ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Chương 5: Tầng giao vận

Đọc trước: Chapter 5-Computer Networks, Tanenbaum

Tổng quan

- 3 tuần trước : Giao thức IP
  - Địa chỉ, gói tin IP
  - ICMP
  - Chọn đường
- Hôm nay: Tầng giao vận
  - Nguyên lý tầng giao vận
  - Giao thức UDP
  - Giao thức TCP

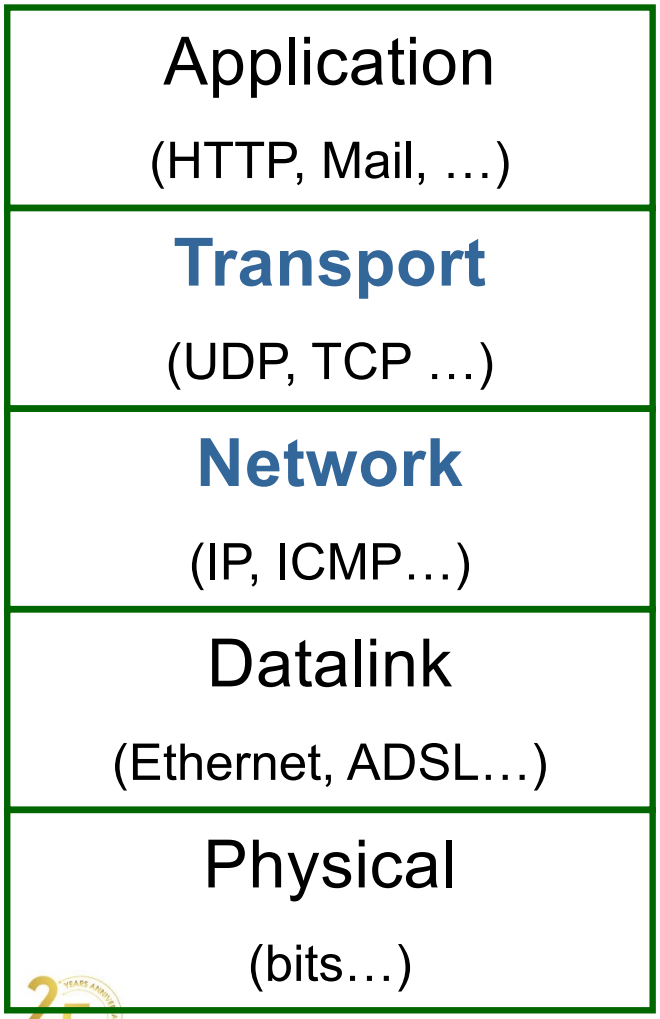
Các khái niệm cơ bản

Vị trí trong kiến trúc phân tầng

Hướng liên kết vs. Không liên kết

UDP & TCP

Vị trí trong kiến trúc phân tầng



Hỗ trợ các ứng dụng trên mạng

Truyền dữ liệu giữa các ứng dụng

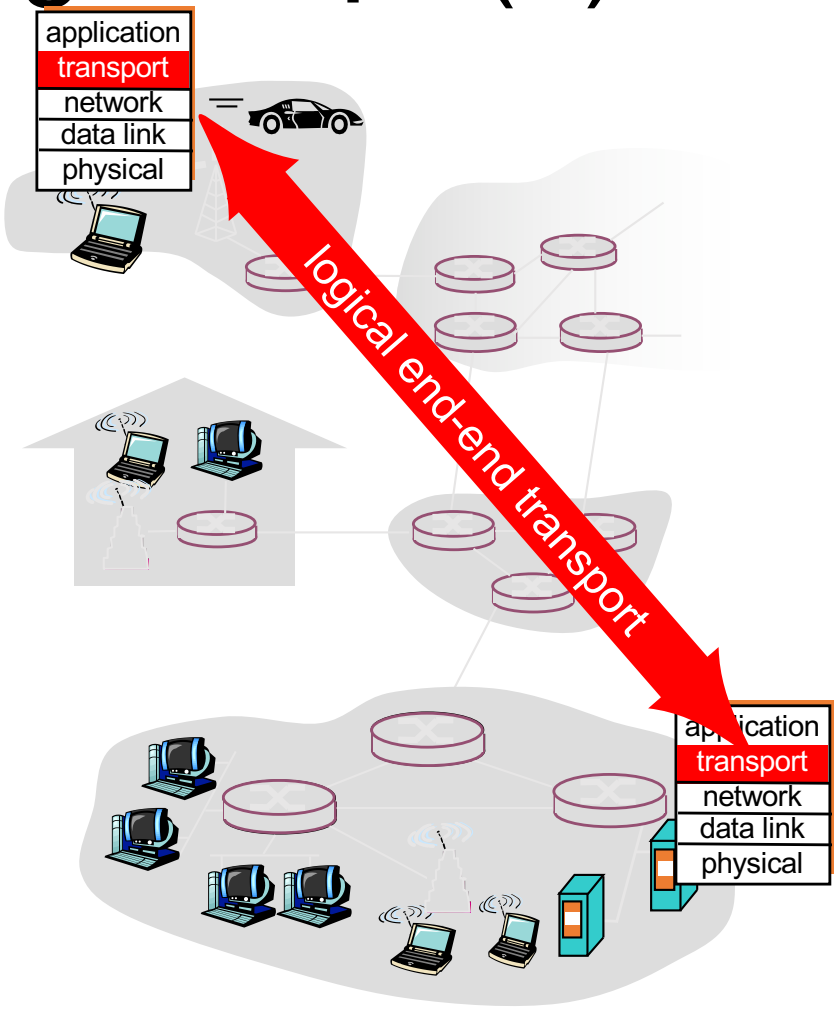
Chọn đường và chuyển tiếp gói tin giữa các máy, các mạng

Hỗ trợ việc truyền thông cho các thành phần kế tiếp trên cùng 1 mạng

Truyền và nhận dòng bit trên đường truyền vật lý

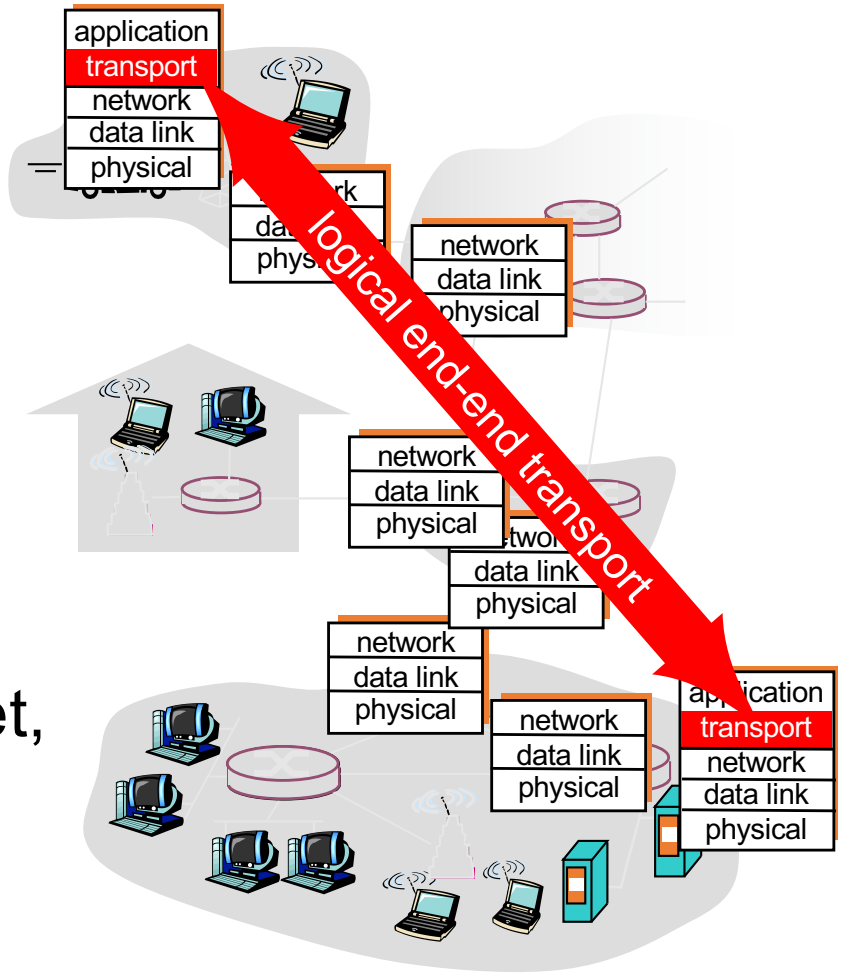
Tổng quan về tầng giao vận (1)

- Cung cấp phương tiện truyền giữa các ứng dụng cuối
  - Các ứng dụng là các tiến trình chạy trên các máy.
- Bên gửi:
  - Nhận dữ liệu từ ứng dụng
  - Đặt dữ liệu vào các đoạn tin và chuyển cho tầng mạng
  - Nếu dữ liệu quá lớn, nó sẽ được chia làm nhiều phần và đặt vào nhiều đoạn tin khác nhau
- Bên nhận:
  - Nhận các đoạn tin từ tầng mạng
  - Tập hợp dữ liệu và chuyển lên cho ứng dụng



Tổng quan về tầng giao vận (2)

- Được cài đặt trên các hệ thống cuối
  - Không cài đặt trên các routers, switches…
- Hai dạng dịch vụ giao vận
  - Tin cậy, hướng liên kết, e.g TCP
  - Không tin cậy, không liên kết, e.g. UDP



Tại sao lại cần 2 loại dịch vụ?

- Các yêu cầu đến từ tầng ứng dụng là đa dạng
- Các ứng dụng cần dịch vụ với 100% độ tin cậy như mail, web…
  - Sử dụng dịch vụ của TCP
- Các ứng dụng cần chuyển dữ liệu nhanh, có khả năng chịu lỗi, e.g. VoIP, Video Streaming
  - Sử dụng dịch vụ của UDP

Ứng dụng và dịch vụ giao vận

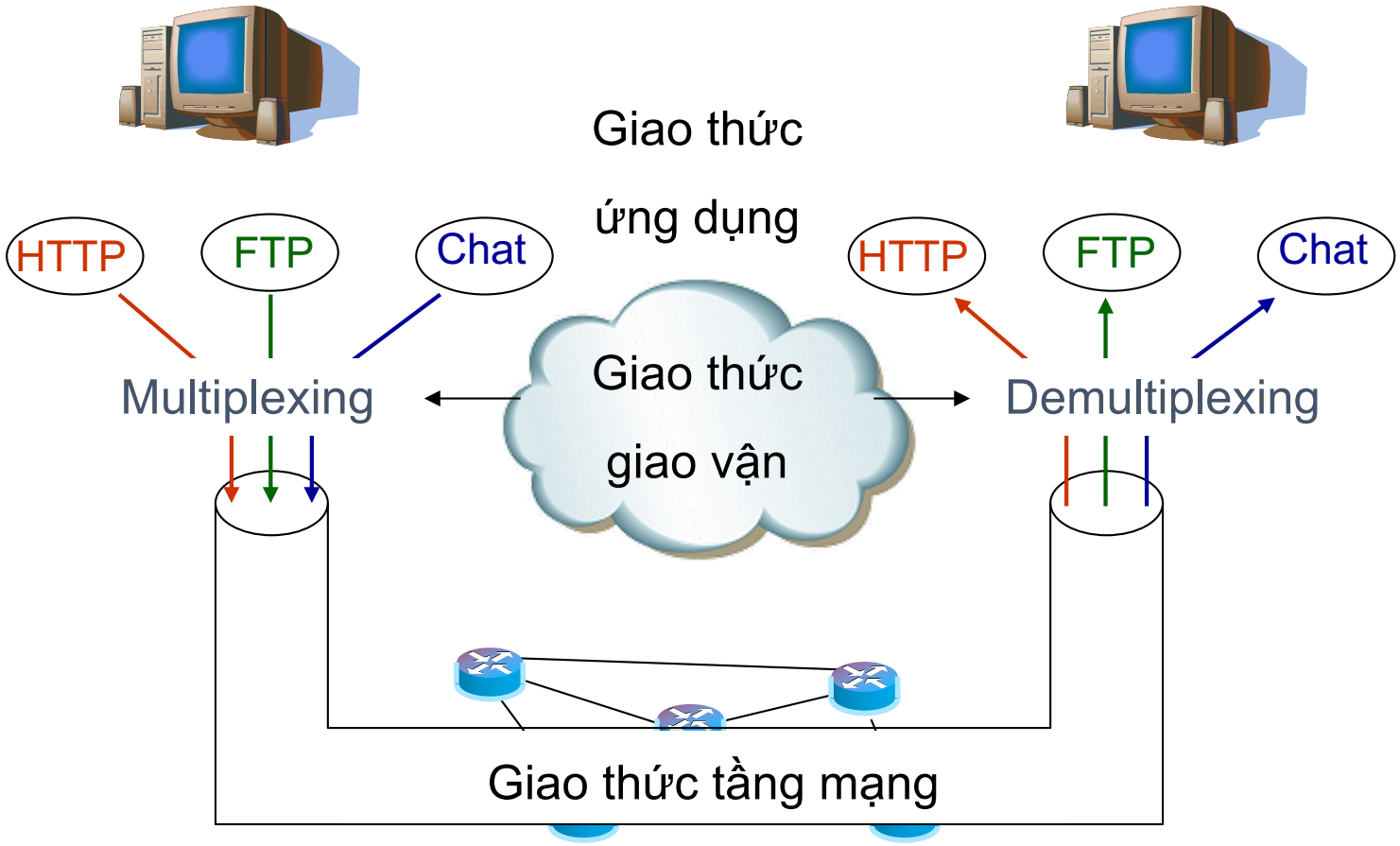
Ứng dụng	Giao thức ứng dụng	Giao thức giao vận
e-mail	SMTP	TCP
remote terminal access	Telnet	TCP
Web	HTTP	TCP
file transfer	FTP	TCP
streaming multimedia	giao thức riêng (e.g. RealNetworks)	TCP or UDP
Internet telephony	giao thức riêng (e.g., Vonage,Dialpad)	thường là UDP

Các chức năng chung

Dồn kênh/phân kênh

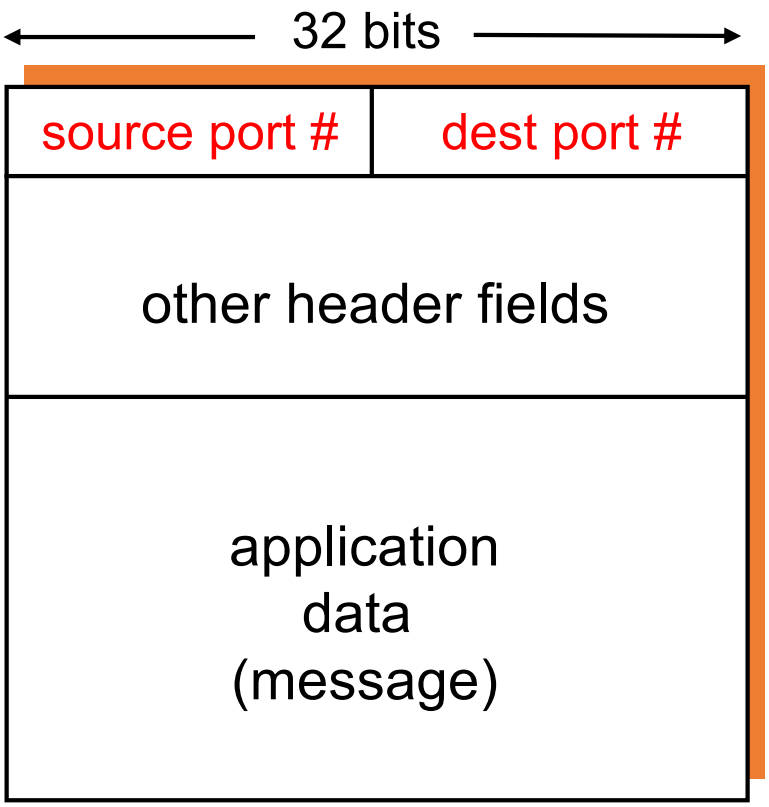
Kiểm soát lỗi

Dồn kênh/phân kênh - Mux/Demux



Mux/Demux hoạt động ntn?

- Tại tầng mạng, gói tin IP được định danh bởi địa chỉ IP
  - Để xác định máy trạm
- Làm thế nào để phân biệt các ứng dụng trên cùng một máy?
  - Sử dụng số hiệu cổng (16 bits)
  - Mỗi tiến trình ứng dụng được gán 1 cổng
- Socket: Một cặp địa chỉ IP và số hiệu cổng



TCP/UDP segment format

Kiểm soát lỗi

- Sử dụng CRC hoặc Checksum
- Checksum
  - Phát hIện lỗi bit trong các đoạn tin/gói tin
  - Nguyên lý giống như checksum (16 bits) của giao thức IP
- Nguyên lý checksum
  - Dữ liệu cần gửi được chia thành các đoạn bằng nhau
  - Các đoạn được tính tổng với nhau, nếu có nhớ thì cộng giá trị nhớ vào tổng
  - Đảo tổng thu được checksum

Checksum: Ví dụ

Sender's End

Frame 1:	11001100
Frame 2:	+ 10101010
Partial Sum:	1 01110110
	+ 1
	01110111
Frame 3:	+ 11110000
Partial Sum:	1 01100111
	+ 1
	01101000
Frame 4:	+ 11000011
Partial Sum:	1 00101011
	+ 1
Sum:	00101100
Checksum:	11010011

Receiver's End

Frame 1:	11001100
Frame 2:	+ 10101010
Partial Sum:	1 01110110
	+ 1
	01110111
Frame 3:	+ 11110000
Partial Sum:	1 01100111
	+ 1
	01101000
Frame 4:	+ 11000011
Partial Sum:	1 00101011
	+ 1
Sum:	00101100
Checksum:	11010011
Sum:	11111111
Complement:	00000000

Hence accept frames.

UDP
User Datagram Protocol

Tổng quan

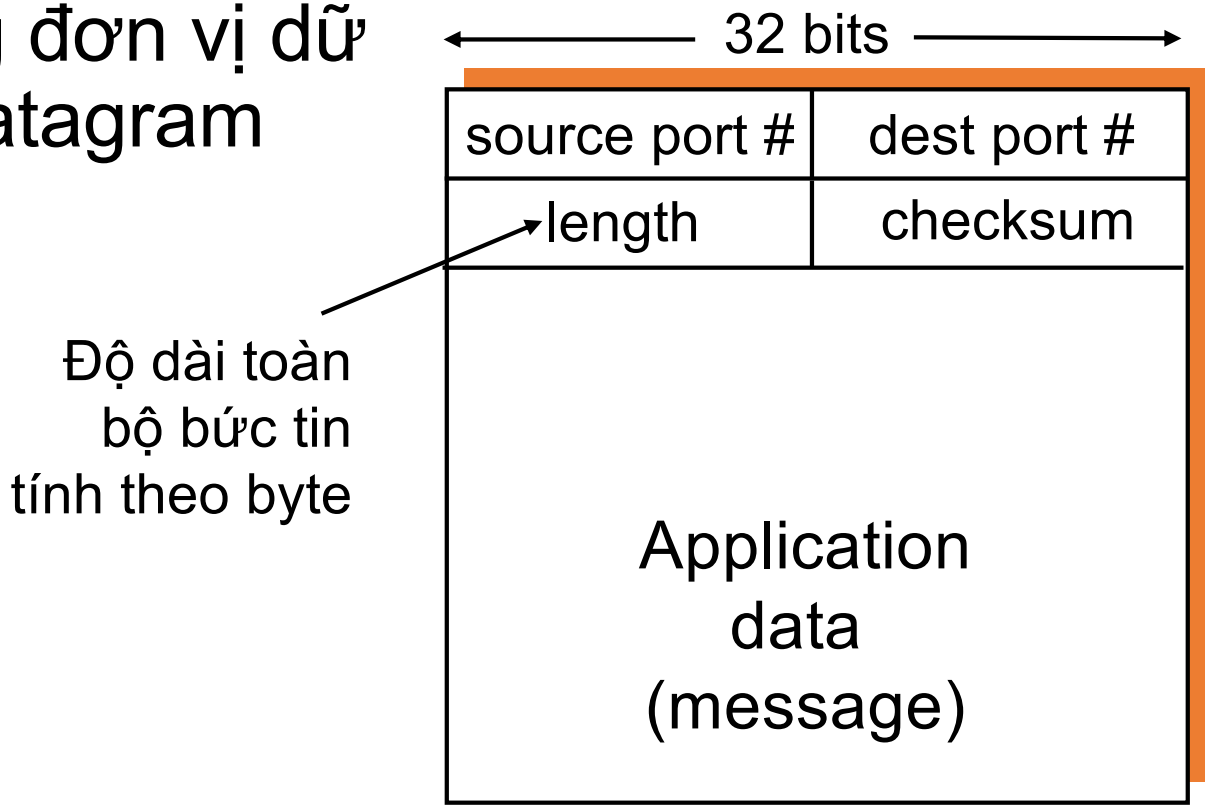
Khuôn dạng gói tin

Giao thức dạng "Best effort"

- Dùng UDP khi nào?
  - Không cần thiết lập liên kết
  - Đơn giản: Không cần lưu lại trạng thái liên kết ở bên gửi và bên nhận
  - Phần đầu đoạn tin nhỏ
  - Không có quản lý tắc nghẽn: UDP cứ gửi dữ liệu nhanh nhất, nhiều nhất nếu có thể
- UDP có những chức năng cơ bản gì?
  - Dồn kênh/phân kênh
  - Phát hiện lỗi bit bằng checksum

Khuôn dạng bức tin (datagram)

- UDP sử dụng đơn vị dữ liệu gọi là –datagram (bức tin)



Khuôn dạng đơn vị dữ liệu của UDP

Các vấn đề của UDP

- Không có kiểm soát tắc nghẽn
  - Làm Internet bị quá tải
- Không bảo đảm được 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

TCP
Transmission Control Protocol

Cấu trúc đoạn tin TCP

Quản lý liên kết

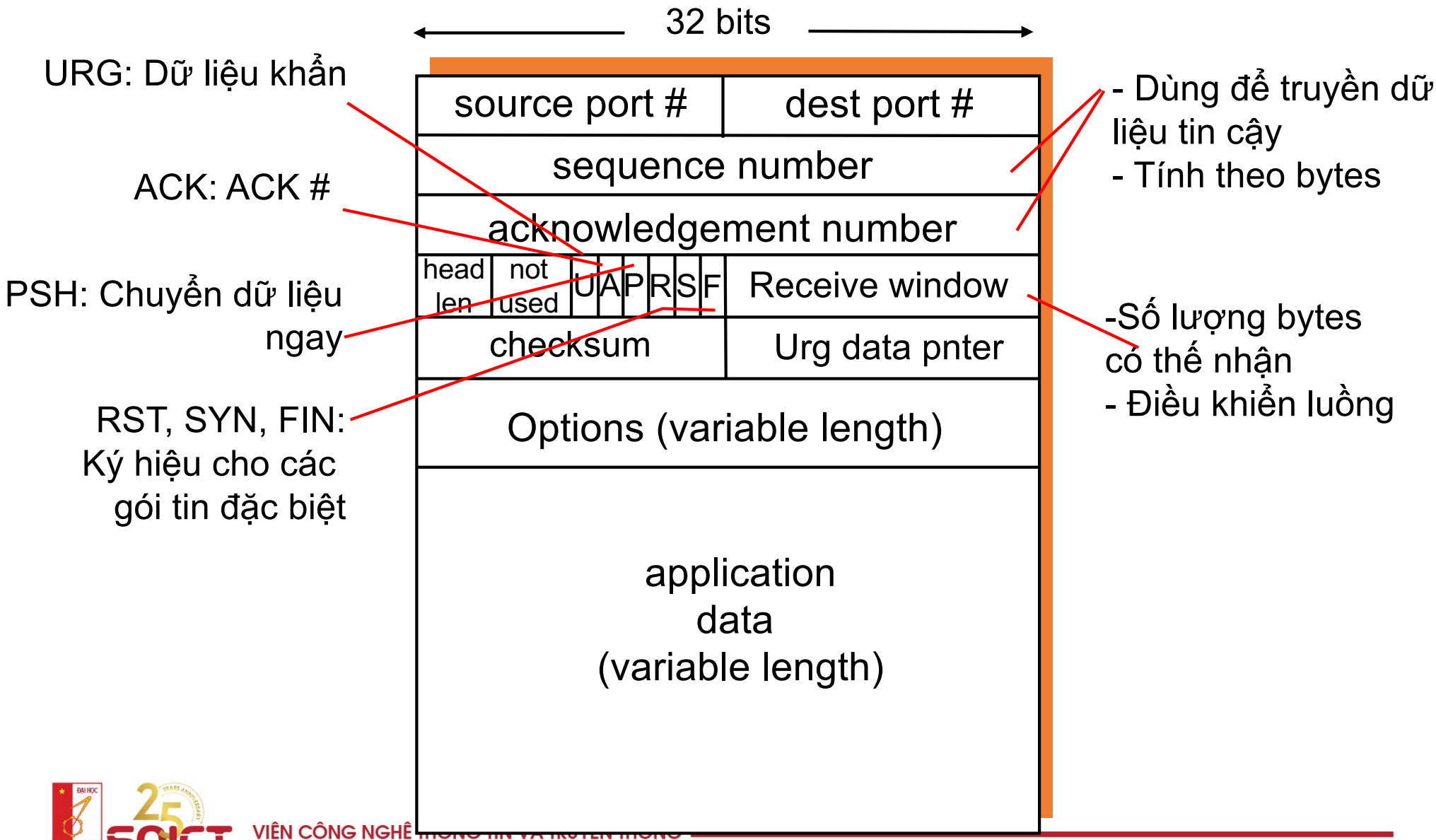
Kiểm soát luồng

Kiểm soát tắc nghẽn

Tổng quan về TCP

- Giao thức hướng liên kết
  - Bắt tay ba bước
- Giao thức truyền dữ liệu theo dòng byte, tin cậy
  - Sử dụng vùng đệm
- Truyền theo kiểu pipeline, sử dụng sliding windows.
  - Tăng hiệu quả
- Kiểm soát luồng
  - Bên gửi không làm quá tải bên nhận (thực tế: quá tải)
- Kiểm soát tắc nghẽn
  - Việc truyền dữ liệu không nên làm tắc nghẽn mạng (thực tế: luôn có tắc nghẽn)

Khuôn dạng đoạn tin - TCP segment



TCP cung cấp dịch vụ tin cậy ntn?

- Kiểm soát dữ liệu đã được nhận chưa:
  - Seq. #
  - Ack
- Chu trình làm việc của TCP:
  - Thiết lập liên kết
    - Bắt tay ba bước
  - Truyền/nhận dữ liệu
  - Đóng liên kết

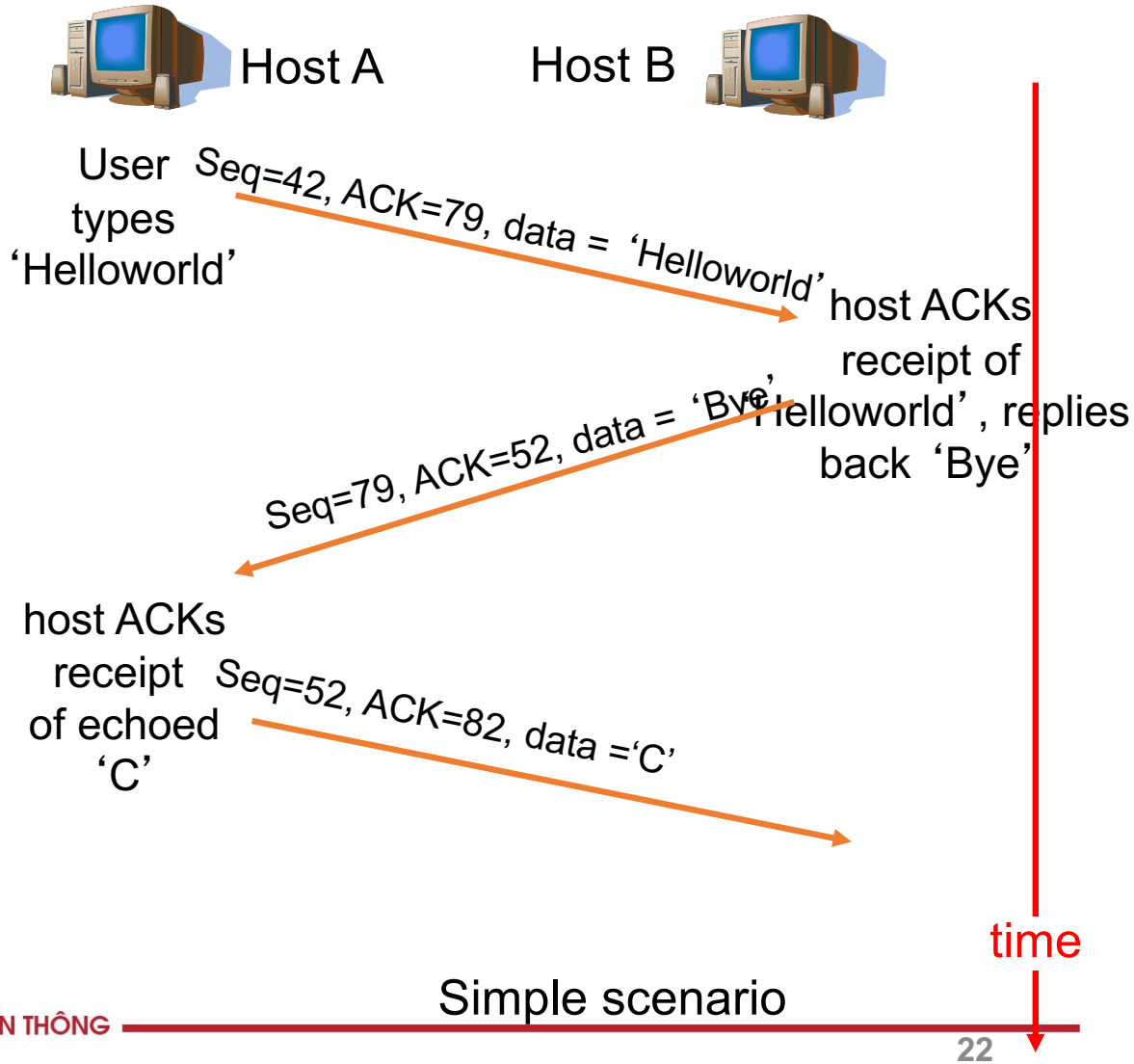
Cơ chế báo nhận trong TCP

Seq. #:

- Số hiệu của byte đầu tiên của gói tin trong dòng dữ liệu

ACK:

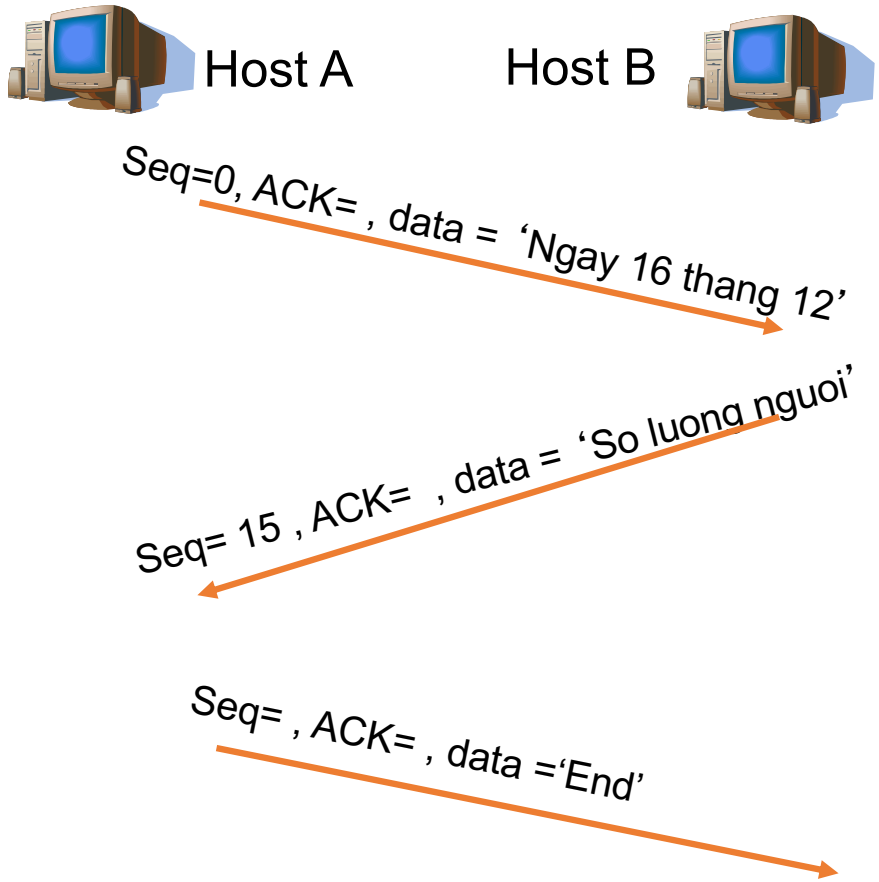
- Số hiệu byte mong muốn nhận từ đối tác
- Ngầm xác nhận đã nhận tốt các byte trước đó.



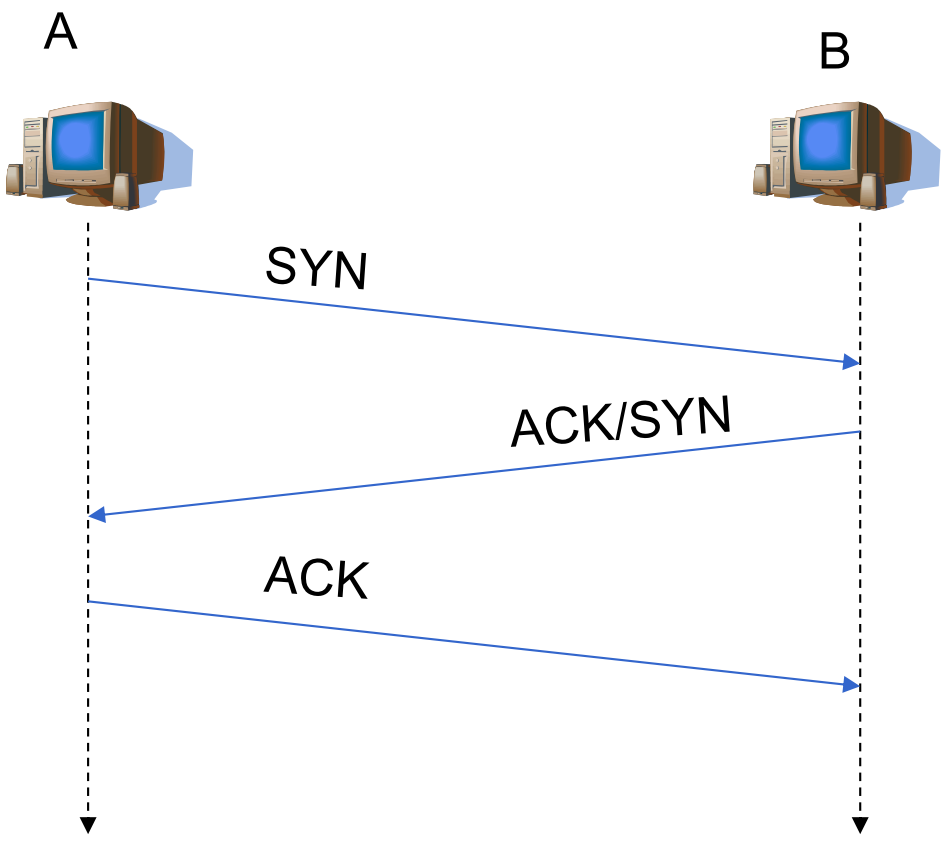
Simple scenario

Bài tập

- Nếu quá trình truyền dữ liệu diễn ra không có sự cố, xác định các trường SEQ, ACK còn thiếu trên gói tin



Thiết lập liên kết TCP : Giao thức bắt tay 3 bước

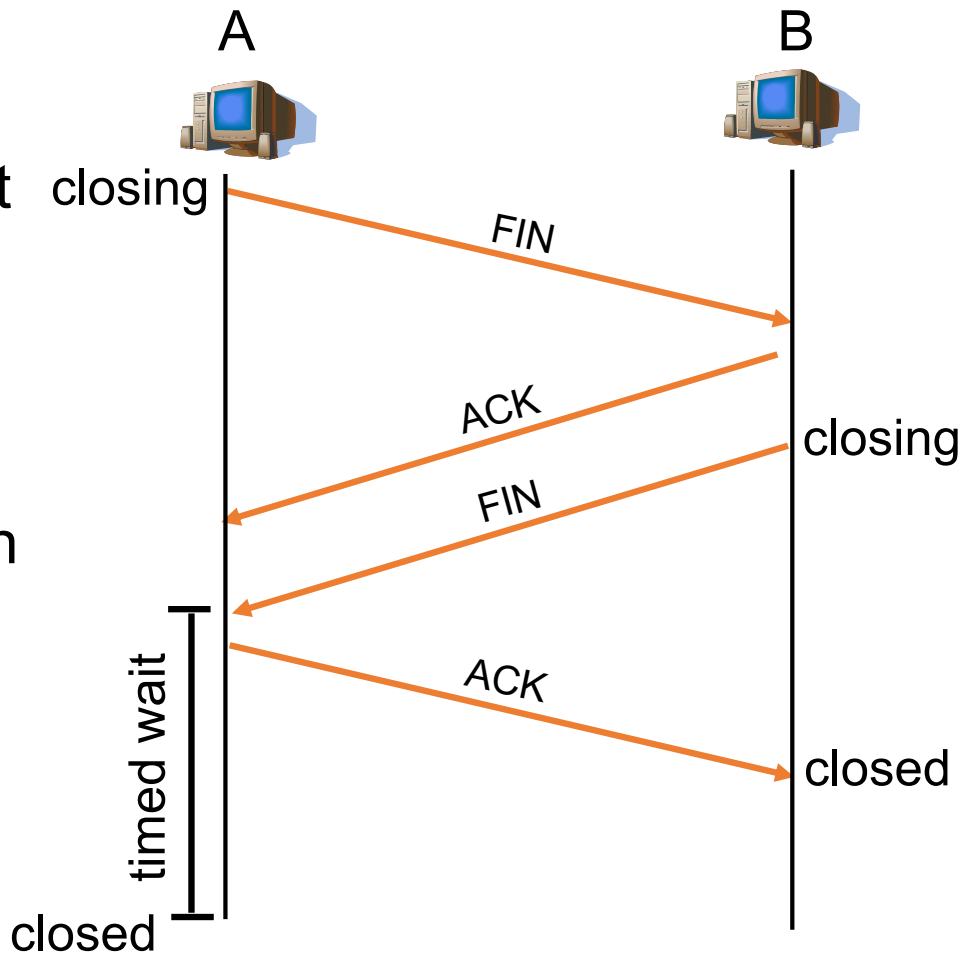


- Bước 1: A gửi SYN cho B
  - chỉ ra giá trị khởi tạo seq # của A
  - không có dữ liệu
- Bước 2: B nhận SYN, trả lời bằng SYNACK
  - B khởi tạo vùng đệm
  - chỉ ra giá trị khởi tạo seq. # của B
- Bước 3: A nhận SYNACK, trả lời ACK, có thể kèm theo dữ liệu

Ví dụ về việc đóng liên kết

- Bước 1: 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



Kiểm soát luồng

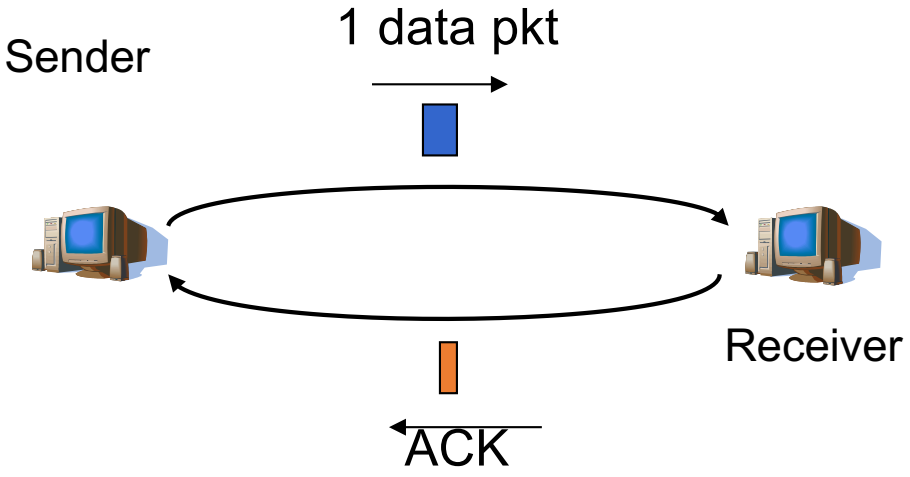
ARQ

Stop-and-wait

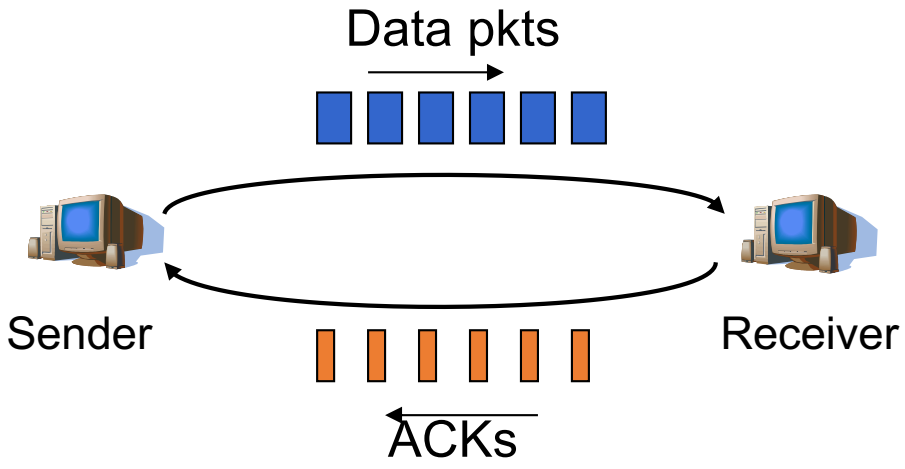
Sliding windows

Cơ chế kiểm soát luồng

- Dùng cơ chế như ở tầng 2:
  - Stop-and-wait
  - Pipeline



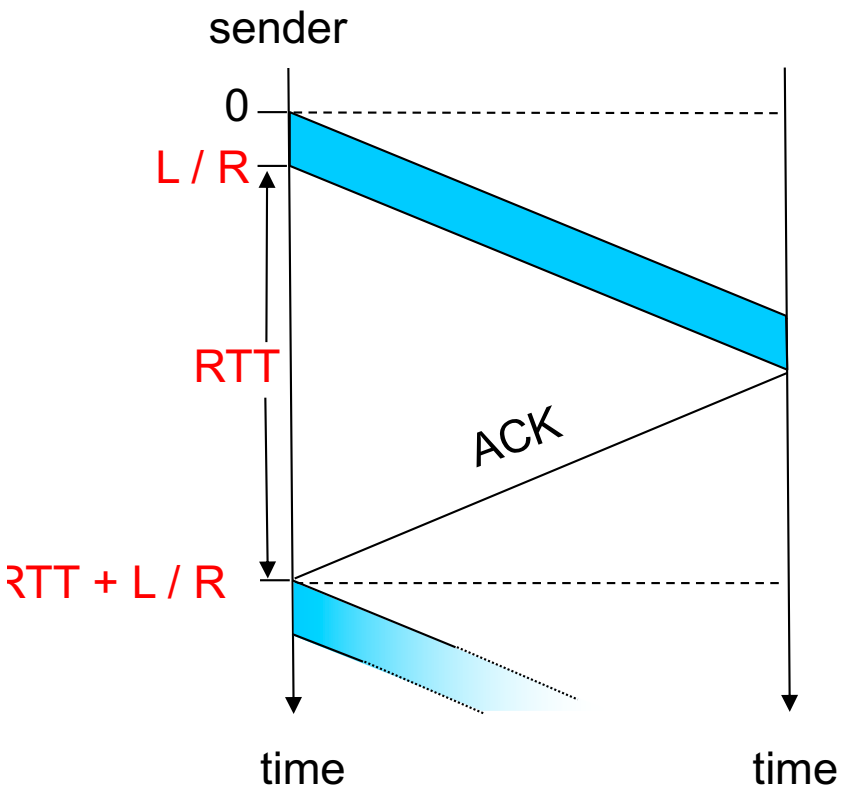
Stop-and-wait



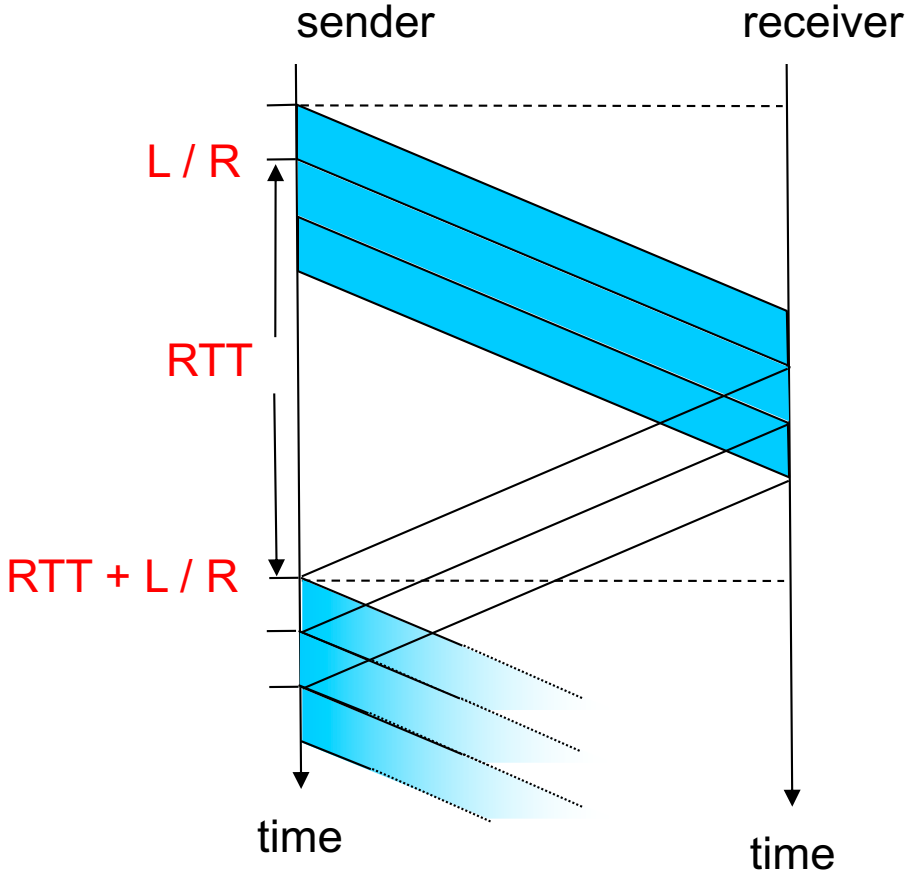
Pipeline

So sánh hiệu quả

stop-and-wait



Pipeline



L: Size of data pkt
R: Link bandwidth
RTT: Round trip time

$$\text{Performance} = \frac{L / R}{RTT + L / R}$$

$$\text{Performance} = \frac{3 * L / R}{RTT + L / R}$$

Kiểm soát luồng trong TCP

- Điều khiển lượng dữ liệu được gửi đi
  - Bảo đảm rằng hiệu quả là tốt
  - Không làm quá tải các bên
- Sử dụng cơ chế cửa sổ trượt.
- Các bên sẽ có cửa sổ kiểm soát
  - Rwnd: Cửa sổ nhận
  - CWnd: Cửa sổ kiểm soát tắc nghẽn
- Lượng dữ liệu gửi đi phải nhỏ hơn min(Rwnd, Cwnd)

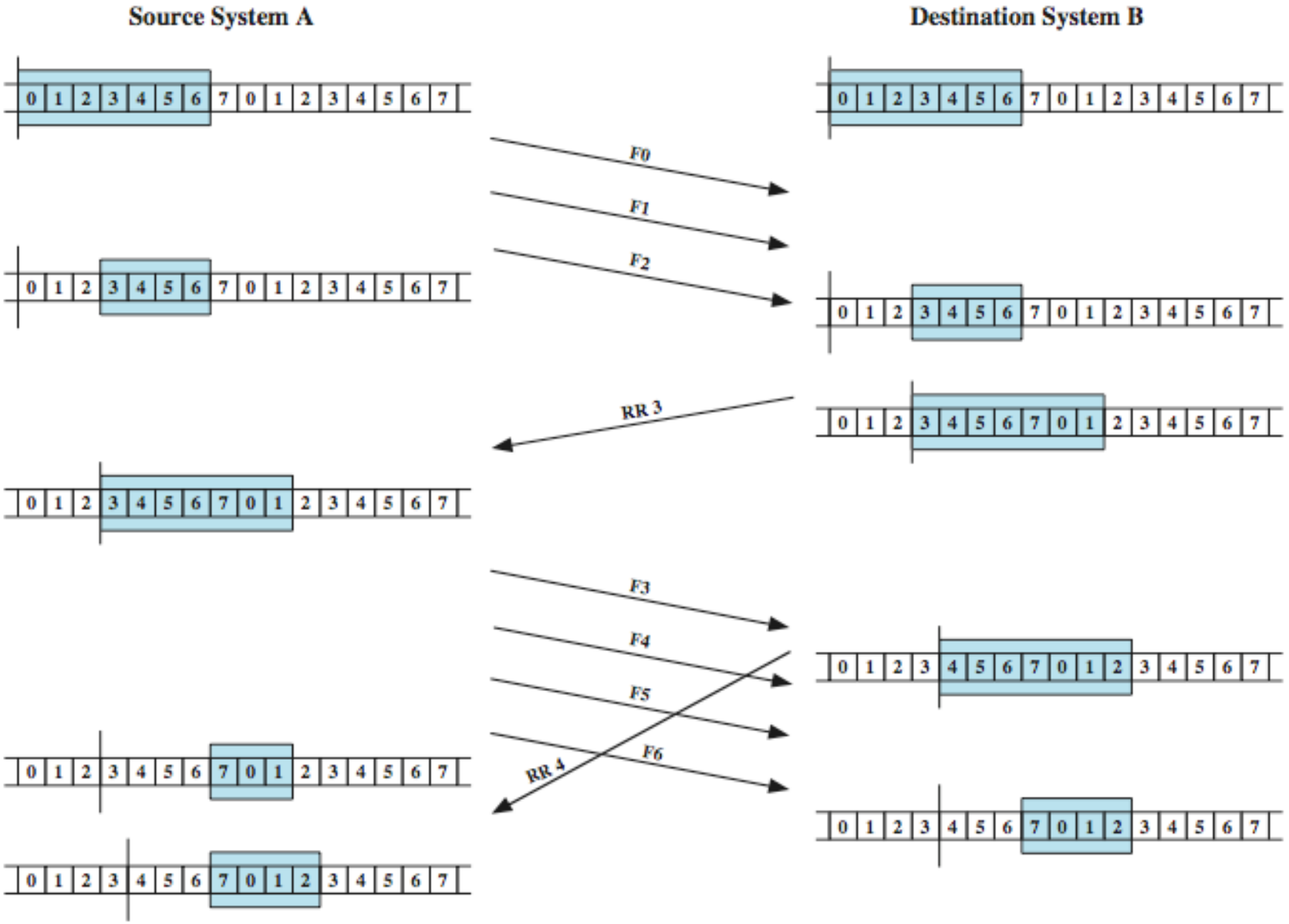
Cơ chế cửa sổ trượt: nguyên tắc

- Bên nhận có vùng đệm lưu các gói tin nhận được mà chưa được xử lý.
- Cửa sổ nhận: kích thước vùng đệm còn trống ở bên nhận.
- Bên gửi: Gửi số gói tin dữ liệu bằng kích thước cửa sổ nhận (vùng đệm còn trống của bên nhận) mà không cần dừng để giảm thời gian chờ.
- Các gói tin đã gửi đi chưa báo nhận được lưu trữ tạm thời trong bộ nhớ đệm bên gửi để phòng phải gửi lại
- Khi nhận được báo nhận
  - giải phóng gói tin dữ liệu đã truyền thành công khỏi bộ nhớ đệm
  - Truyền tiếp các gói tin bằng với số gói tin đã truyền thành công

Cơ chế cửa sổ trượt: Nguyên tắc

- Xét hai trạm A, B kết nối bằng một đường truyền song công
  - Bên A có vùng đệm chứa gói đã phát
  - Bên A có cửa sổ phát, phản ánh các gói được phép phát đi
  - B có vùng đệm chứa dữ liệu nhận được
  - Bên B có cửa sổ nhận, phản ánh số gói có thể nhận được.
  - Mỗi khi A phát một gói tin, kích thước cửa sổ phát giảm đi 1
  - Mỗi khi A nhận được một phản hồi, kích thước cửa sổ phát tăng lên 1.
  - Mỗi khi B nhận được 1 gói tin, kích thước cửa sổ nhận giảm đi 1.
  - Mỗi khi B báo nhận 1 gói tin, kích thước cửa sổ nhận tăng lên 1.
- Báo nhận
  - B báo nhận một gói bằng cách báo số gói dữ liệu mà B đang chờ nhận, ngầm định đã nhận tất cả các gói trước đó
  - Cơ chế báo nhận cho phép đồng bộ cửa sổ phát của A và cửa sổ nhận của B.

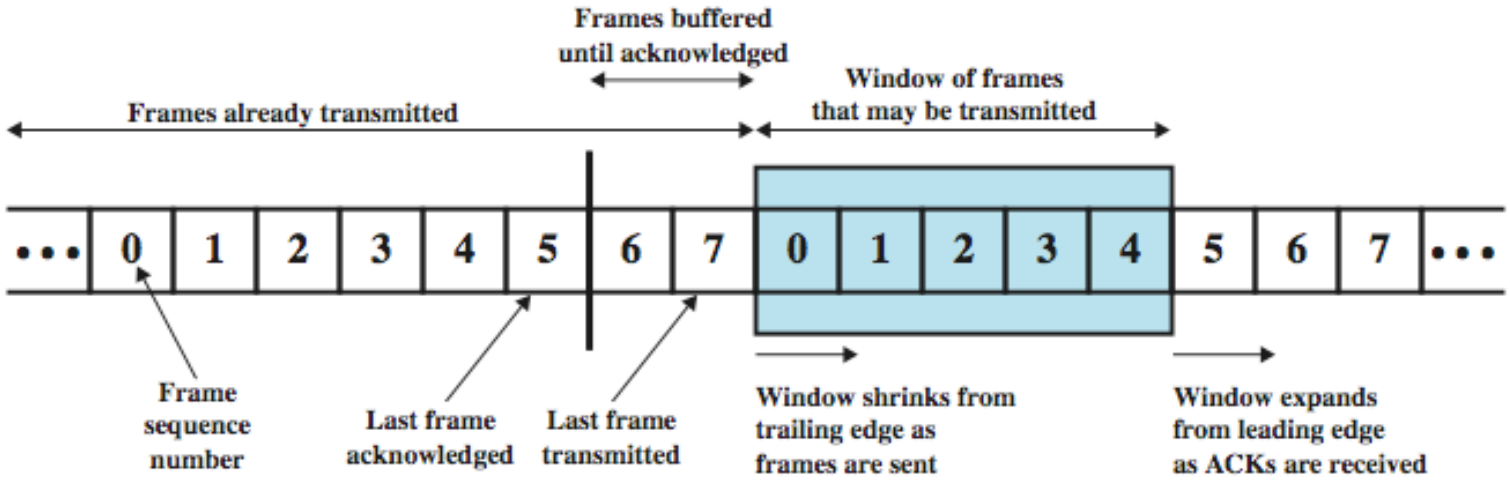
Cơ chế cửa sổ trượt



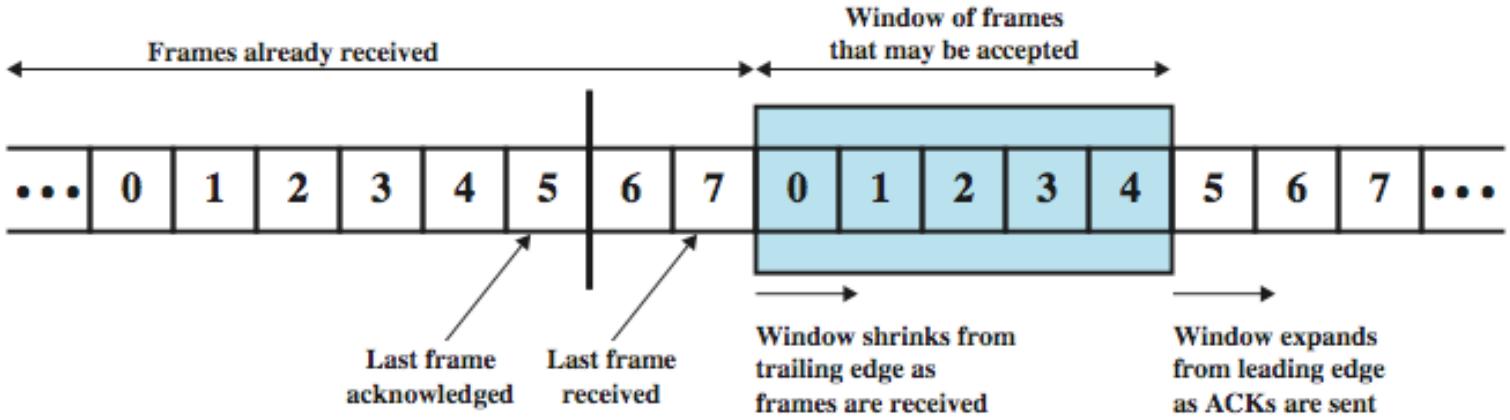
Trong cửa sổ là các gói sẽ phát

Trong cửa sổ là các gói chờ nhận

Cơ chế cửa sổ trượt



(a) Sender's perspective



(b) Receiver's perspective

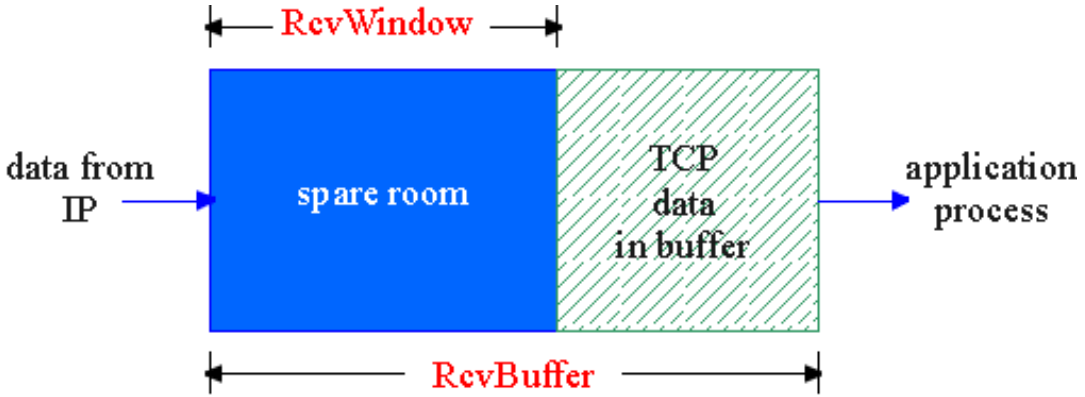
Cơ chế cửa sổ trượt

- Các khung dữ liệu đang được gửi đi được đánh số. Số thứ tự phải lớn hơn hoặc bằng kích thước cửa sổ
- Các khung dữ liệu được báo nhận bằng thông báo có đánh số
- Được báo gộp. Nếu 1,2,3,4 được nhận thành công, chỉ gửi báo nhận 4
- Khi đã nhận được thông báo nhận được khung k, có nghĩa là tất cả các khung k-1, k-2.. đã nhận được

Cơ chế cửa sổ trượt

- Nguồn quản lý
  - Các khung đã gửi đi thành công
  - Các khung đã gửi đi chưa báo nhận
  - Các khung có thể gửi đi ngay
  - Các khung chưa thể gửi đi ngay
- Đích quản lý
  - Các khung đã nhận được
  - Các khung đang chờ nhận

Kích thước cửa sổ

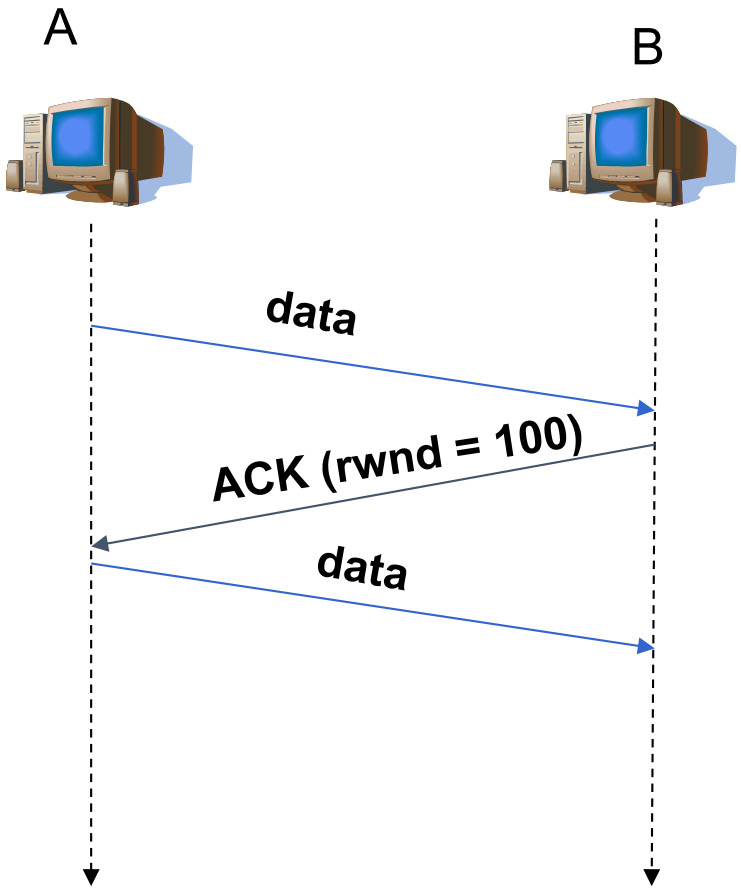


- Kích thước vùng đệm trống

= Rwnd

= **RcvBuffer-[LastByteRcvd - LastByteRead]**

Trao đổi thông tin về Rwnd



- Bên nhận sẽ báo cho bên gửi biết Rwnd trong các đoạn tin
- Bên gửi đặt kích thước cửa sổ gửi theo Rwnd

Bài tập

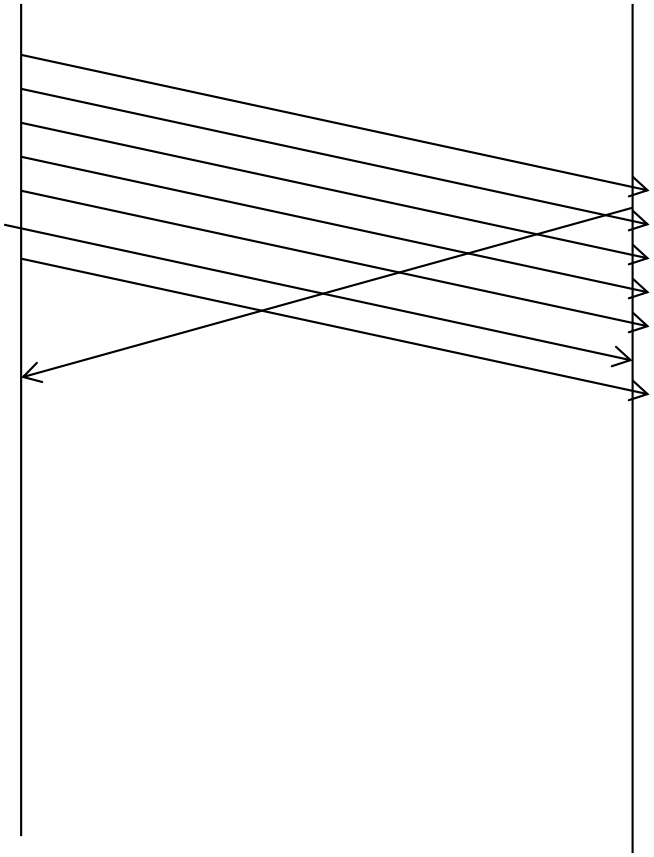
- Giả sử cần truyền 1 file
  - Kích thước *O* =100KB trên kết nối TCP
  - S là kích thước mỗi gói TCP, *S* = 536 byte, bỏ qua header. Bỏ qua kích thước ACK.
  - *RTT* = 3 *ms*.
- Giả sử vùng đệm bên nhận của TCP là cố định với kích thước W gói.

  1) Hỏi thời gian truyền hết dữ liệu với phương pháp Stop-and-wait ?

  2) Thời gian truyền với phương pháp cửa sổ trượt với kích thước cửa sổ =7 gói ?

  3)Hỏi kích thước cửa sổ nhận là bao nhiêu để thời gian chờ đợi ít nhất?
  - Nếu tốc độ đường truyền là R = 10 Mbit/s; R= 100 Mbit/s

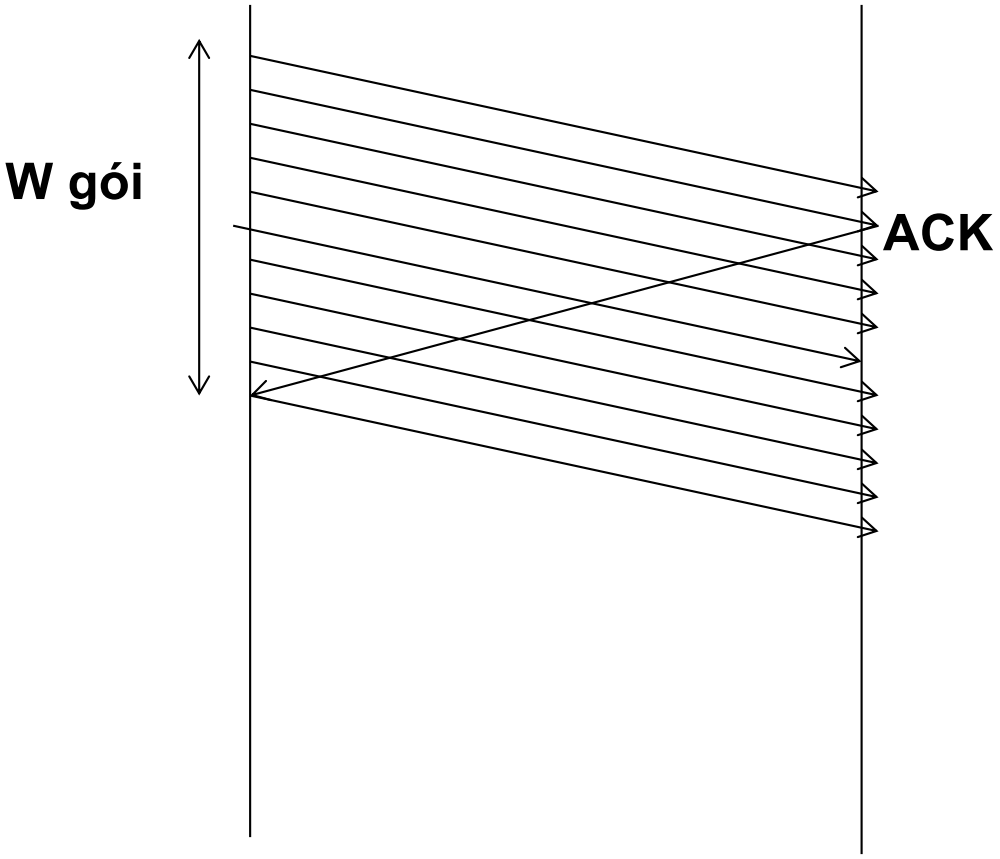
Cơ chế cửa sổ trượt



Thời gian truyền với cửa sổ 7

- T truyền nhanh nhất = (T phát 7 gói + chờ) *số lần.
- 1 lần Chờ = (T phát 1 gói + RTT) – T phát 7 gói
- Số lần chờ = số gói /7

Thời gian truyền nhanh nhất với cửa sổ trượt



- Truyền nhanh nhất đạt được khi nguồn phát xong gói cuối của cửa sổ thì đã nhận được ACK của gói đầu tiên.
- Kích thước cửa sổ W
- T phát (W gói) >= T phát gói đầu + RTT

Thời gian truyền nhanh nhất với cửa sổ trượt (cont.)

- T phát (W gói) = W * S/R
- Phát ngắn nhất khi không phải chờ:
- => (W-1)*S/R >= RTT
- => W >= RTT*R/S +1
- Thời gian phát hết dữ liệu L = L/R + RTT
- R=100 Mbps
  - W>= 3ms * 100 Mbps/ (536*8) + 1

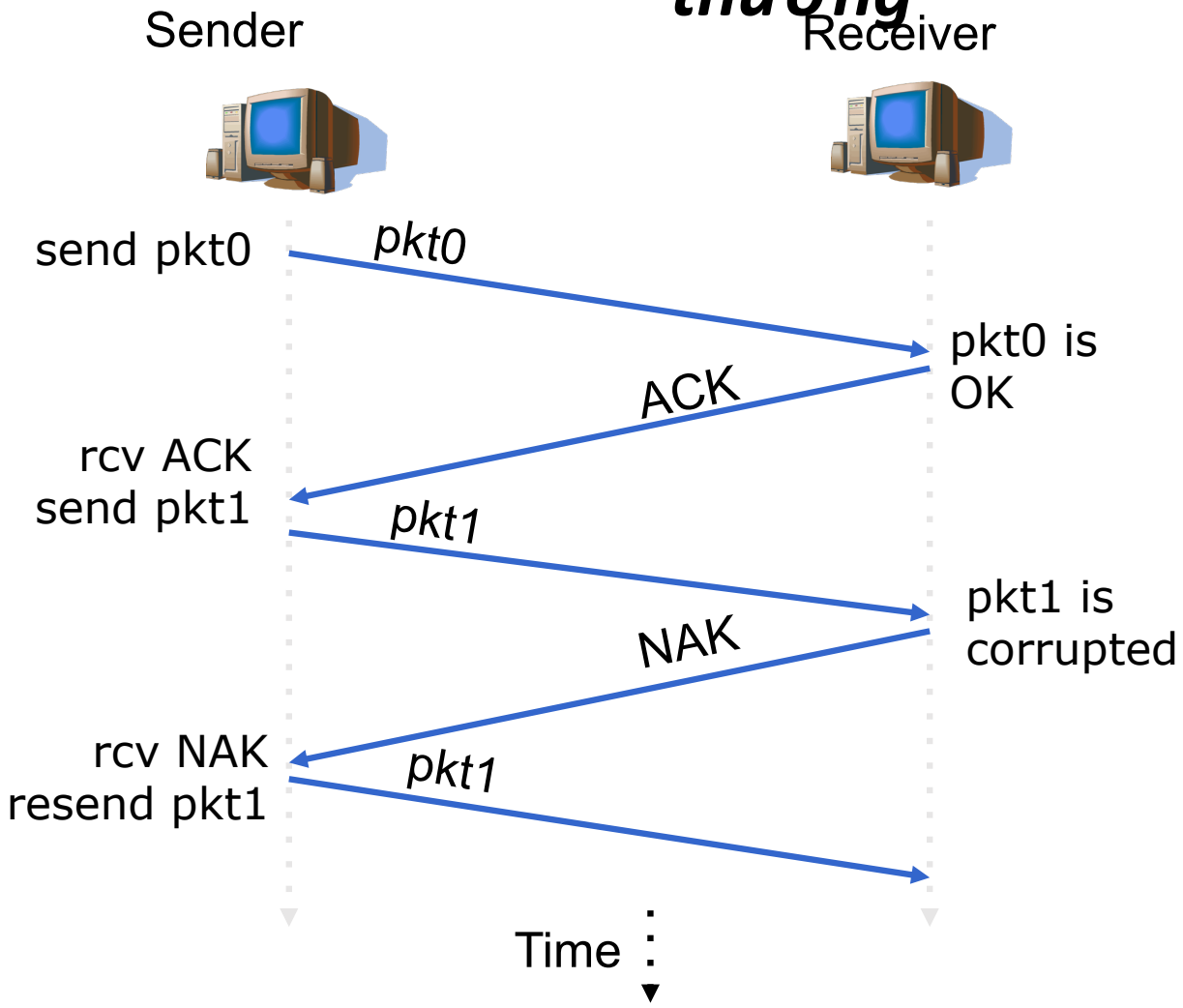
Vấn đề báo nhận và phát lại

- Phát hiện lỗi?
  - Checksum, CRC
- Làm thế nào để báo cho bên gửi?
  - ACK (*acknowledgements*):
  - NAK (*negative acknowledgements*): báo cho người gửi về một gói tin bị lỗi
- Phản ứng của bên gửi?
  - Truyền lại nếu là NAK
  - Nếu không nhận được cả ACK và NAK?

Các kỹ thuật báo nhận, phát lại

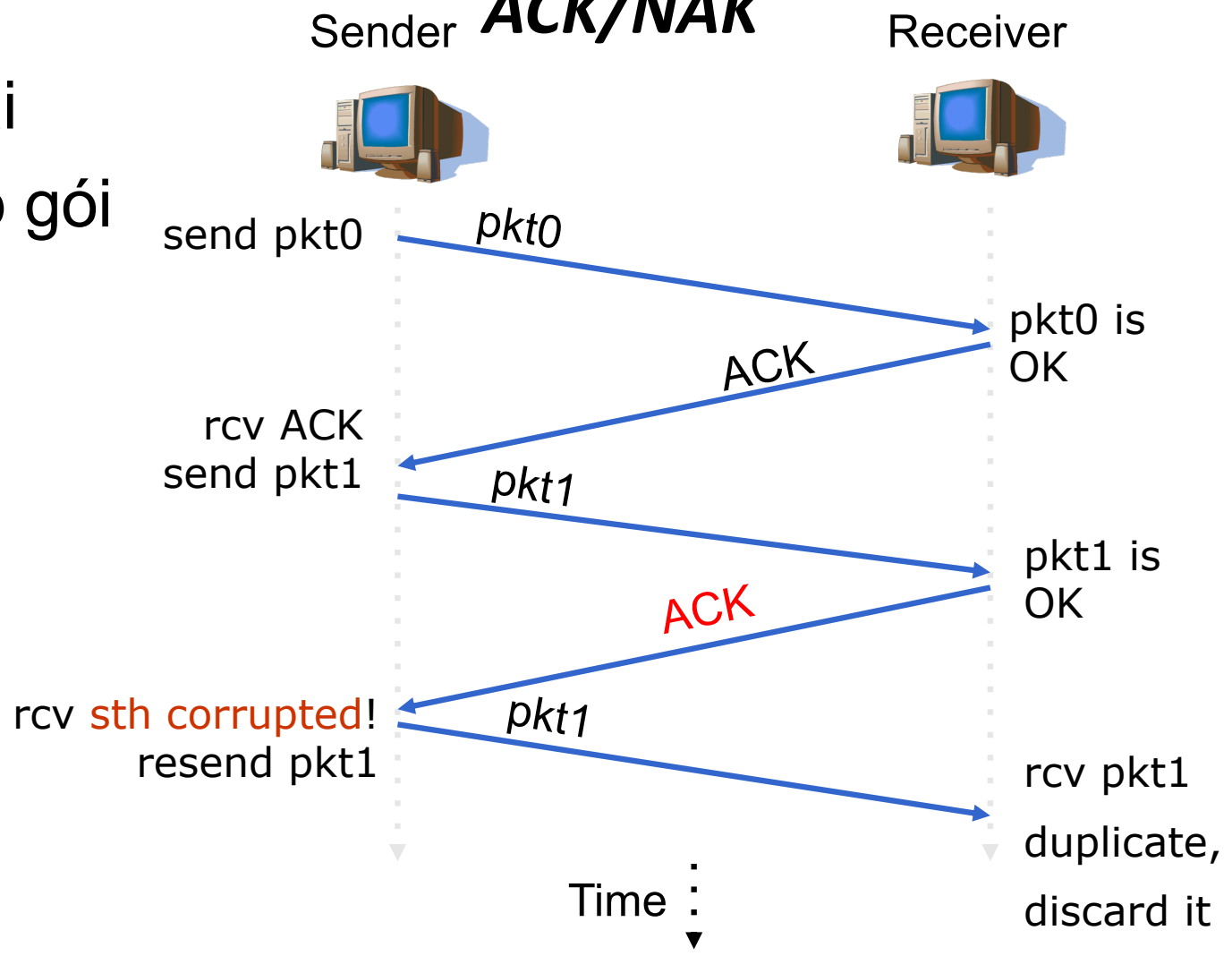
- **Kỹ thuật tự động truyền lại (ARQ automatic repeat request)**.
- Có 3 phiên bản chuẩn hóa
  - Dừng và chờ (Stop and Wait) ARQ
    - Đã xem ở tầng 2
  - Quay lại N (Go Back N) ARQ
    - Sẽ xem ở tầng 4
  - Loại bỏ chọn lọc (Selective Reject) ARQ
    - Sẽ xem ở tầng 4

Stop-and-wait ARQ *Trường hợp thông thường*

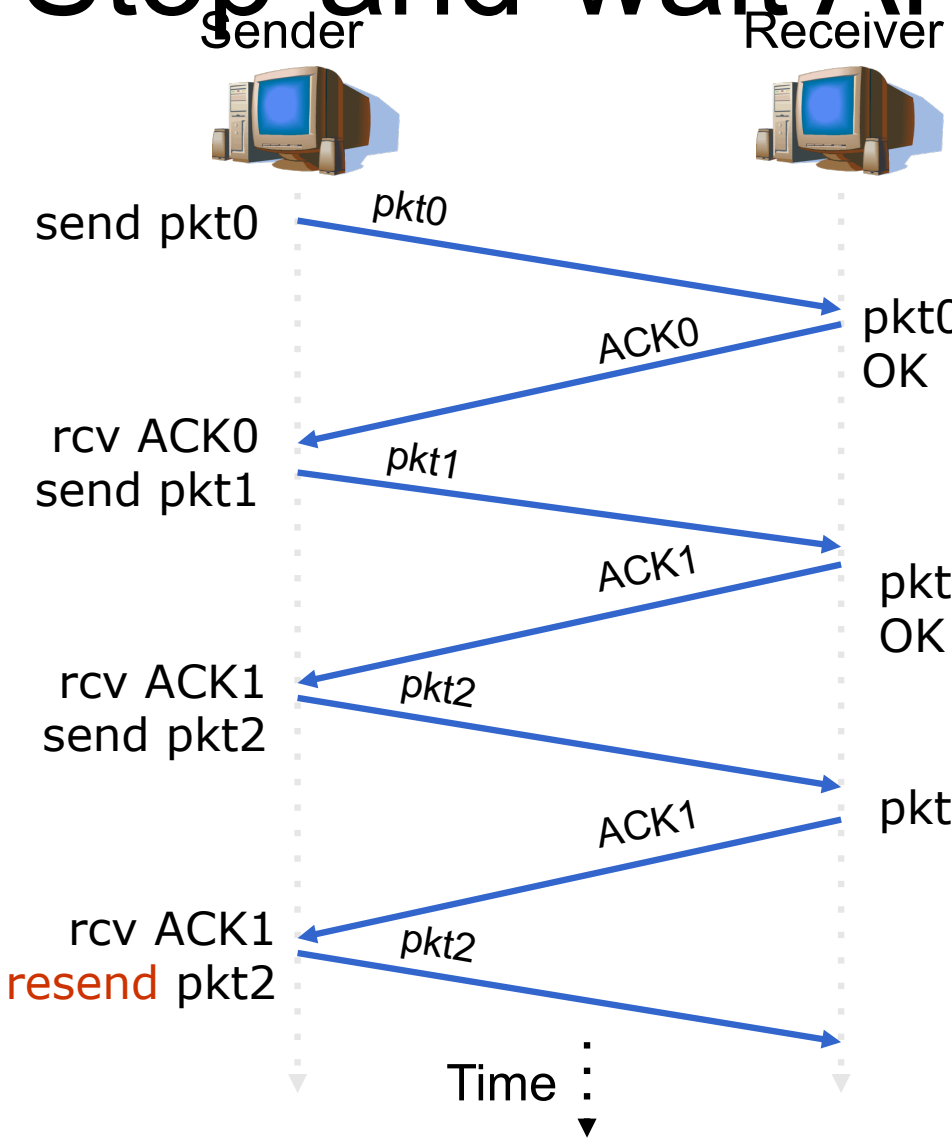


Stop-and-wait ARQ *Trường hợp lỗi ACK/NAK*

- Cần truyền lại
- Xử lý việc lặp gói tin ntn?
- Thêm Seq.#



Stop-and-wait ARQ



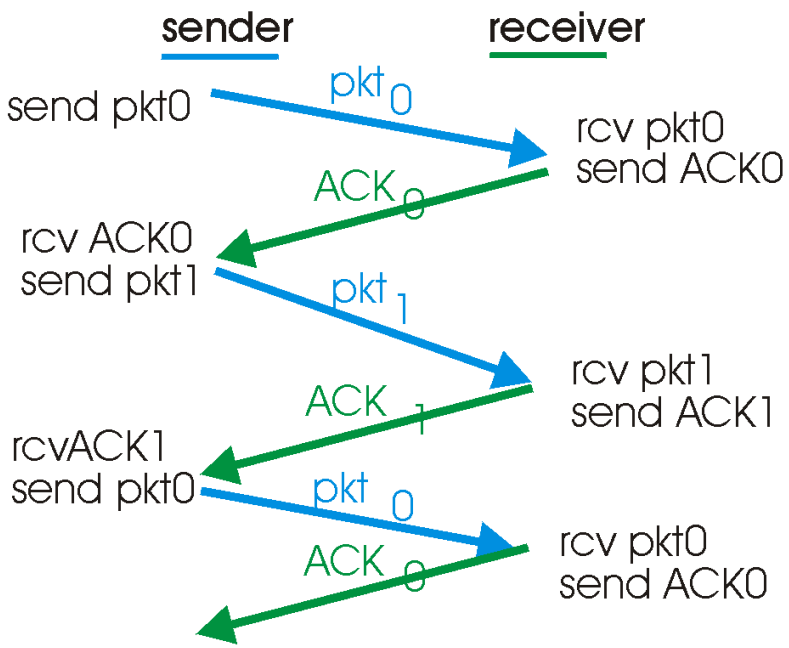
Giải pháp không dùng NAK

- ACK có kèm theo số hiệu gói tin được báo nhận
- ACK với số hiệu báo nhận n thể hiện đã nhận tốt tất cả gói tin số hiệu < =n.

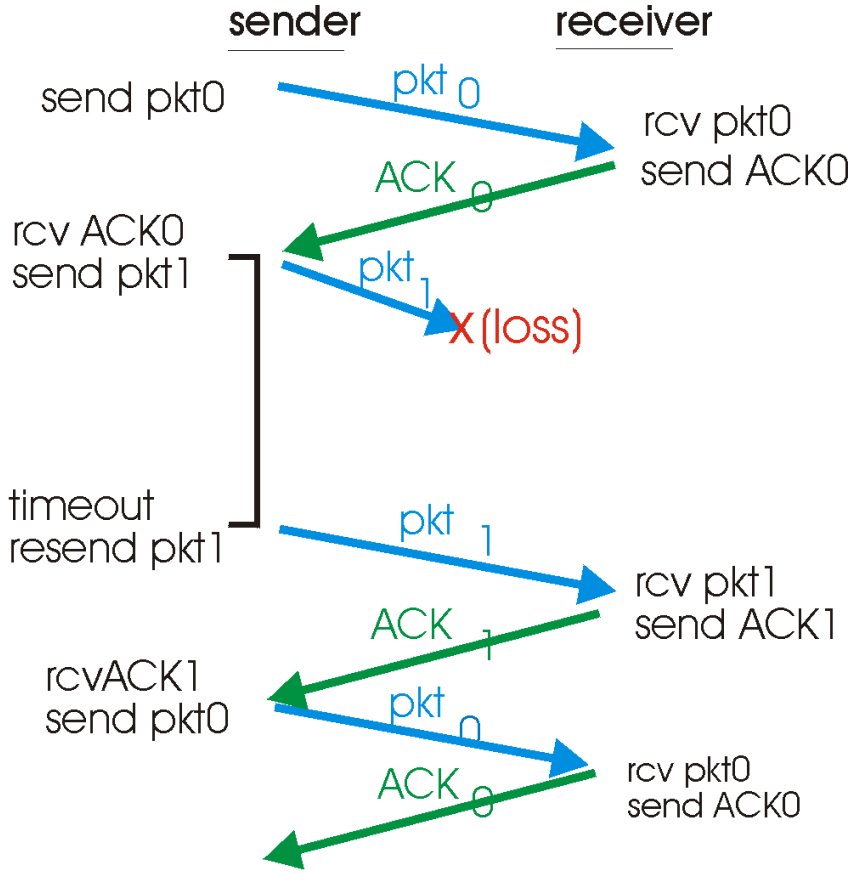
Stop-and-wait ARQ: Khi ACK bị mất

- Dữ liệu và ACK có thể bị mất
  - Nếu không nhận được ACK?
  - Truyền lại như thế nào?
- Cơ chế
  - Các gói được đánh số và bên gửi tự động truyền lại cho đến khi nhận được ACK của gói.
  - Chờ hết Timeout mới truyền lại
- Timeout nên dài bao lâu?
  - Ít nhất là 1 RTT (Round Trip Time)
  - Mỗi gói tin gửi đi cần 1 timer
- Nếu gói tin vẫn đến đích, chỉ ACK bị mất?
  - Phía nhận lọc gói tin dư căn cứ số hiệu gói tin trùng lặp

Minh họa ARQ với timeout

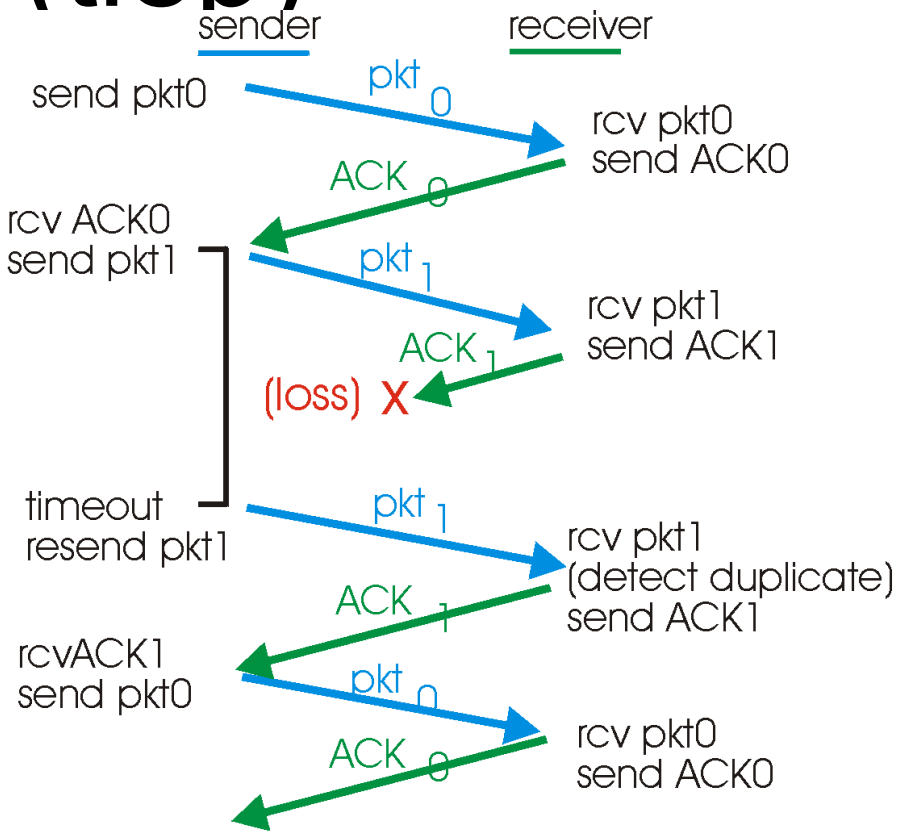


(a) operation with no loss

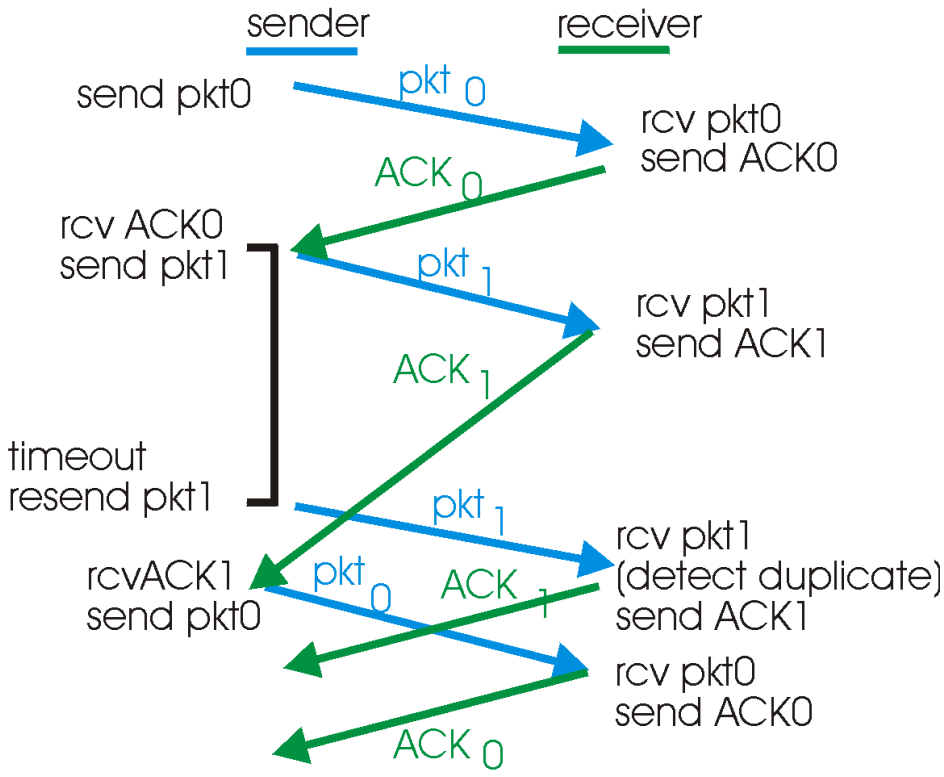


(b) lost packet

Minh họa ARQ với timeout (tiếp)



(c) lost ACK

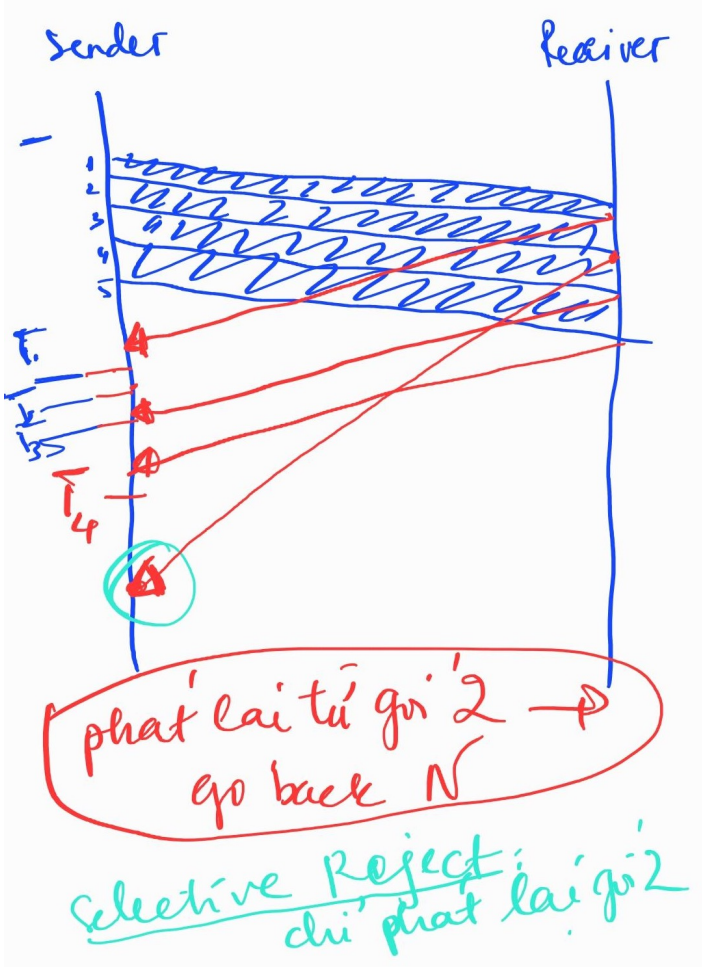


(d) premature timeout

Kỹ thuật tự động truyền lại ARQ

- Quay lại N (Go Back N) ARQ
  - Phiên bản đơn giản của Sliding windows.
  - Phía gửi phát liên tục các gói tin (max W gói)
  - Phía nhận phát liên tục các ACK nếu gói được nhận tốt
  - Phía gửi đặt timeout phải nhận ACK cho từng gói tin
  - Khi gặp timeout phía gửi **tự động phát lại tất cả các gói tin bắt đầu từ gói đầu tiên không nhận được ACK**
- Loại bỏ chọn lọc (Selective Reject) ARQ
  - Phía gửi phát liên tục các gói tin (max W gói)
  - Phía nhận phát liên tục các ACK nếu gói được nhận tốt
  - Phía gửi đặt timeout phải nhận ACK cho từng gói tin
  - Phía gửi **tự động phát lại chỉ gói tin không nhận được ACK sau timeout**

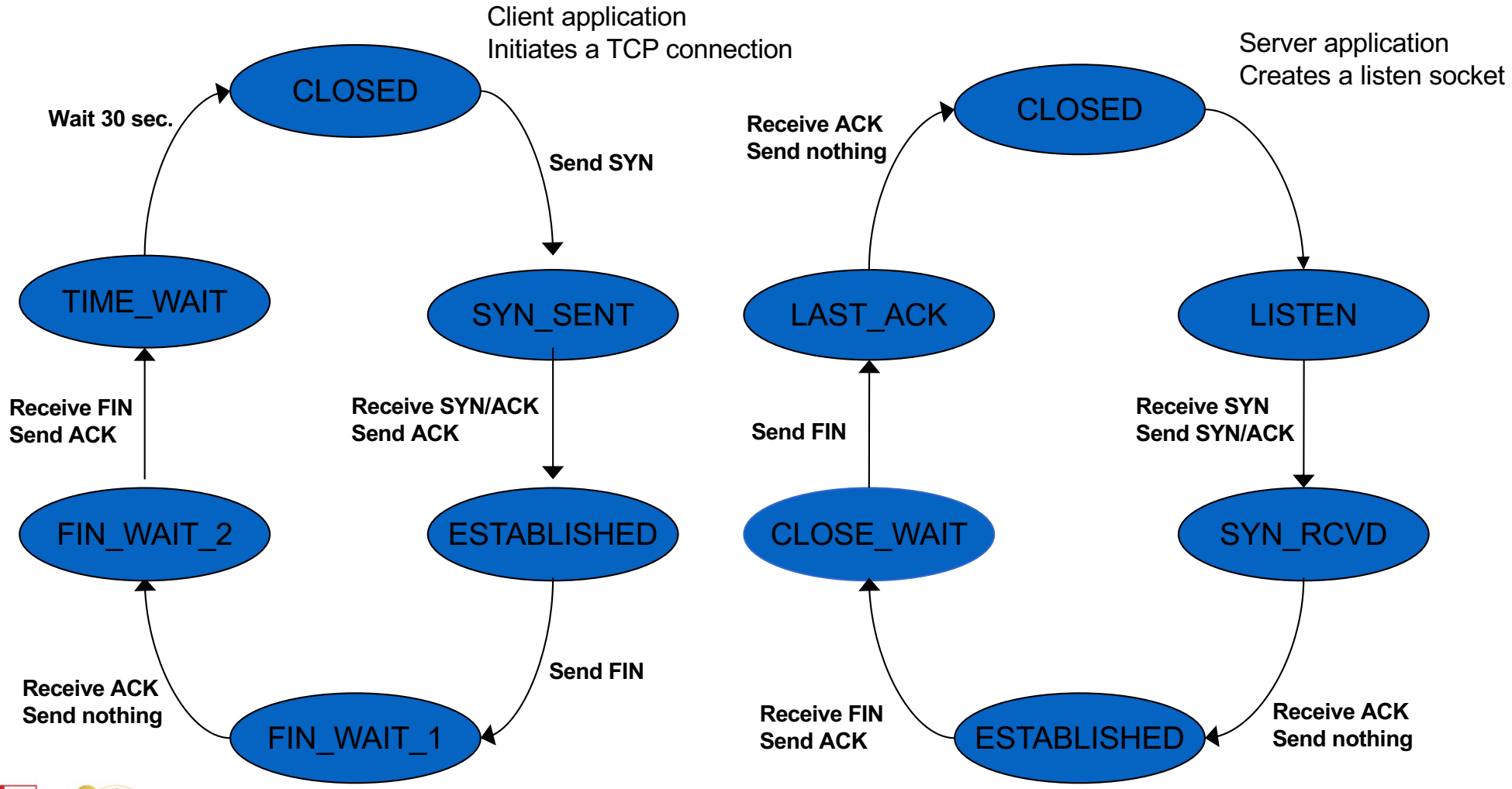
Kỹ thuật tự động truyền lại ARQ



Kỹ thuật tự động truyền lại ARQ

- Các kỹ thuật ARQ được sử dụng trong cả kiểm soát luồng và kiểm soát lỗi
  - Khi gói tin đến đích bị lỗi cần phát lại
  - Khi gói tin đến đích không đúng thứ tự cần phát lại
  - Khi gói tin bị mất hoặc ACK bị mất.

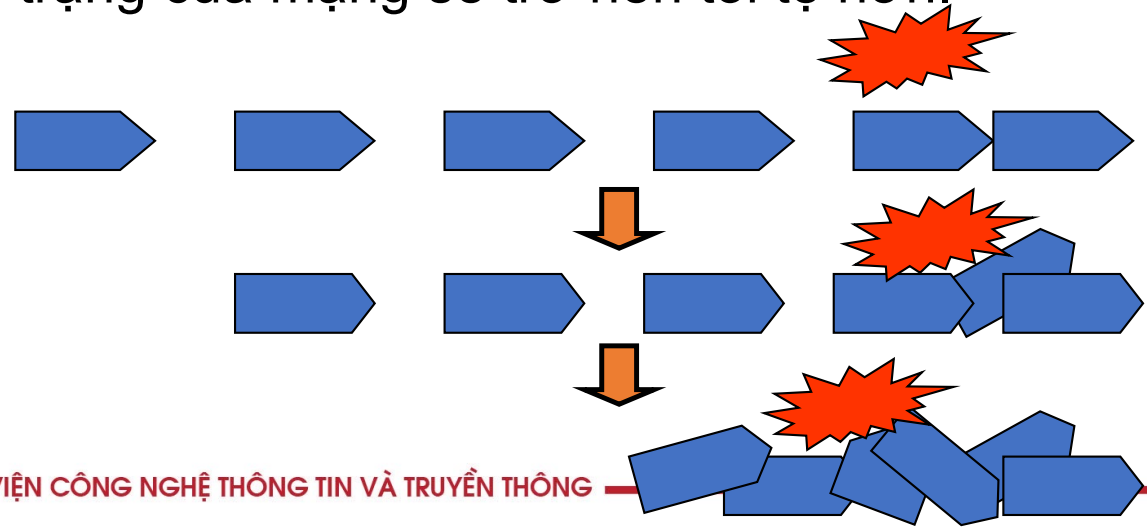
Chu trình sống của TCP (đơn giản hóa)



Điều khiển tắc nghẽn trong TCP

Tổng quan về tắc nghẽn

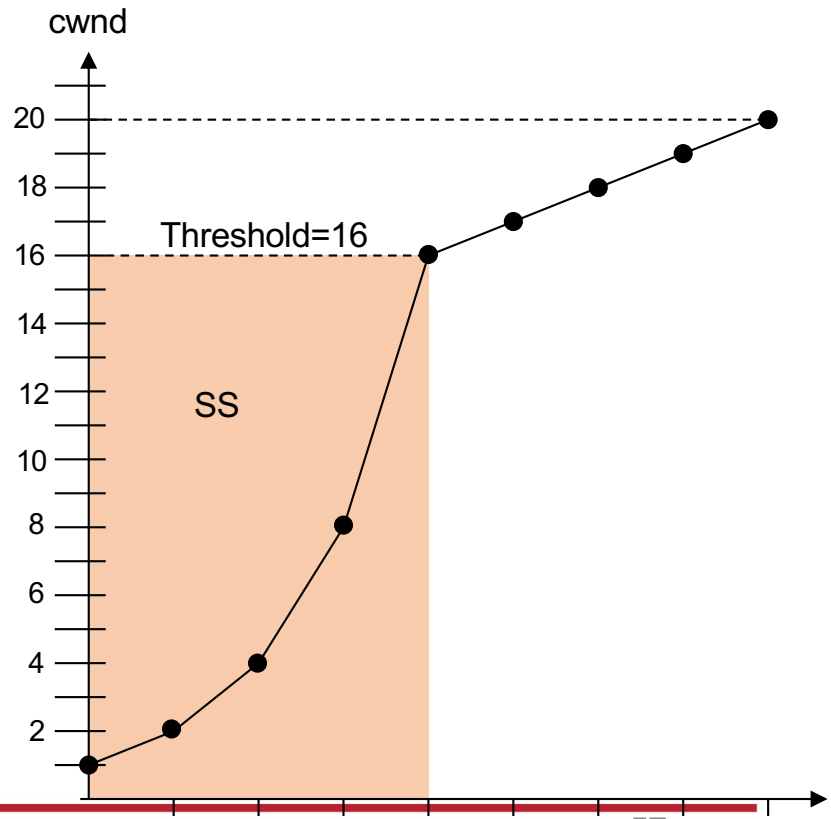
- Khi nào tắc nghẽn xảy ra ?
  - Quá nhiều cặp gửi-nhận trên mạng
  - Truyền quá nhiều làm cho mạng quá tải
    - Các thiết bị trung gian như router bị quá tải.
- Hậu quả của việc nghẽn mạng
  - Mất gói tin
  - Thông lượng giảm, độ trễ tăng
  - Tình trạng của mạng sẽ trở nên tồi tệ hơn.



Congestion occur

Nguyên lý kiểm soát tắc nghẽn

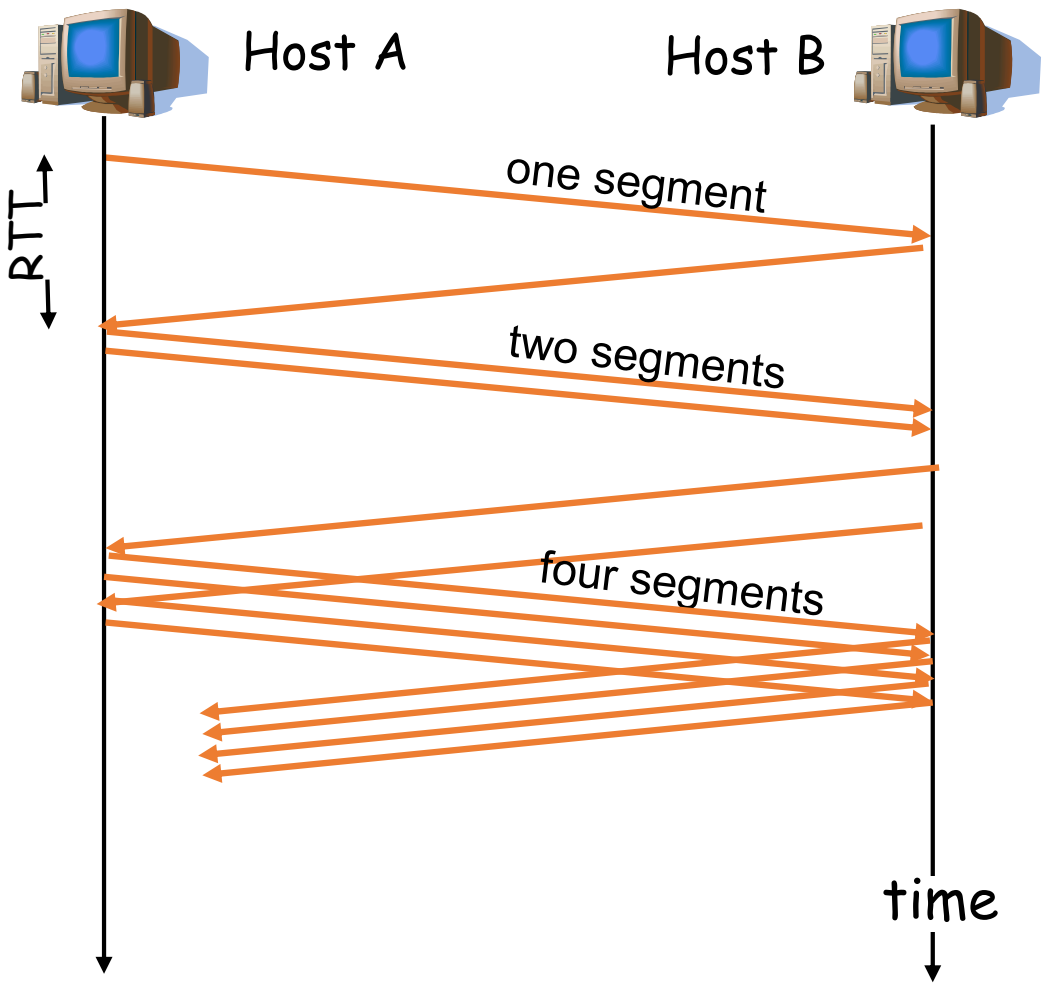
- Slow-start
  - Tăng tốc độ theo hàm số mũ
  - Tiếp tục tăng đến một ngưỡng nào đó
- Tránh tắc nghẽn (congestion avoidance)
  - Tăng dẫn tốc độ theo hàm tuyến tính cho đến khi phát hiện tắc nghẽn
- Phát hiện tắc nghẽn
  - Nếu gói tin bị mất



TCP Slow Start (1)

- Ý tưởng cơ bản
  - Điều khiển tốc độ bằng cwnd: Cửa sổ cwnd quy định chỉ được gửi lượng dữ liệu bằng kích thước cwnd mà không cần chờ báo nhận.
    - Kết hợp Rwnd, suy ra số gói tin được phát min (Rwnd, Cwnd)
  - Đặt cwnd bằng 1 MSS (Maximum segment size)
  - Tăng cwnd lên 1 MSS sau mỗi lần bên gửi nhận được 1 gói tin ACK
    - Bắt đầu chậm,
    - Tốc độ tăng theo hàm mũ sau mỗi RTT
- Tăng cho đến một ngưỡng: ssthresh
  - Sau đó, TCP chuyển sang trạng thái tránh tắc nghẽn

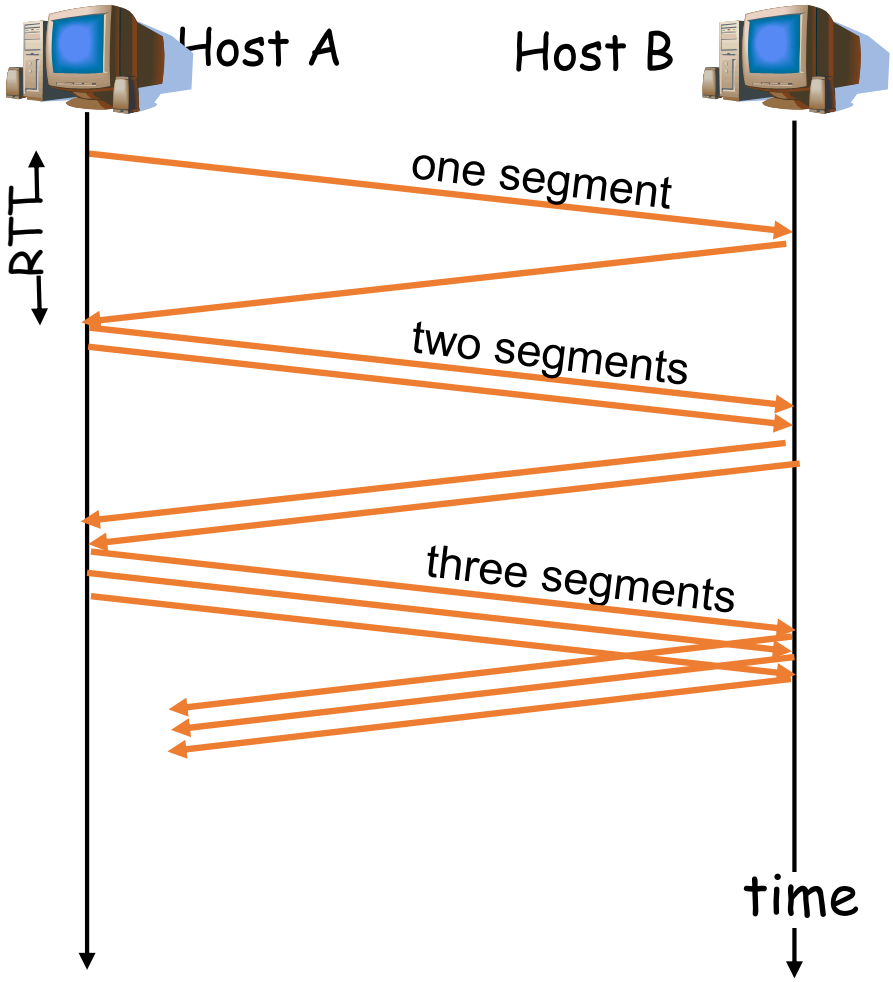
TCP Slow Start (2)



Tránh tắc nghẽn - Congestion avoidance

- Sau mỗi RTT tăng cwnd thêm 1 MSS

➔ Tăng cwnd theo cấp số cộng sau khi nó đạt tới ssthresh



Phát hiện tắc nghẽn và phản ứng của TCP (1)

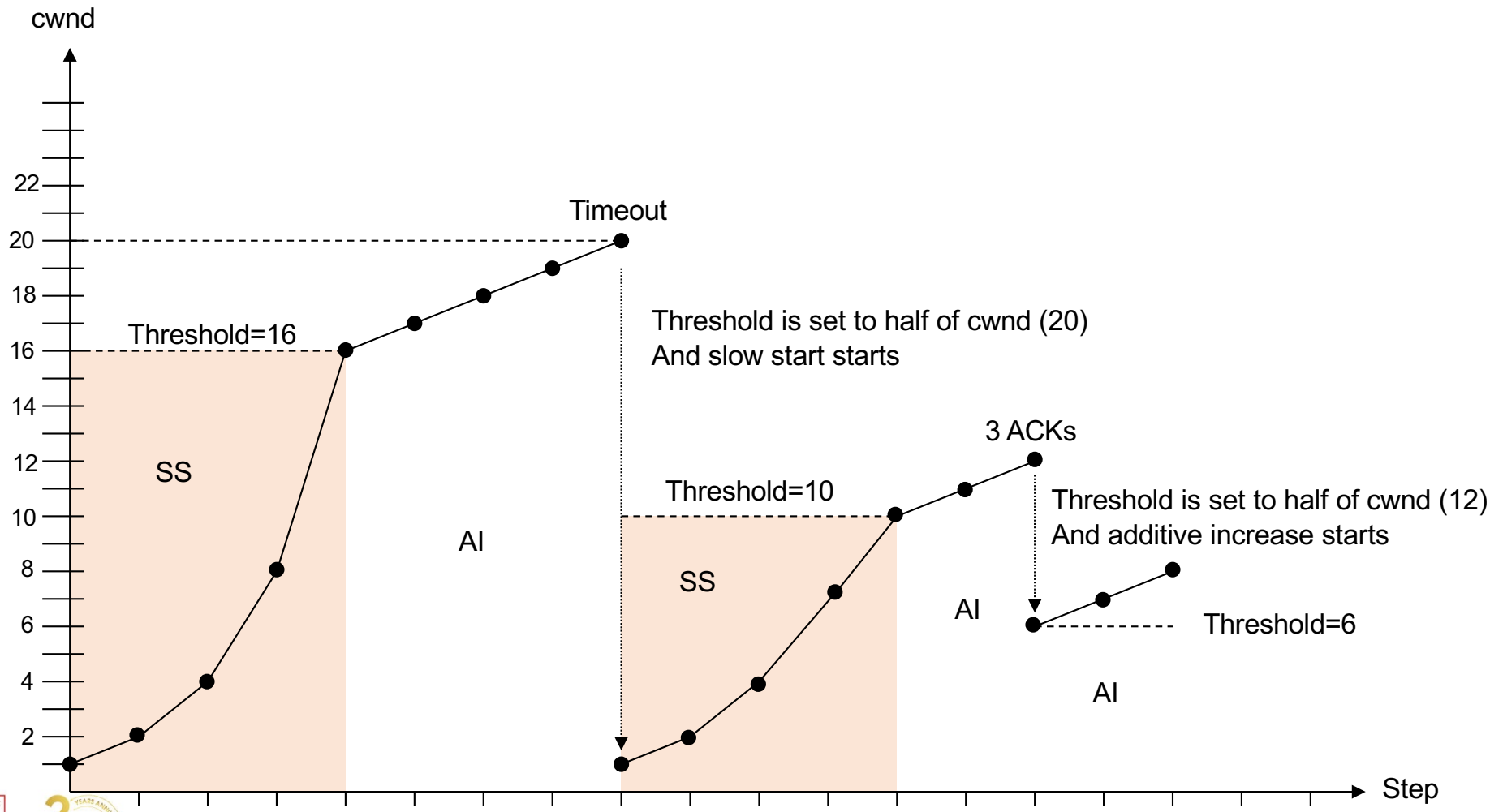
- Phát hiện tắc nghẽn?
  - Nếu như phải truyền lại
    - Có thể suy ra là mạng "tắc nghẽn"
- Dấu hiệu phát hiện tắc nghẽn của TCP
  - Timeout!
  - Nhận được 3 gói tin ACK giống nhau
- Khi phát hiện tắc nghẽn thì giảm tốc độ gửi

Phát hiện tắc nghẽn và phản ứng của TCP(2)

- Khi có timeout của bên gửi
  - TCP đặt ngưỡng ssthresh xuống còn một nửa giá trị hiện tại của cwnd
  - TCP đặt cwnd về 1 MSS
  - TCP chuyển về slow start

- Nếu nhận được 3 ACK giống nhau
  - TCP đặt cửa sổ cwnd bang một nửa giá trị hiện tại
  - TCP chuyển trạng thái “congestion avoidance”

Kiểm soát tắc nghẽn – minh họa



Bài tập

- Giả sử cần truyền 1 file
  - Kích thước *O* =100KB trên kết nối TCP
  - S là kích thước mỗi gói TCP, *S* = 536 byte
  - *RTT* = 100 *ms*.
- Giả sử cửa sổ tắc nghẽn của TCP hoạt động theo cơ chế slow-start.
- Cửa sổ đạt đến giá trị bao nhiêu thì truyền hết file.
- Hỏi thời gian chờ đợi để truyền hết file là bao nhiêu? Nếu R = 10 Mbit/s; R= 100 Mbits/s.

Tổng kết

- Còn rất nhiều chi tiết về TCP!

- Có hai dạng giao thức giao vận
  - UDP và TCP
  - Best effort vs. reliable transport protocol
- Các cơ chế bảo đảm độ tin cậy
  - Báo nhận
  - Truyền lại
  - Kiểm soát luồng và kiểm soát tắc nghẽn

Tuần tới: Application Layer

- Application service model
  - Client-server vs. P2P
- Typical applications and protocols
  - HTTP
  - Mail
  - FTP
  - P2P file sharing
  - ……
  - and your applications?

Acknowledgment

- Bài giảng có sử dụng các hình vẽ từ
  - Tài liệu của trường đại học Keio và Ritsumekan
  - Tài liệu “Computer Network, a top down approach” của J.F Kurose và K.W. Ross