TRƯỜNG ĐẠI HỌC THỦY LỢI

KHOA CÔNG NGHỆ THÔNG TIN

Bộ môn: Kỹ thuật máy tính và mạng

MÔN HỌC: MẠNG MÁY TÍNH

Giảng viên: Trần Văn Hội

Email: hoitv@tlu.edu.vn

Điện thoại: 0944.736.007

NỘI DUNG MÔN HỌC

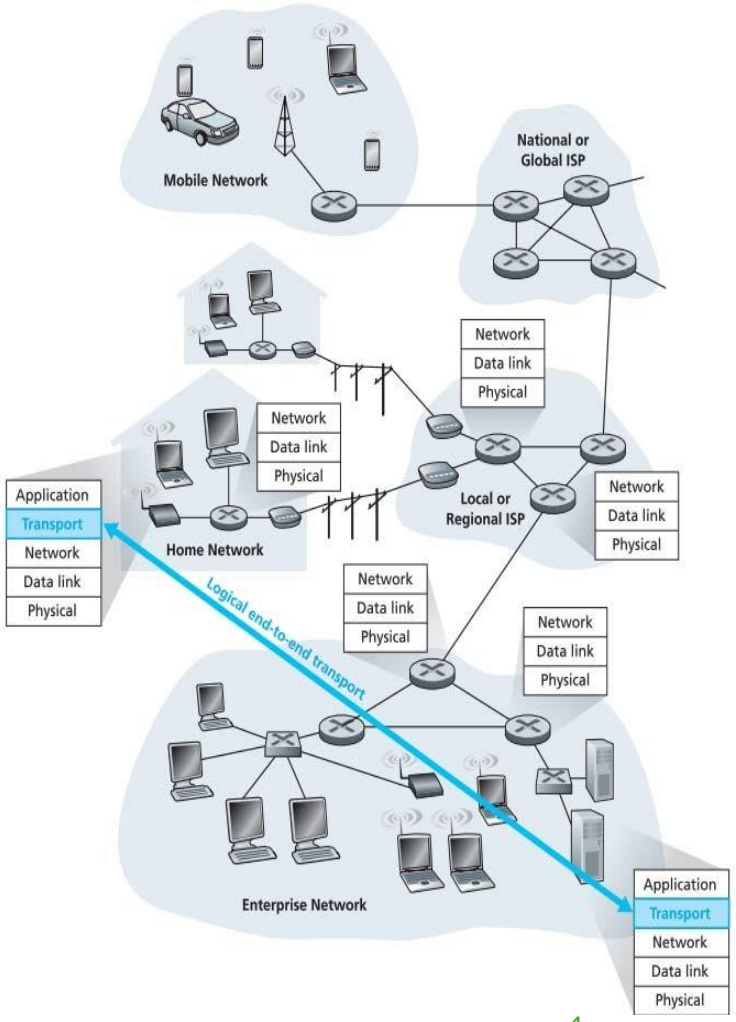
- Chương 1: Tổng quan về mạng máy tính
- Chương 2: Tầng ứng dụng
- Chương 3: Tầng giao vận
- Chương 4: Tầng mạng
- Chương 5: Tầng truy nhập mạng

CHƯƠNG 3: TẦNG GIAO VẬN

1. Các dịch vụ tầng giao vận
2. Dịch vụ dồn kênh/ Phân kênh
3. Giao thức UDP
4. Giao thức TCP
5. Điều khiển luồng/ tắc nghẽn trong TCP

DỊCH VỤ VÀ GIAO THỨC TẦNG GIAO VẬN

- Giao thức giao vận cung cấp *kênh truyền lôgic* giữa các tiến trình ứng dụng chạy trên các host khác nhau
- Giao thức giao vận chạy trong các end system
  - Phía gửi: chia app message thành các *segment*, chuyển tới tầng mạng
  - Phía nhận: ghép các segment lại thành message, chuyển tới tầng ứng dụng
- Có nhiều hơn một giao thức giao vận cho các ứng dụng
  - Internet: TCP và UDP



QUAN HỆ GIỮA TẦNG GIAO VẬN VÀ TẦNG MẠNG

- *Tầng giao vận:* cung cấp đường truyền lôgic giữa các tiến trình chạy trên các host
- *Tầng mạng:* cung cấp đường truyền lôgic giữa các host

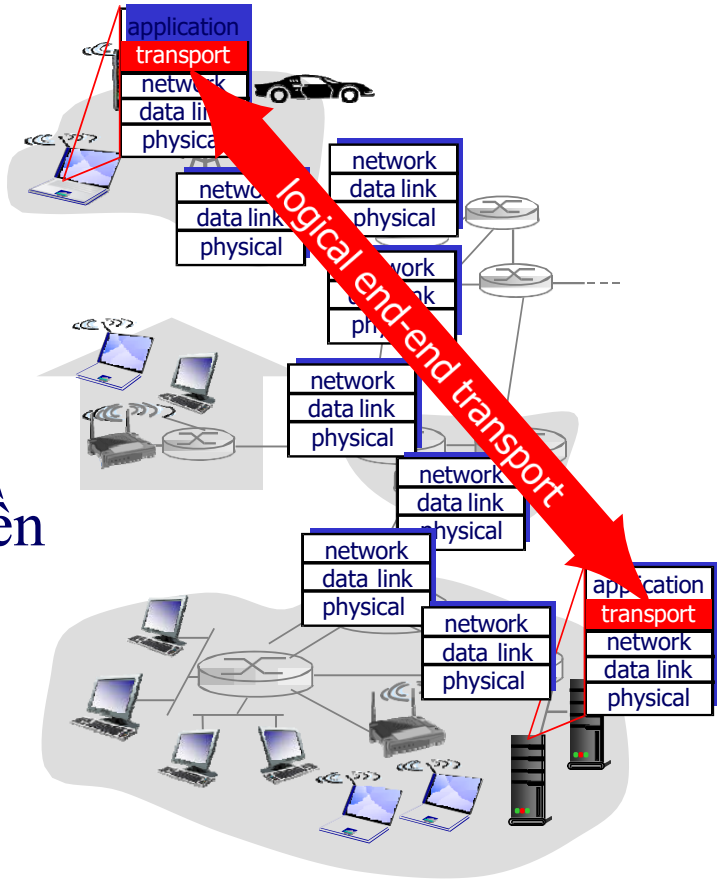
một so sánh:

12 trẻ trong nhà của Anh gửi thư cho 12 trẻ trong nhà của Hoàng:

- host = nhà
- tiến trình = trẻ
- Thông điệp = thư trong phong bì thư
- Giao thức giao vận = Anh và Hoàng
- Giao thức tầng mạng = dịch vụ thư bưu điện

TẦNG GIAO VẬN CỦA INTERNET

- TCP (Transmission Control Protocol):
  - Truyền dữ liệu tin cậy
  - Thiết lập kết nối trước khi truyền
  - Điều khiển luồng
  - Điều khiển tắc nghẽn
- UDP (User Datagram Protocol):
  - Truyền dữ liệu không tin tin cậy
  - Không thiết lập kết nối trước khi truyền
  - Không có điều khiển luồng/tắc nghẽn
- Các dịch vụ không cung cấp ở tầng giao vận:
  - Đảm bảo độ trễ
  - Đảm bảo băng thông



TẦNG GIAO VẬN CỦA INTERNET

- **Ứng dụng và dịch vụ giao vận**

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	TCP
Remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
File transfer	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (e.g., YouTube)	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	UDP or TCP

CHƯƠNG 3: TẦNG GIAO VẬN

1. Các dịch vụ tầng giao vận
2. Dịch vụ dồn kênh/ Phân kênh
3. Giao thức UDP
4. Giao thức TCP
5. Điều khiển luồng/ tắc nghẽn trong TCP

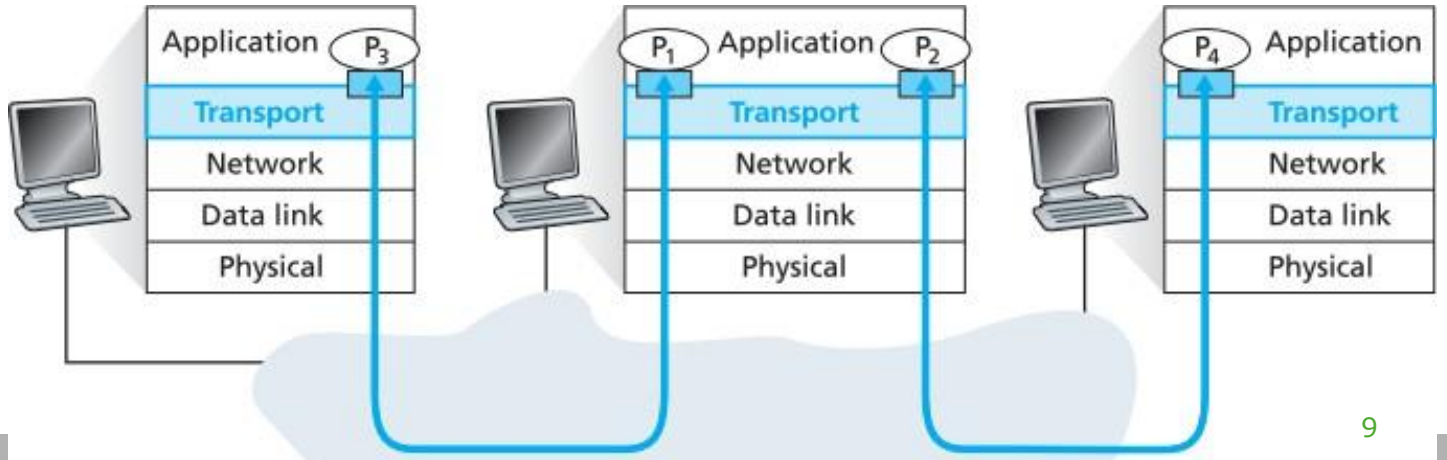
DỒN KÊNH VÀ PHÂN KÊNH

Dồn kênh (multiplexing) tại nút gửi

Tầng giao vận nhận dữ liệu từ nhiều tiến trình khác nhau, tạo segment chứa dữ liệu và thêm thông tin số hiệu cổng nguồn và số hiệu cổng đích vào tiêu đề segment và gửi segment xuống tầng mạng

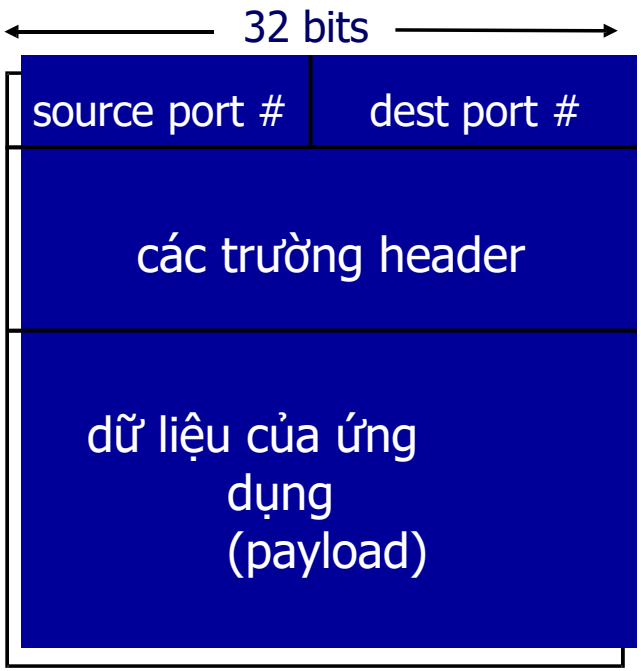
Phân kênh (demultiplexing) tại nút nhận

Tầng giao vận sử dụng thông tin header để chuyển các segment đã nhận tới đúng tiến trình ứng dụng nhận



DỒN KÊNH VÀ PHÂN KÊNH HOẠT ĐỘNG NHƯ THẾ NÀO?

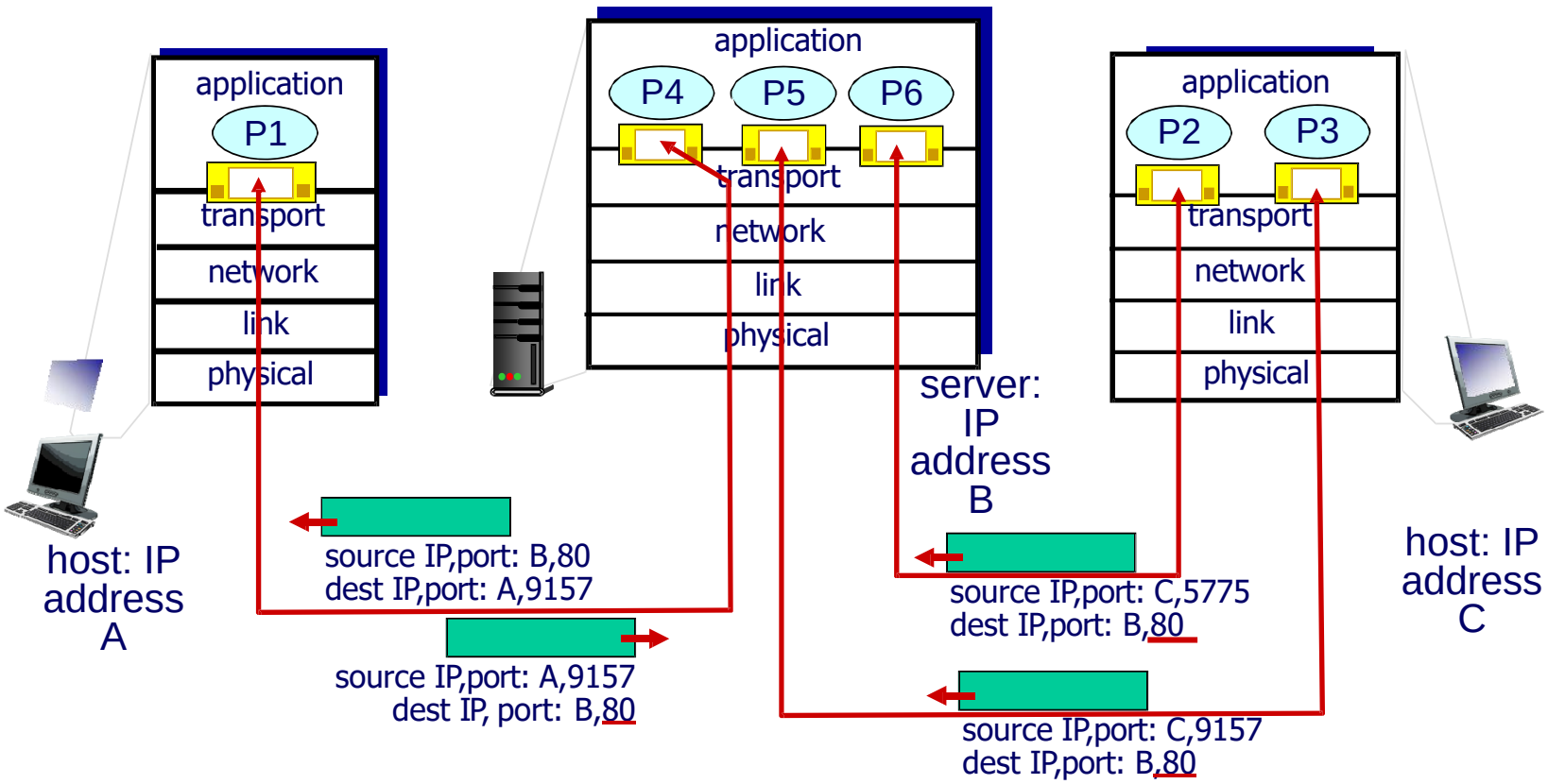
- Dồn kênh và phân kênh nhờ hai trường đặc biệt ở đầu segment
  - Định danh cổng tiến trình gửi (source port number): là một giá trị chưa được sử dụng bởi tiến trình nào
  - Định danh cổng tiến trình nhận (dest port number): là số hiệu cổng của ứng dụng, ví dụ: Web là 80
- Mỗi datagram ở tầng mạng có: Địa chỉ IP nguồn và IP đích để xác định host gửi và host nhận



cấu trúc TCP/UDP segment

- Host sử dụng địa chỉ *IP và port number* để chuyển segment tới socket thích hợp

MINH HỌA VỀ DỒN KÊNH VÀ PHÂN KÊNH



3 segments, gửi tới IP address: B,
dest port: 80 được tách kênh tới các socket khác nhau

CHƯƠNG 3: TẦNG GIAO VẬN

1. Các dịch vụ tầng giao vận
2. Dịch vụ dồn kênh/ Phân kênh
3. Giao thức UDP
4. Giao thức TCP
5. Điều khiển luồng/ tắc nghẽn trong TCP

GIAO THỨC UDP

- RFC 768
- Giao thức UDP (User Datagram Protocol) là giao thức không hướng kết nối (connectionless), không cần thiết lập kết nối
- UDP không yêu cầu về độ tin cậy cao, không có cơ chế xác nhận ACK, không đảm bảo chuyển giao các gói dữ liệu đến đích và theo đúng thứ tự và không thực hiện loại bỏ các gói tin trùng lặp.
- Tiêu đề gói tin nhỏ: 8 bytes (so với 20 bytes của TCP)
- UDP có những chức năng cơ bản gì?
  - Dồn kênh/ phân kênh
  - Phát hiện lỗi bit đơn giản (dùng checksum)

UDP CHECKSUM

Mục đích: phát hiện lỗi bit trong segment đã gửi

Bên gửi:

- Coi dữ liệu của segment, bao gồm cả header gồm các từ 16 bít
- Tính giá trị bù một của tổng các từ 16 bit trong segment (RFC 1071)
- Giá trị checksum nhận được được đặt vào trường checksum trong gói dữ liệu UDP

Bên nhận:

- tính checksum của segment đã nhận được
- kiểm tra xem checksum đã tính có bằng với giá trị của trường checksum không:
  - Không: phát hiện lỗi
  - Có - không phát hiện lỗi.

VÍ DỤ: INTERNET CHECKSUM

- Giả sử có 2 từ 16 bit

- Lưu ý: khi cộng các số, bit nhớ cao nhất cần được cộng vào kết quả

VÍ DỤ: INTERNET CHECKSUM

- Dữ liệu truyền đi: 2 từ 16 bit + 16 bit checksum

1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	Từ 1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	Từ 2
0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	Checksum

- Dữ liệu nhận được không có lỗi: cộng tất cả các từ 16 bit kể cả checksum thì thu được tổng gồm các bit 1

Từ 1		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
Từ 2	+	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
bit dư thừa		(1)	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
checksum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
	+	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

VÍ DỤ: INTERNET CHECKSUM

- Dữ liệu truyền đi: 2 từ 16 bit + 16 bit checksum

1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	Từ 1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	Từ 2
0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	Checksum

- Dữ liệu nhận được có lỗi: cộng tất cả các từ 16 bit kể cả checksum thì thu được tổng có một bit nào đó bằng 0

Từ 1		1	1	1	0	0	1	1	0	0	1	0	0	0	1	1	0
Từ 2	+	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
bit dư thừa		(1)	1	0	1	1	1	0	1	1	1	0	0	1	1	0	1	1
		1	0	1	1	1	0	1	1	1	0	0	1	1	1	0	0
checksum	+	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1
		1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1

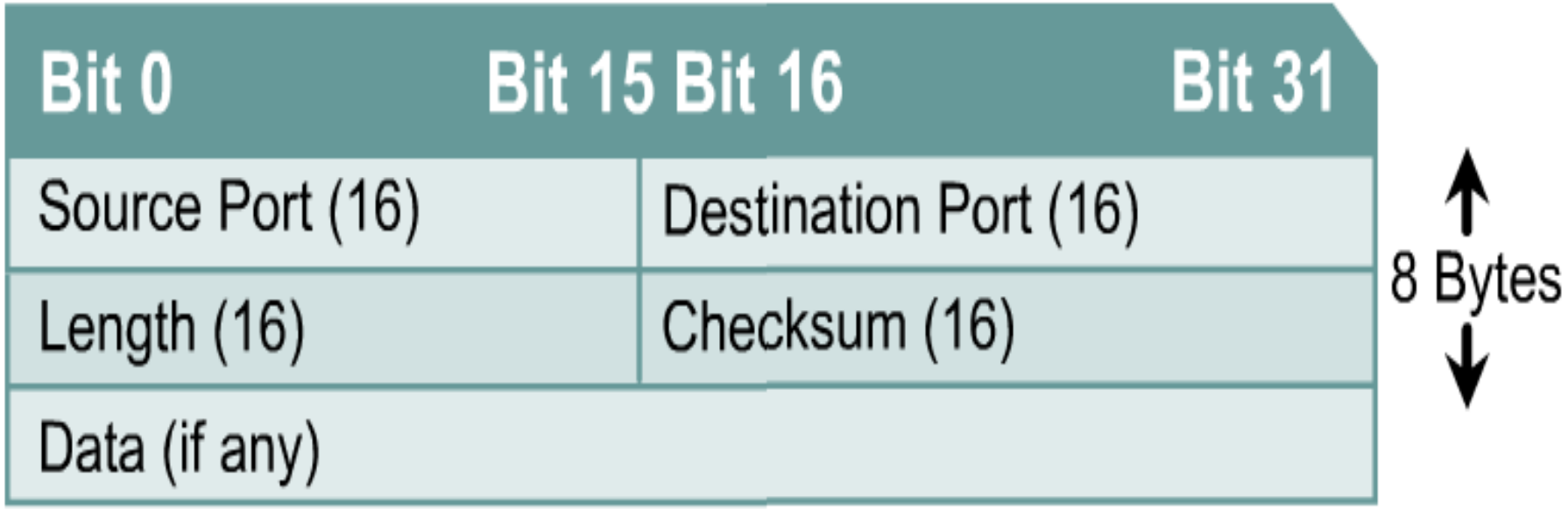
BÀI TẬP

1. **Cho hai số 16 bit 513 và 1025 tính checksum của hai số là**

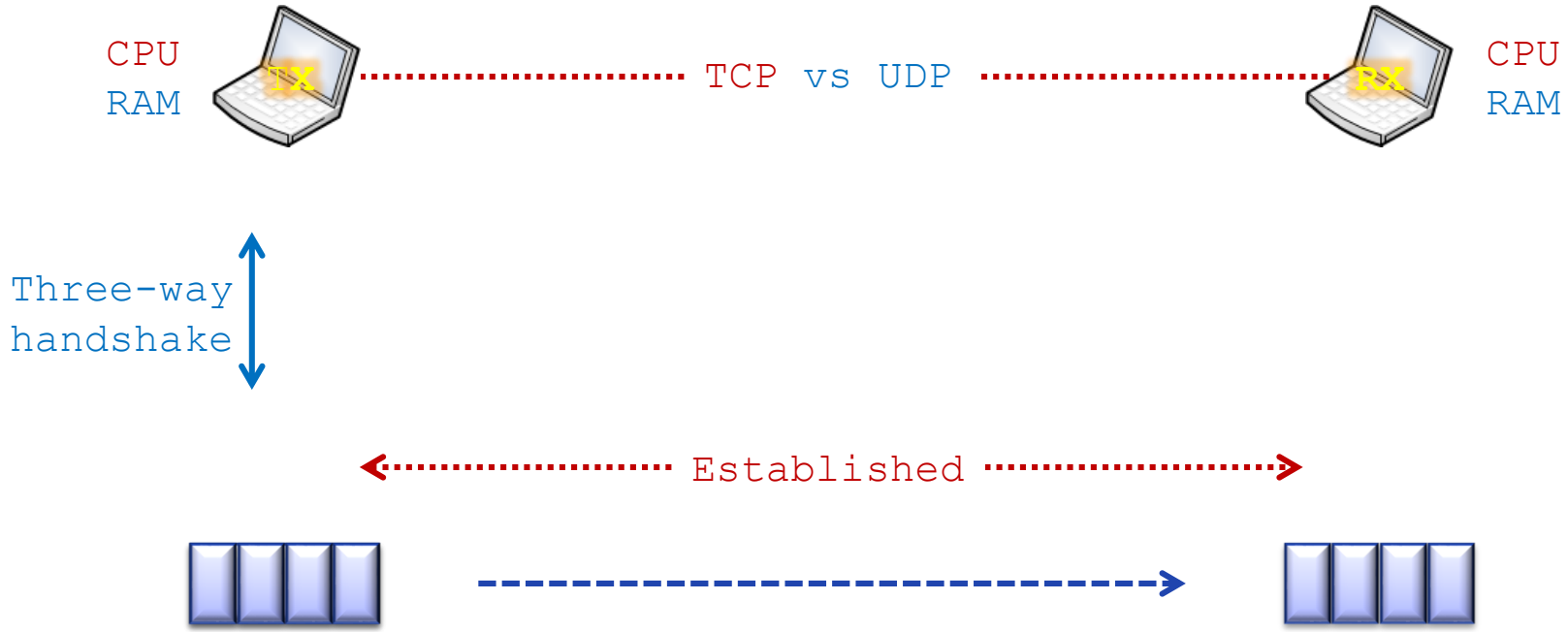
2. Cho hai số AEDC và 2F7A tính checksum của hai số là

0010 0001 1010 1001

KHUÔN DẠNG GÓI TIN UDPV4



GIAO THỨC UDP



S.PORT VS D.PORT

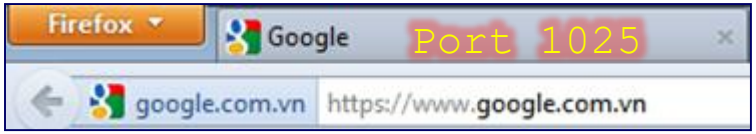
Established

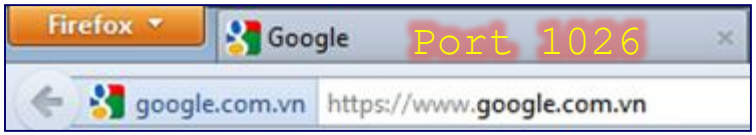
Port 53

Port 443

Segment	S.Port 1025	D.Port 443
Segment	S.Port 443	D.Port 1025

Segment	S.Port 1025	D.Port 443
Segment	S.Port 443	D.Port 1025





Established

Port 1025

Established

CÁC VẤN ĐỀ CỦA UDP

- Không có kiểm soát tắc nghẽn
  - UDP cứ gửi dữ liệu nhanh nhất, nhiều nhất có thể

  $\Rightarrow$ làm Internet quá tải
- Không đảm bảo độ tin cậy
  - Các ứng dụng phải cài đặt cơ chế tự kiểm soát độ tin cậy
  - Việc phát triển ứng dụng sẽ phức tạp hơn

CHƯƠNG 3: TẦNG GIAO VẬN

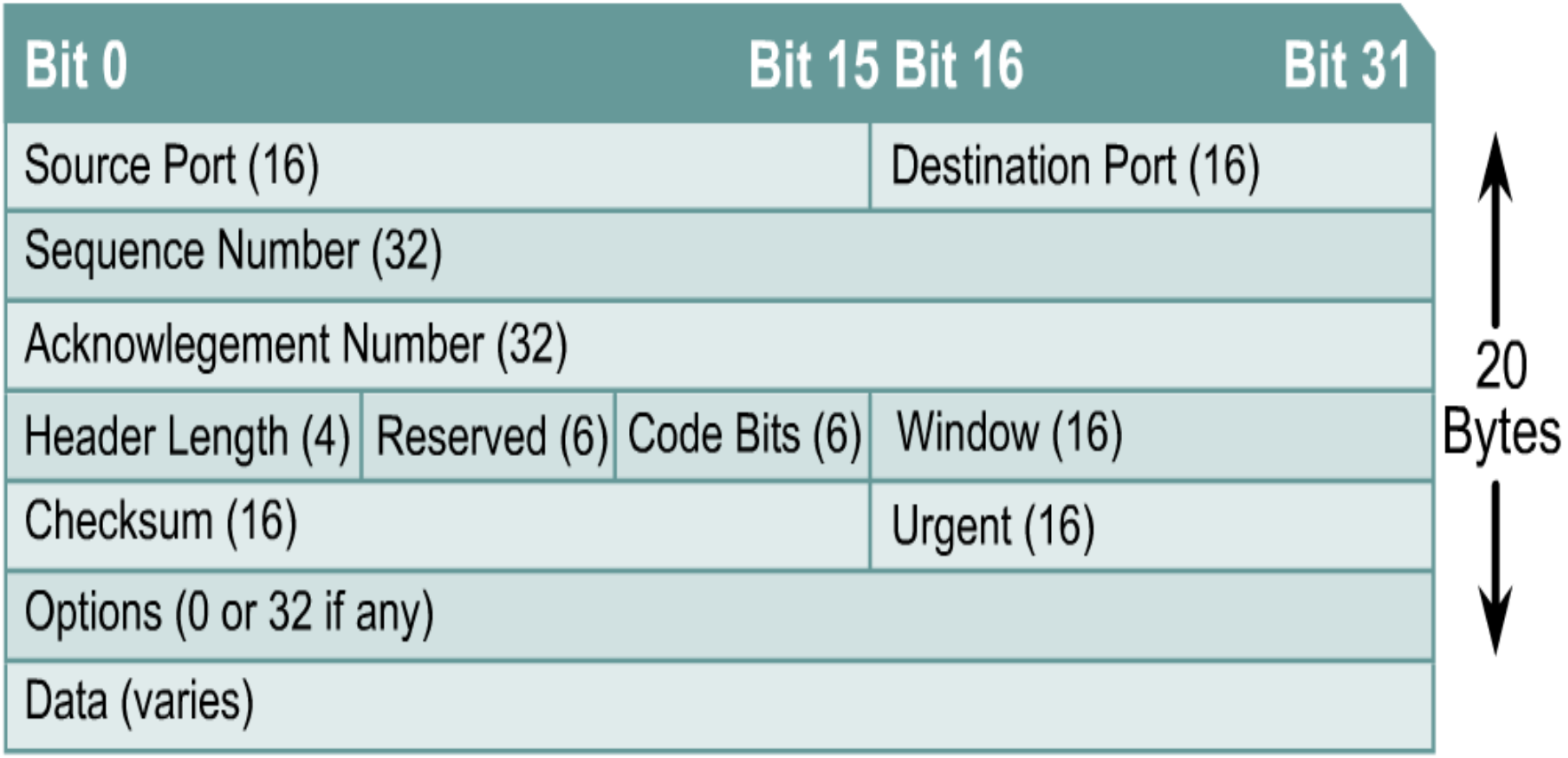
1. • Các dịch vụ tầng giao vận
2. • Dịch vụ dồn kênh/ Phân kênh
3. • Giao thức UDP
4. • Giao thức TCP
5. • Điều khiển luồng/ tắc nghẽn trong TCP

GIAO THỨC TCP

- TCP (Transmission Control Protocol) hướng kết nối (connection-oriented):
  - Bắt tay ba bước (handshaking)
- Giao thức truyền dữ tin cậy, truyền theo chuỗi byte đảm bảo thứ tự:
  - Sử dụng vùng đệm
- Truyền theo kiểu liên tục (pipelined):
  - Tăng hiệu quả

- **Điều khiển luồng:**
  - Bên gửi không làm quá tải bên nhận
- **Điều tắc nghẽn:**
  - Việc truyền dữ liệu không nên làm tắc nghẽn mạng

CẤU TRÚC GÓI TIN TCP



CHỨC NĂNG CÁC TRƯỜNG TRONG TCP

- Cổng nguồn (Source Port): 16 bít, số hiệu cổng nguồn.
- Cổng đích (Destination Port): Độ dài 16 bít, chứa số hiệu cổng đích.
- Sequence Number: 32 bits, số thứ tự của gói số liệu khi phát.
- Giả sử rằng một tiến trình trong máy A muốn gửi một luồng dữ liệu đến một tiến trình trong máy B qua kết nối TCP. TCP trong máy A sẽ đánh số thứ tự cho từng byte của luồng dữ liệu. Mỗi luồng chia thành nhiều segment, mỗi segment có số thứ tự là số thứ tự của byte đầu tiên của segment

CHỨC NĂNG CÁC TRƯỜNG TRONG TCP

- Giả sử luồng dữ liệu gồm 500.000 byte, mỗi segment là 1.000 byte. TCP xây dựng 500 segment dữ liệu.
  - Segment đầu tiên được gán số thứ tự 0. Segment thứ hai được gán số thứ tự 1.000. Segment thứ ba số thứ tự 2.000, v.v.
  - Mỗi số thứ tự như vậy được chèn vào trường Sequence number trong tiêu đề của phân đoạn TCP thích hợp.

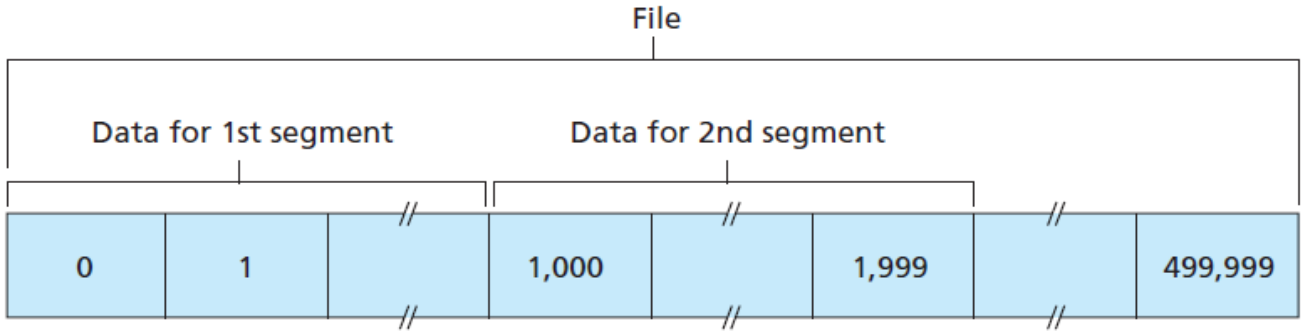


Figure 3.30 ♦ Dividing file data into TCP segments

CHỨC NĂNG CÁC TRƯỜNG TRONG TCP

- TCP là kênh truyền song công, nên bên A có thể nhận được dữ liệu từ B, trong khi A gửi dữ liệu cho B (trên cùng kết nối TCP)
- Acknowledgement number (32 bits): là số thứ tự của byte tiếp theo mà máy A đang chờ máy B gửi tới.
- Ví dụ:
  - A đã nhận được tất cả các byte được đánh số từ 0 đến 535 từ B và giả sử rằng A sắp gửi một đoạn đến B.
  - A đang chờ byte từ 536 và các byte tiếp theo trong luồng dữ liệu của B. Vì vậy, A đặt 536 vào trường Acknowledgement number của đoạn mà nó gửi đến B.

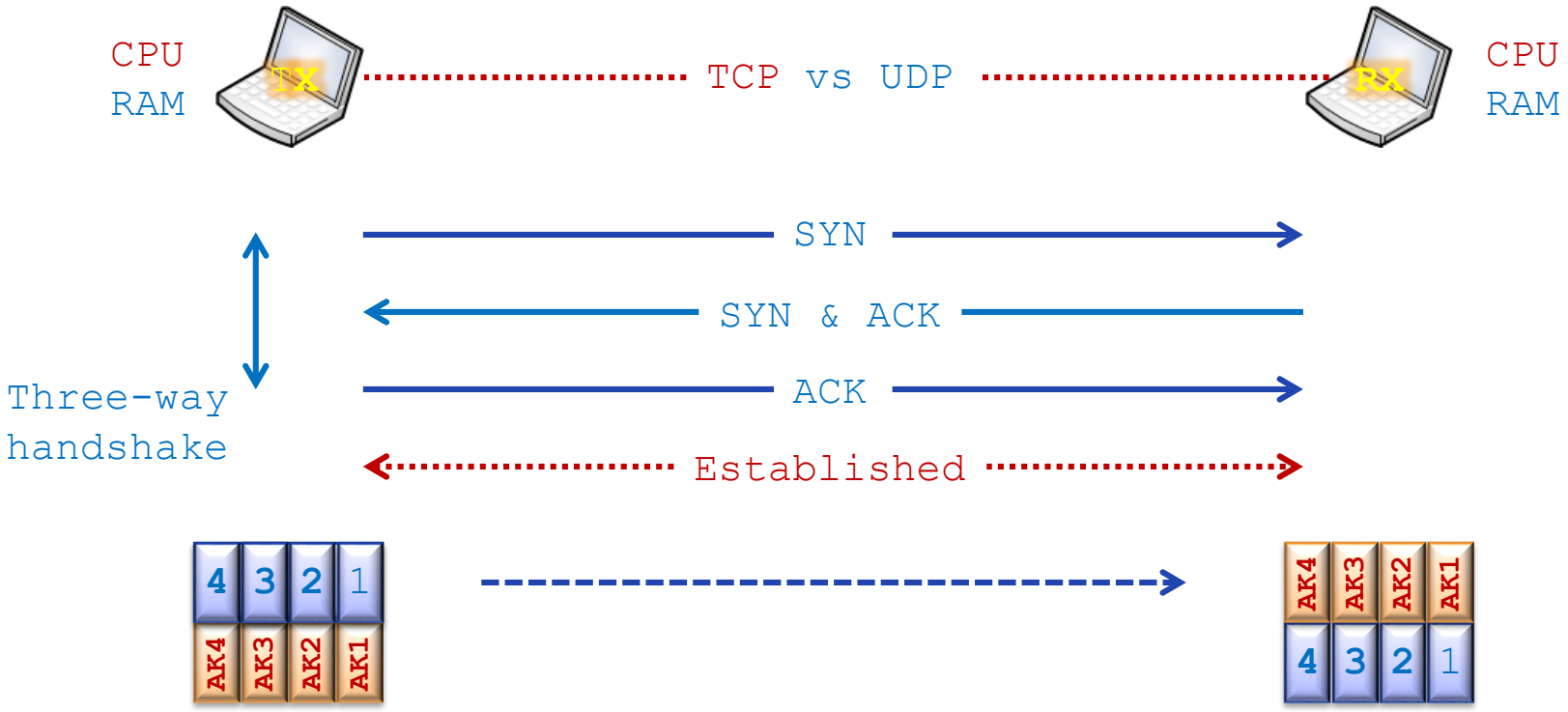
CHỨC NĂNG CÁC TRƯỜNG TRONG TCP

- Header length (4 bíts): Xác định độ dài Header gói tin TCP.
- Reserved (6 bít): Để dành cho tương lai lấp đầy bằng 0.
- Code bits: Các bits điều khiển (U, A, P, R, S, F)
- Window (16 bits): Số lượng các Byte dữ liệu trong vùng cửa sổ bên phát mà bên nhận sẵn sàng chấp nhận.
- Checksum (16 bits): Mã kiểm soát lỗi (theo phương pháp CRC).
- Urgent Pointer (16 bits): Số thứ tự của Byte dữ liệu khẩn, khi URG được thiết lập (trong Code bits).
- Option (độ dài thay đổi): Khai báo độ dài tối đa của TCP Data trong một Segment.

TRUYỀN DỮ LIỆU TIN CẬY CỦA TCP

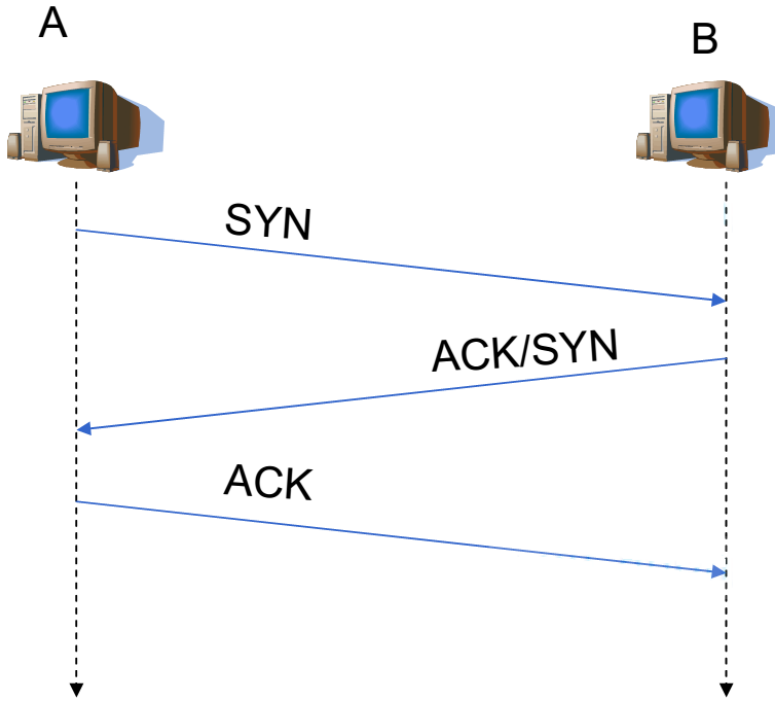
- TCP kiểm soát dữ liệu đã được nhận chưa
  - Seq. #
  - ACK (Acknowledgement number)
- Chu trình làm việc của TCP:
  - Thiết lập kết nối
    - Bắt tay ba bước
  - Truyền/nhận dữ liệu
  - Đóng kết nối

TRUYỀN DỮ LIỆU TIN CẬY CỦA TCP



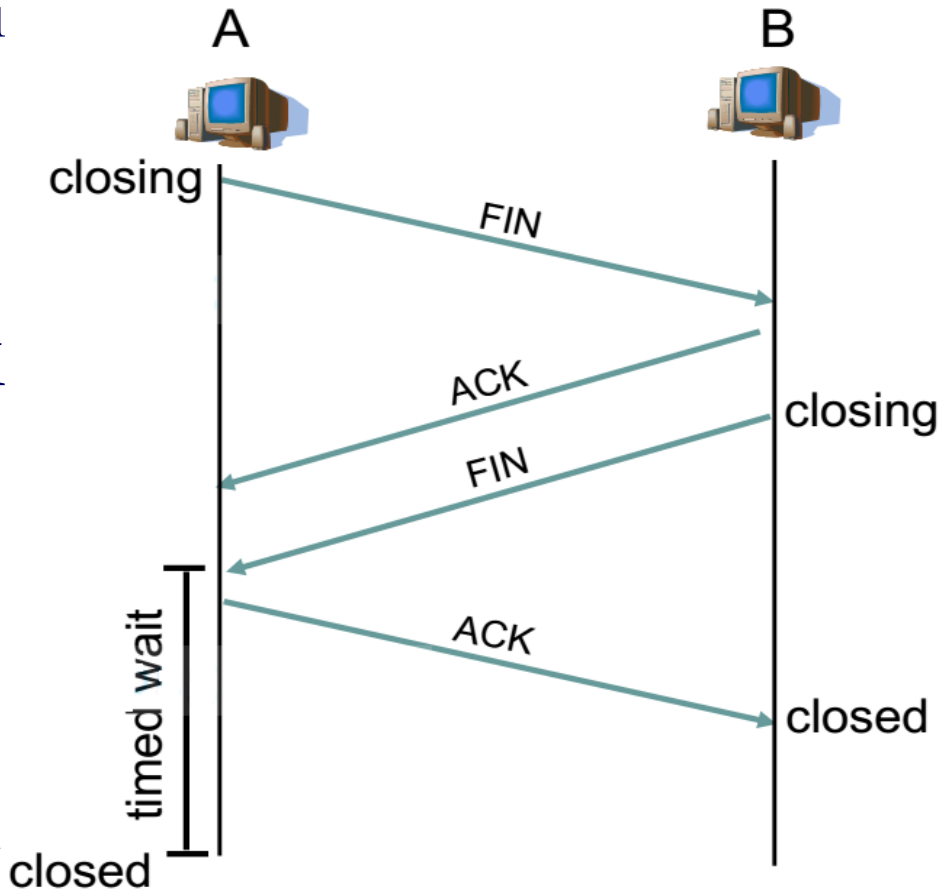
THIẾT LẬP KẾT NỐI TCP: GIAO THỨC BẮT TAY BA BƯỚC

- Bước 1: A gửi SYN cho B
  - Chỉ ra giá trị khởi tạo seq # của A không có dữ liệu
- Bước 2: B nhận SYN, trả lời bằng SYN/ACK
  - B khởi tạo vùng đệm chỉ ra giá trị khởi tạo seq. # của B
- Bước 3: A nhận SYN/ACK, trả lời ACK, có thể kèm theo dữ liệu.



VÍ DỤ VỀ ĐÓNG KẾT NỐI TCP

- Bước 1: A Gửi FIN cho B
- Bước 2: B nhận được FIN, trả lời ACK, đồng thời đóng liên kết và gửi FIN.
- Bước 3: A nhận FIN, trả lời ACK vào trạng thái "chờ".
- Bước 4: B nhận ACK. Đóng liên kết.
- Lưu ý: Cả hai bên đều có thể chủ động đóng liên kết



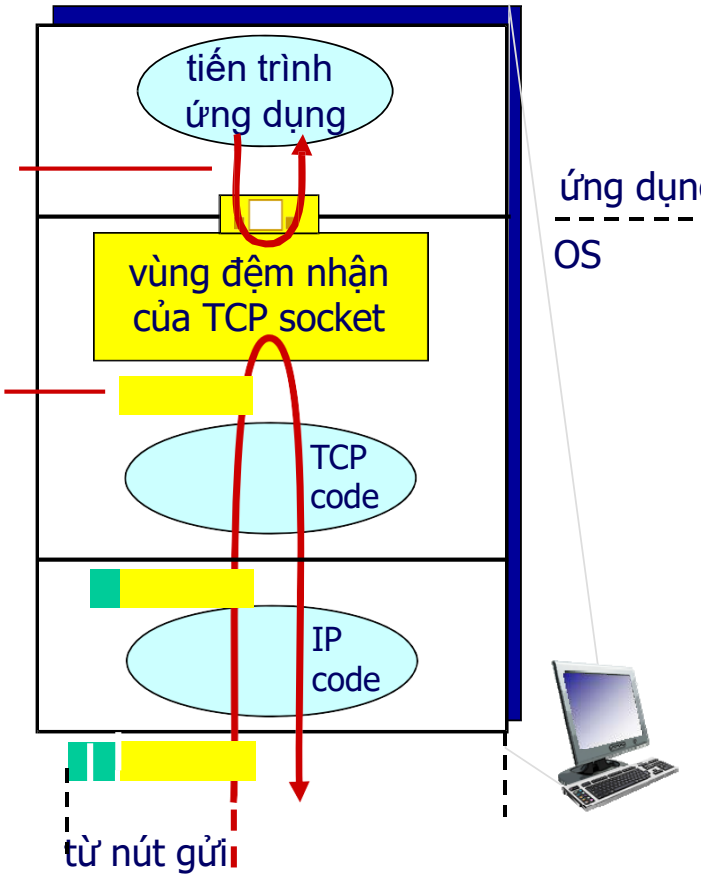
CHƯƠNG 3: TẦNG GIAO VẬN

1. • Các dịch vụ tầng giao vận
2. • Dịch vụ dồn kênh/ Phân kênh
3. • Giao thức UDP
4. • Giao thức TCP
5. • Điều khiển luồng/ tắc nghẽn trong TCP

ĐIỀU KHIỂN LUỒNG CỦA TCP

Ứng dụng có thể lấy dữ liệu khỏi TCP socket buffers ….

… chậm hơn TCP bên nhận đang chuyển (sender đang gửi)



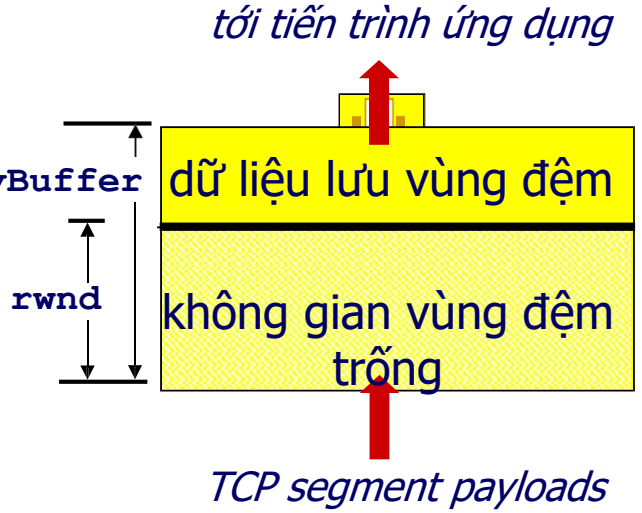
Ngăn xếp giao thức tại nút nhận

Điều khiển luồng

nút nhận điều khiển nút gửi, để nút gửi không làm tràn vùng đệm của nút nhận bởi truyền quá nhiều và quá nhanh.

ĐIỀU KHIỂN LUỒNG CỦA TCP

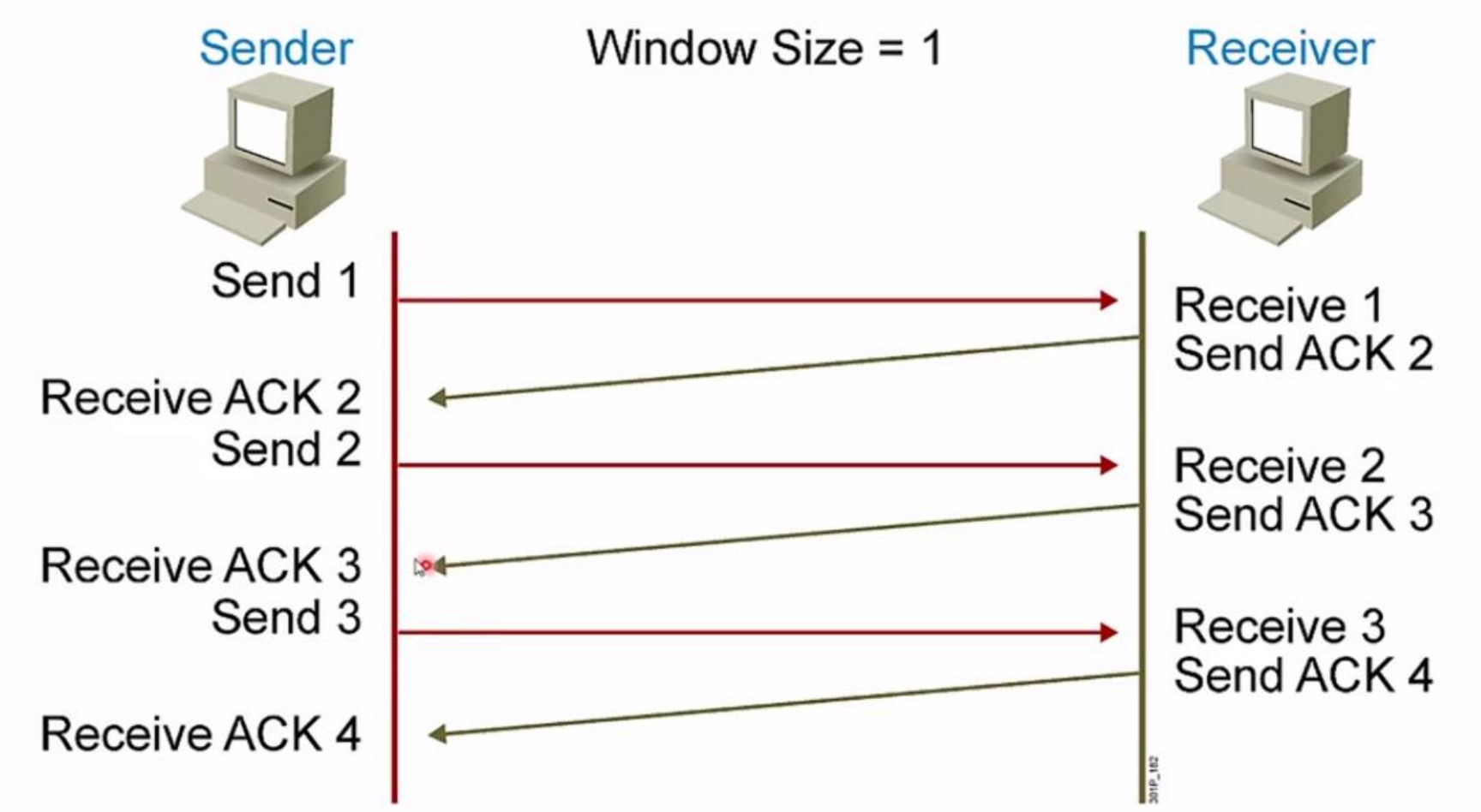
- Nút nhận thông báo không gian vùng đệm còn trống bằng cách đưa giá trị **rwnd (receive window)** trong TCP header của segment gửi từ nút nhận tới nút gửi.
  - **Kích thước của RcvBuffer** được đặt thông qua tùy chọn của socket (thường mặc định 4096 byte).
  - Nhiều hệ điều hành tự động điều chỉnh **RcvBuffer.**



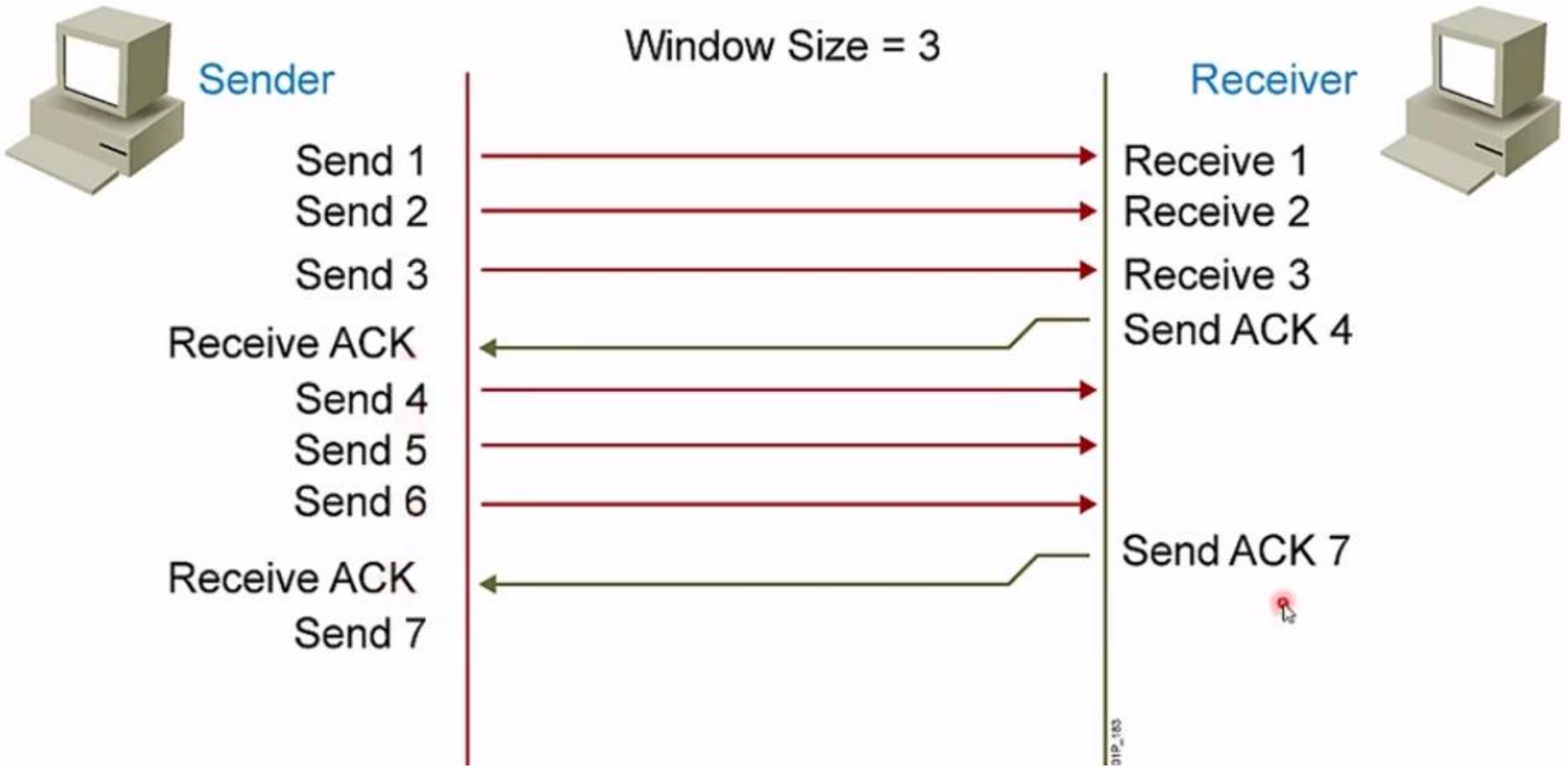
Bộ đệm phía nút nhận

- Nút gửi giới hạn dữ liệu chưa được ack bằng giá trị **rwnd** của nút nhận
- Đảm bảo vùng đệm nhận không bị tràn.

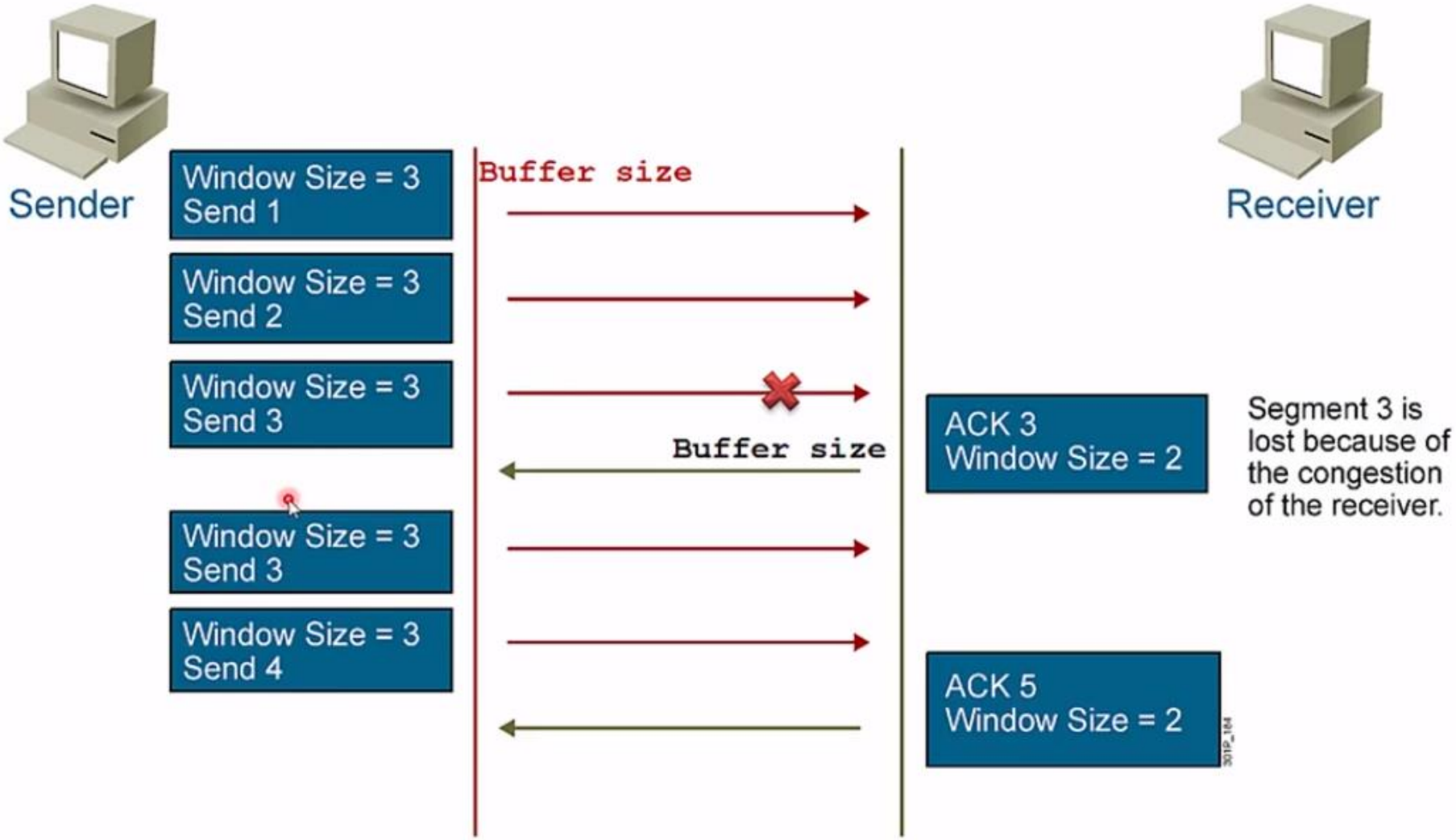
ĐIỀU KHIỂN LUỒNG CỦA TCP



ĐIỀU KHIỂN LUỒNG CỦA TCP



ĐIỀU KHIỂN LUỒNG CỦA TCP

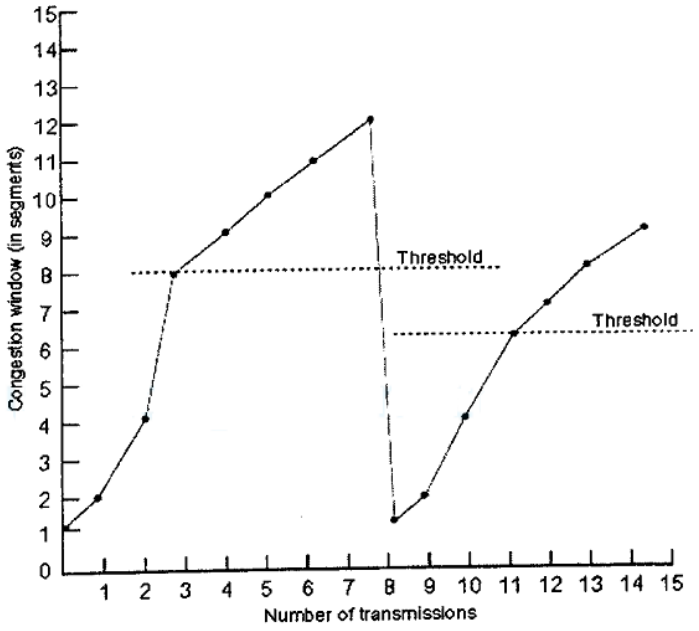


ĐIỀU KHIỂN TẮC NGHẼN

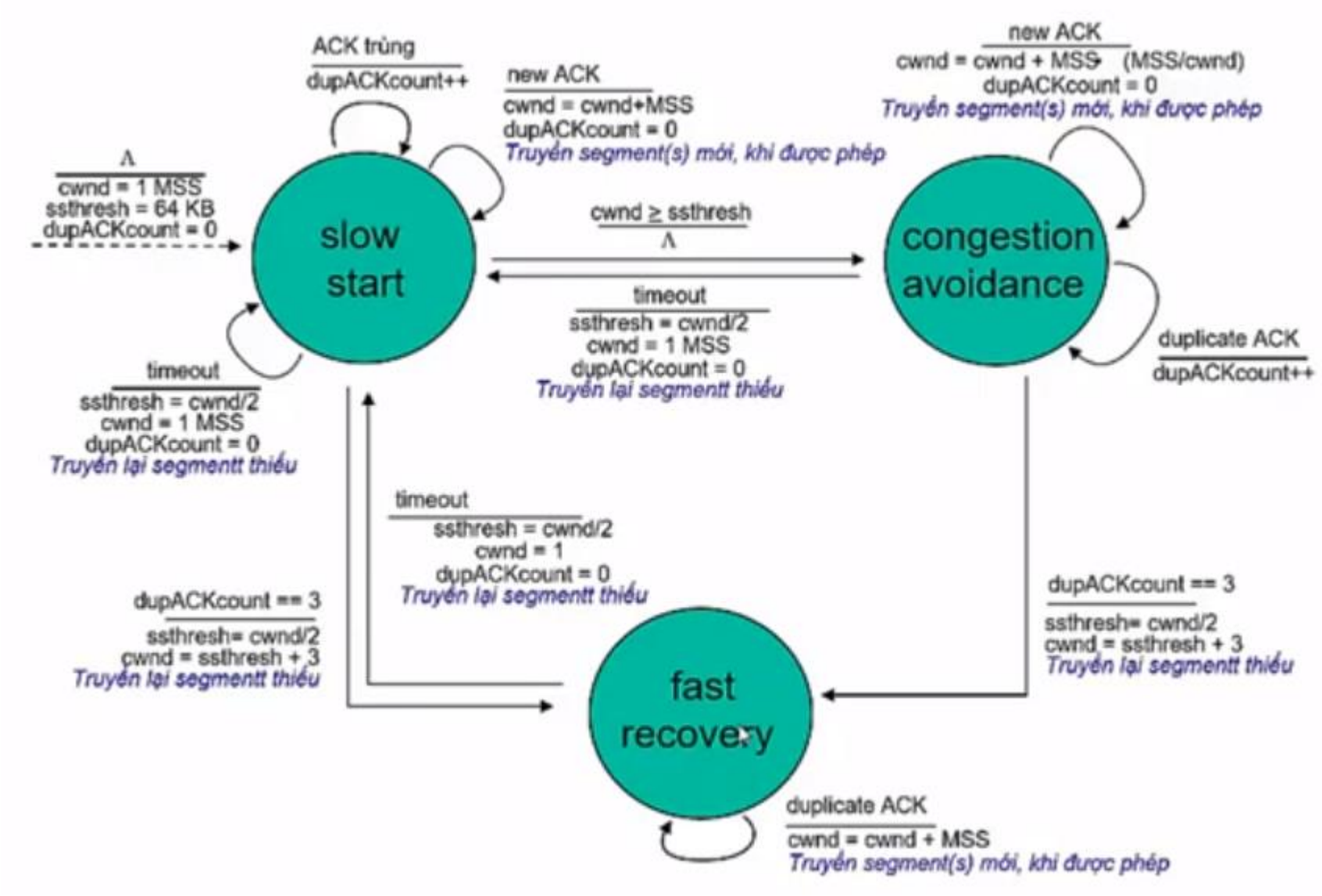
- *Khi nào tắc nghẽn* xảy ra?
  - Quá nhiều cặp gửi - nhận trên mạng
  - Truyền quá nhiều làm cho mạng quá tải
  - Khác với điều khiển luồng!
- Hậu quả của việc nghẽn mạng:
  - Mất gói tin (tràn vùng đệm tại router)
  - Độ trễ lớn (đợi trong vùng đệm của router)
- Bổ sung thêm thông tin cho điều khiển tắc nghẽn
  - Congestion window (cwnd): Số lượng dữ liệu tối đa mà người gửi có thể gửi qua kết nối.
  - Threshold (ngưỡng)

NGUYÊN TẮC ĐIỀU KHIỂN TẮC NGHẼN

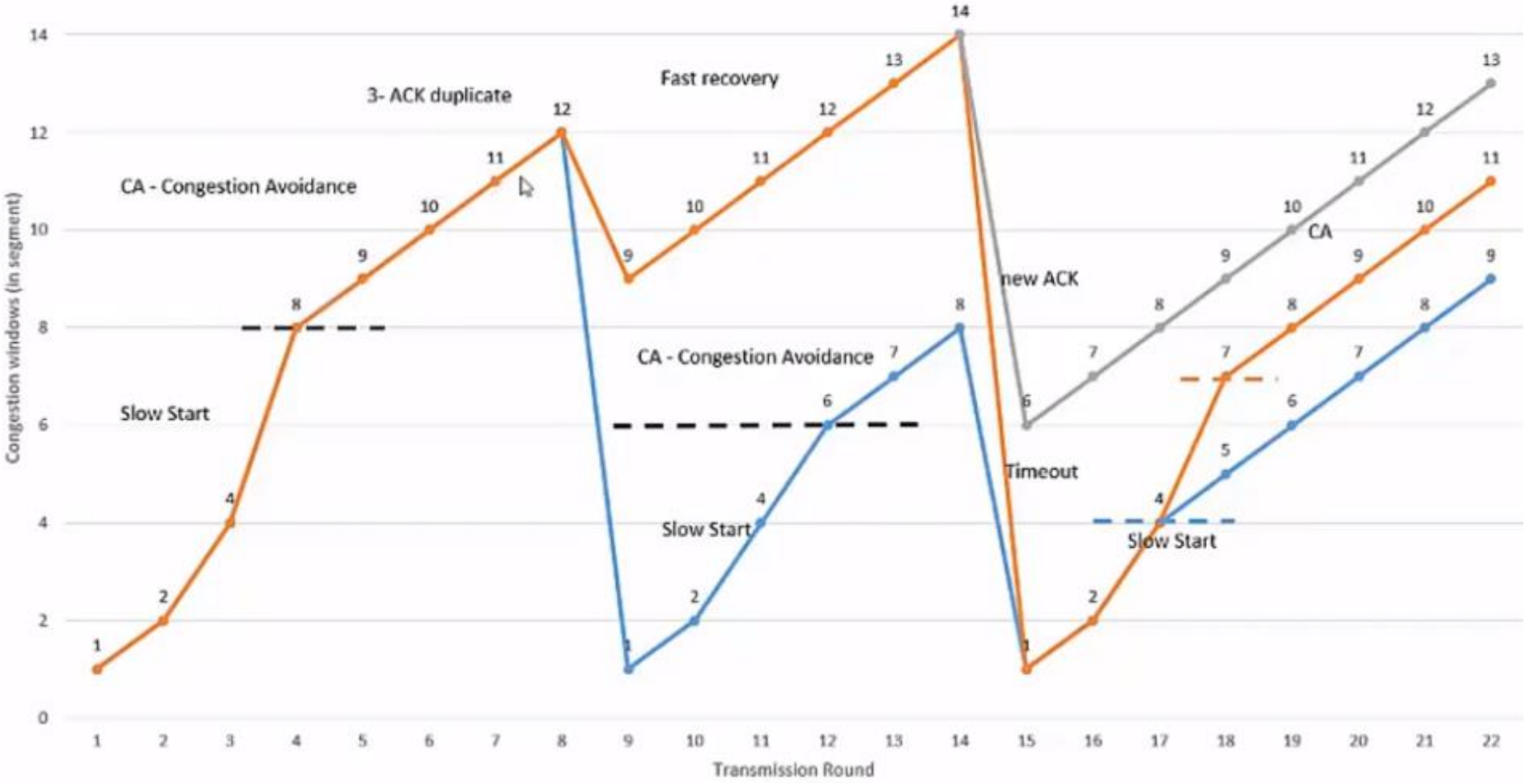
- Tăng dần tốc độ (Slow - Start)
  - Tăng tốc độ theo hàm số mũ
  - Tiếp tục tăng đến một ngưỡng nào đó
- Tránh tắc nghẽn (Congestion Avoidance)
  - Tăng dần tốc độ theo hàm tuyến tính cho đến khi phát hiện tắc nghẽn
- Khôi phục tắc nghẽn nhanh (Fast Recovery)
  - Nếu gói tin bị mất



ĐIỀU KHIỂN TẮC NGHẼN



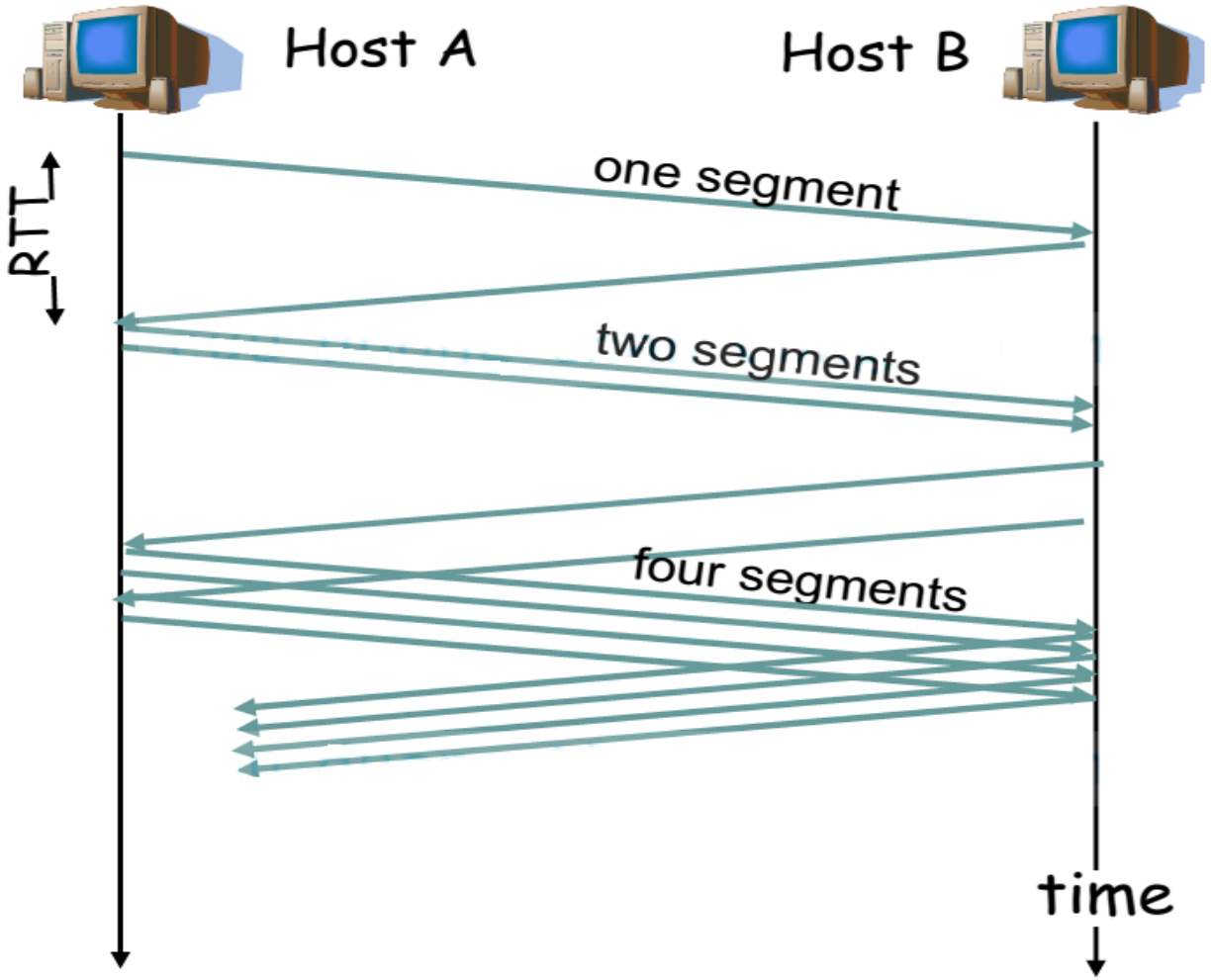
ĐIỀU KHIỂN TẮC NGHẼN



TCP SLOW-START

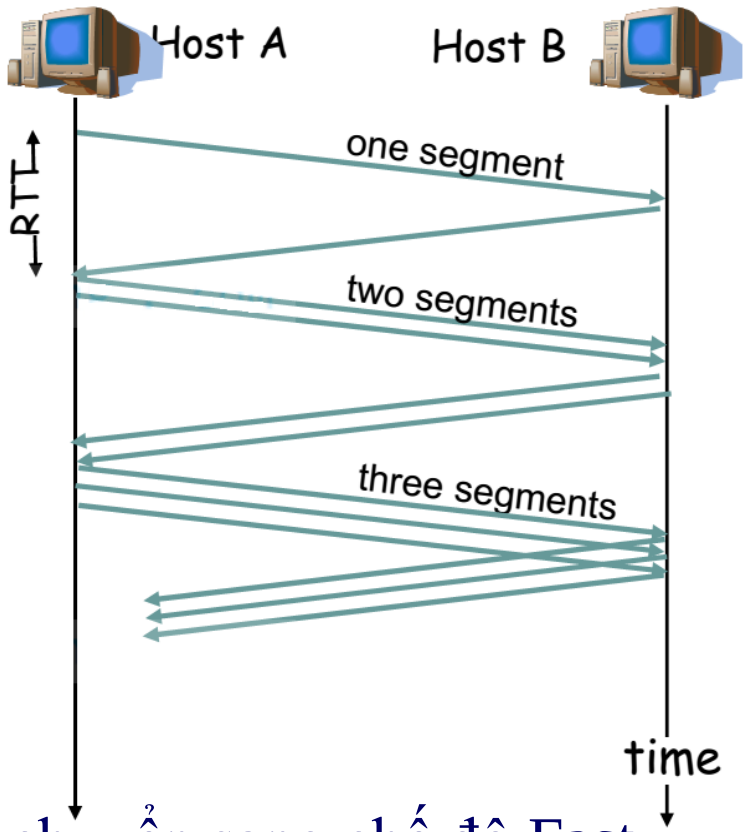
- Ý tưởng cơ bản
  - Đặt cwnd bằng 1 MSS (Maximum segment size)
  - Tăng cwnd lên gấp đôi
    - Khi nhận được ACK
  - Bắt đầu chậm, nhưng tăng theo hàm mũ
- Tăng cho đến một ngưỡng: threshold
  - Sau đó, TCP chuyển sang trạng thái tránh tắc nghẽn (congestion avoidance)

TCP SLOW-START: MINH HỌA



TCP CONGESTION AVOIDANCE

- Ý tưởng cơ bản
  - Đặt cwnd theo cấp số cộng sau khi nó đạt tới threshold
  - Khi bên gửi nhận được ACK
    - Tăng cwnd thêm 1 MSS (Maximum segment size)



- Khi ACK trùng 3 lần thì thực hiện chuyển sang chế độ Fast Recovery
- Khi timeout thì chuyển sang chế độ Slow Start

TCP FAST RECOVERY

- Giảm tốc độ gửi bằng cách cho Threshold = cwnd/2 và cwnd mới = Threshold + 3.
- Khi nhận được ACK thì tăng cwnd lên 1 MSS
- Phát hiện tắc nghẽn?
  - Nếu như phải truyền lại -> Có thể suy ra là mạng "tắc nghẽn"
- Khi nào thì phải truyền lại?
  - Timeout!
  - Cùng một số hiệu gói tin trong ACK

TCP FAST RECOVERY (TIẾP)

- Khi có timeout của bên gửi
  - TCP đặt ngưỡng threshold = cwnd/2
  - TCP đặt cwnd về 1 MSS
  - TCP chuyển về slow-start
- Nếu nhận được 3 ACK giống nhau
  - TCP đặt ngưỡng threshold = cwnd/2
  - TCP đặt cwnd về giá trị hiện tại của ngưỡng cũ
  - TCP chuyển trạng thái tránh tắc nghẽn (congestion avoidance)

Quan sát giao thức TCP - UDP

1. Chạy phần mềm Wireshark để ghi lại dữ liệu gửi nhận (Chọn Capture -> Start)
2. Mở trình duyệt vào trang web: tlu.edu.vn
3. Dừng việc ghi dữ liệu của phần mềm Wireshark
4. Gõ Filter: tcp.port == 80 và udp.port == 80
5. Quan sát các gói tin TCP và UDP