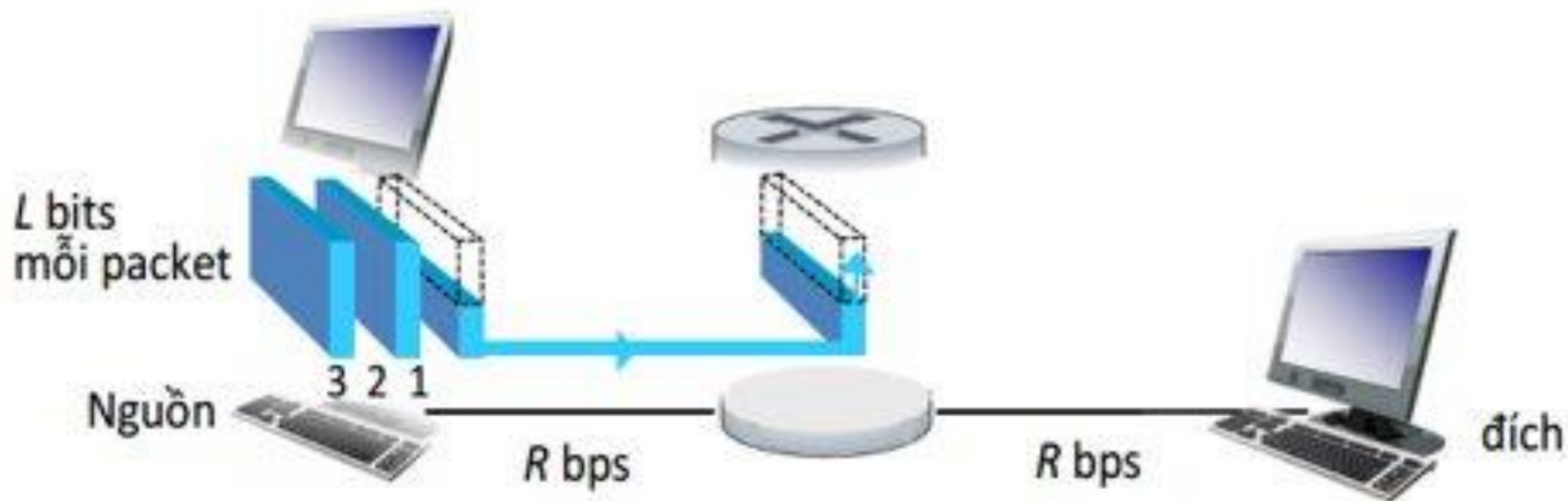
Các hệ thống đầu cuối (host) chia nhỏ dữ liệu tầng ứng dụng thành các packet.





CHUYỂN MẠCH GÓI (PACKET SWITCHING)

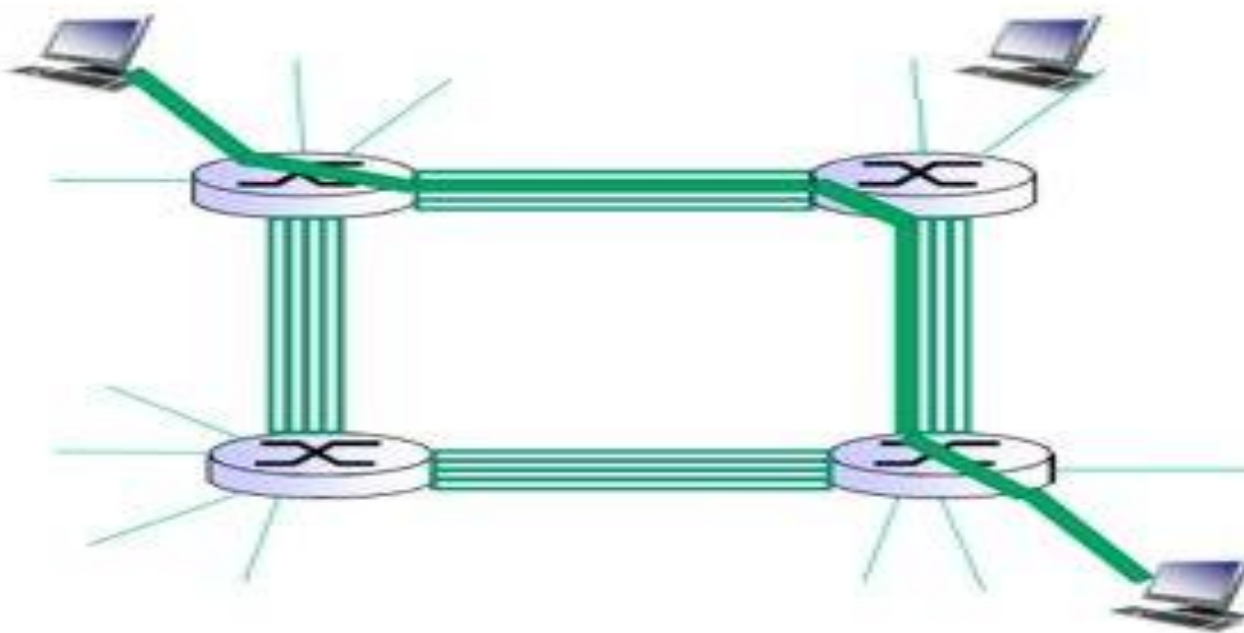
Lưu và chuyển tiếp (store-and-forward): toàn bộ các gói phải đến bộ định tuyến trước khi nó có thể được truyền tải trên đường liên kết tiếp theo.





CHUYỂN MẠCH KÊNH (Circuit switching) :

Các tài nguyên phân bổ và dành riêng cho “cuộc gọi” giữa nguồn và đích.



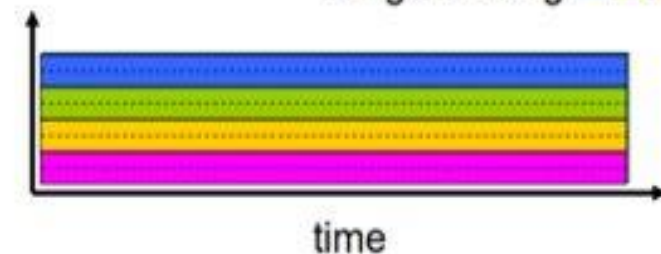
CHUYỂN MẠCH KÊNH (Circuit switching):

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.

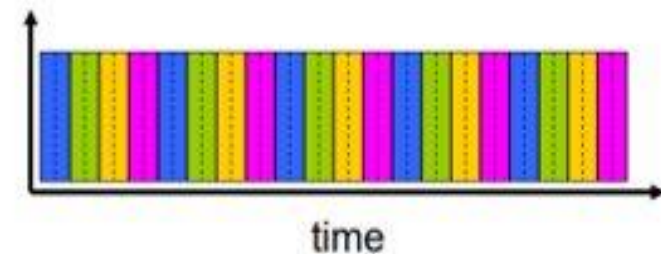
FDM

Tần số



TDM

tần số





SO SÁNH

Chuyển mạch gói	Chuyển mạch kênh
<ul style="list-style-type: none">- Không cần thiết lập kênh- Không chiếm dụng đường truyền nên hiệu suất truyền tin cao.- Băng thông không đảm bảo do luôn được sử dụng và dùng chung với tất cả- Độ tin cậy không cao, dễ gây tắc nghẽn, lỗi mất gói tin- Độ trễ đường truyền lớn- Có tính bảo mật không cao- Phải có cơ chế khắc phục lỗi	<ul style="list-style-type: none">- Phải thiết lập kết nối kênh truyền- Thực hiện trao đổi thông tin theo thời gian thực- Nội dung của thông tin không chứa địa chỉ bên gửi và nhận- Chất lượng đường truyền tốt, ổn định, có độ trễ thấp- Sử dụng tài nguyên của mạng không hiệu quả, lãng phí đường truyền- Độ bảo mật cao- Khả năng nâng cấp kém



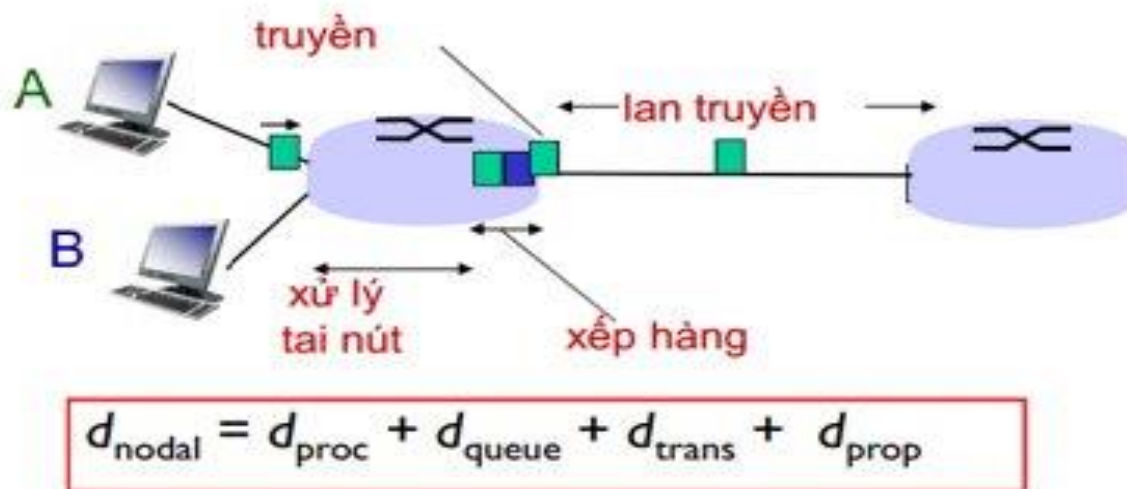


4. TRỄ GÓI, MẤT GÓI, THÔNG LƯỢNG MẠNG:

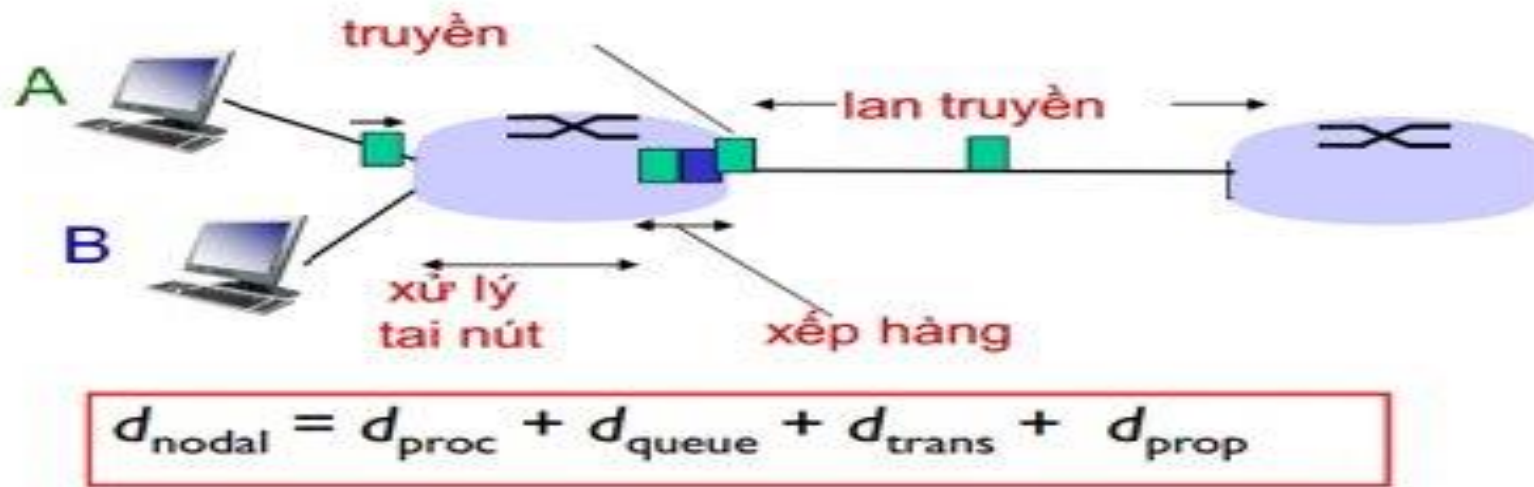
ĐỘ TRỄ:

Có 4 nguồn gây ra độ trễ:

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



ĐỘ TRỄ



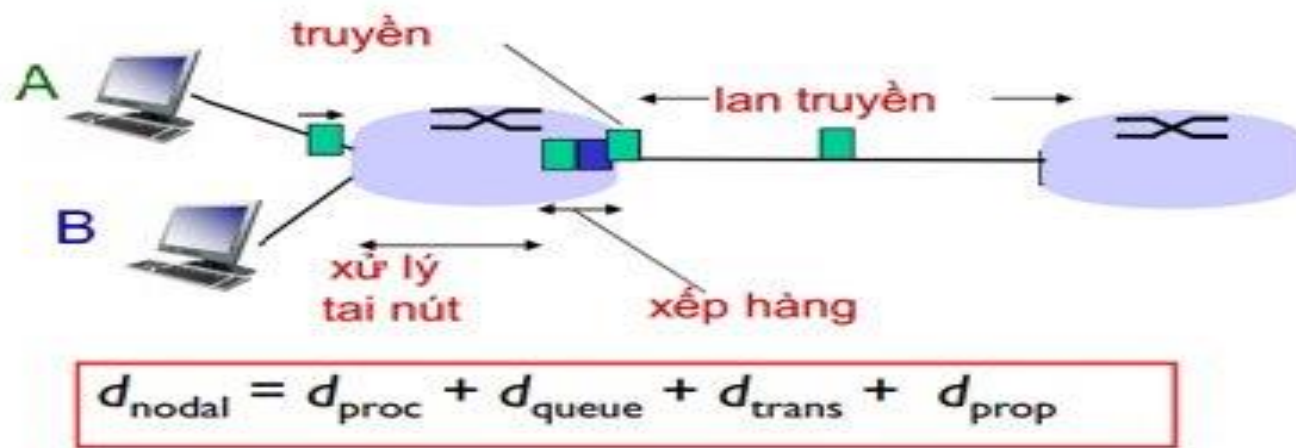
d_{proc} : xử lý tại nút

- 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



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

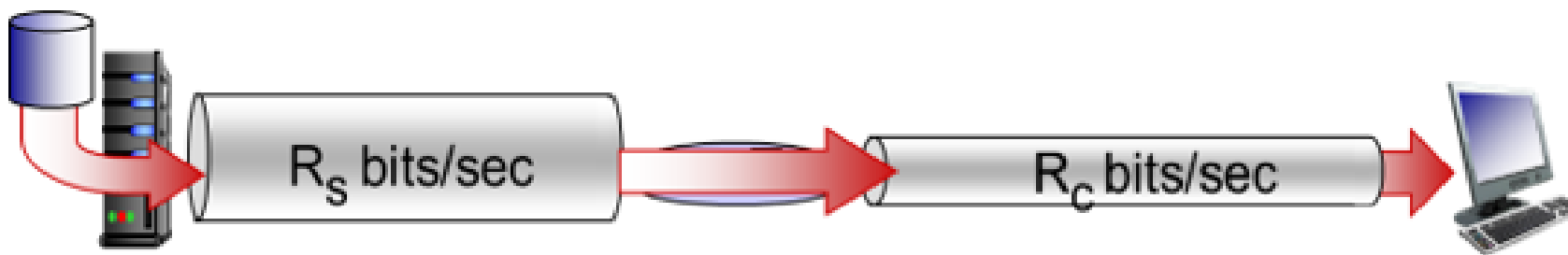
- L: chiều dài gói (bits)
- R: băng thông đường liên kết (bps)

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

- 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 một 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.





5. PHÂN TẦNG GIAO THỨC VÀ DỊCH VỤ:

Mô hình TCP/IP





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): Chuyển các bit trên đường truyền.





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 các ý nghĩa của dữ liệu.

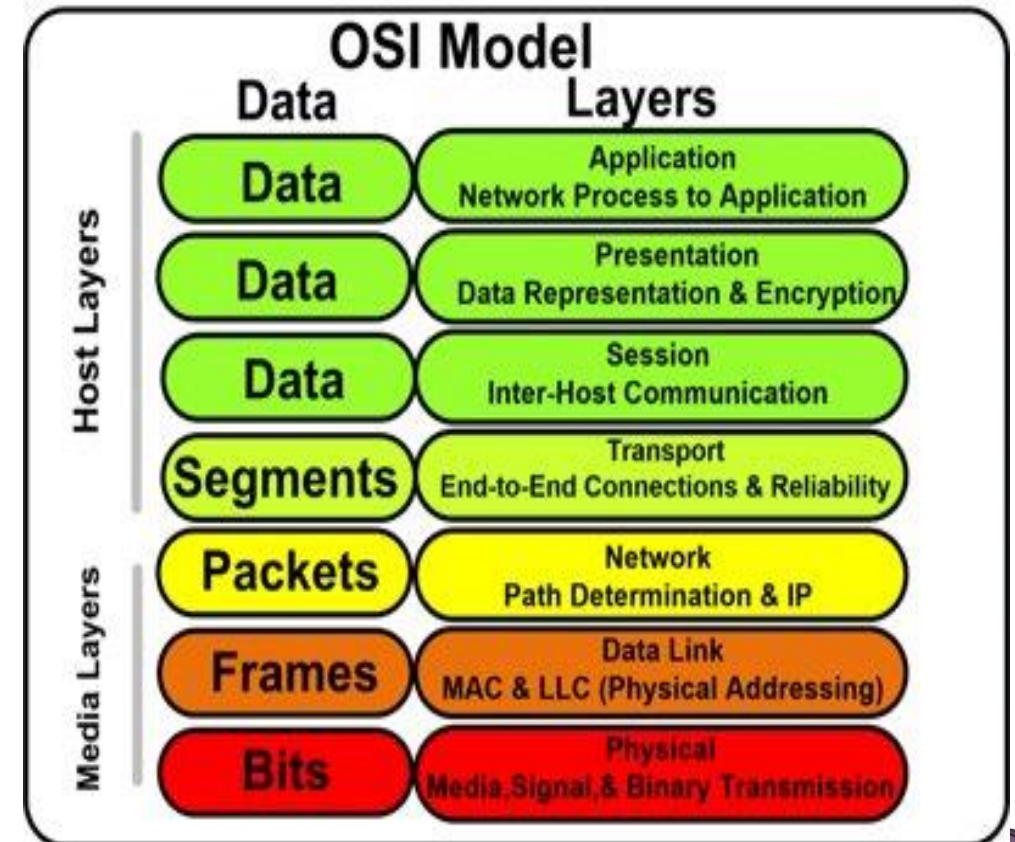
VD: mã hóa, nén, những quy ước chuyên biệt.

- Phiên (session): sự đồng bộ hóa, khả năng chịu lỗi, phục hồi sự trao đổi dữ liệu. Nói cách khác là thiết lập “Các giao dịch” giữa các thực thể đầu cuối.



PHÂN TẦNG MẠNG

- Xử lý các hệ thống dễ dàng
- Gói tin khi người gửi gửi đi sẽ được đóng gói (**encapsulation**)
- Đến tay người nhận sẽ bắt đầu quá trình ngược lại là mở gói (**de-encapsulation**)





Trắc nghiệm

Câu 1: Mô hình TCP/IP gồm mấy tầng?

A.4

B.5

C.6

D.7





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





Lời giải:

Ta có: $d_{trans} = L/R$

Chú ý: 1 byte = 8 bits

$$1000 \text{ Kbps} = 10^6 \text{ bps}$$

$$\Rightarrow d_{trans} = \frac{1000 \cdot 8}{10^6} = \frac{1}{125} \text{ s}$$

\Rightarrow Mỗi 1/125 giây sẽ có 1 gói tin được đẩy lên đường truyền

\Rightarrow 1 giây có 125 gói tin





Câu 3: Để tải 1 tài liệu văn bản với tốc độ 100 trang mỗi giây, ta giả sử rằng một trang tài liệu trung bình có 24 dòng với 80 ký tự (mỗi ký tự sử dụng mã 8 bit) trên mỗi dòng. Băng thông tối thiểu của kênh truyền là bao nhiêu?

- A. 160000 bps
- B. 586000 bps
- C. 1586 bps
- D. 1536000 bps





Lời giải:

Cứ 1 trang tài liệu có 24 dòng với 1 dòng chứa 80 ký tự

=> Tốc độ tải: $100 \times 24 \times 80 = 192000$ ký tự/s

Với mỗi ký tự sử dụng mã 8 bit

=> Tốc độ tải: $192000 \times 8 = 1536000$ bit/s

=> Băng thông tối thiểu của kênh truyền là 1536000 bps.





CHƯƠNG 2: TẦNG ỨNG DỤNG (APPLICATION LAYER)



1. KIẾN TRÚC ỨNG DỤNG MẠNG

KIẾN TRÚC CLIENT – SERVER



Máy chủ (serve):

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

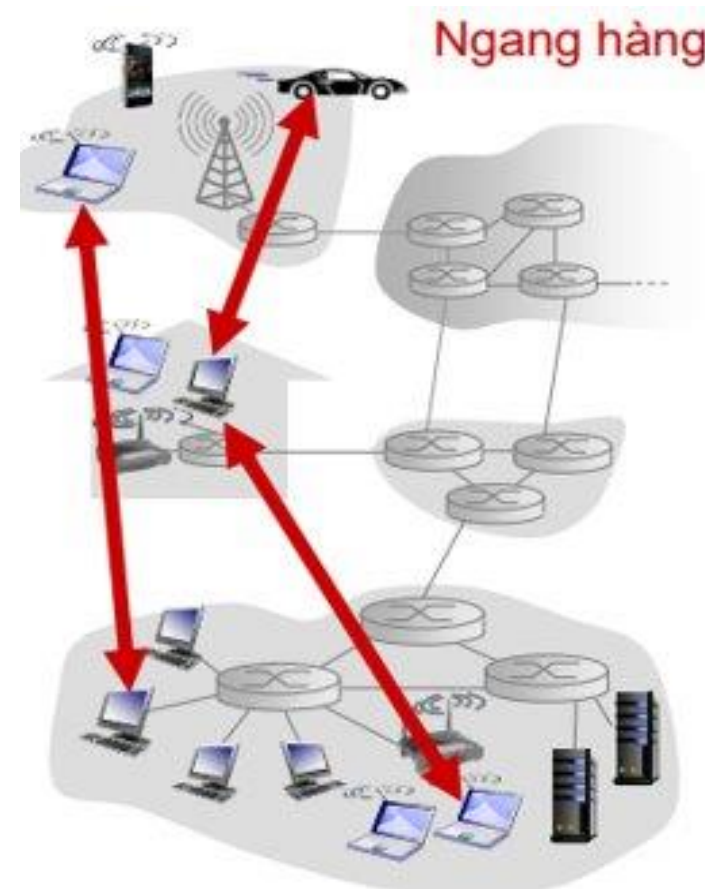
Máy khách (client):

- Giao tiếp trực tiếp với server
- Không giao tiếp trực tiếp với các máy khách khác
- Hoạt động không liên tục
- Có thể thay đổi địa chỉ IP.



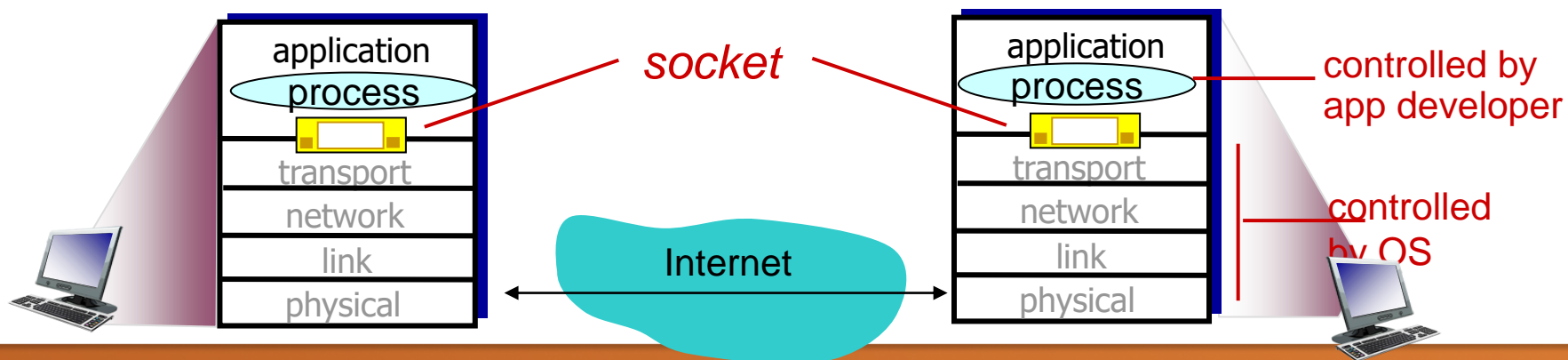
KIẾN TRÚC P2P (peer – to – peer)

- Không có máy chủ luôn hoạt động.
- Các host bất kỳ giao tiếp trực tiếp với nhau.
- Khả năng tự mở rộng.
- Các peer (bên):
 - Cung cấp dịch vụ cho nhau.
 - Kết nối không liên tục.
 - IP động.



Socket

- Tiến trình **gửi/nhận** thông điệp **đến/từ** socket của nó
- Socket hoạt động tương tự như cánh cửa
 - Tiến trình gửi đẩy thông điệp ra khỏi cửa
 - Tiến trình gửi dựa trên hạ tầng vận chuyển bên kia của cánh cửa để phân phối thông điệp đến socket tại tiến trình nhận
- Hai socket có liên quan: một socket ở mỗi bên
- 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 (Chậm mà chắc) , 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)





2. WEB VÀ HTTP (Hyper Text Transfer Protocol)

HTTP (Hyper Text Transfer Protocol) là gì ?

- Là giao thức thuộc tầng Application trong TCP/IP
- Máy khách khởi tạo kết nối TCP (tạo socket) đến cổng 80 của máy chủ
- HTTP “không lưu trạng 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 (dung 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
2RTT cho mỗi đối tượng	Chỉ cần một RTT cho tất cả các đối tượng được tham chiếu (Yêu cầu kết nối mở)



THÔNG ĐIỆP HTTP

Có hai loại thông điệp HTTP:

- Request: gồm dòng yêu cầu, các dòng header, thân.
- Response: gồm dòng trạng thái, header, các dòng 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 thân thông điệp).

HEAD: Yêu cầu máy chủ loại bỏ đối tượng được yêu cầu ra khỏi thông điệp phản hồi.

PUT (HTTP 1.1): Tải tập tin trong thân 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

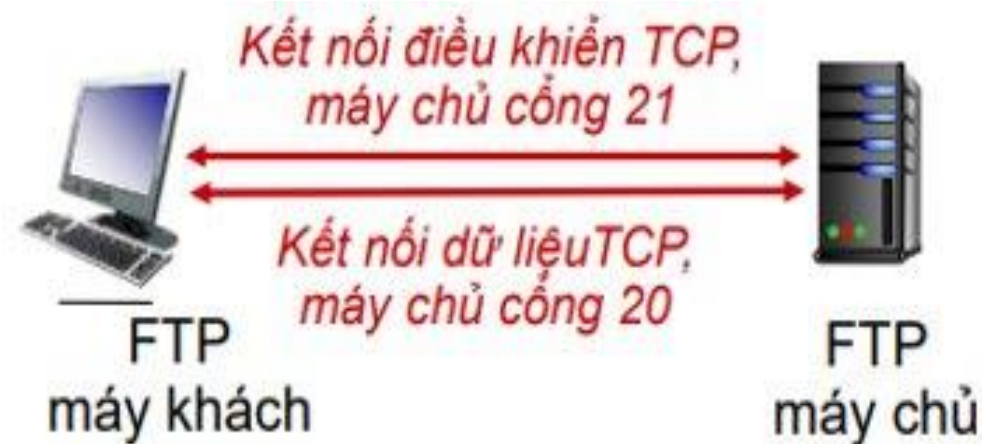
- 200 OK: Yêu cầu thành công, đối tượng được yêu cầu sau ở trong thông điệp này.
- 301 Moved Permanently: Đối tượng được yêu cầu đã được di chuyển, vị trí mới được xác định sau trong thông điệp này (trường Location:)
- 400 Bad Request: Máy chủ không hiểu thông điệp yêu cầu.
- 404 Not Found :Thông tin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





3. FTP (File Transfer Protocol)

- FTP là một giao thức chuyên dụng để **truyền tải file từ máy tính này sang máy tính khác.**
- FTP dùng 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 từ máy chủ
- STOR filename: lưu trữ dữ liệu trên máy chủ





4. Thư điện tử: SMTP

Các thành phần chính:

- User agents: Soạn thảo, sửa đổi, đọc thông điệp mail, các thông điệp đi và đến được lưu lại trên serve.

Vd: Outlook, iPhone mail client,...

- Mail servers:

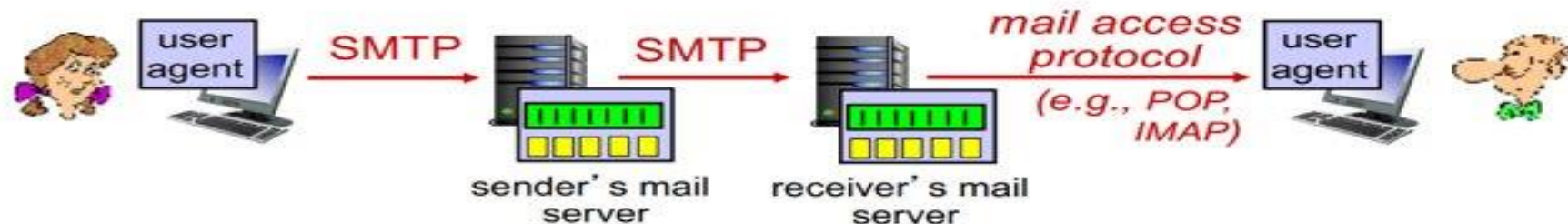
- Hộp thư (mailbox)
 - Hàng thông điệp (message queue)
- SMTP: Simple Mail Transfer Protocol.



SMTP: Giao thức gửi tin

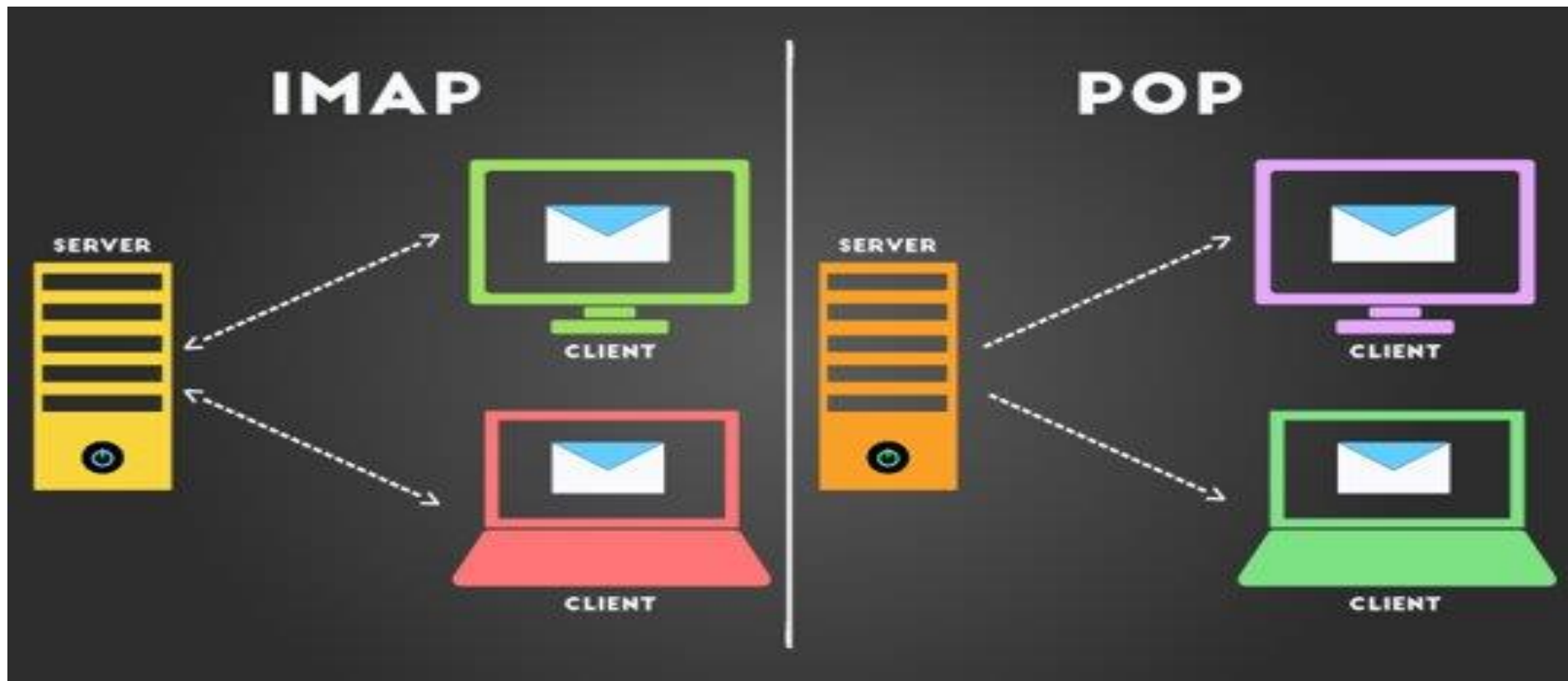
- Sử dụng TCP để gửi thông điệp mail từ client đến server (port: 25).
- Truyền trực tiếp server gửi đến server nhận.
- Gồm 3 giai đoạn truyền (bắt tay, truyền thông điệp, đóng).
- Tương tác lệnh/phản hồi (như HTTP, FTP):
 - Lệnh: Văn bản ASCII.
 - Phản hồi: Mã trạng thái và cụm từ.
- Thông điệp phải ở dạng mã ASCII 7 bit.

Một số lệnh SMTP thông dụng: HELO, MAIL FROM, RCPT TO, DATA, QUIT.





POP3 và IMAP: Giao thức đọc tin





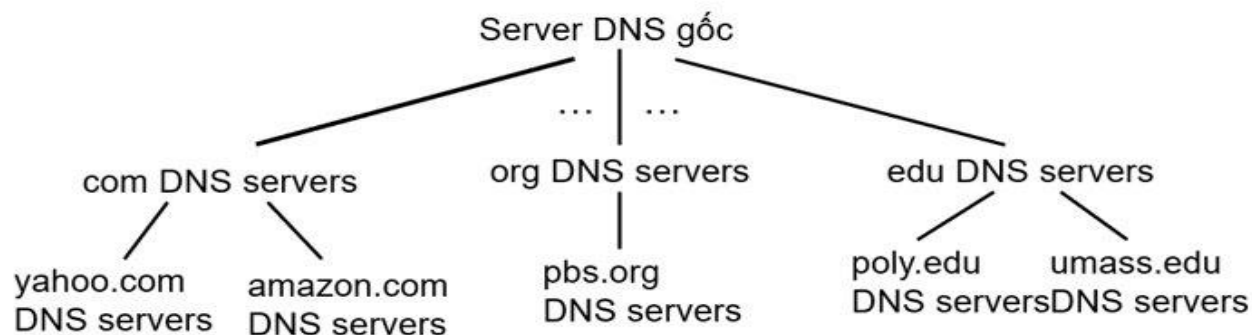
5.DNS (Domain Name System)

- 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: Lưu các tên gốc, bí danh tương ứng.
 - + Bí danh mail servers.
 - + Cân bằng tải: nhiều địa chỉ IP tương ứng cho 1 tên miền





Cở sở dữ liệu phân cấp, phân tán



Client muốn địa chỉ IP của www.amazon.com; lần đầu tiên sẽ:

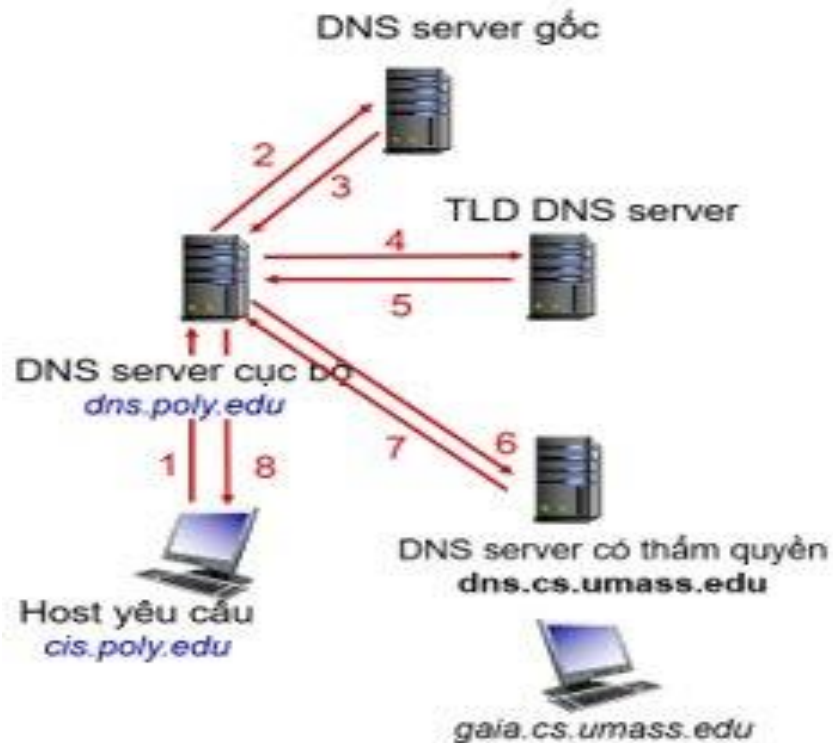
- Client truy vấn máy chủ gốc (root) để tìm máy chủ DNS .com
- Client truy vấn máy chủ DNS .com để tìm máy chủ DNS amazon.com
- Client truy vấn máy chủ DNS amazon.com để lấy địa chỉ IP của www.amazon.com



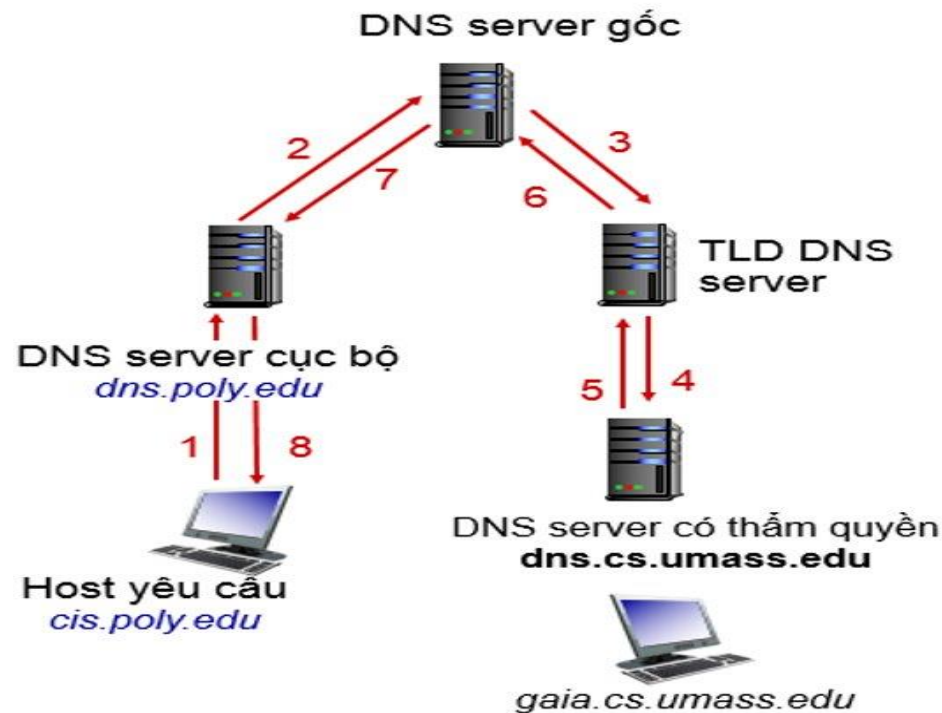


Phân giải tên miền

Truy vấn tuần tự



Truy vấn đệ quy



6. Lập trình socket với UDP và TCP

Datagram Socket (UDP)

Máy chủ (chạy với địa chỉ máy chủ IP)



Tạo socket, port= x:
`serverSocket =
socket(AF_INET,SOCK_DGRAM)`

↓
Đọc dữ liệu từ
`serverSocket`

↓
Ghi phản hồi đến
`serverSocket`
hỉ định địa chỉ IP và số cổng

Máy khách



Tạo socket:
`clientSocket =
socket(AF_INET,SOCK_DGRAM)`

↓
Tạo datagram với địa chỉ IP máy chủ
Và port = x; Gửi datagram qua
`clientSocket`

↓
Đọc dữ liệu từ
`clientSocket`

↓
Đóng
`clientSocket`

Stream Socket (TCP)



Máy chủ (chạy trên HostID)



Máy khách

Tạo socket,
port=**x**, cho các yêu cầu được gửi đến:

`serverSocket = socket()`

Chờ các yêu cầu kết nối
được gửi đến

`connectionSocket =
serverSocket.accept()`

Đọc yêu cầu từ
`connectionSocket`

Viết phản hồi đến
`connectionSocket`

Đóng
`connectionSocket`

**TCP
connection
setup**

Tạo socket,
Kết nối đến `hostid`, port=**x**
`clientSocket = socket()`

Gửi yêu cầu sử dụng
`clientSocket`

Đọc phản hồi từ
`clientSocket`

Đóng
`clientSocket`



PORT CỦA CÁC GIAO THỨC PHỔ BIẾN

TCP 20 : FTP (Data)

TCP 21 : FTP (Control)

TCP 22 : SSH

TCP 23 : Telnet

TCP 25 : SMTP

UDP 53 : DNS

UDP 67 : DHCP (Server)

UDP 68 : DHCP (Client)

TCP 80 : HTTP

TCP 110 : POP3

UDP 123 : NTP

TCP 443 : HTTPS





Câu 1. TCP không hỗ trợ chức năng nào sau đây?

- A. Kiểm soát lượng gói tin từ bên gửi sang bên nhận, tránh việc làm tràn bộ đệm phía nhận (flow control)
- B. Thiết lập kết nối giữa client – server (connection – oriented)
- C. Đảm bảo thông lượng tối thiểu cho đường truyền (minimum throughput guarantees)
- D. Đảm bảo gửi gói tin 1 cách tin cậy trên đường truyền (reliable transport)





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 *  
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()
```





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).





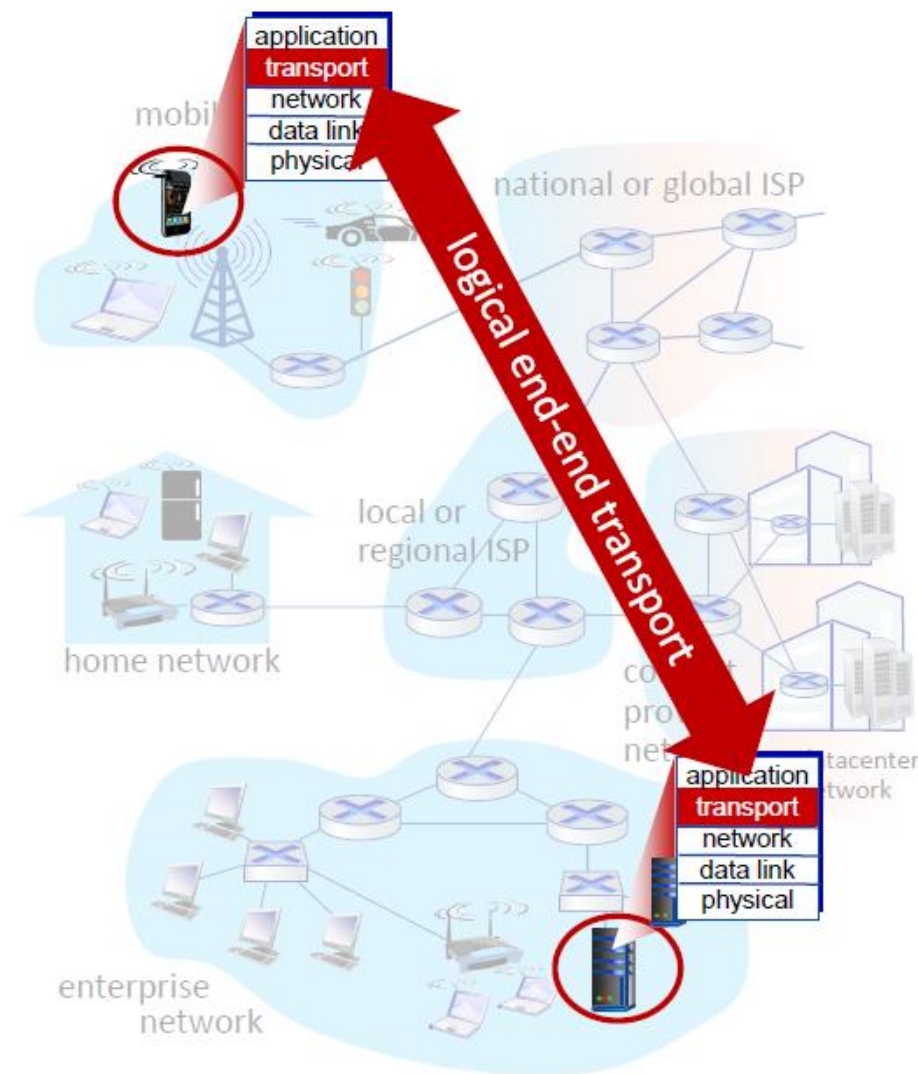
CHƯƠNG 3: TÀNG VẬN CHUYỂN

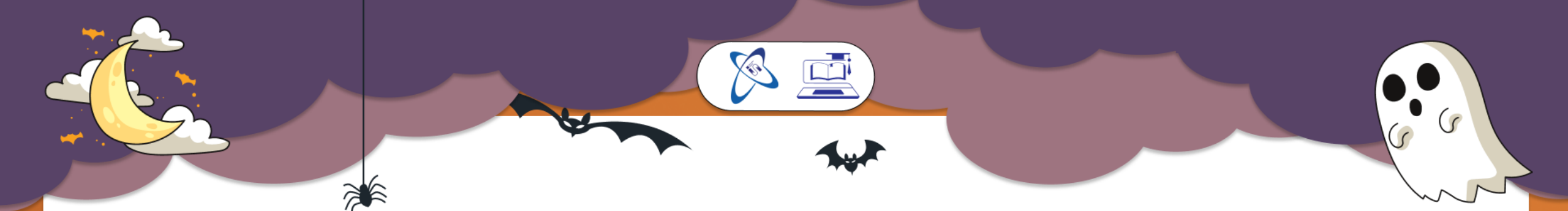
Nội dung

- Các dịch vụ tầng vận chuyển
- Multiplexing and demultiplexing
- UDP
- Nguyên lý truyền tin cậy
- TCP
- TCP - Điều khiển tắc nghẽn
- Sự phát triển của các tính năng của tầng vận chuyển

Giao thức và dịch vụ

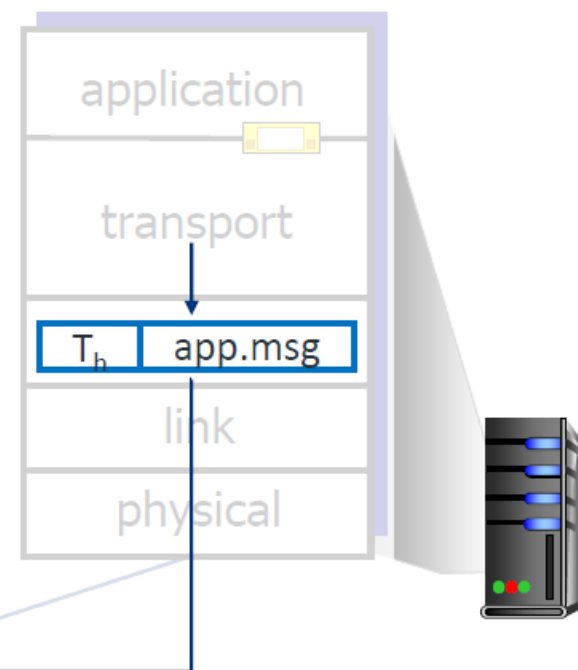
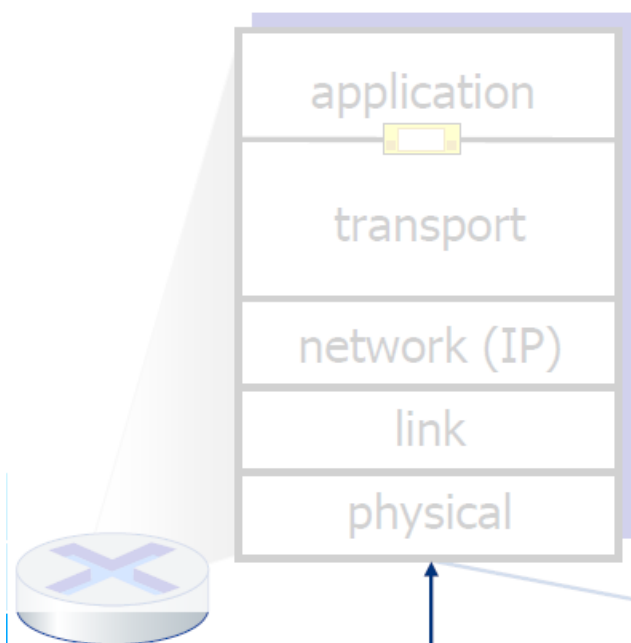
- Cung cấp “truyền thông luận lý (logical communication)” giữa các tiến trình trên các “host” khác nhau
- Các giao thức vận chuyển hoạt động trên các thiết bị đầu cuối:
- Bên gửi: chia các “messages – thông điệp” thành các “segments” và chuyển xuống tầng mạng.
- Bên nhận: ghép các segment thành messages, chuyển đến tầng ứng dụng.
- 2 giao thức chính: TCP và UDP





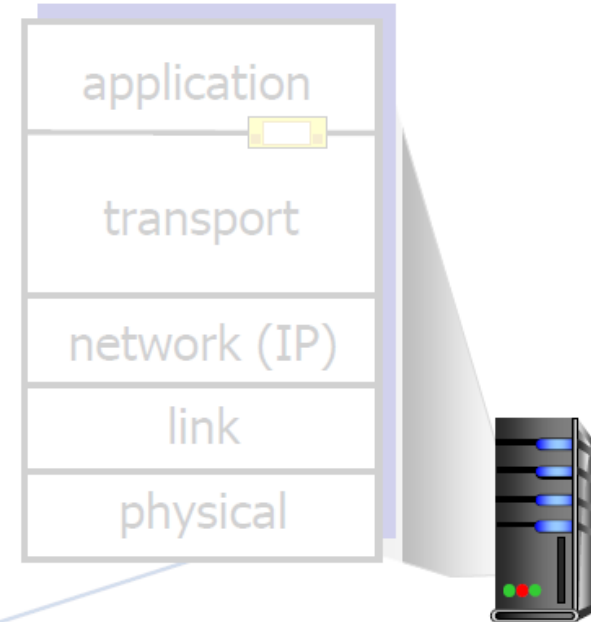
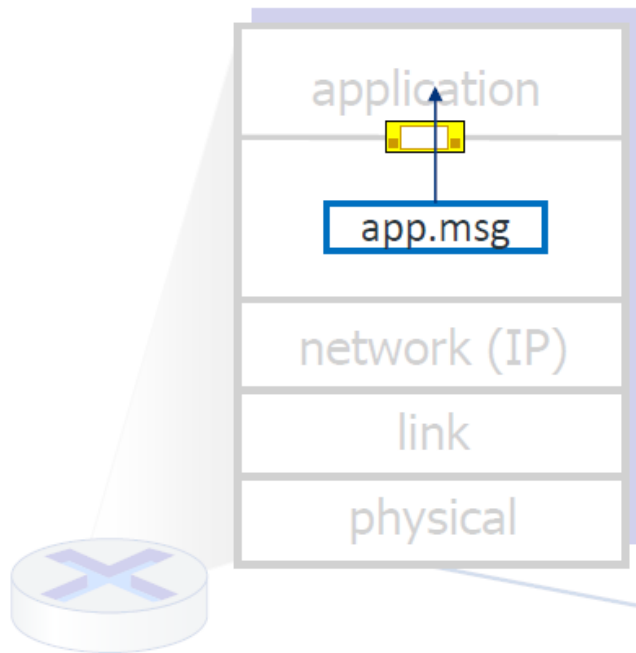
Bên gửi:

- Nhận message từ tầng ứng dụng (nhận thư)
- Xác định giá trị header (thông tin phong bì)
- Tạo segment (bỏ thư vào phong bì)
- Chuyển segment đến tầng mạng



Bên nhận:

- Nhận segment từ tầng mạng
- Kiểm tra giá trị header
- Bỏ header, trích xuất thông điệp của tầng ứng dụng
- Chuyển thông điệp đến tầng ứng dụng qua socket



Giao thức tầng vận chuyển

❑ TCP: Transmission Control Protocol

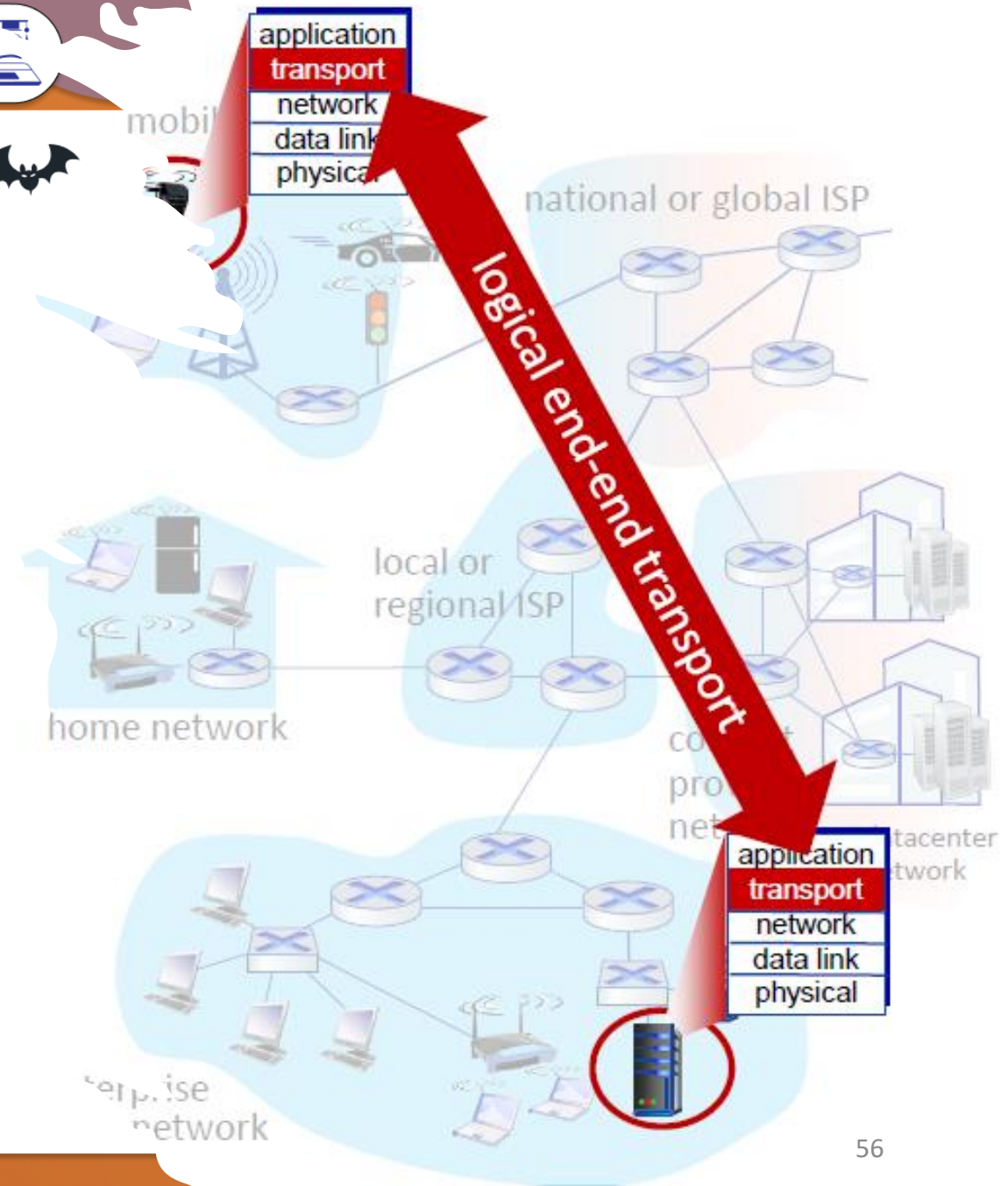
- Tin cậy, vận chuyển đúng thứ tự
- Điều khiển tắc nghẽn
- Điều khiển luồng
- Thiết lập kết nối

❑ UDP: User Datagram Protocol

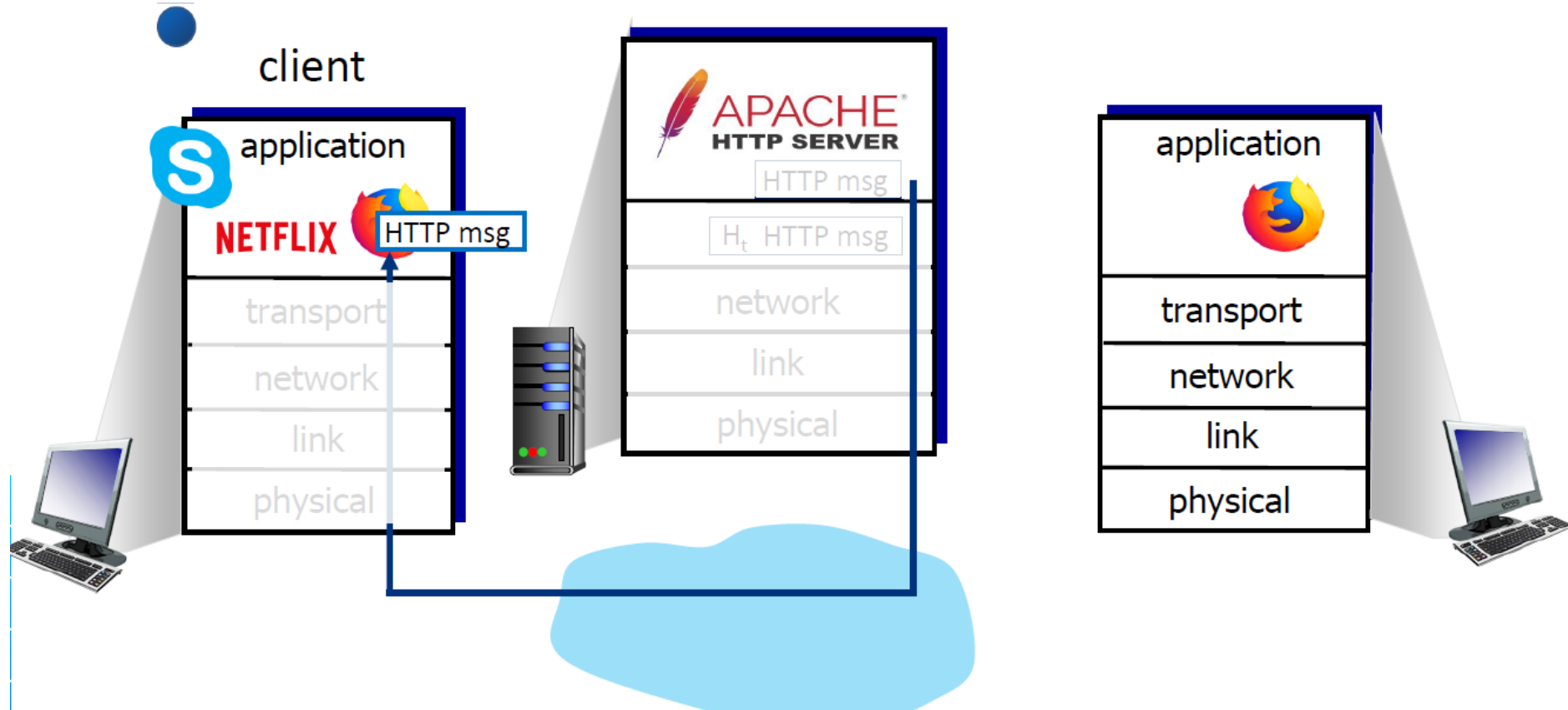
- Không tin cậy, truyền nhận không đúng thứ tự
- Phần mở rộng của giao thức IP “best-effort”

Không cung cấp các dịch vụ sau:

- Đảm bảo độ trễ
- Đảm bảo băng thông



Q: Làm thế nào mà tầng vận chuyển biết gửi đúng message tới trình duyệt Firefox thay vì Netflix hoặc Skype?



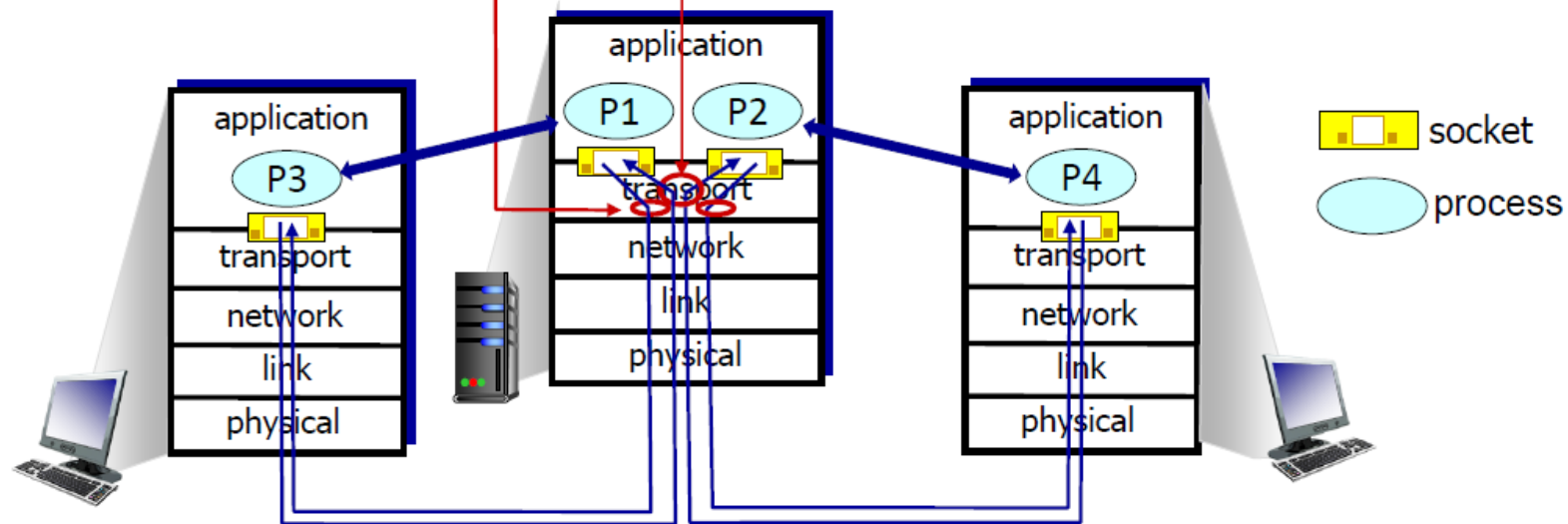
Multiplexing/demultiplexing

multiplexing tại bên gửi:

Nhận dữ liệu từ socket, thêm header của tầng vận chuyển

demultiplexing tại bên nhận:

Sử dụng thông tin trong header để chuyển segment nhận được đến đúng socket.



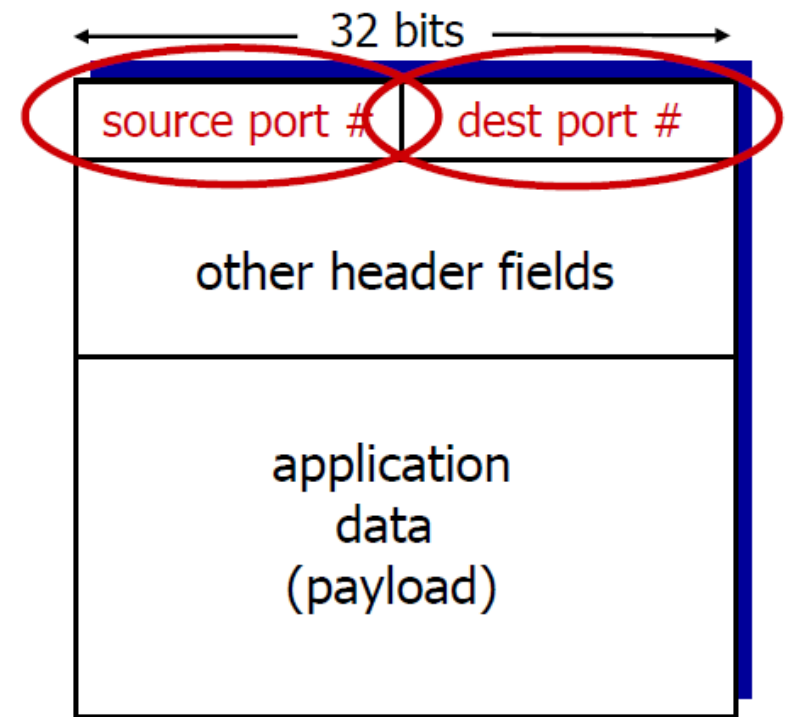
Demultiplexing làm việc thế nào?

Khi host nhận được một IP “datagram”

- Mỗi datagram có địa chỉ IP nguồn và IP đích
- Mỗi datagram này chứa 1 segment(đơn vị dữ liệu của tầng vận chuyển)

Mỗi segment có port nguồn và port đích

- “Host” dùng địa chỉ IP & số port để chuyển segment đến đúng socket tương ứng



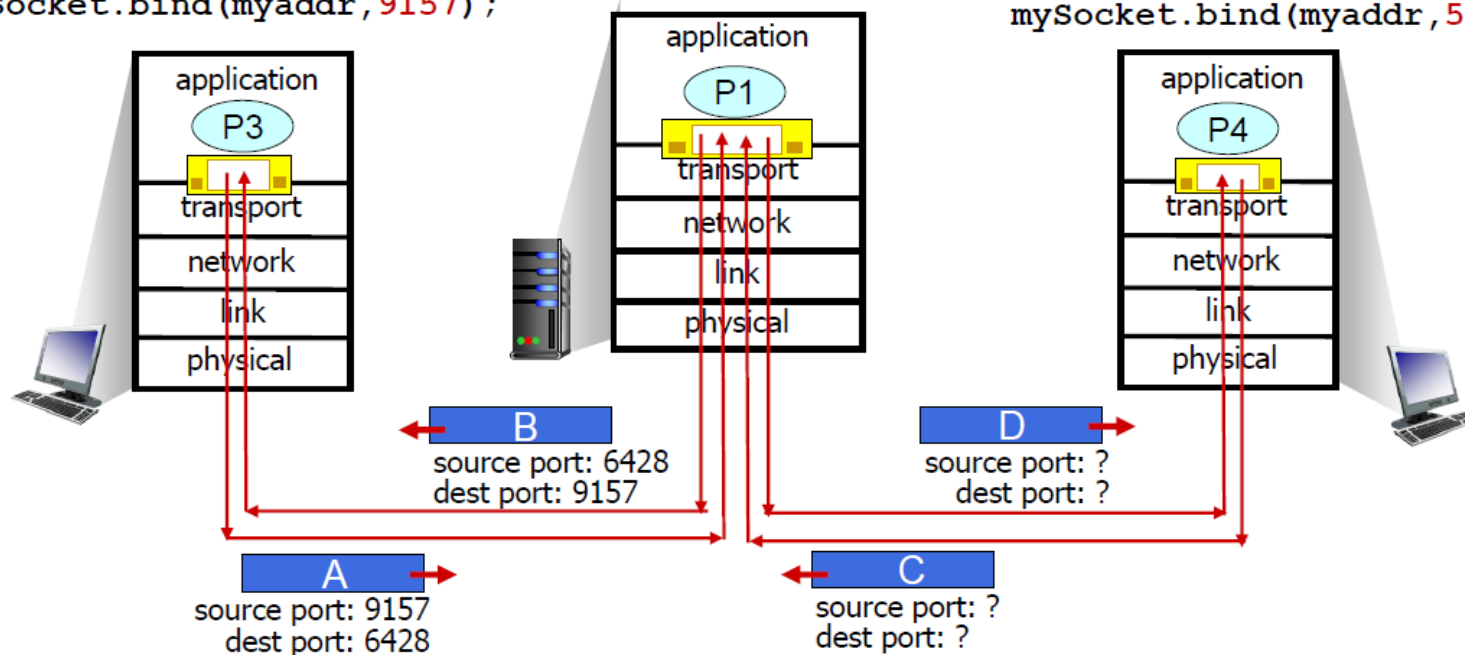
TCP/UDP segment format

Ví dụ về “connectionless - không kết nối”

```
mySocket =  
    socket(AF_INET, SOCK_DGRAM)  
mySocket.bind(myaddr, 6428);
```

```
mySocket =  
    socket(AF_INET, SOCK_STREAM)  
mySocket.bind(myaddr, 9157);
```

```
mySocket =  
    socket(AF_INET, SOCK_STREAM)  
mySocket.bind(myaddr, 5775);
```





Hướng kết nối

❑ TCP socket được xác định bởi **4-tuple**

(4 thông tin chính):

- Source IP address
- Source port number
- Dest IP address
- Dest port number

❑ Demux: bên nhận sử dụng cả 4 thông tin chính để chuyển segment đến đúng socket

❑ Server có thể hỗ trợ nhiều TCP socket cùng lúc:

- Mỗi socket được xác định bởi 4 thông tin
- Mỗi socket tương ứng với một client đang kết nối với nó



Tổng kết về Multiplexing, demultiplexing

- Multiplexing, demultiplexing: dựa trên giá trị trong header của segment, datagram
 - **UDP**: demultiplexing chỉ sử dụng destination port number
 - **TCP**: demultiplexing sử dụng 4 thông tin: source và destination IP addresses, và port numbers
- Multiplexing/demultiplexing hoạt động ở tất cả các tầng



UDP: User Datagram Protocol

- Một giao thức đơn giản
- Cung cấp dịch vụ “best effort”
 - Segment có thể mất
 - Segment có thể đến không đúng thứ tự
- Tại sao vẫn dùng UDP?
 - Không có bước thiết lập kết nối (bước này làm tăng độ trễ)
 - Đơn giản: không lưu trữ trạng thái kết nối
 - Header nhỏ gọn
 - Không điều khiển tắc nghẽn
 - Segment đi nhanh nhất có thể
- “connectionless”:
 - Không có bước thiết lập kết nối
 - Mỗi segment được xử lý độc lập



UDP: User Datagram Protocol

UDP được sử dụng bởi:

streamingmultimediaapps(cho phép mất mát, cần tốc độ)

- DNS
- SNMP
- HTTP/3

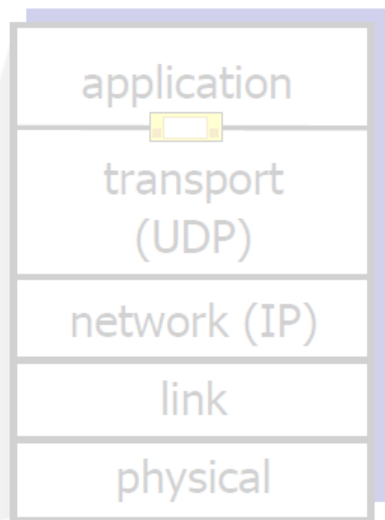
Nếu cần truyền tin cậy qua UDP(e.g., HTTP/3): Thêm xử lý tin cậy ở tầng ứng dụng

- Thêm điều khiển tắc nghẽn ở tầng ứng dụng

UDP: Hoạt động

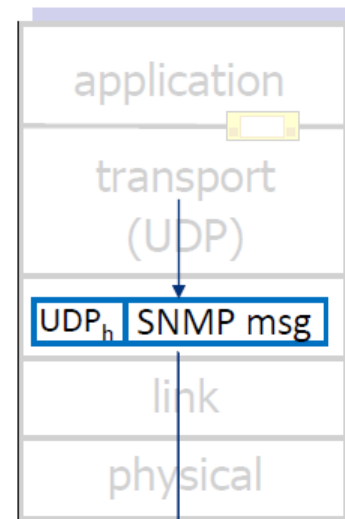
Bên gửi:

SNMP client

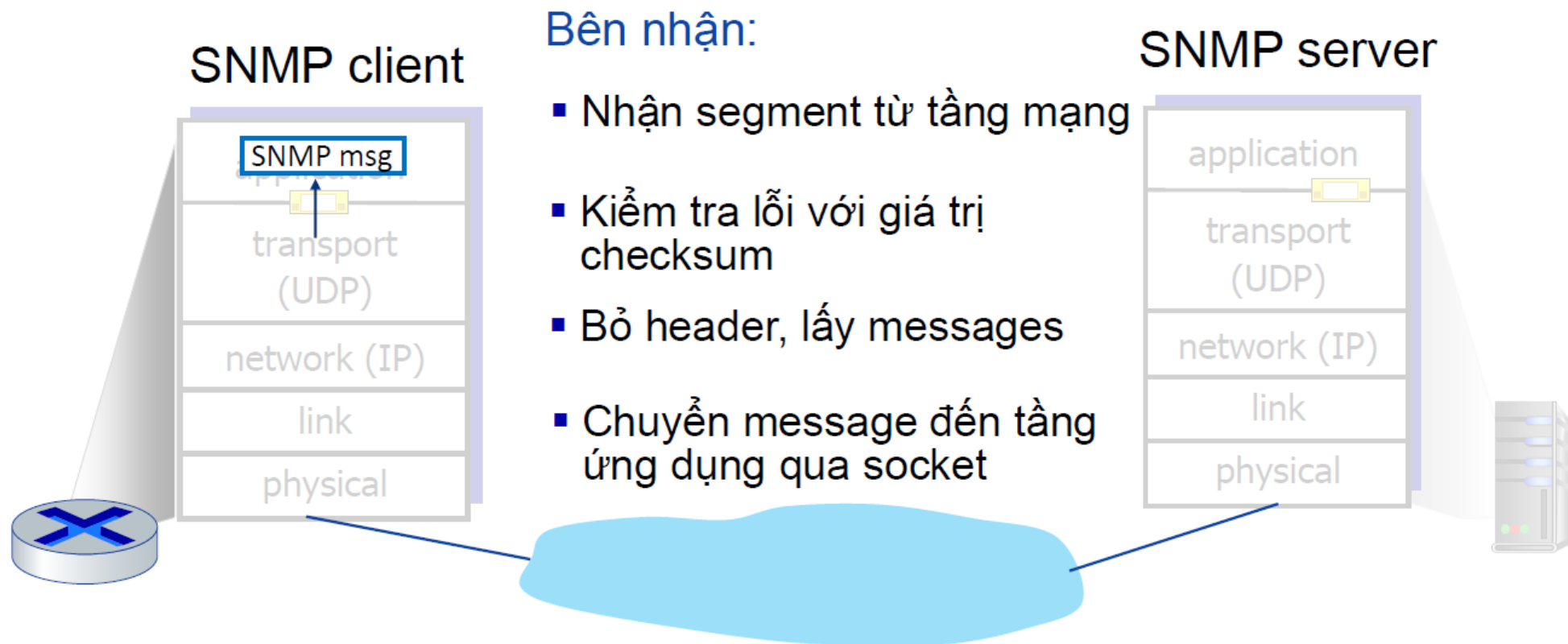


- Nhận message từ tầng ứng dụng
- Xác định giá trị header
- Tạo segment
- Chuyển segment đến tầng mạng

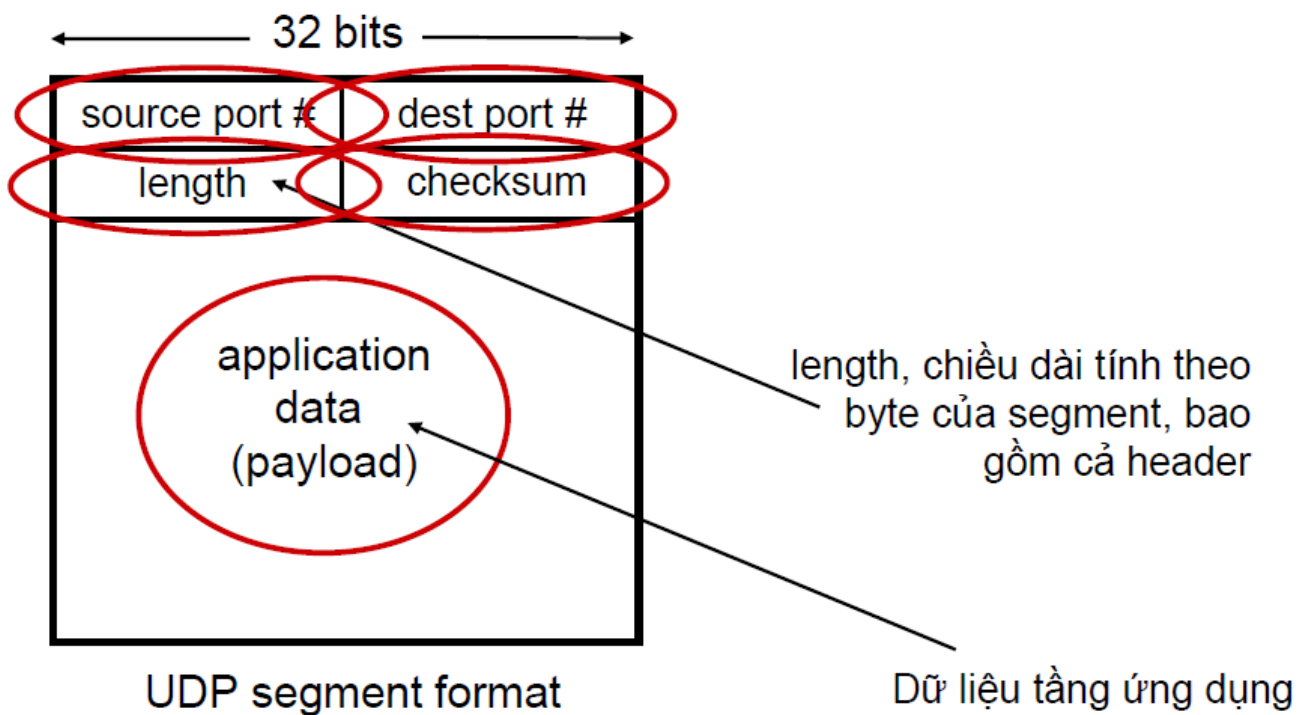
SNMP server



UDP: Hoạt động







UDP header



UDP checksum

Mục tiêu: phát hiện lỗi trong quá trình truyền (các bit bị lỗi, từ 0 thành 1 hoặc ngược lại)

	1 st number	2 nd number	sum
Gửi:	5	6	11
			
Nhận:	4	6	11
			
	receiver-computed checksum		sender-computed checksum (as received)
			



- **Mục tiêu:** phát hiện lỗi trong quá trình truyền (các bit bị lỗi, từ 0 thành 1 hoặc ngược lại)

- **sender:**

- Chia nội dung của segment(bao gồm các trường tiêu đề UDP và địa chỉ IP) dưới dạng các chuỗi 16 bit
- **checksum:** Tính tổng bù 1 (one's complement sum)
- Đặt kết quả vào phần UDP checksum trong header

Internet checksum

- **receiver:**

- Tính checksum của segment nhận được
- So sánh kết quả tính được và nội dung phần checksum
 - Không bằng nhau – có lỗi
 - Bằng nhau – không phát hiện lỗi.
Tuy nhiên...





Internet checksum: ví dụ

Ví dụ: cộng 2 chuỗi 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
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1
sum	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Note: nếu kết quả là 17 bits, lấy bit cuối cùng bên trái cộng vào kết quả.





Internet checksum: điểm yếu!

- Ví dụ: cộng 2 chuỗi 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
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1
sum	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Khi có 2 bit bị thay đổi, checksum không thay đổi





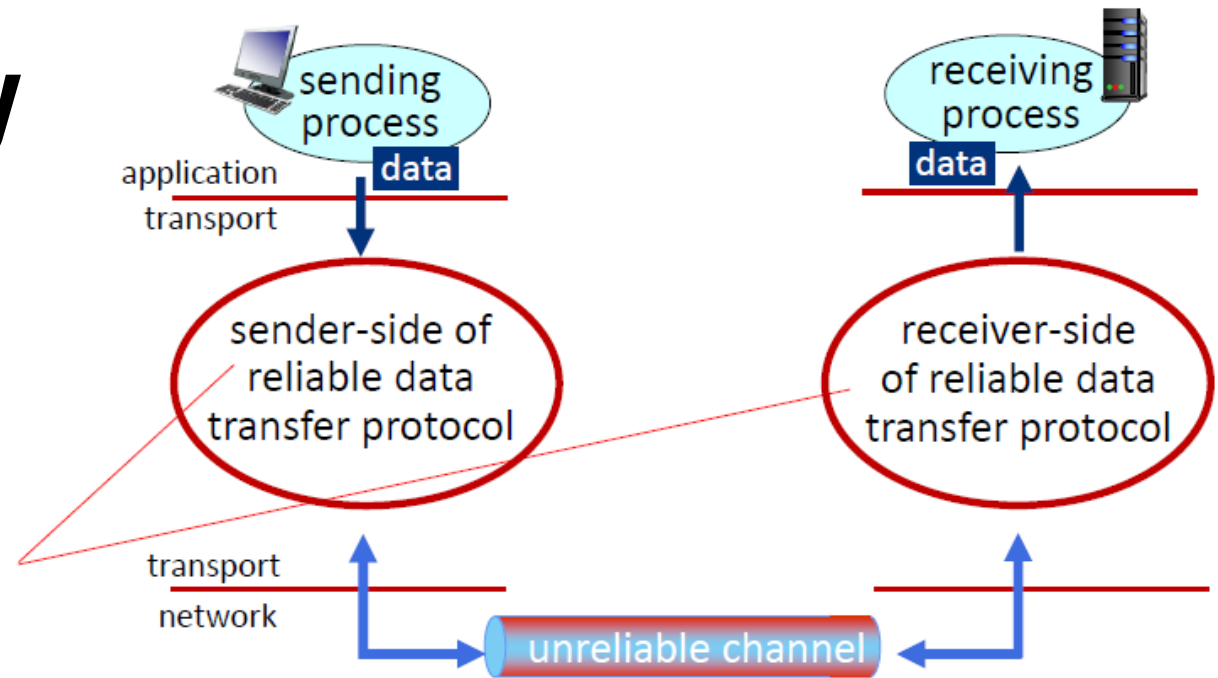
Tổng kết: UDP

- Giao thức đơn giản:
 - Segment có thể bị mất, sai thứ tự
 - “Best-effort”: gửi và hy vọng có kết quả tốt nhất
- UDP vẫn có điểm cộng:
 - Không có quá trình thiết lập kết nối
 - Có checksum để kiểm tra lỗi
- Các tính năng bổ sung thì đưa lên tầng ứng dụng (e.g., HTTP/3)



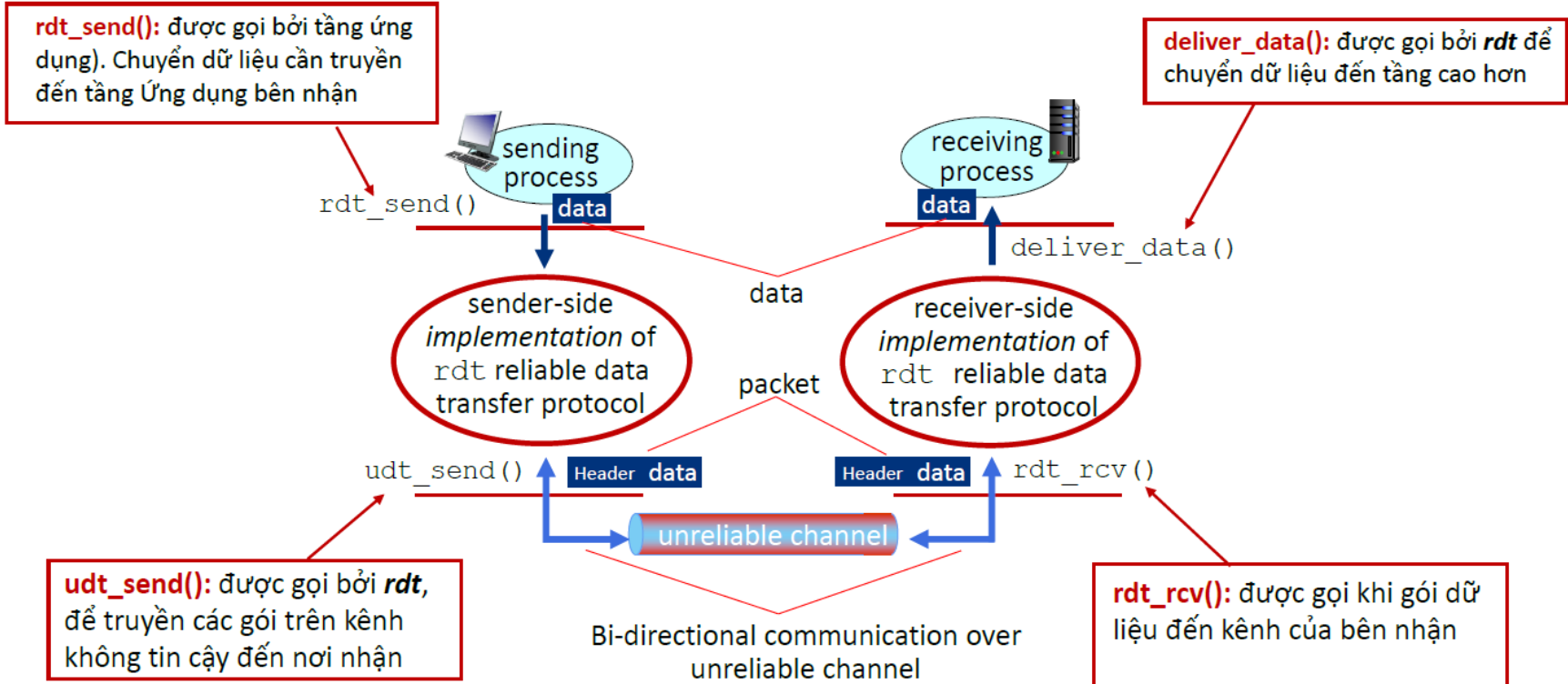
Nguyên lý truyền tin cậy

Độ phức tạp của giao thức truyền dữ liệu đáng tin cậy sẽ phụ thuộc (mạnh mẽ) vào các đặc điểm của kênh không đáng tin cậy (mất, hỏng, sắp xếp lại dữ liệu?)



Triển khai cụ thể của dịch vụ truyền tin cậy

Reliable data transfer protocol (rdt): interfaces

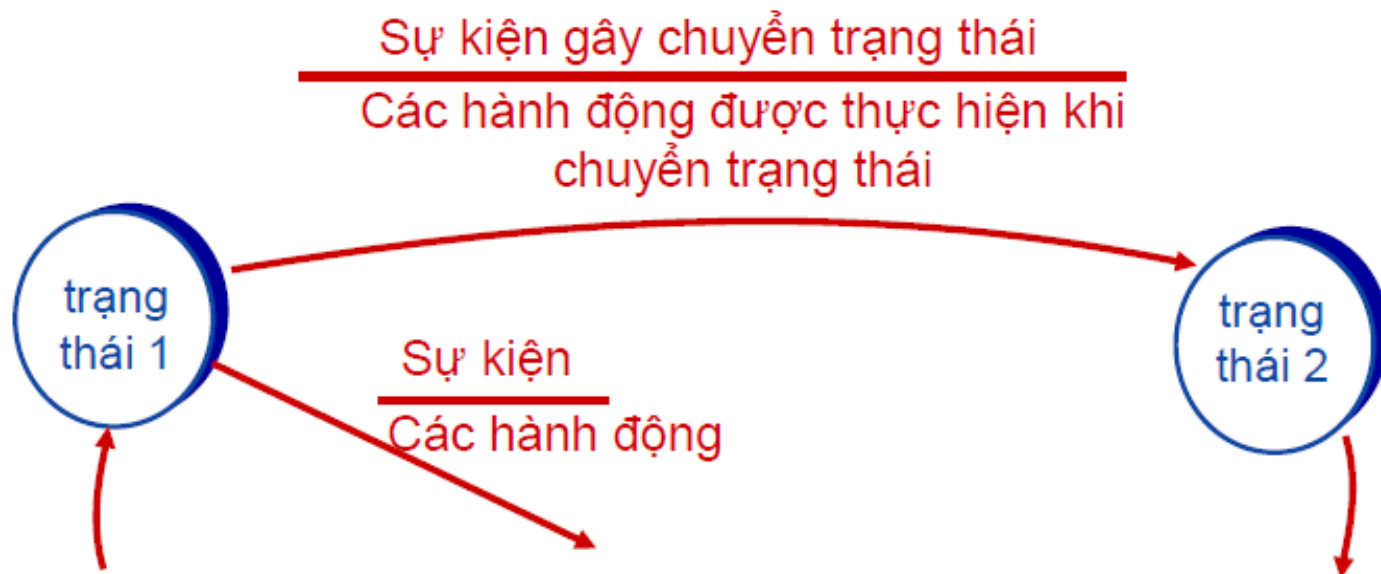


Truyền dữ liệu tin cậy: bắt đầu

Chúng ta sẽ:

- Từng bước phát triển truyền dữ liệu tin cậy (rdt) bên phía người gửi và nhận
- Chỉ xem xét việc truyền dữ liệu theo một chiều (bên gửi đến bên nhận)
- Sử dụng finite state machines (FSM) để xác định bên gửi và nhận

Trạng thái: khi ở “trạng thái” này thì trạng thái kế tiếp được xác định duy nhất bởi sự kiện kế tiếp



Tổng quan

RDT1.0

RDT2.0

RDT2.1

RDT2.2

RDT3.0



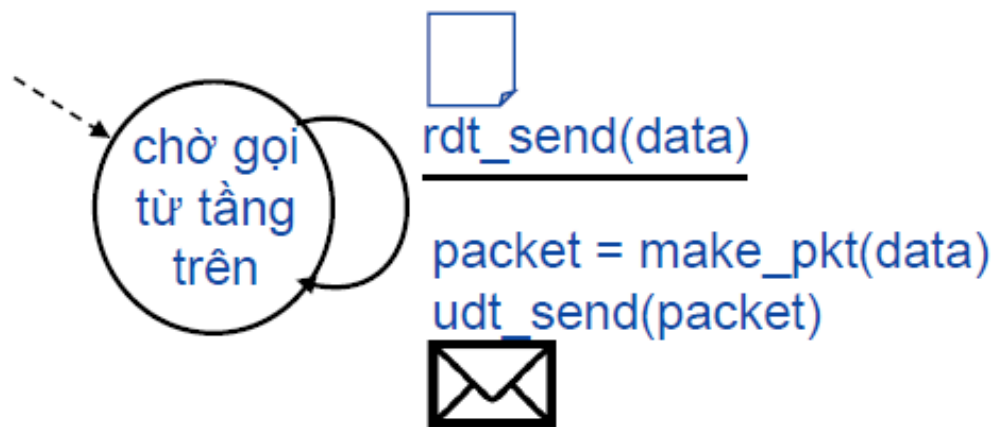
Truyền dữ liệu tin cậy

RDT1.0

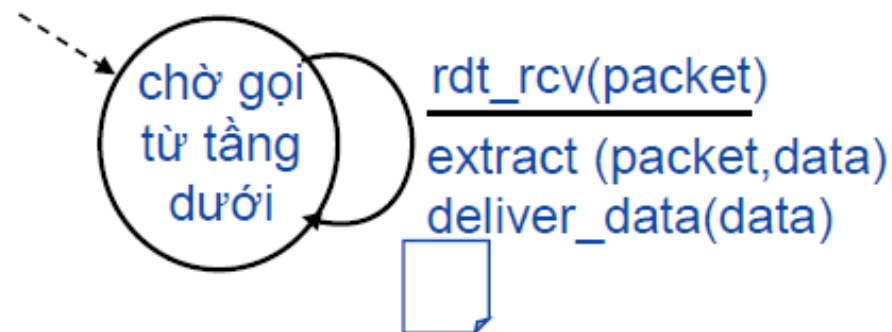
- Kênh truyền tin cậy hoàn toàn



rdt1.0 - Truyền dữ liệu tin cậy hoàn toàn



bên gửi



bên nhận



Truyền dữ liệu tin cậy

RDT1.0

- Kênh truyền tin cậy hoàn toàn

RDT2.0

- Vấn đề: Kênh truyền có lỗi

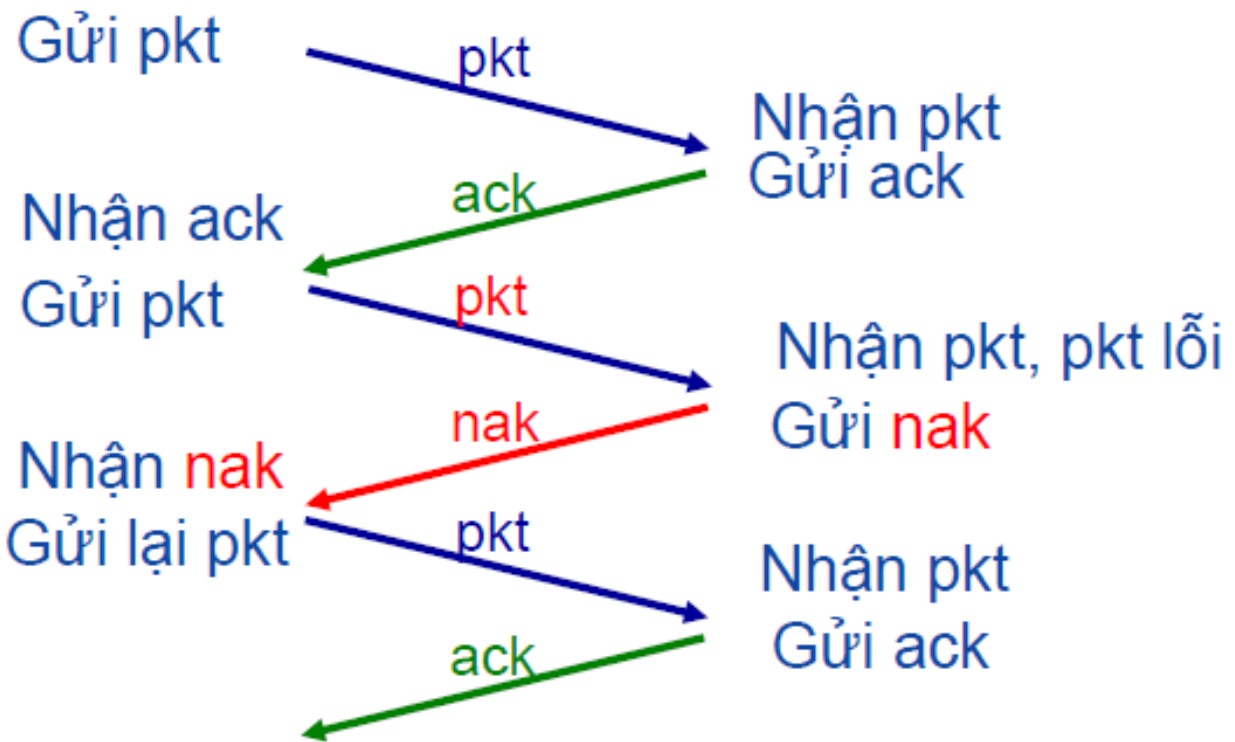




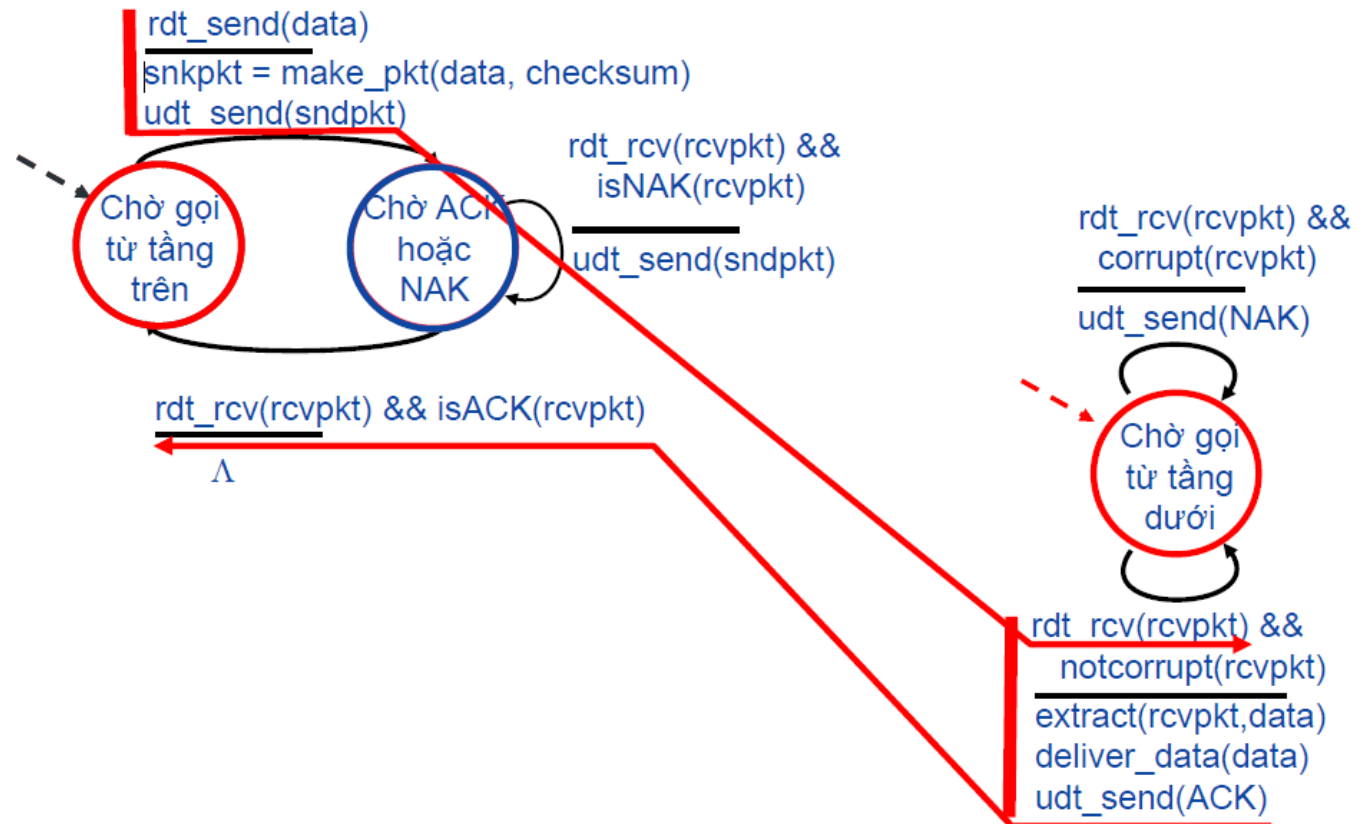
rdt2.0: Kênh truyền xảy ra lỗi

bên gửi

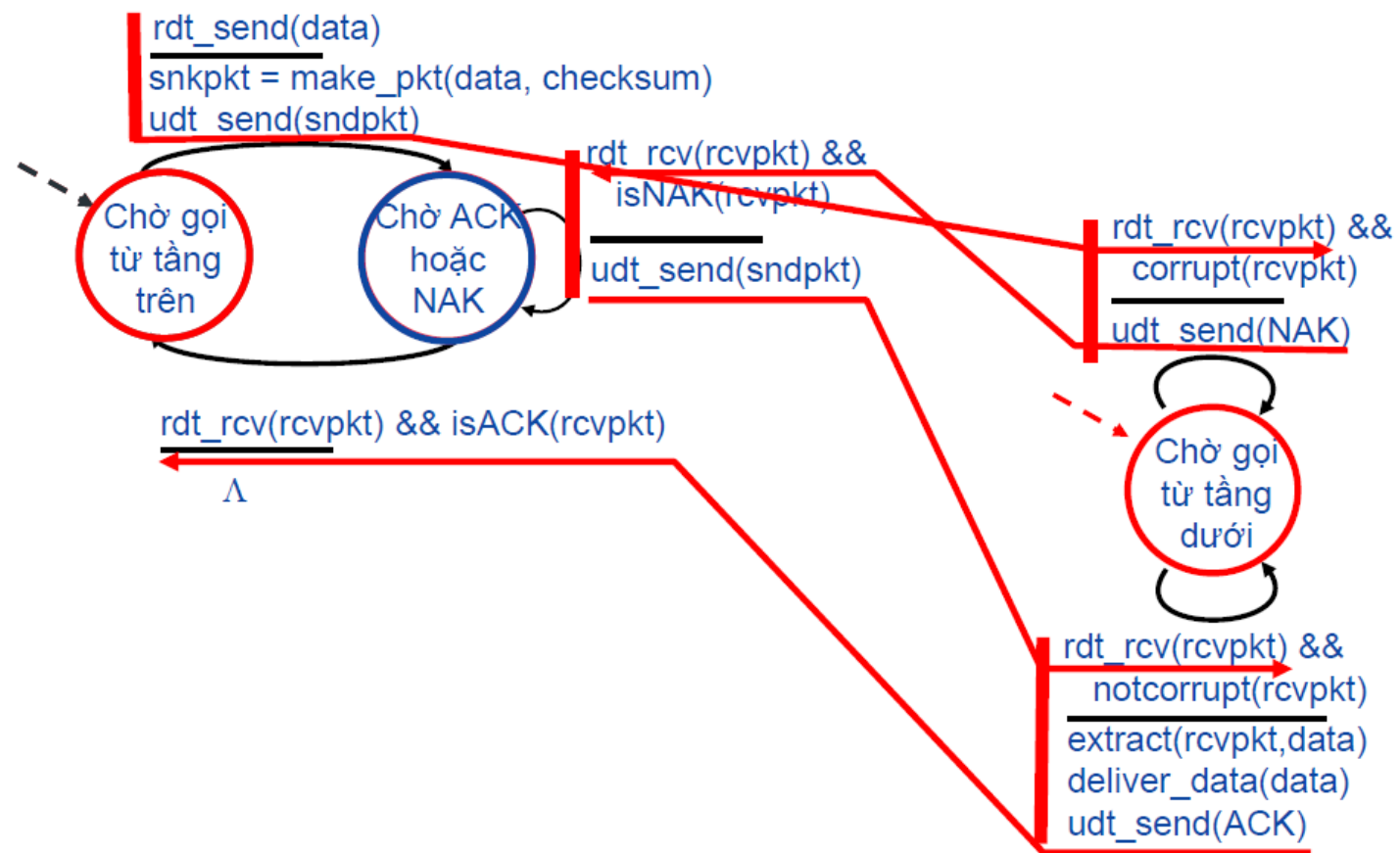
bên nhận



rdt2.0: hoạt động khi không lỗi



rdt2.0: hoạt động khi có lỗi





Truyền dữ liệu tin cậy

RDT1.0

- Kênh truyền tin cậy hoàn toàn

RDT2.0

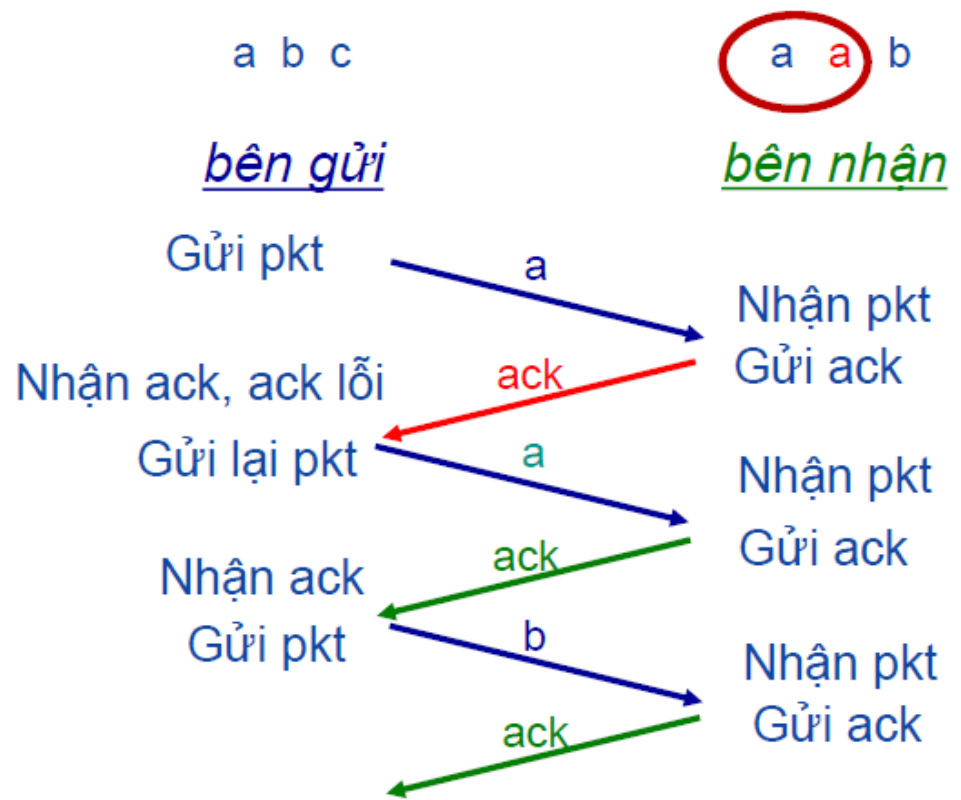
- Vấn đề: Kênh truyền có lỗi
- Giải quyết: ACK và NAK

RDT2.1

- Vấn đề: ACK/NAK lỗi

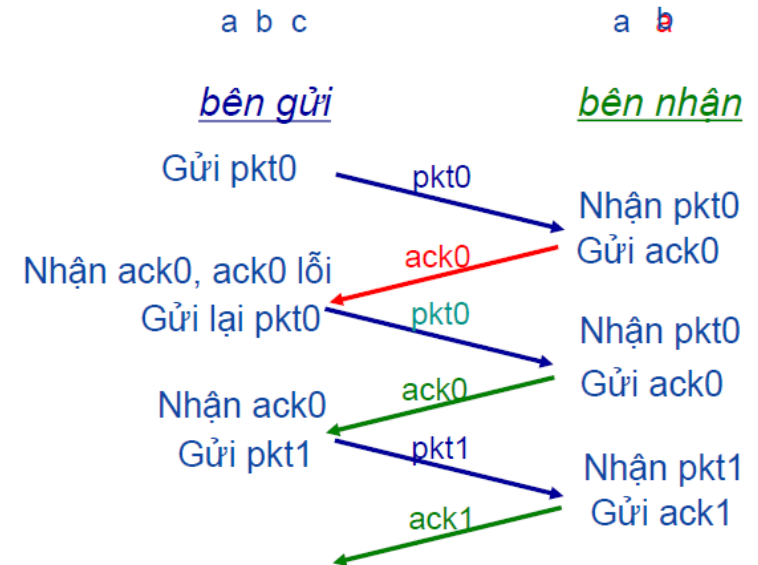
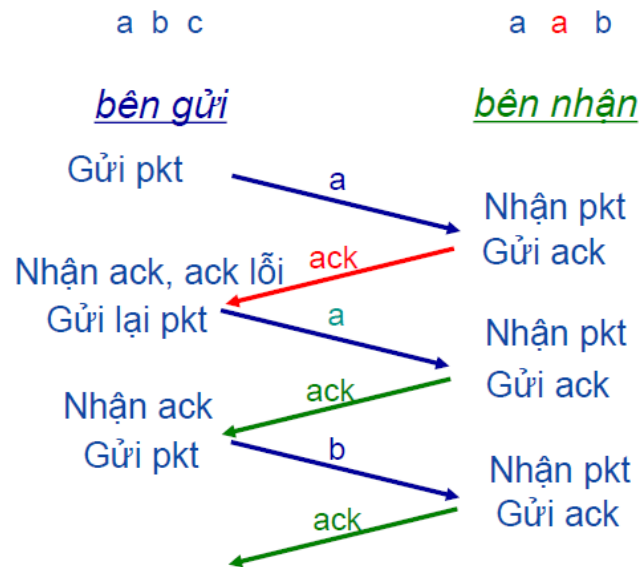


- rdt2.0 có lỗi hỏng nghiêm trọng!





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





rdt2.2: một giao thức không cần NAK

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





The diagram illustrates the Stop-and-Wait protocol between a sender (*bên gửi*) and a receiver (*bên nhận*).

- Scenario 1 (Successful):** The sender sends *pkt0* (blue arrow). The receiver receives *Nhận pkt0* and sends *Gửi ack0* (red arrow). The sender receives *Nhận ack0* and sends *Gửi pkt1* (blue arrow).
- Scenario 2 (Packet Loss):** The sender sends *pkt1* (blue arrow). The receiver receives *Nhận pkt1, pkt1 lỗi* (red arrow). The sender receives *Nhận ack0* and sends *Gửi pkt1* (blue arrow).
- Scenario 3 (Acknowledgment Receipt):** The sender sends *pkt1* (blue arrow). The receiver receives *Nhận pkt1* and sends *Gửi ack1* (green arrow). The sender receives *Nhận ack1* and sends *Gửi pkt1* (blue arrow).



RDT1.0

- Kênh truyền tin cậy hoàn toàn

RDT2.0

- Vấn đề: Kênh truyền có lỗi
- Giải quyết: ACK và NAK

RDT2.1

- Vấn đề: ACK/NAK lỗi
- Giải quyết: Đánh số gói tin $\{0, 1\}$

RDT2.2

- Không cần NAK

RDT3.0

- Vấn đề: Mất gói tin

Truyền dữ liệu tin cậy

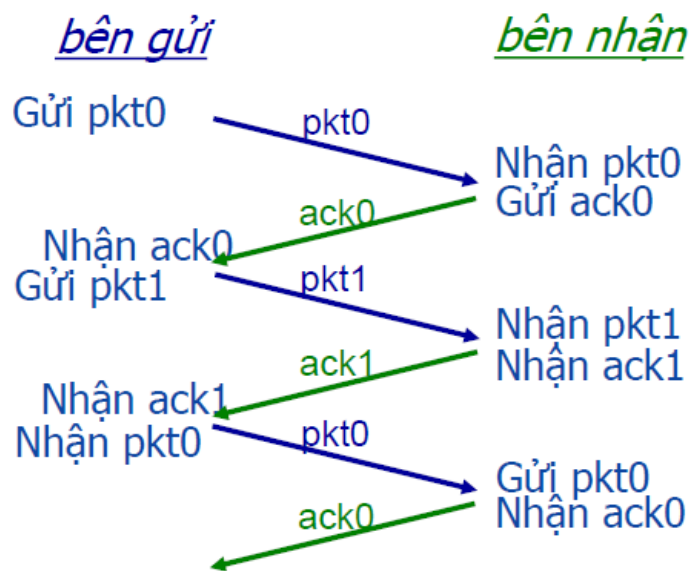




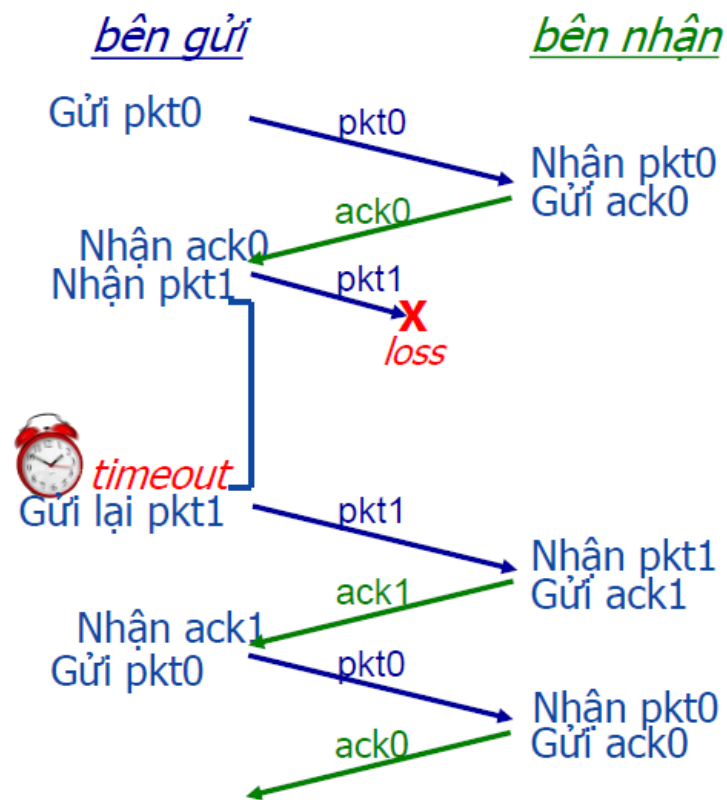
rdt3.0: các kênh với lỗi và mất mát

- **Cách tiếp cận:** bên gửi chờ ACK trong khoảng thời gian “hợp lý”
 - Truyền lại nếu không nhận được ACK trong khoảng thời gian này
 - Nếu gói (hoặc ACK) chỉ trễ (không mất):
 - Việc truyền lại sẽ gây trùng, nhưng số thứ tự đã xử lý trường hợp này
 - Bên nhận phải xác định số thứ tự của gói vừa gửi ACK
- Yêu cầu bộ định thì (timer)
- **Giả định mới:** kênh truyền cũng có thể làm mất gói (dữ liệu, các ACK)
 - checksum, số thứ tự, các ACK, việc truyền lại sẽ hỗ trợ... nhưng không đủ

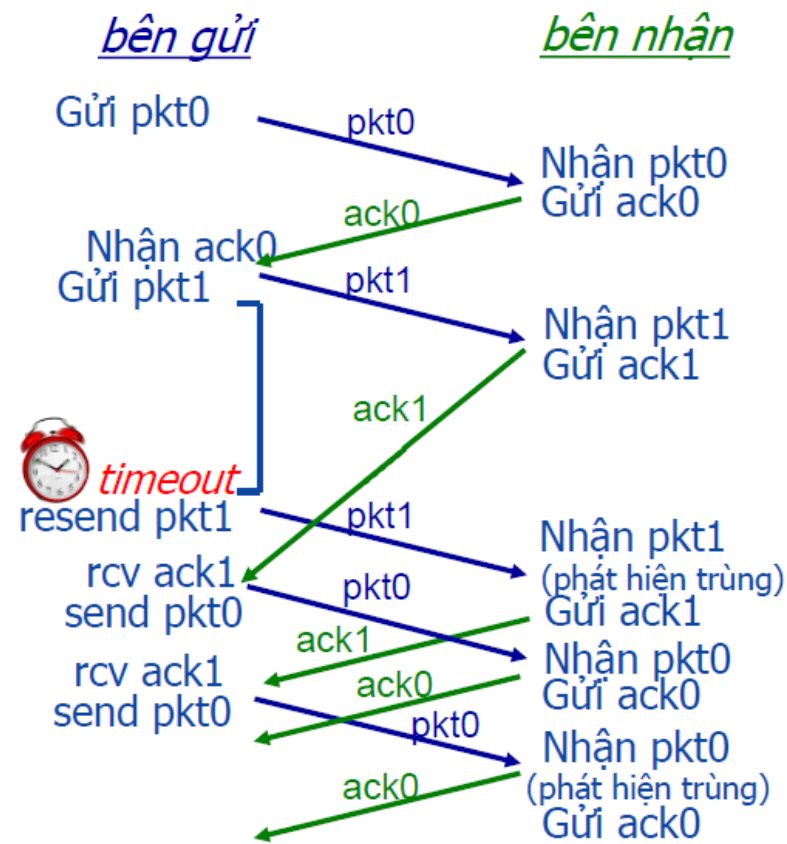
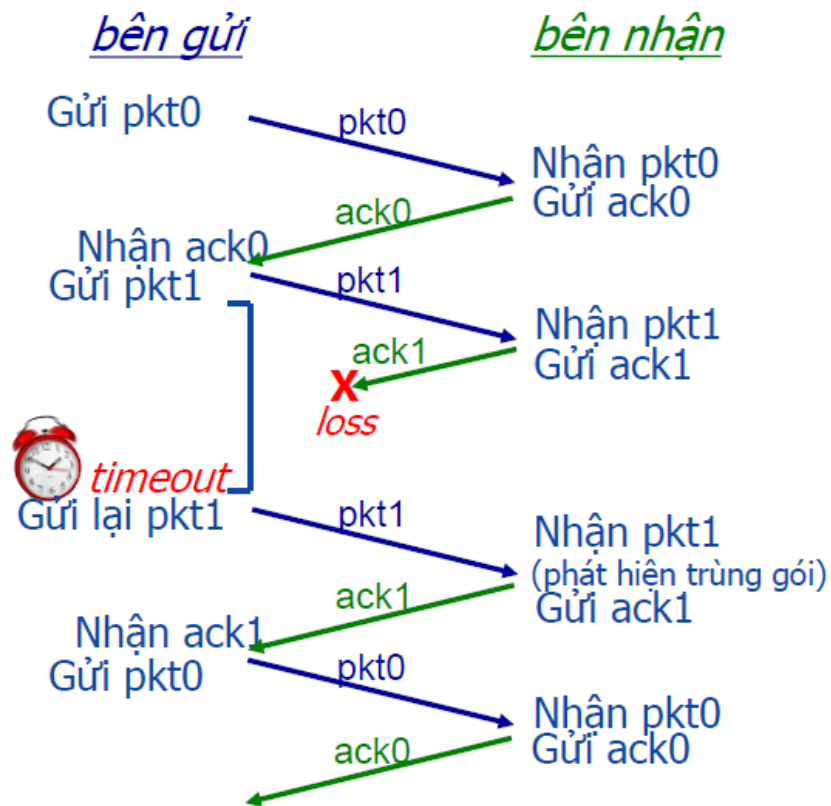




(a) Không mất mát



(b) Mất gói





RDT1.0

- Kênh truyền tin cậy hoàn toàn

RDT2.0

- Vấn đề: Kênh truyền có lỗi
- Giải quyết: ACK và NAK

RDT2.1

- Vấn đề: ACK/NAK lỗi
- Giải quyết: Đánh số gói tin $\{0,1\}$

RDT2.2

- Không cần NAK

RDT3.0

- Vấn đề: Mất gói tin
- Giải quyết: Thêm biến timer

Tổng kết về
Truyền dữ
liệu tin cậy





Hiệu suất của rdt3.0 (stop-and-wait)

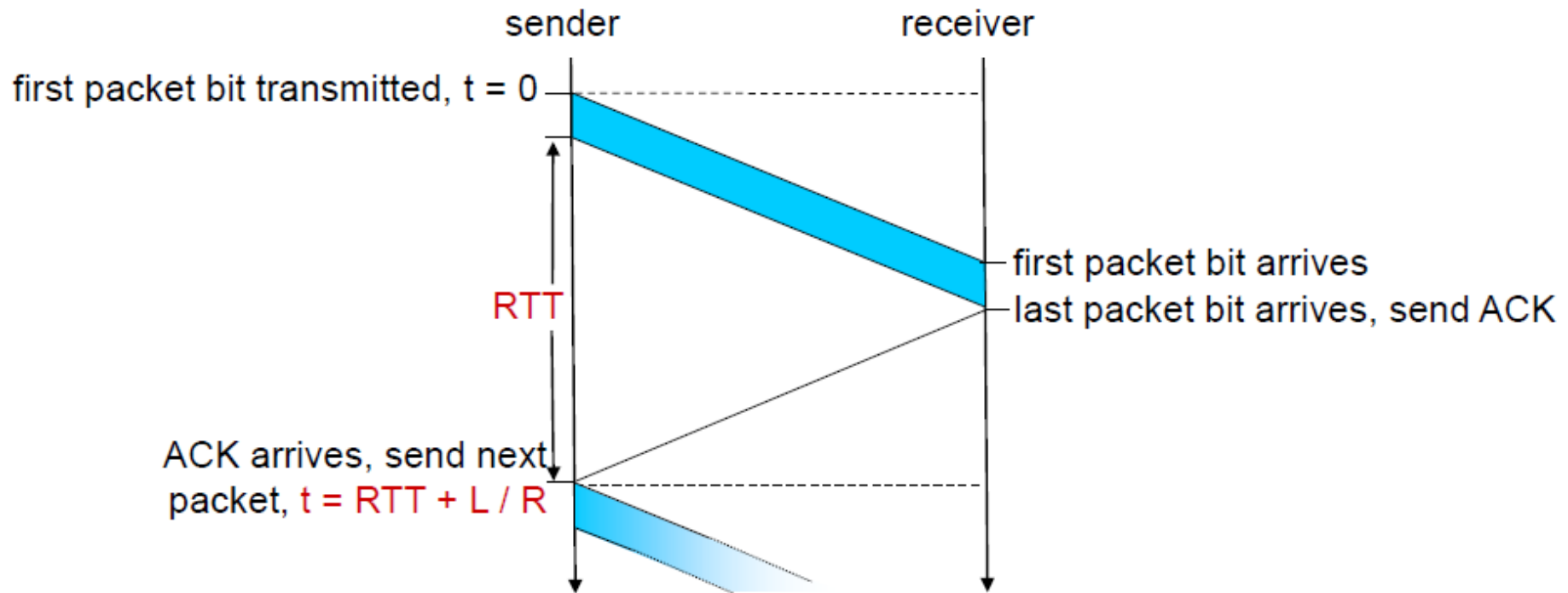
- *U sender. utilization* – tỉ lệ thời gian truyền của bên gửi
- Ví dụ: 1 Gbps link, truyền một gói tin 8000 bit trên kênh truyền có độ trễ lan truyền 8 ms

Thời gian truyền gói tin đến kênh truyền:

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

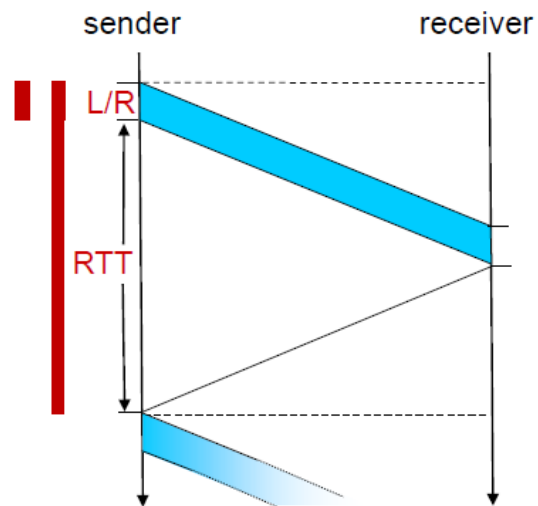


rdt3.0: hoạt động stop-and-wait





$$\begin{aligned}U_{\text{sender}} &= \frac{L / R}{RTT + L / R} \\&= \frac{.008}{30.008} \\&= 0.00027\end{aligned}$$



rdt3.0: hoạt
động stop-
and-wait

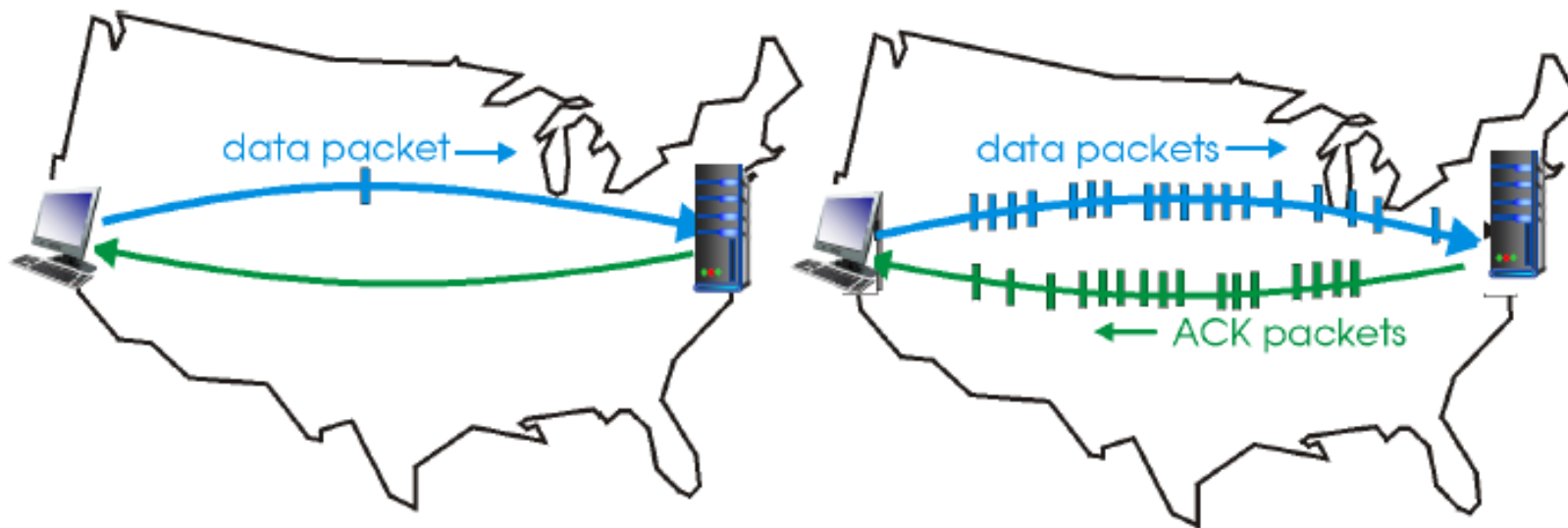
- Hiệu suất giao thức rdt 3.0 kém
- Giao thức giới hạn hiệu suất của cơ sở hạ tầng cơ bản (kênh truyền)



rdt3.0: hoạt động giao thức “pipelined”

pipelining: bên gửi gửi nhiều packet cùng một lúc mà không cần chờ ACK

- Dãy số thứ tự phải được tăng lên
- Đưa vào bộ đệm tại bên gửi hoặc bên nhận



(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation



QR Thi Thử





QR Điểm Danh

