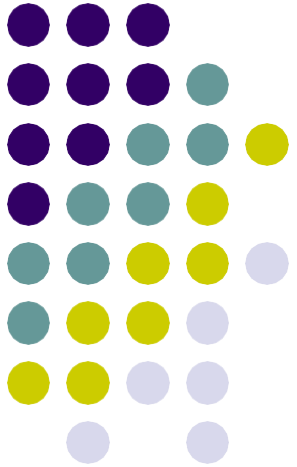
Chương 8:
Tầng giao vận

Giảng viên: Nguyễn Đức Toàn

Bộ môn Truyền thông và Mạng máy tính
Viện CNTT&TT - ĐHBK Hà Nội



Tổng quan

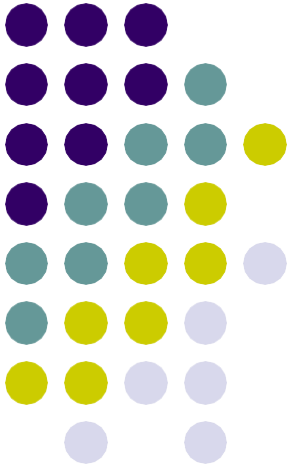
- Các tuần trước : Giao thức IP
  - ðịa chỉ, gói tin IP
  - ICMP
  - Chọn đường
- Hôm nay: Tầng giao vận
  - Nguyên lý tầng giao vận
  - Giao thức UDP
  - Giao thức TCP

Các khái niệm cơ bản

Nhắc lại kiến trúc phân tầng

Hướng liên kết vs. Không liên kết

UDP & TCP



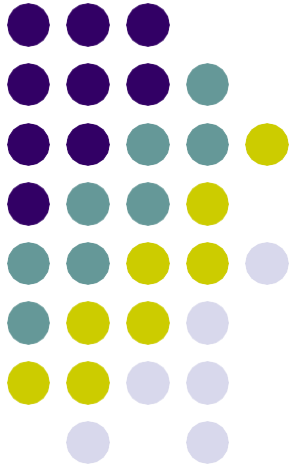
Nhắc lại về kiến trúc phân tầng

Application (HTTP, Mail, …)	Hỗ trợ các ứng dụng trên mạng
Transport (UDP, TCP …)	**Truyền dữ liệu giữa các ứng dụng**
Network (IP, ICMP…)	Chọn đường và chuyển tiếp gói tin giữa các máy, các mạng
Datalink (Ethernet, ADSL…)	Hỗ trợ việc truyền thông cho các thành phần kế tiếp trên cùng 1 mạng
Physical (bits…)	Truyền và nhận dòng bit trên đường truyền vật lý

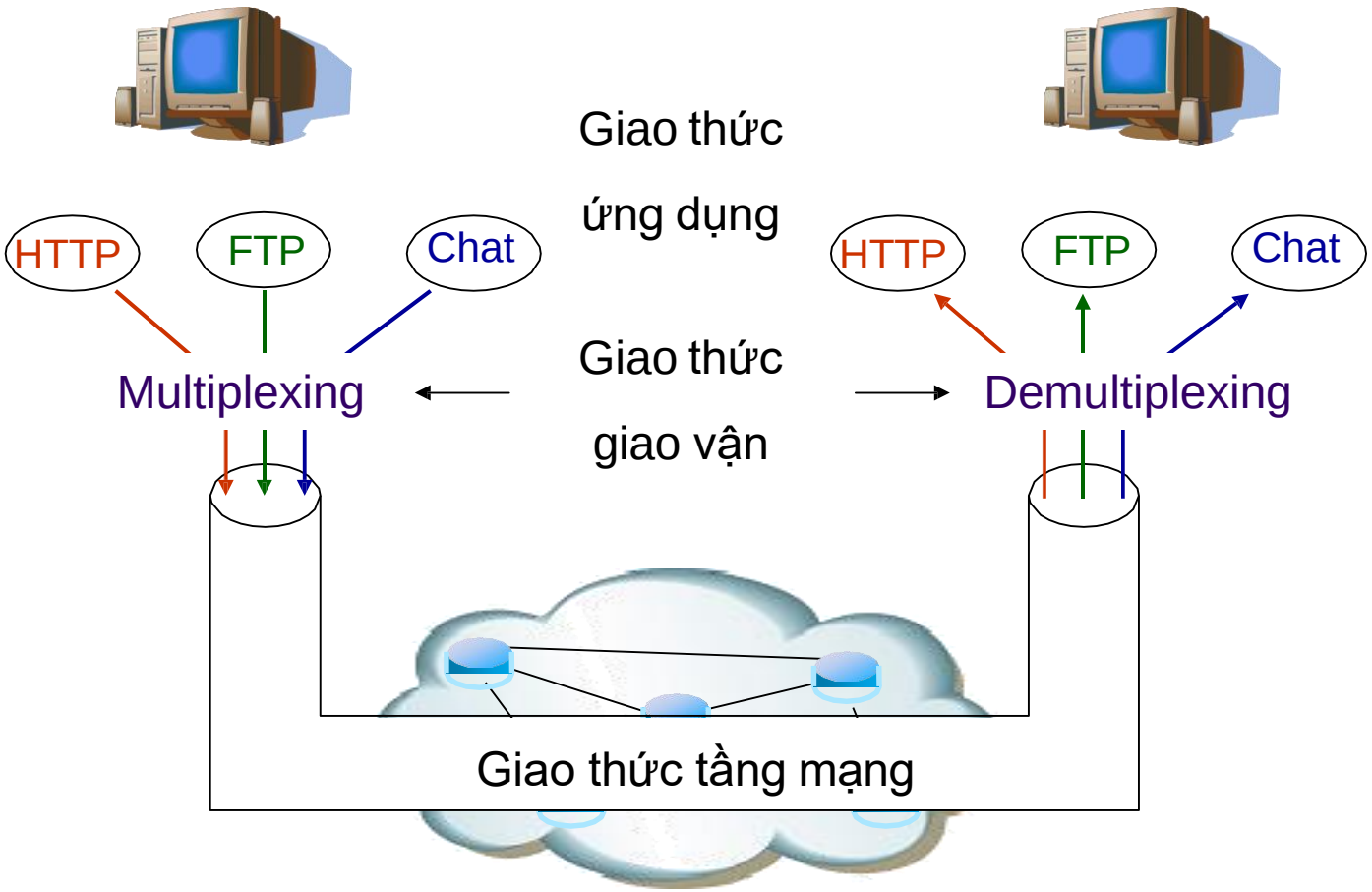
Các chức năng chung

Dồn kênh/phân kênh

Mã kiểm soát lỗi

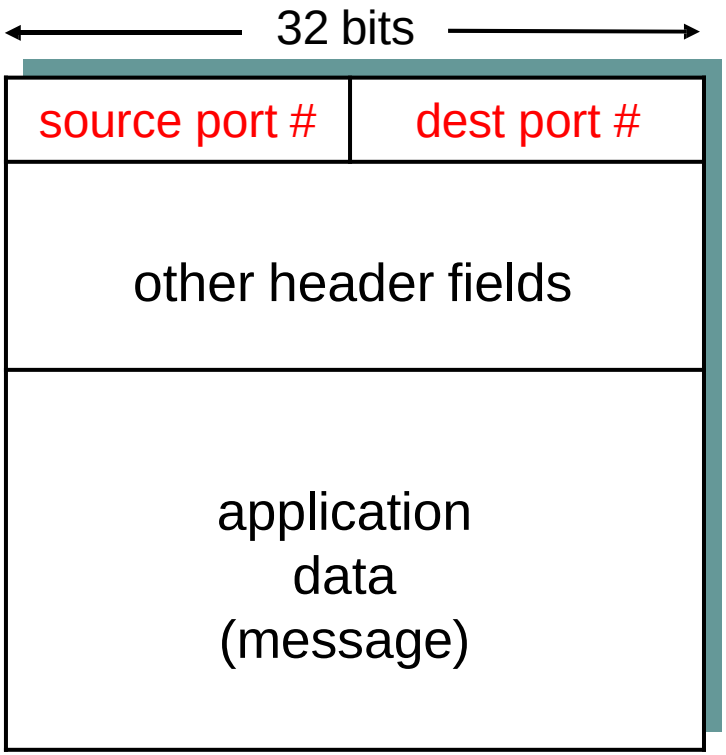


Dồn kênh/phân kênh - Mux/Demux



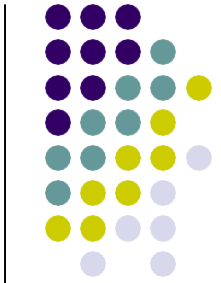
Mux/Demux hoạt động ntn?

- 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

Checksum



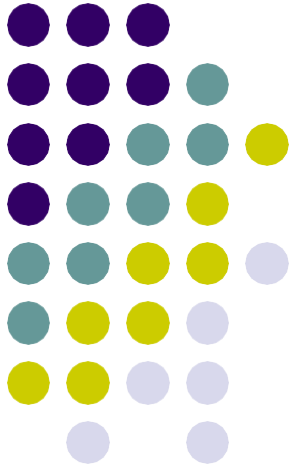
- Phát hiện lỗi bit trong các đoạn tin/gói tin
- Nguyên lý giống như checksum (16 bits) của giao thức IP
- Ví dụ:

	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
	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
Tổng	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

UDP
User Datagram Protocol

Tổng quan

Khuôn dạng gói tin

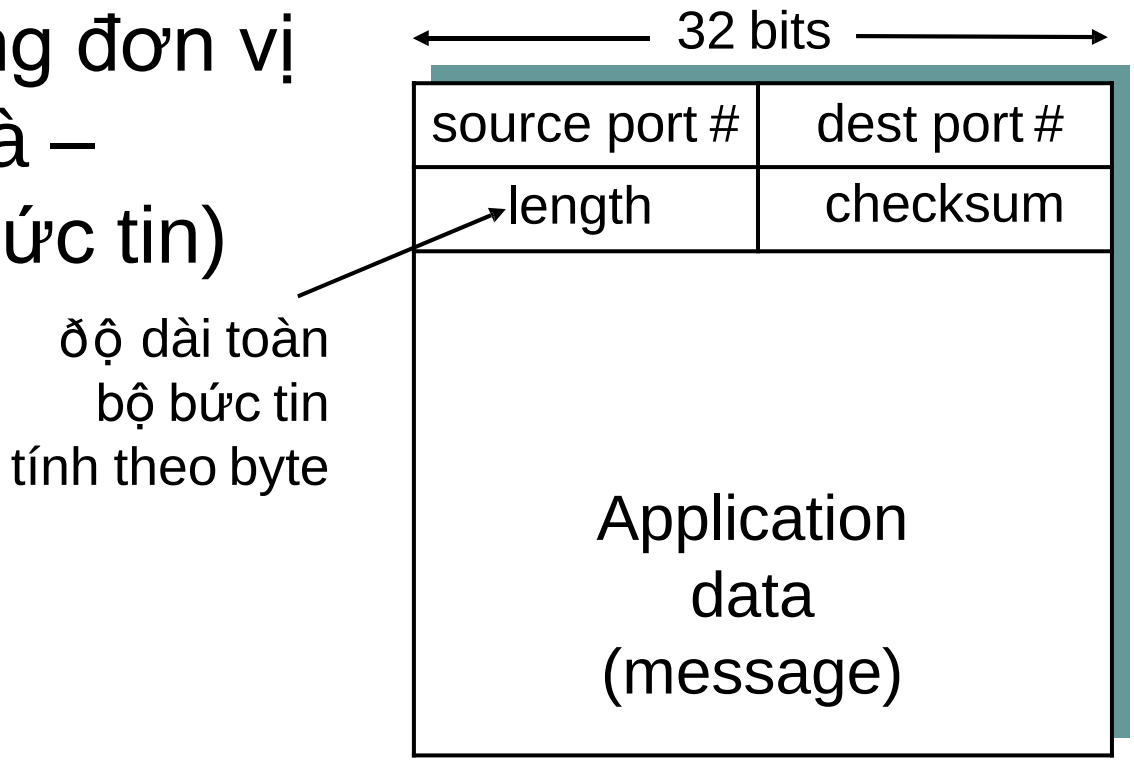


Giao thức dạng "Best effort"

- Vì sao cần UDP?
  - Không cần thiết lập liên kết (tăng độ trễ)
  - 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)

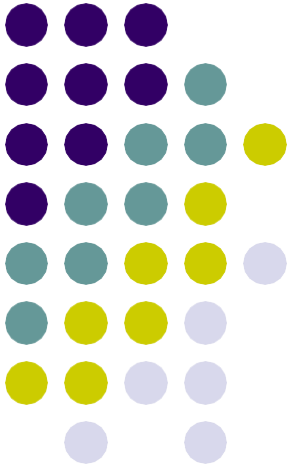


Khuôn dạng đơn vị dữ liệu của UDP

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

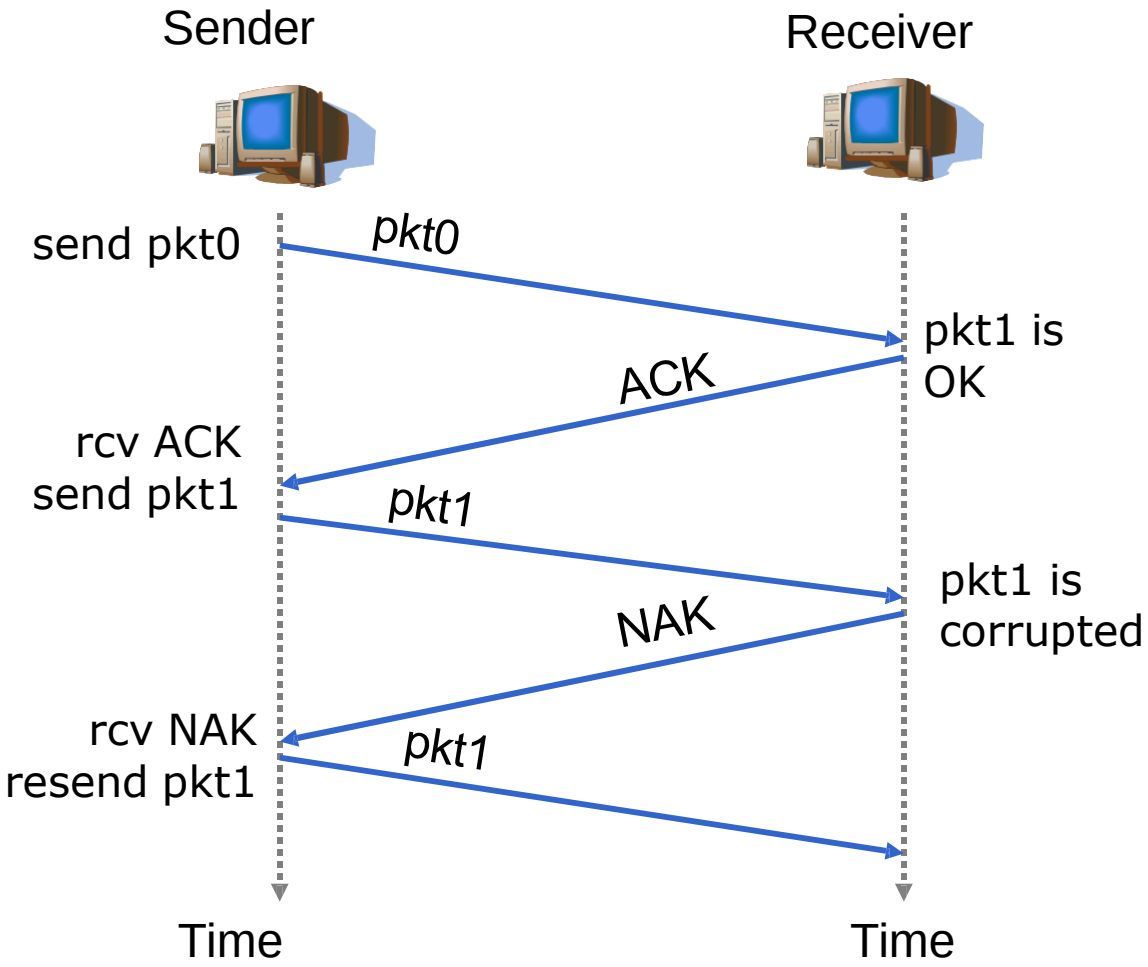
Khái niệm về truyền thông tin cậy



Kênh có lỗi bit, không bị mất tin

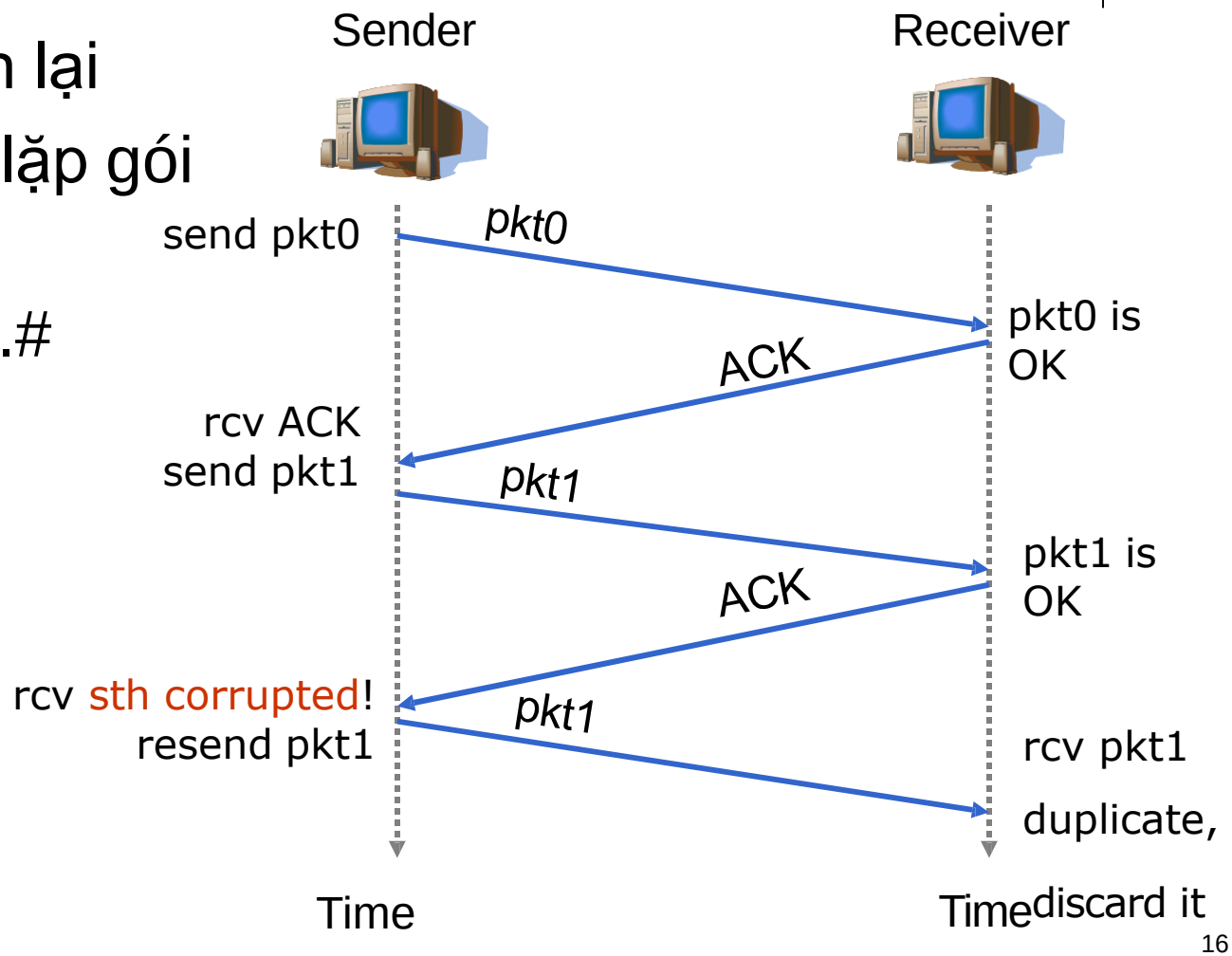
- Phát hiện lỗi?
  - Checksum
- Làm thế nào để báo cho bên gửi?
  - ACK (*acknowledgements*):
  - NAK (*negative acknowledgements*): báo cho bên nhận rằng pkt bị lỗi
- Phản ứng của bên gửi?
  - Truyền lại nếu là NAK

Hoạt động

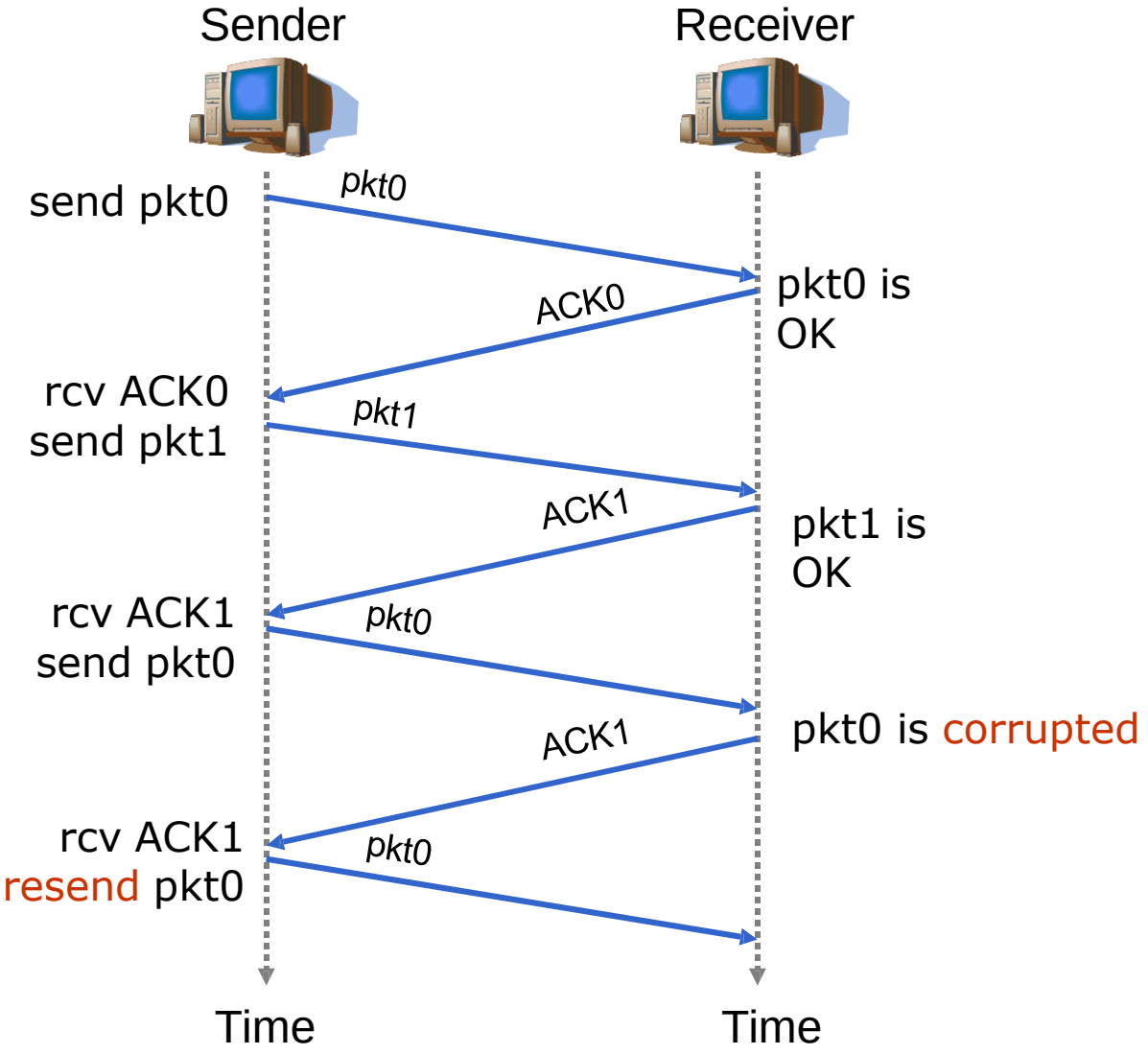

Sender
Receiver
send pkt0
pkt0
pkt1 is
OK
ACK
rcv ACK
send pkt1
pkt1
pkt1 is
corrupted
NAK
rcv NAK
resend pkt1
pkt1
Time
Time

Lỗi ACK/NAK

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



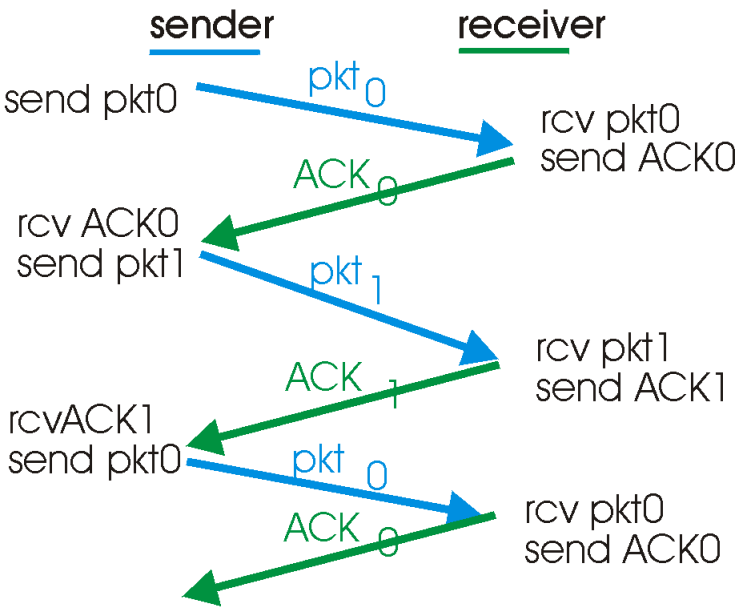
Giải pháp không dùng NAK



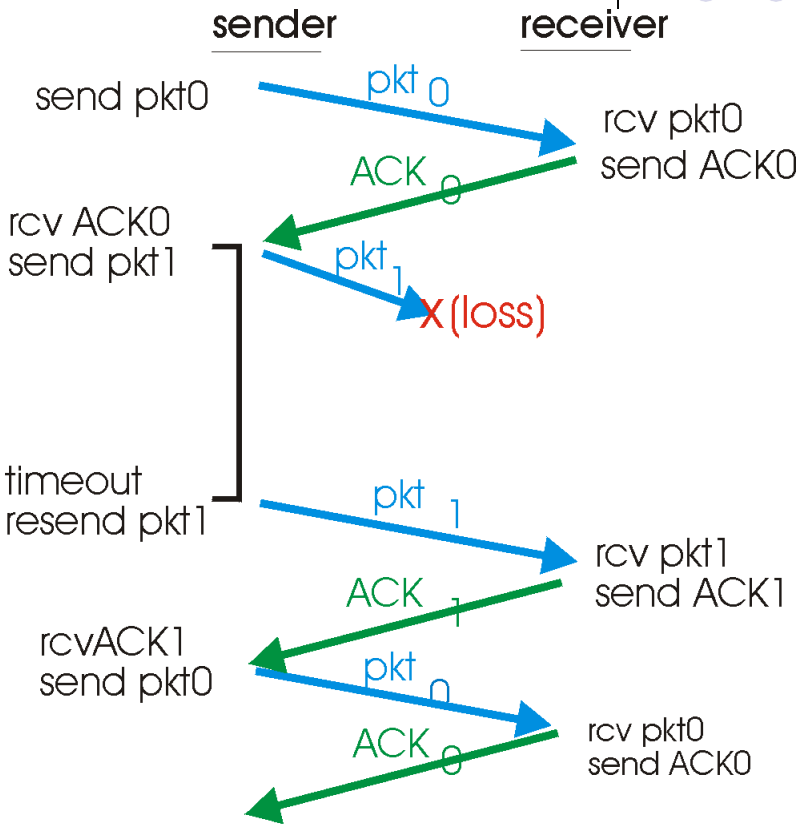
Kênh có lỗi bit và mất gói tin

- Dữ liệu và ACK có thể bị mất
  - Nếu không nhận được ACK?
  - Truyền lại như thế nào?
  - Timeout!
- Thời gian chờ là bao lâu?
  - Ít nhất là 1 RTT (Round Trip Time)
  - Mỗi gói tin gửi đi cần 1 timer
- Nếu gói tin vẫn đến đích và ACK bị mất?
  - Dùng số hiệu gói tin

Minh họa

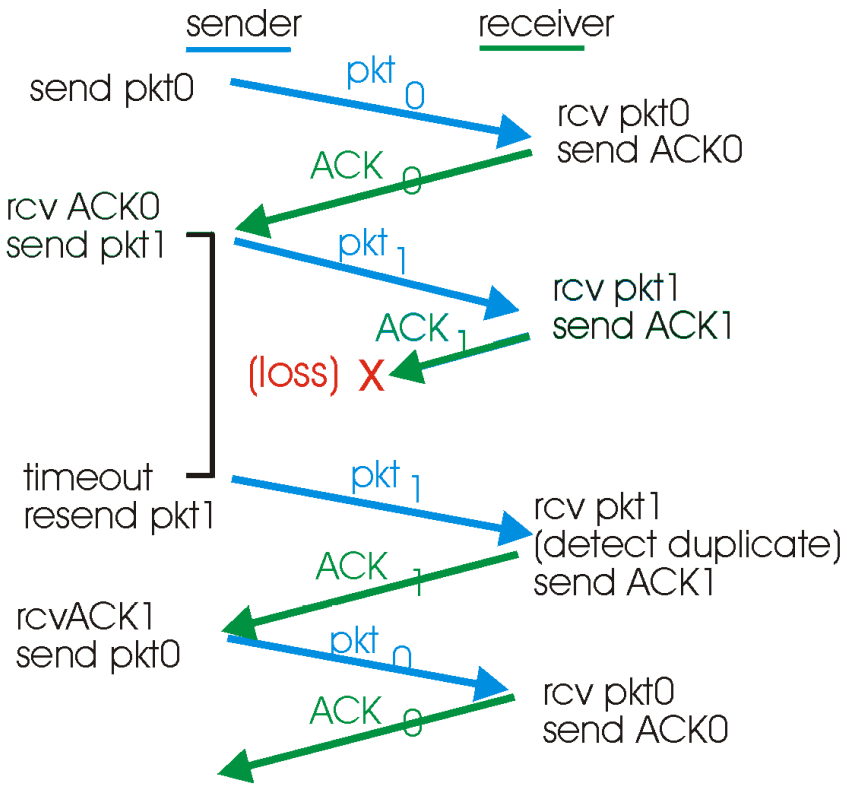


(a) operation with no loss

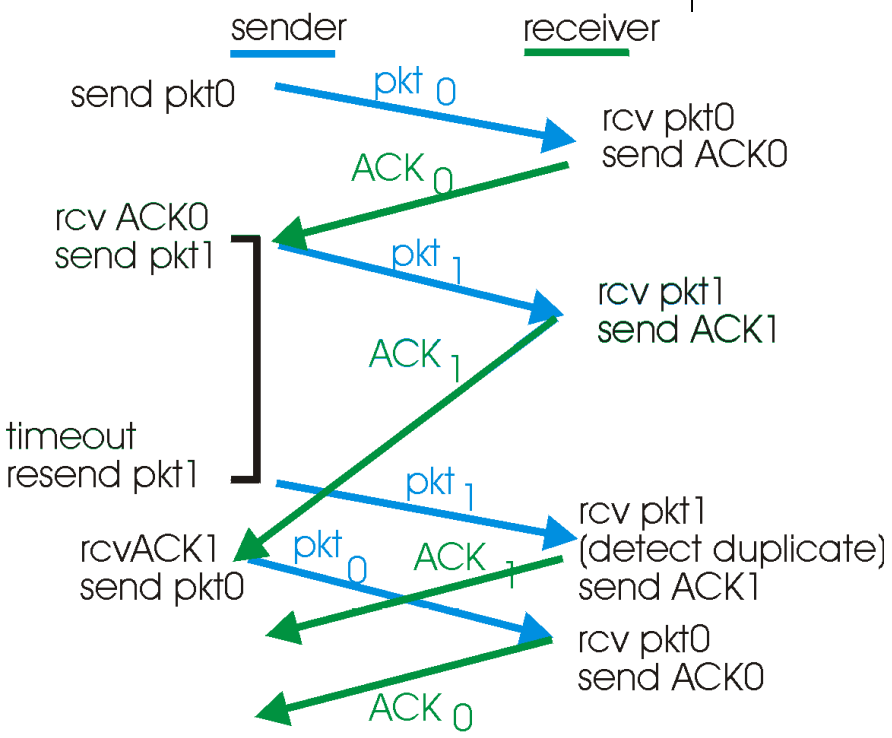


(b) lost packet

Minh họa

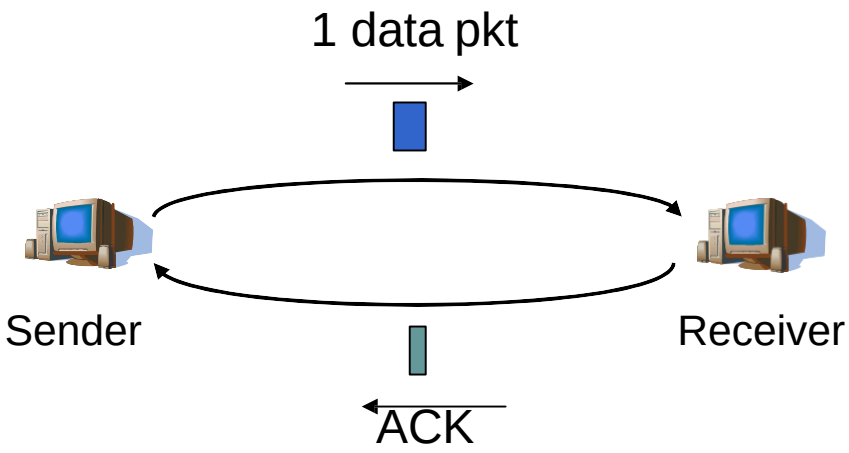


(c) lost ACK

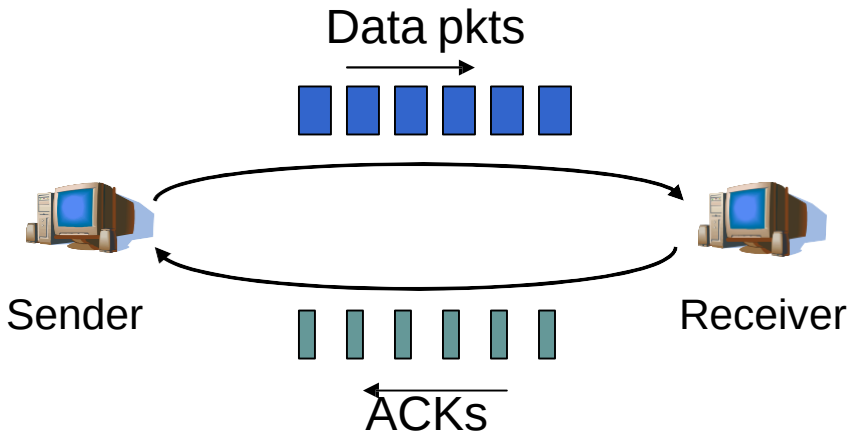


(d) premature timeout

Truyền theo kiểu pipeline



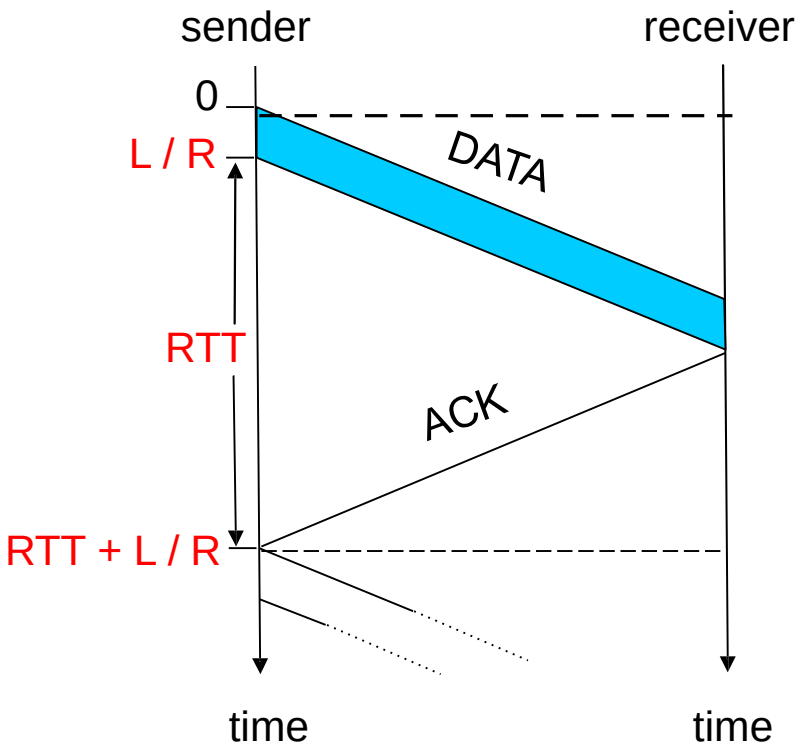
stop-and-wait



Pipeline

So sánh hiệu quả

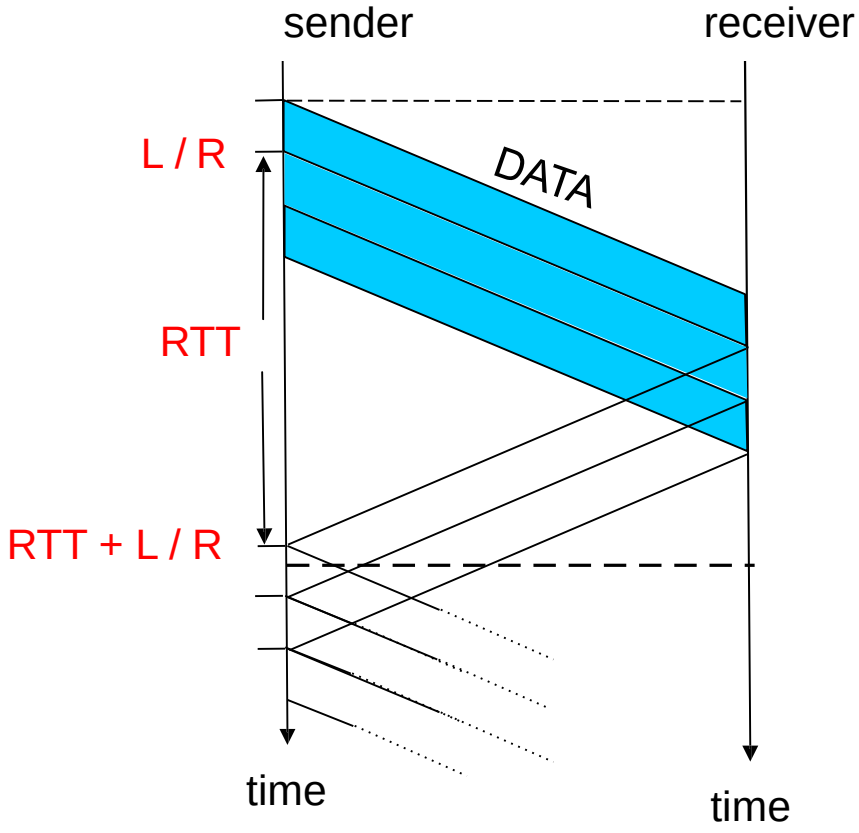
stop-and-wait



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

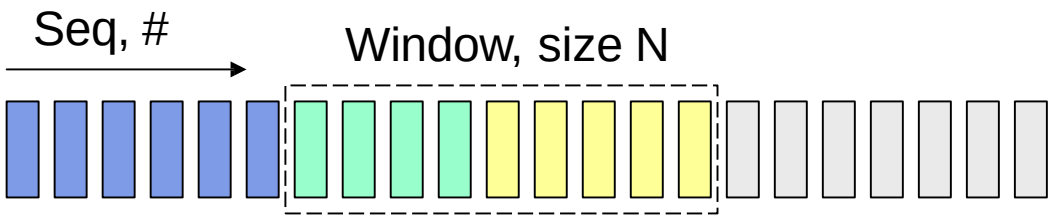
$$\text{Performance} = \frac{L / R}{RTT + L / R}$$

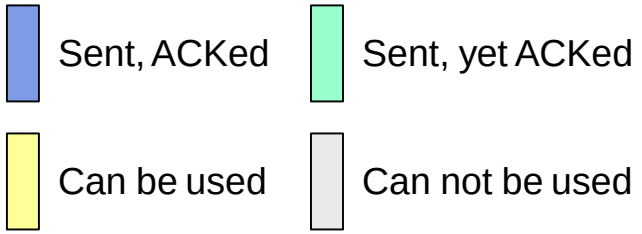
Pipeline



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

Go-back-N





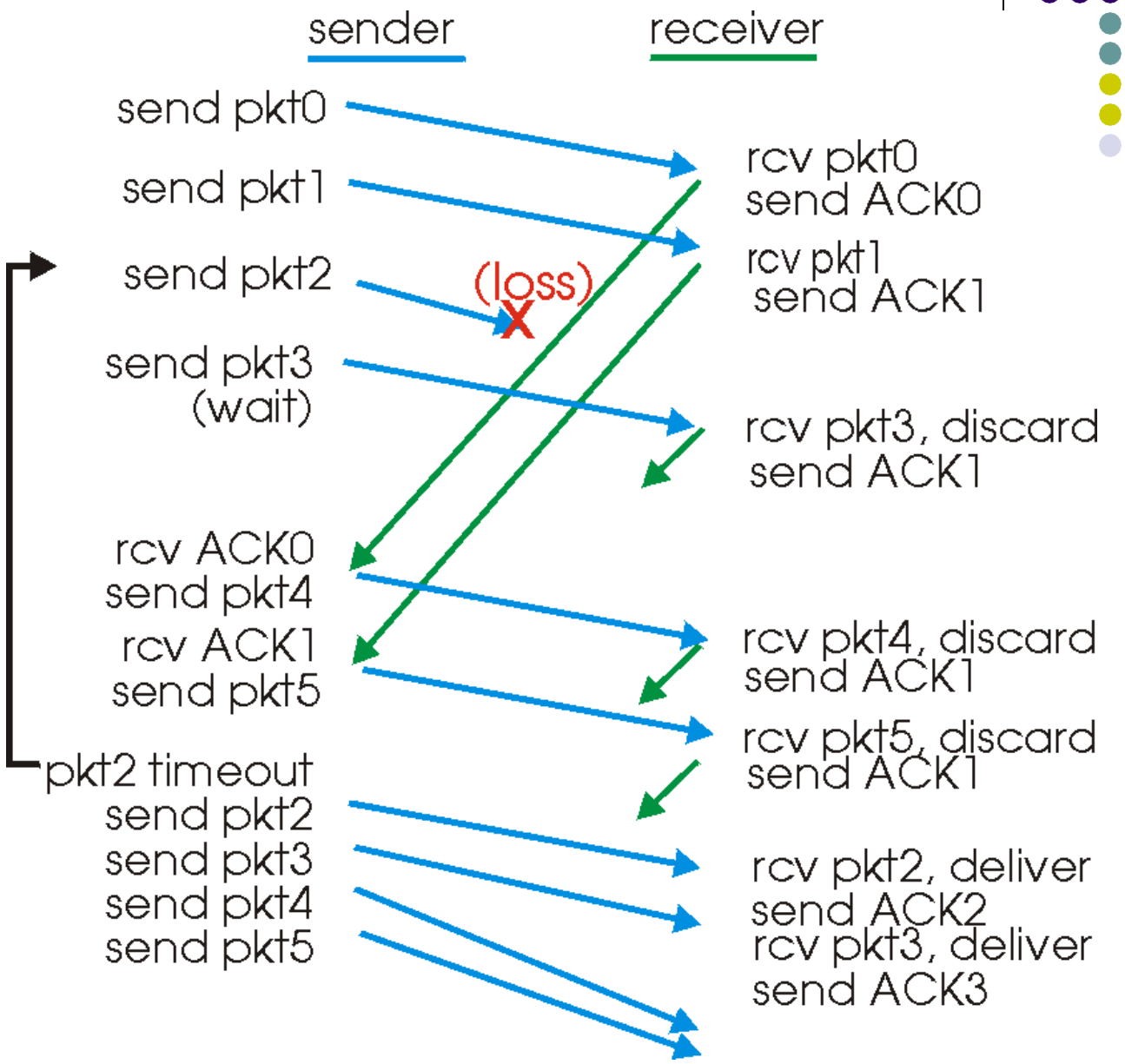
Sender

- Chỉ gửi các gói tin với số hiệu trong cửa sổ, "dịch" cửa sổ sang phải mỗi khi nhận được ACK
- ACK(n): xác nhận cho các gói tin với số hiệu cho đến n
- Khi có timeout: truyền lại tất cả các gói tin có số hiệu lớn hơn n trong cửa sổ.

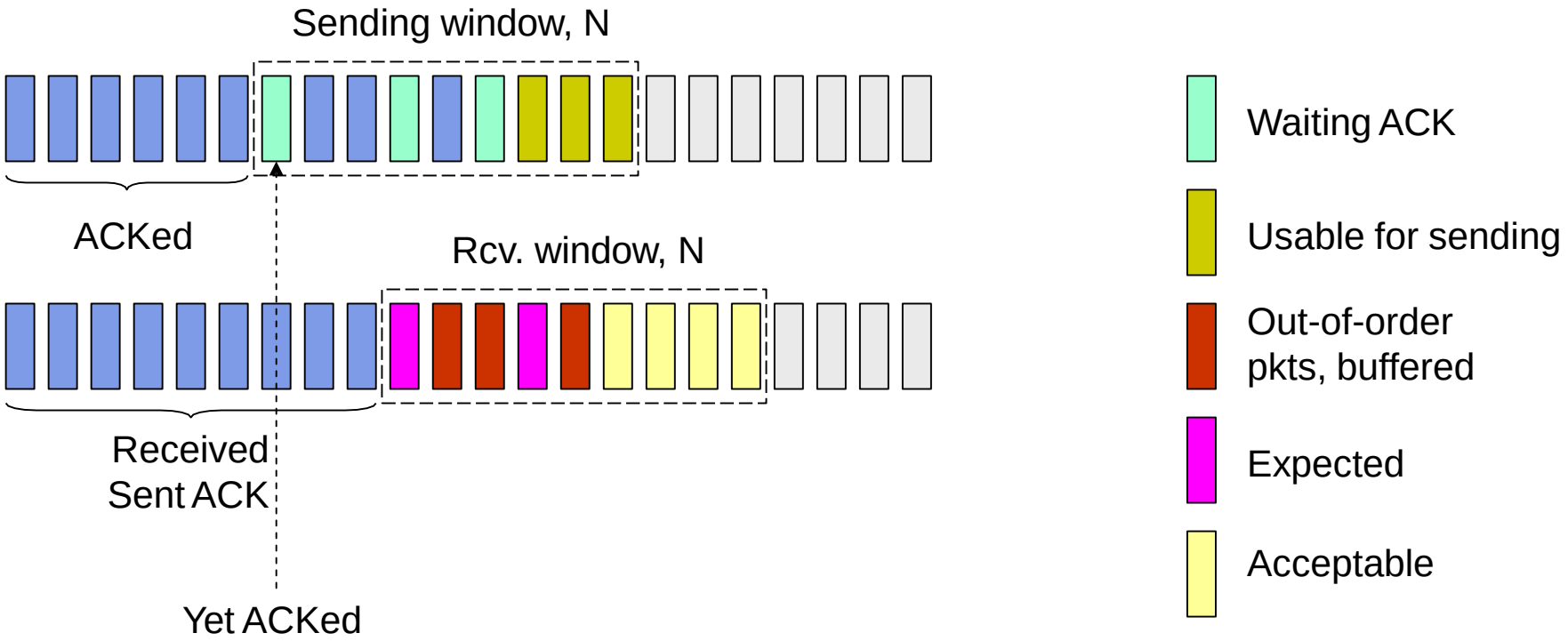
Receiver

- Chỉ gửi 1 xác nhận ACK cho gói tin có số hiệu lớn nhất đã nhận được theo đúng thứ tự.
- Với các gói tin không theo thứ tự:
  - Hủy bỏ -> không lưu vào vùng đệm
  - Xác nhận lại gói tin với số hiệu lớn nhất còn đúng thứ tự

Ví dụ về GBN

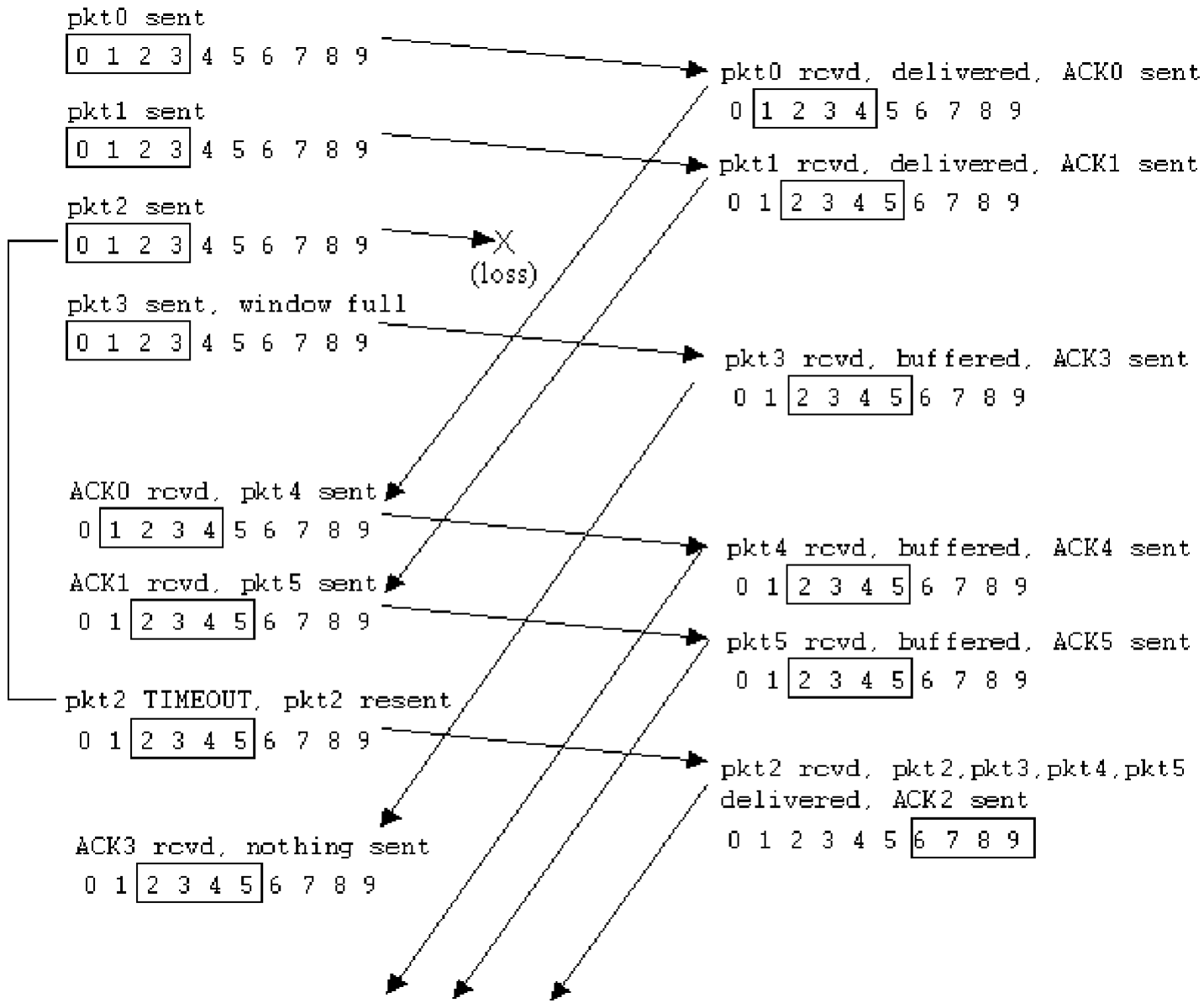

sender
receiver
send pkt0
send pkt1
send pkt2
(loss)
send pkt3
(wait)
rcv pkt0
send ACK0
rcv pkt1
send ACK1
rcv pkt3, discard
send ACK1
rcv ACK0
send pkt4
rcv ACK1
send pkt5
rcv pkt4, discard
send ACK1
rcv pkt5, discard
send ACK1
pkt2 timeout
send pkt2
send pkt3
send pkt4
send pkt5
rcv pkt2, deliver
send ACK2
rcv pkt3, deliver
send ACK3

Selective Repeat



- Bên nhận xác nhận riêng rẽ cho từng gói tin
- Chỉ gửi lại các gói tin chưa được xác nhận bị timeout
- Tổ chức vùng đệm để sắp xếp các gói tin theo đúng thứ tự để chuyển cho tầng trên

Ví dụ về Selective Repeat



So sánh 2 phương pháp

	GO-BACK-N	SELECTIVE REPEAT
Nguyên lý	Truyền lại tất cả frame sau frame lớn nhất	Chỉ truyền lại frame được cho là bị mất
Bandwidth	Tốn bang thông	Ít tốn băng thông
Complexity	Đơn giản	Phức tạp
Window size	N-1	<= (N+1)/2
Sorting	Không cần sắp xếp ở cả 2 phía truyền và nhận. Không cần lưu thông tin frames sau frame bị mất	Bên nhận phải sắp xếp được. Lưu thông tin frame trong buffer
Searching	Không có tính năng tìm kiếm	Bên gửi phải có tính năng tìm kiếm

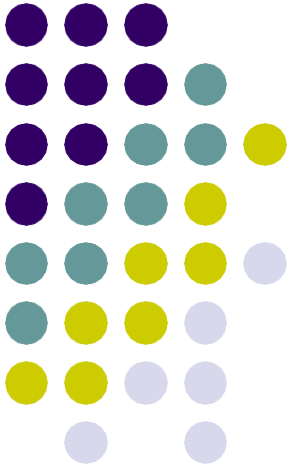
TCP
Transmission Control Protocol

Cấu trúc đoạn tin TCP

Quản lý liên kết

Kiểm soát luồng

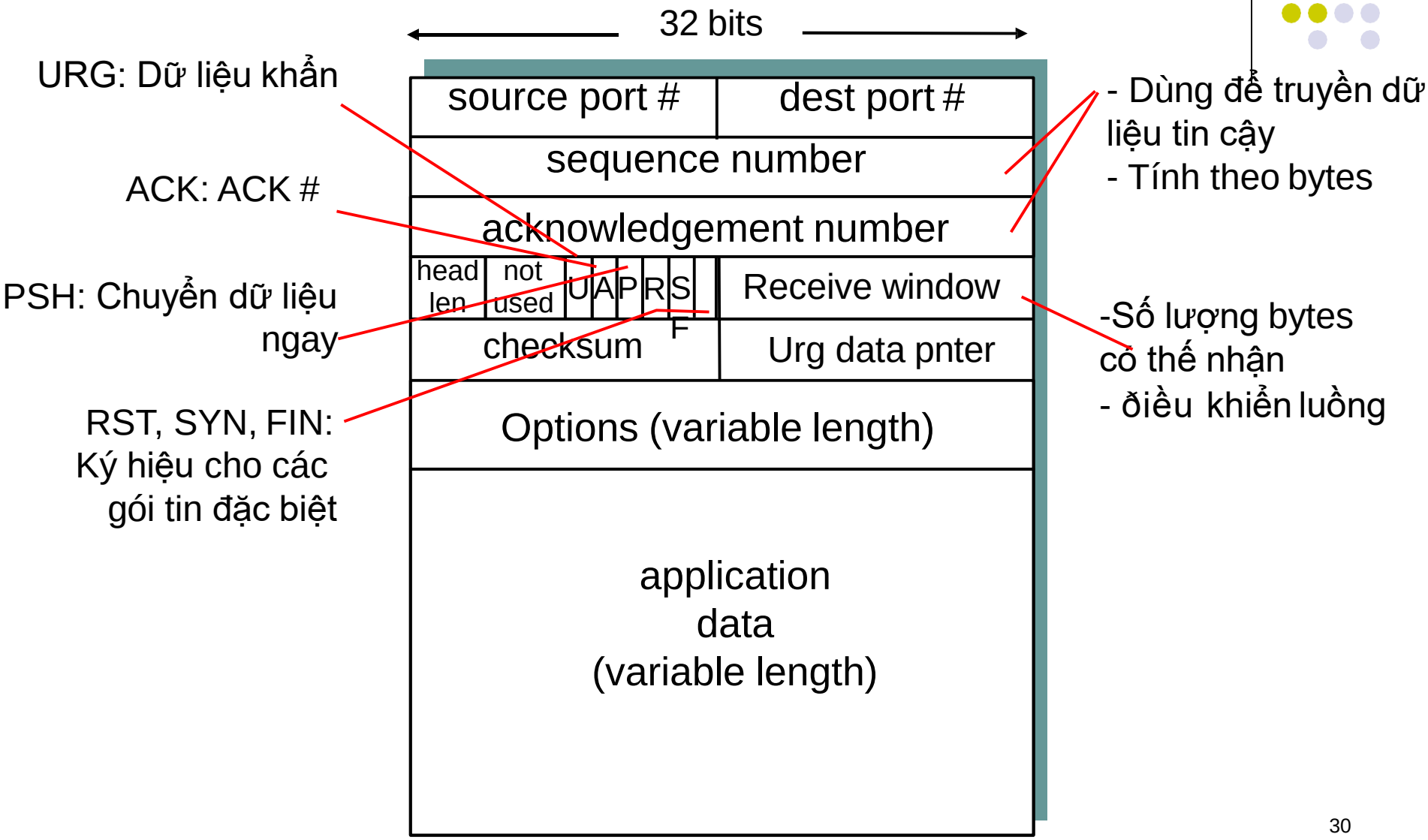
Kiểm soát tắc nghẽn



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
  - 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

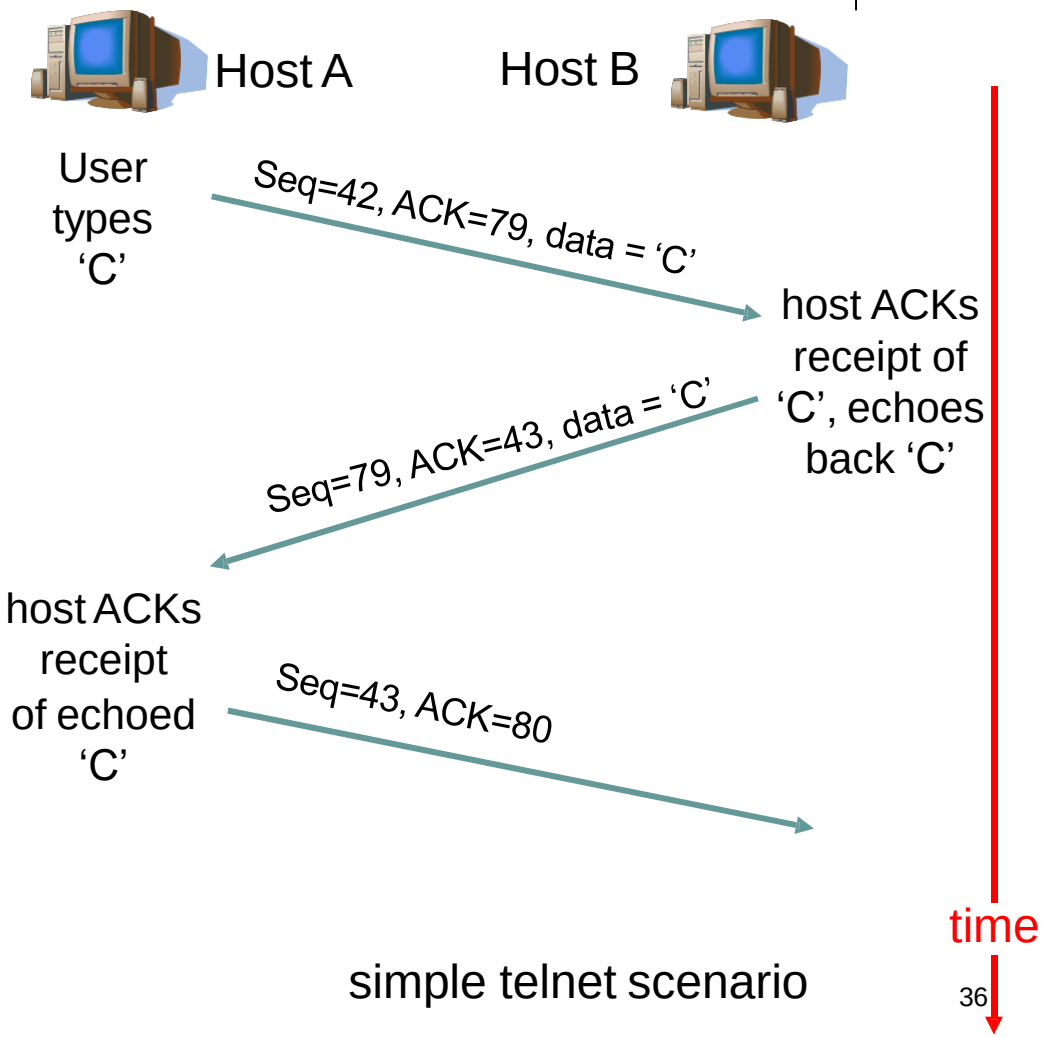
Cơ chế báo nhận trong TCP

Seq. #:

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

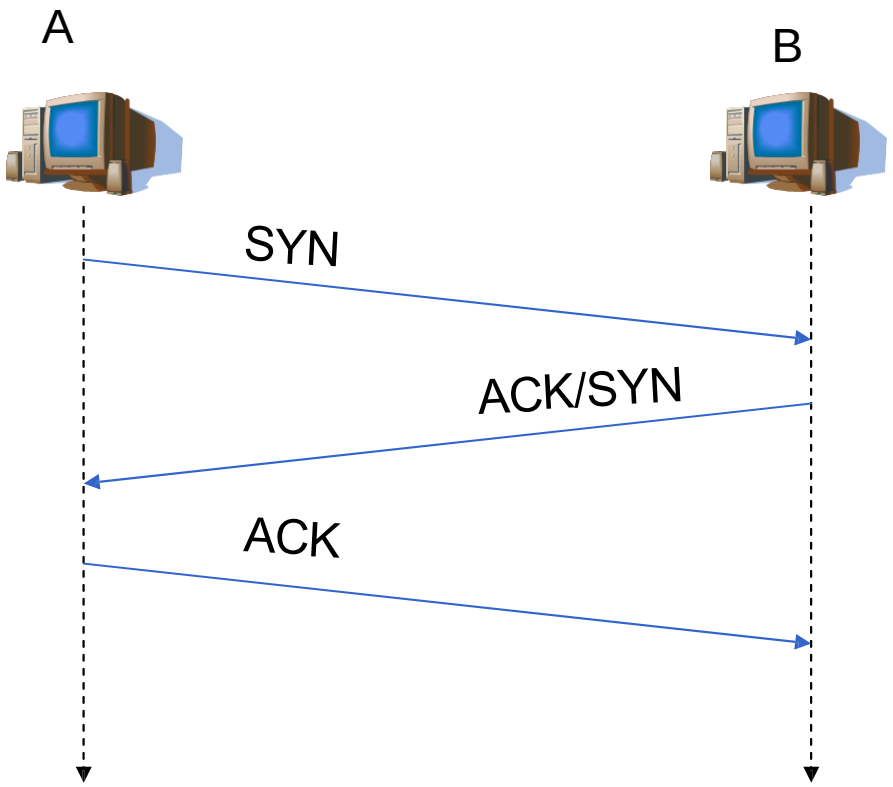
ACK:

- Số hiệu byte đầu tiên mong muốn nhận từ đối tác



simple telnet scenario

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

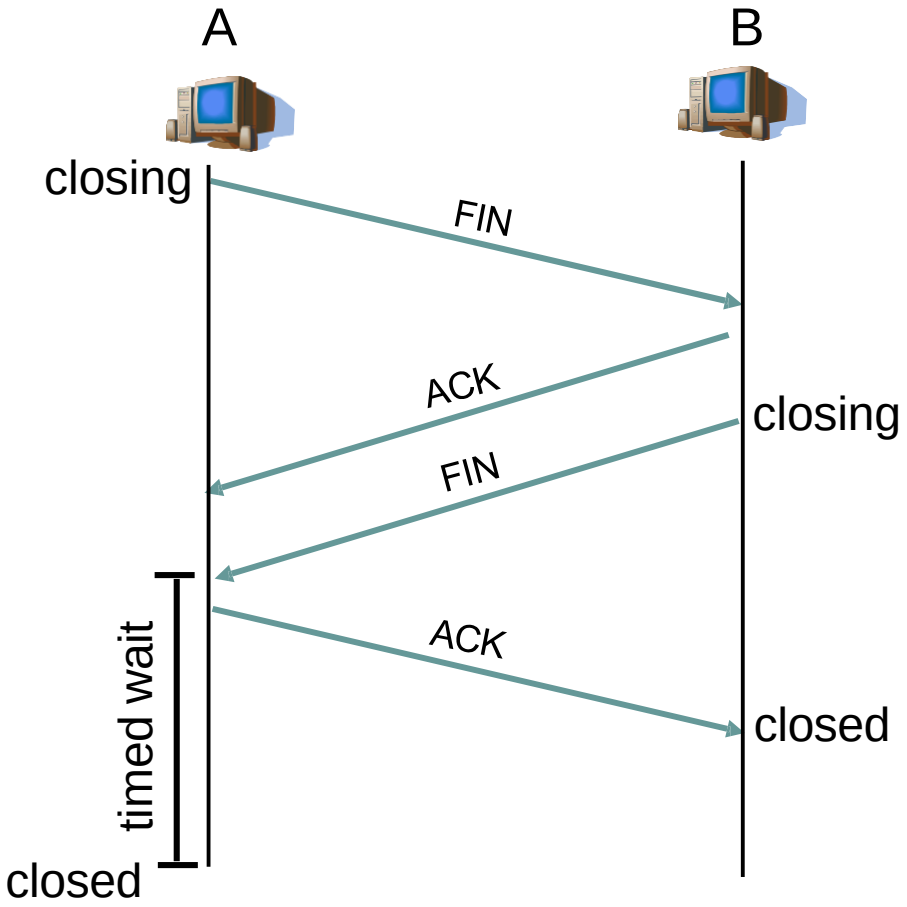


- Bước 1: A gửi SYN cho B
  - chỉ ra giá trị khởi tạo seq # của A
  - không có dữ liệu
- Bước 2: B nhận SYN, trả lời bằng SYNACK
  - B khởi tạo vùng đệm
  - chỉ ra giá trị khởi tạo seq. # của B
- Bước 3: A nhận SYNACK, trả lời ACK, có thể kèm theo dữ liệu

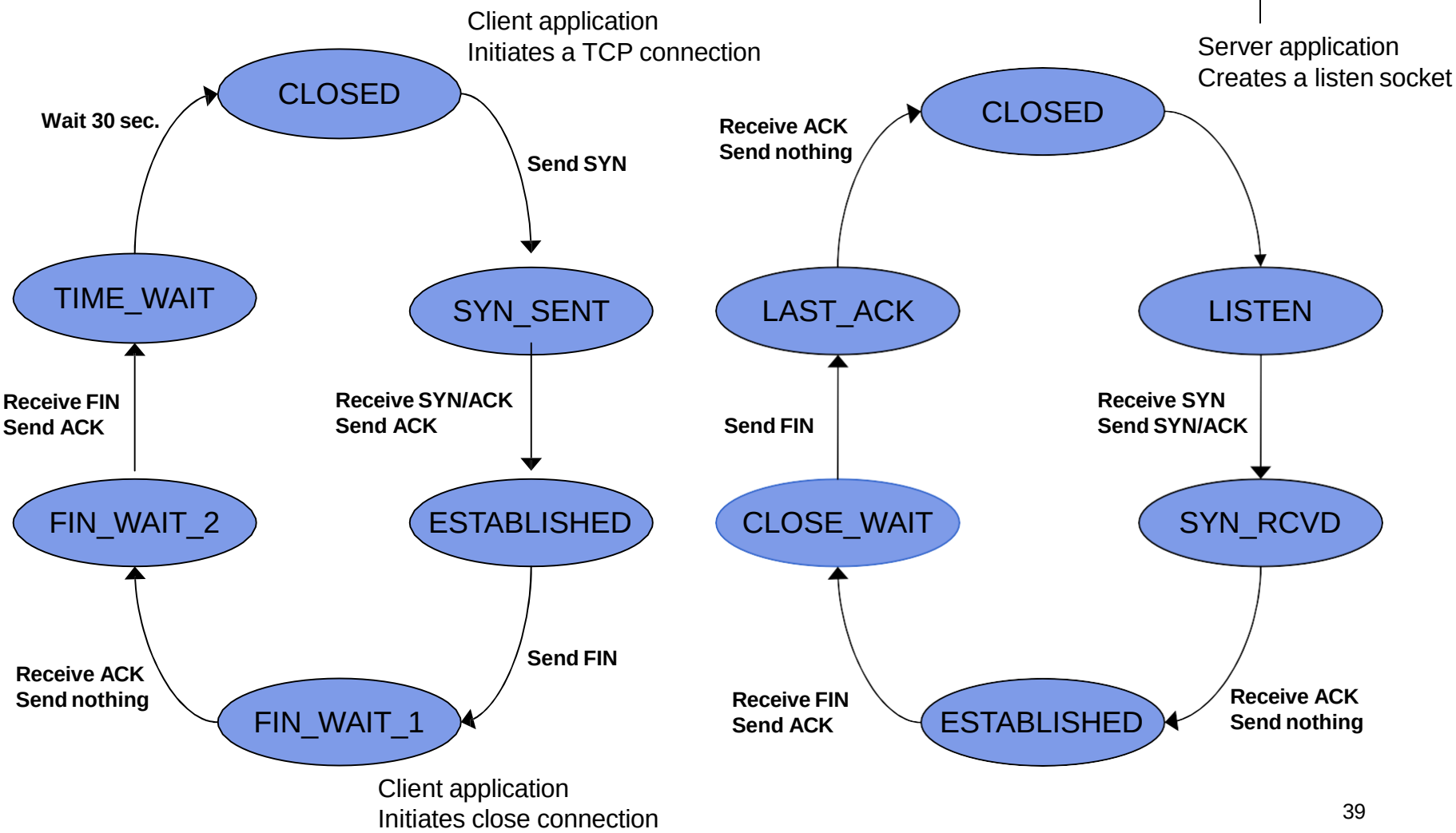
Ví dụ về việc đóng liên kết

- 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.

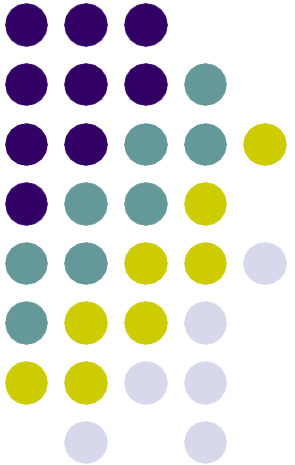
Lưu ý: Cả hai bên đều có thể chủ động đóng liên kết



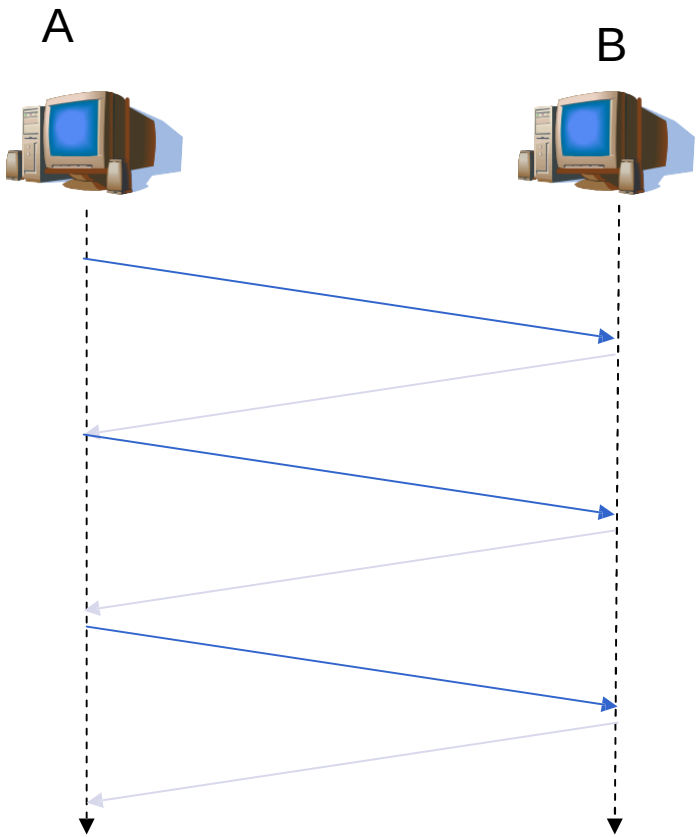
Chu trình sống của TCP (đơn giản hóa)



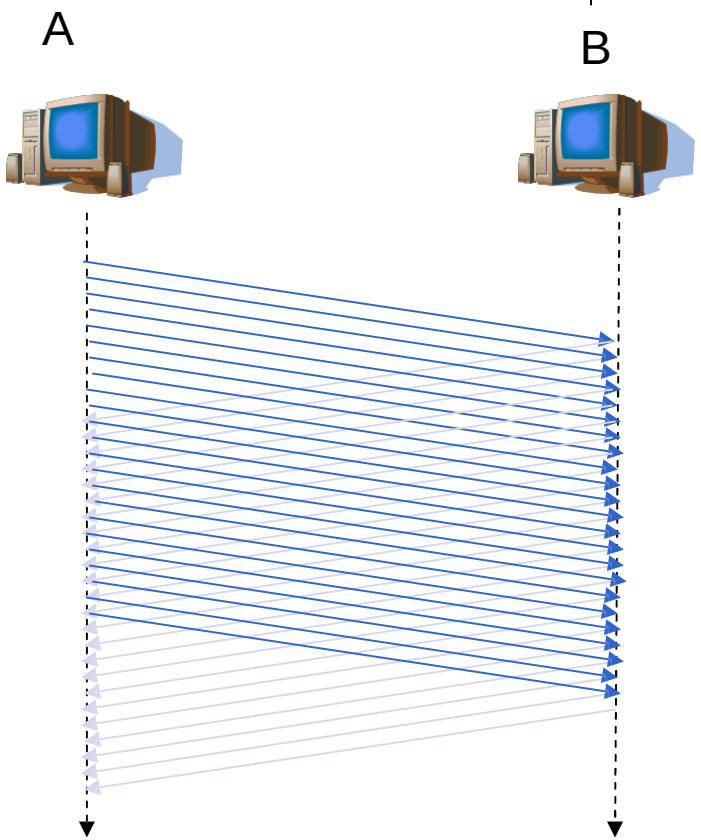
Kiểm soát luồng



Kiểm soát luồng – Quá tải



Chậm

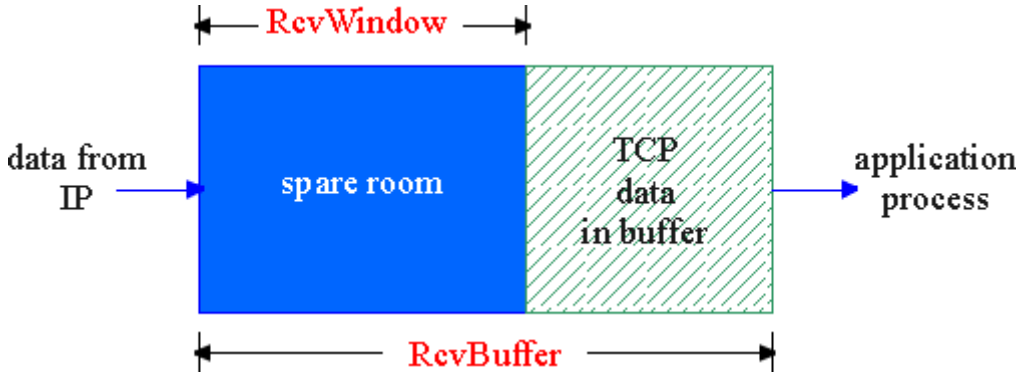


Quá tải

Kiểm soát luồng – Nguyên lý

- điều khiển lượng dữ liệu được gửi đi
  - Bảo đảm rằng hiệu quả là tốt
  - Không làm quá tải các bên
- 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)

Kiểm soát luồng trong TCP

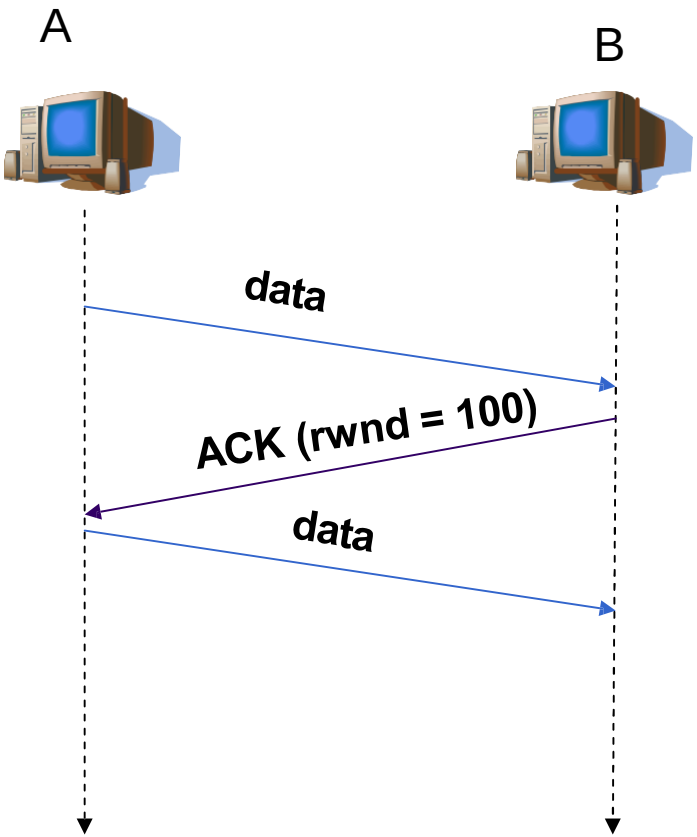


- Kích thước vùng đệm trống

= Rwnd

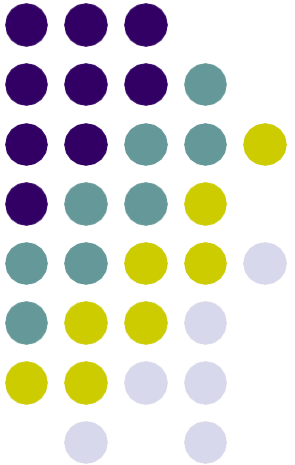
= **RcvBuffer-[LastByteRcvd - LastByteRead]**

Trao đổi thông tin về Rwnd



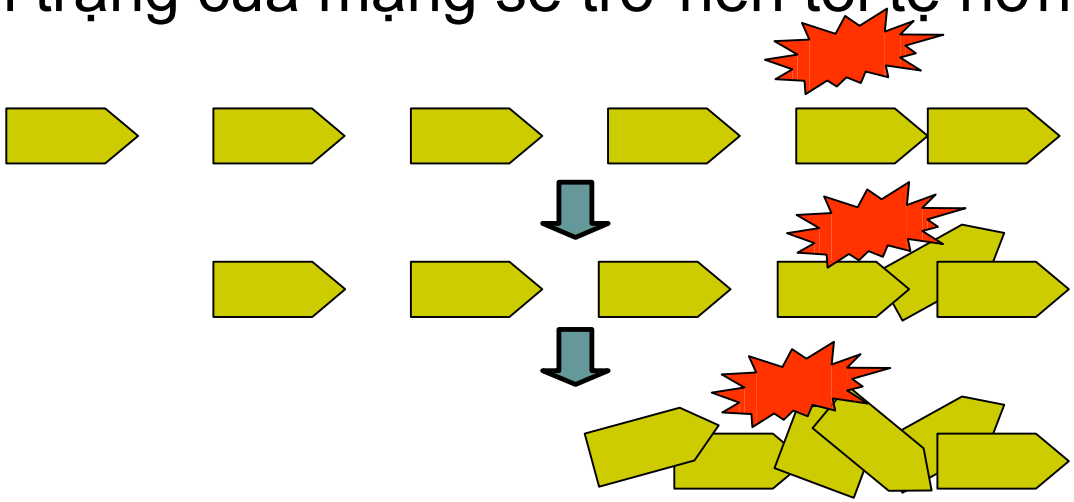
- Bên nhận sẽ báo cho bên gửi biết Rwnd trong các đoạn tin
- Bên gửi đặt kích thước cửa sổ gửi theo Rwnd

điều khiển tắc nghẽn trong TCP



Tổng quan về tắc nghẽn

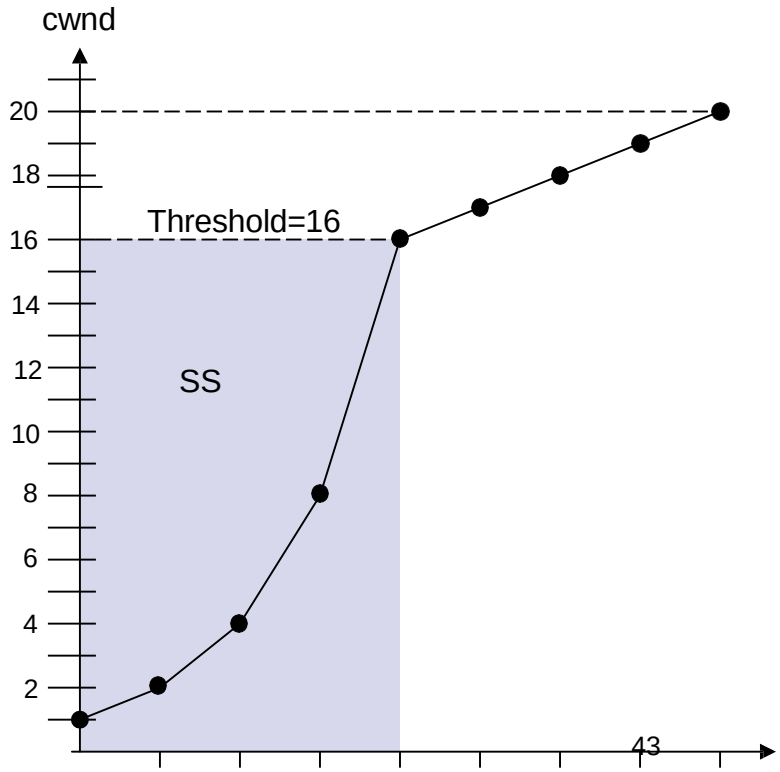
- Khi nào tắc nghẽn xảy ra ?
  - Quá nhiều cặp gửi-nhận trên mạng
  - Truyền quá nhiều làm cho mạng quá tải
- Hậu quả của việc nghẽn mạng
  - Mất gói tin
  - Thông lượng giảm, độ trễ tăng
  - Tình trạng của mạng sẽ trở nên tồi tệ hơn.



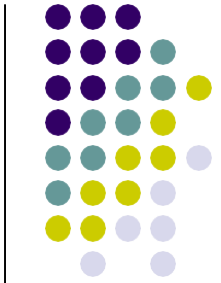
Congestion occur

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
  - Tăng dẫn tốc độ theo hàm tuyến tính cho đến khi phát hiện tắc nghẽn
- Phát hiện tắc nghẽn
  - Nếu gói tin bị mất

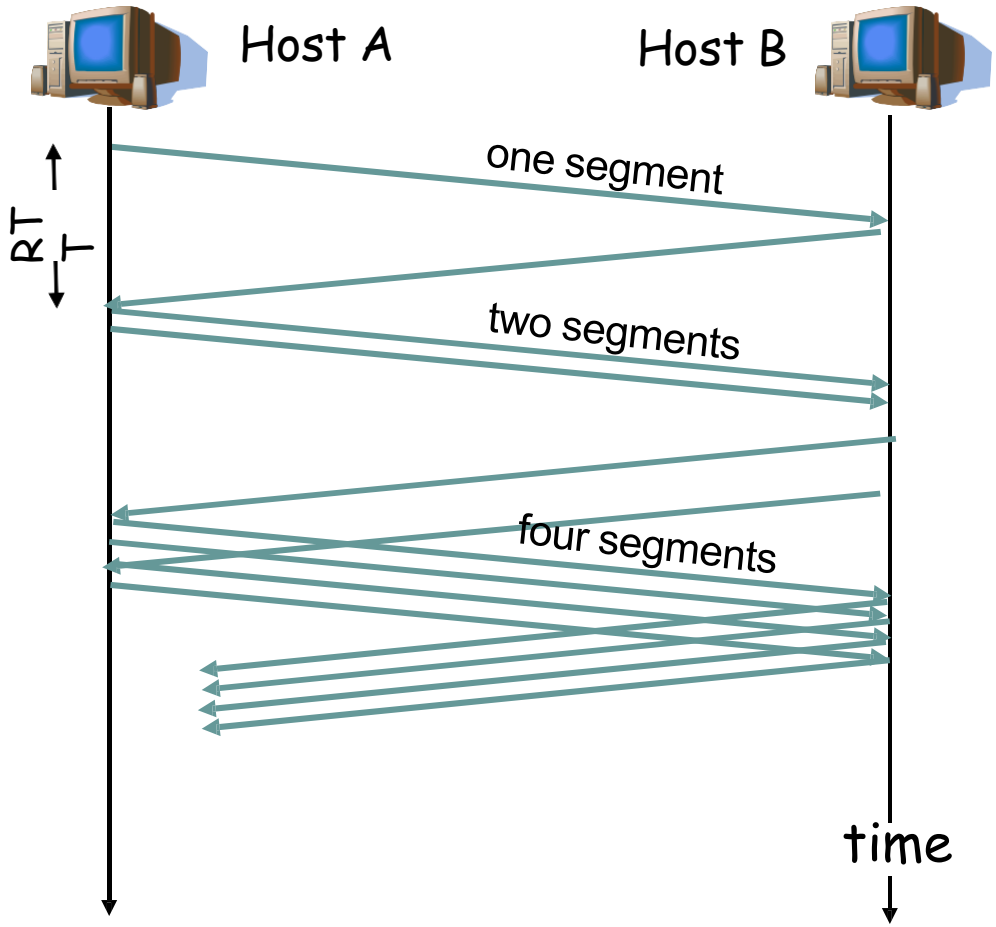


TCP Slow Start (1)



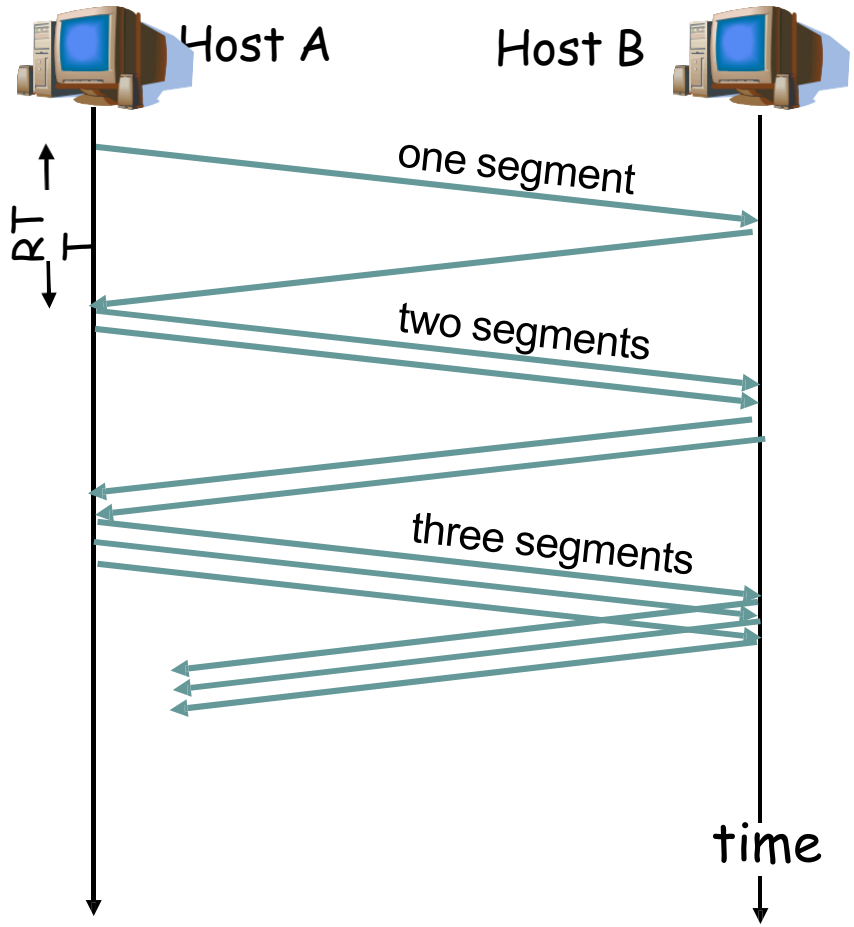
- Ý tưởng cơ bản
  - đặt cwnd bằng 1 MSS (Maximum segment size)
  - Tăng cwnd lên gấp đôi
    - Khi nhận được ACK
  - Bắt đầu chậm, nhưng tăng theo hàm mũ
- Tăng cho đến một ngưỡng: ssthresh
  - Sau đó, TCP chuyển sang trạng thái tránh tắc nghẽn

TCP Slow Start (2)



Tránh tắc nghẽn - Congestion avoidance

- ý tưởng cơ bản
  - Tăng cwnd theo cấp số cộng sau khi nó đạt tới ssthresh
  - Khi bên gửi nhận được ACK
    - Tăng cwnd thêm 1 MSS



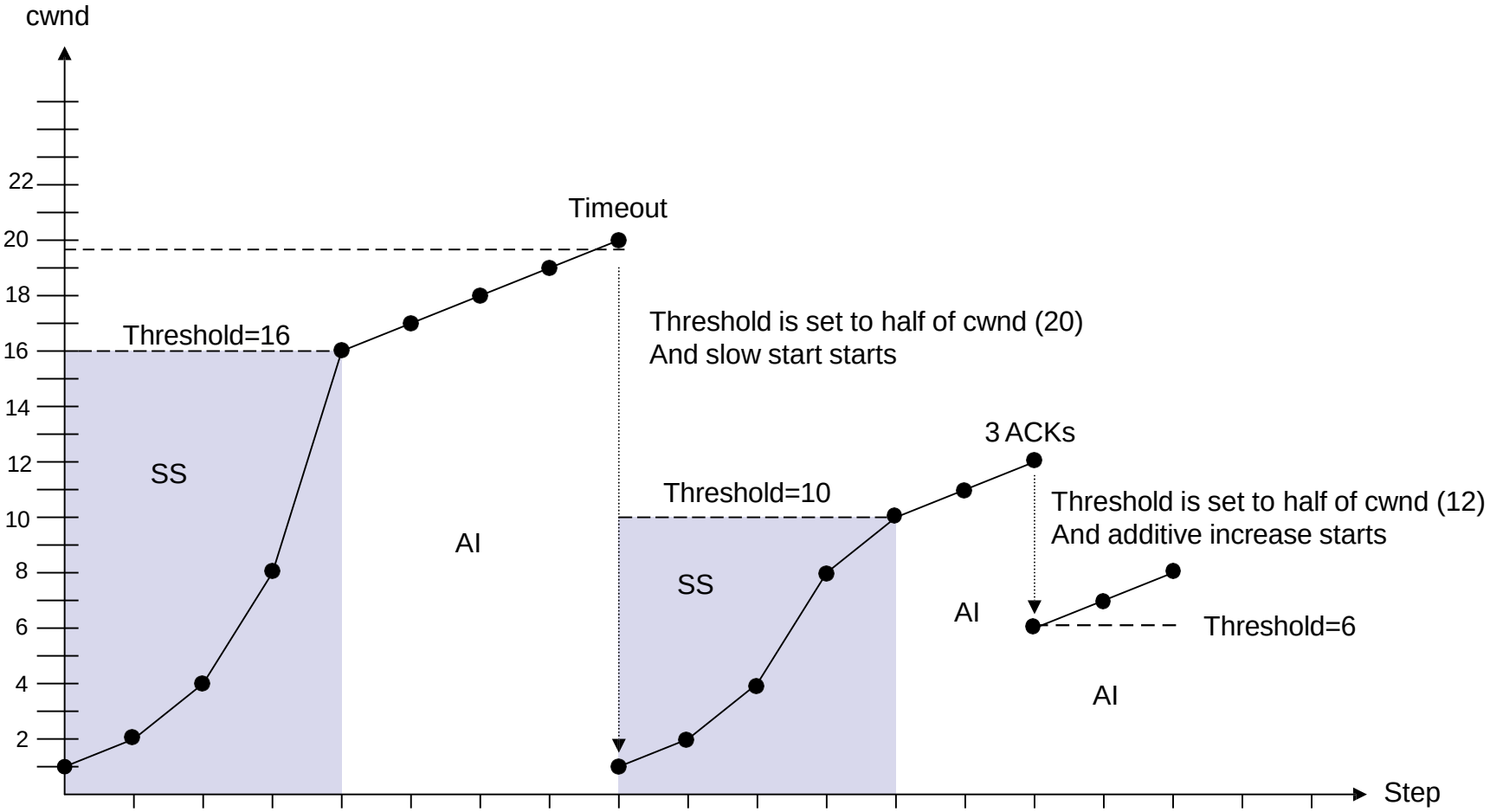
Phản ứng của TCP (1)

- Giảm tốc độ gửi
- 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 gói tin số hiệu gói tin trong ACK

Phản ứng của TCP (2)

- Khi có timeout của bên gửi
  - TCP đặt ngưỡng 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 ngưỡng xuống còn một nửa giá trị hiện tại của cwnd
  - TCP đặt cwnd về giá trị hiện tại của ngưỡng cũ
  - TCP chuyển trạng thái “congestion avoidance”

Kiểm soát tắc nghẽn – minh họa



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