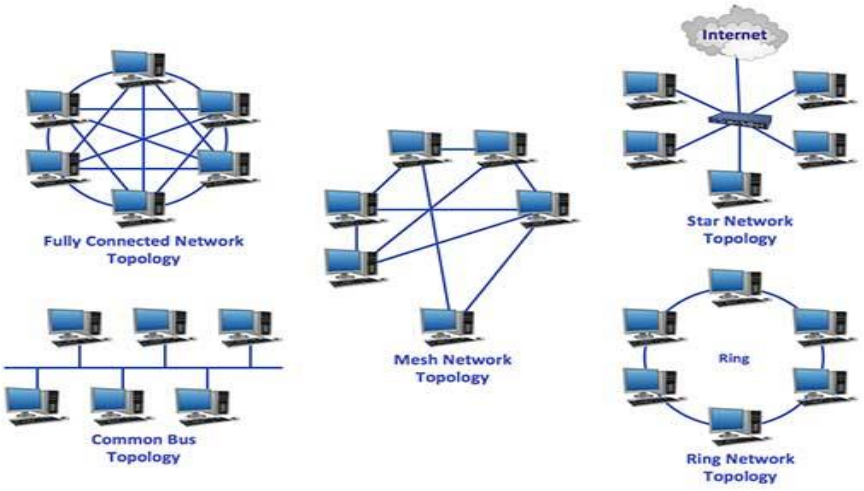
MẠNG MÁY TÍNH

CHƯƠNG 7. TẦNG VẬN CHUYỂN



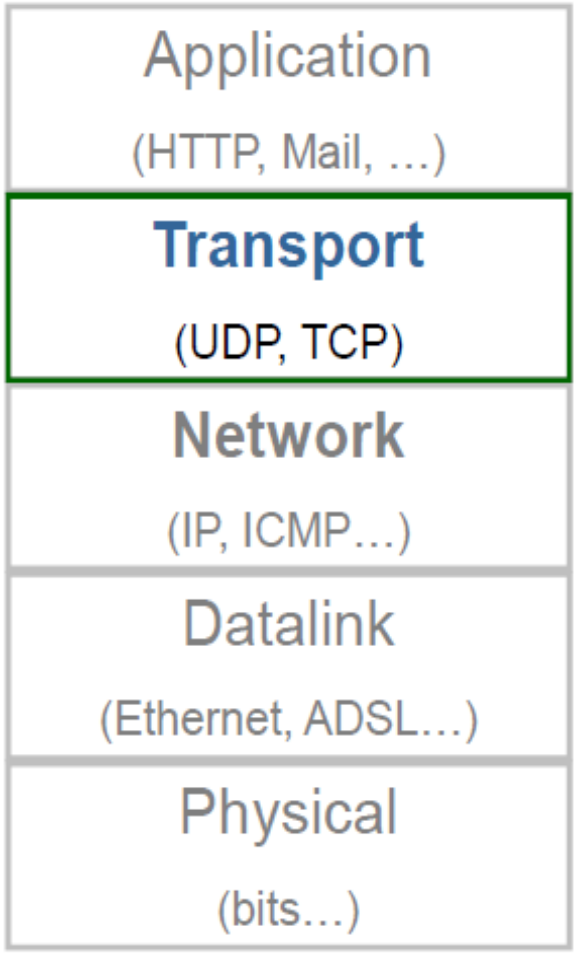
NỘI DUNG

- Tổng quan
- Chức năng
- UDP
- Vấn đề vận chuyển dữ liệu an toàn
- TCP

NỘI DUNG

- **Tổng quan**
- Chức năng
- UDP
- Vấn đề vận chuyển dữ liệu an toàn
- TCP

Giới thiệu



Hỗ trợ các ứng dụng trên mạng

Điều khiển truyền dữ liệu giữa các tiến trình của tầng 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

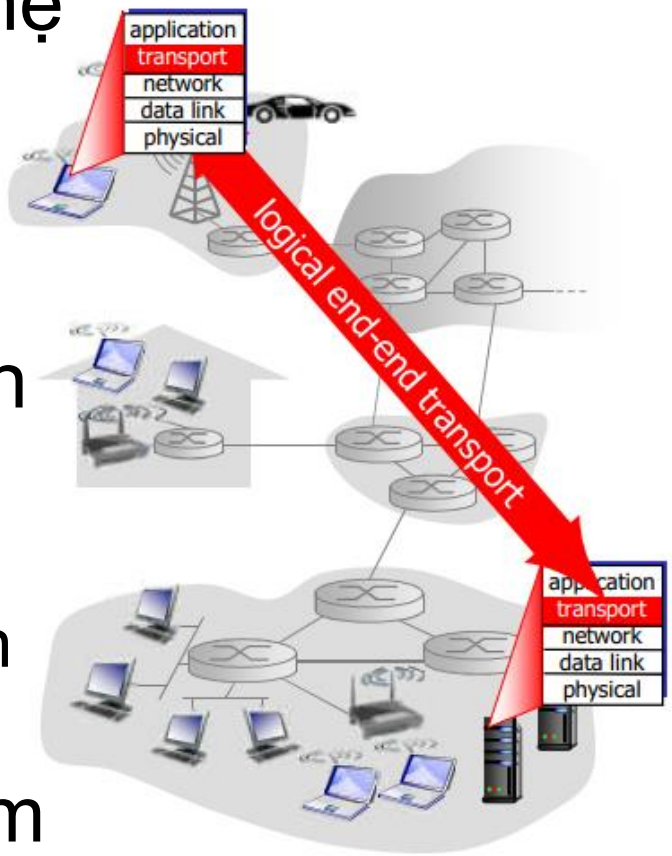
Truyền và nhận dòng bit trên đường truyền vật lý

Tổng quan

- Cung cấp phương tiện truyền giữa các ứng dụng cuối
- Bên gửi:
  - Nhận dữ liệu từ ứng dụng
  - Đặt dữ liệu vào các gói 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

- Được cài đặt trên các hệ thống cuối
- Không cài đặt trên các routers, switches…
- 2 dạng dịch vụ giao vận
  - Tin cậy, hướng liên kết: TCP
  - Không tin cậy, không liên kết: UDP
- Đơn vị truyền: datagram (UDP), segment (TCP)



Tại sao cần 2 dạng dịch vụ giao vận (TCP, UDP)

- Từ 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

Các ứng dụng và dịch vụ

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

NỘI DUNG

- Tổng quan
- **Chức năng**
- UDP
- Vấn đề vận chuyển dữ liệu an toàn
- TCP

Chức năng

Dồn kênh và phân kênh

- Dồn kênh (Multiplexing):
  - Thực hiện tại bên gởi
  - Thu thập dữ liệu từ các socket
  - Dán nhãn dữ liệu với 1 header
- Phân kênh (Demultiplexing):
  - Thực hiện tại bên nhận
  - Phân phối các segment nhận được cho socket tương ứng

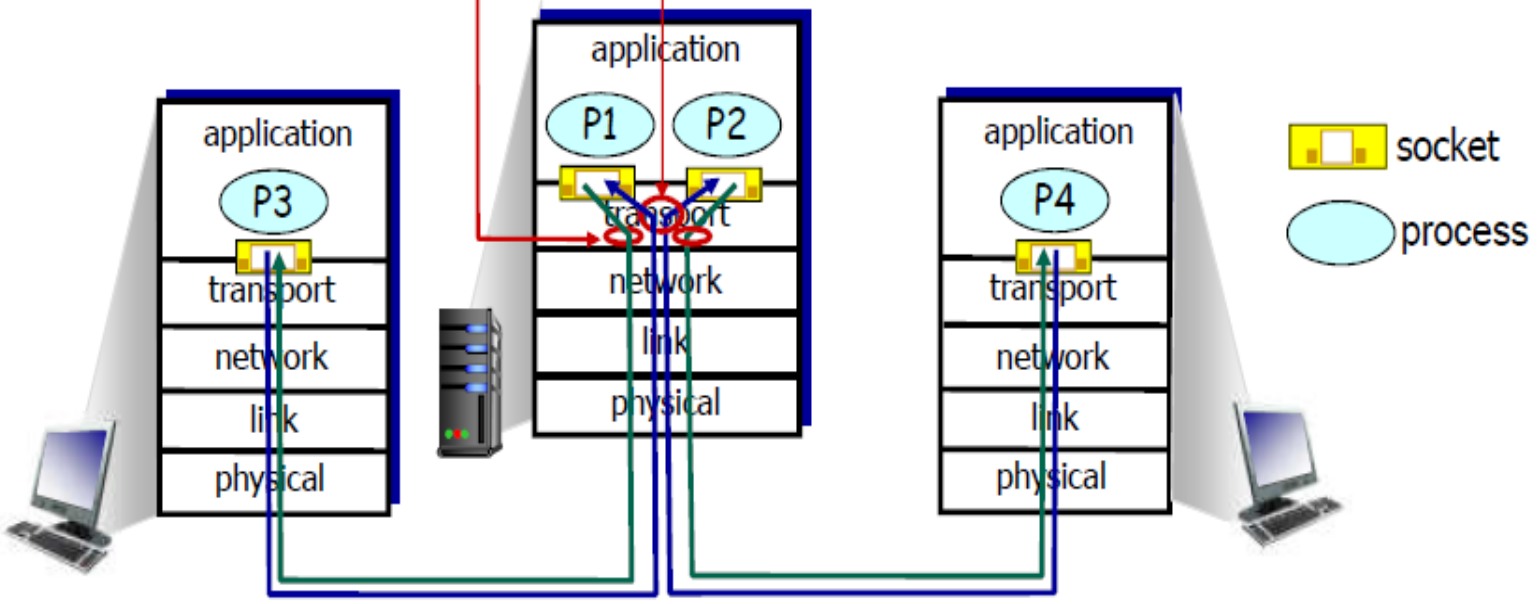
Chức năng

Gửi: Dồn kênh

Nhận dữ liệu từ các tiến trình tầng ứng dụng khác nhau (qua socket), đóng gói theo giao thức tầng giao vận và gửi trên liên kết mạng

Nhận: Phân kênh

Sử dụng thông tin trên tiêu đề gói tin để gửi dữ liệu tới đúng socket

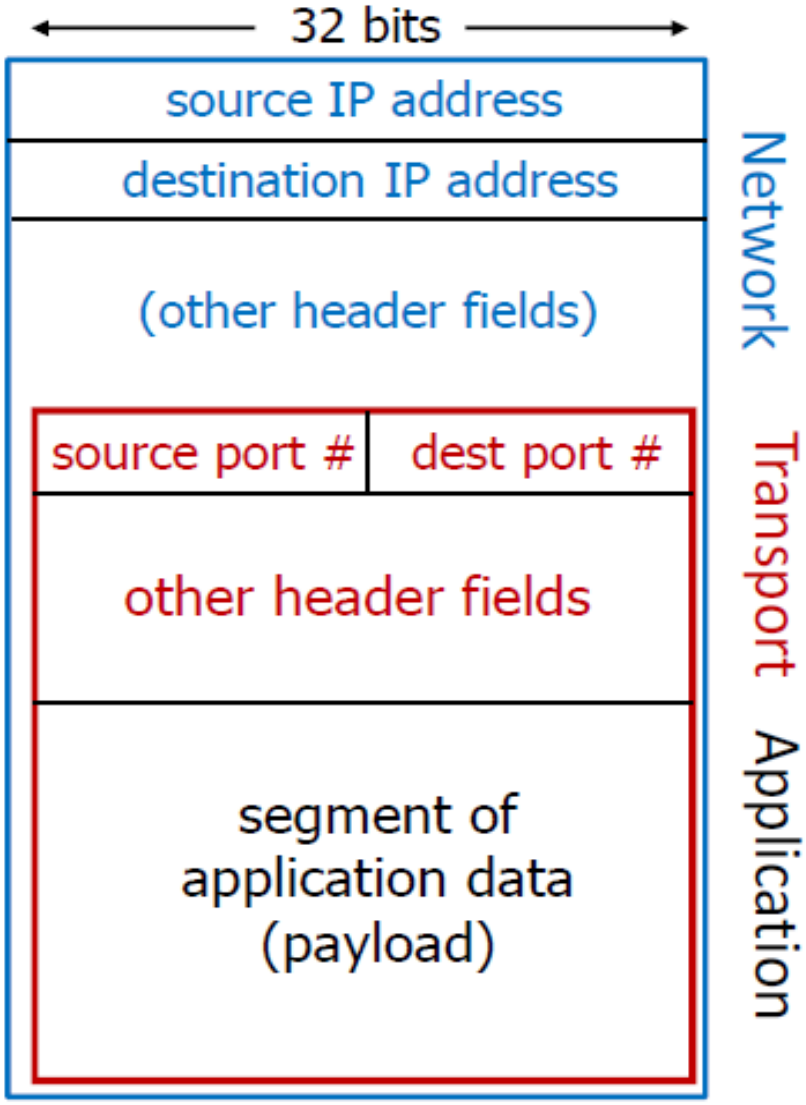


Chức năng

Mux/Demux hoạt động như thế nào?

- Nút mạng nhận gói tin với các địa chỉ:
  - Địa chỉ IP nguồn
  - Địa chỉ IP đích
  - Số hiệu cổng nguồn
  - Số hiệu cổng đích
- Địa chỉ IP và số hiệu cổng được sử dụng để xác định socket nhận dữ liệu

Chức năng


32 bits
source IP address
destination IP address
(other header fields)
source port #
dest port #
other header fields
segment of application data (payload)
Network
Transport
Application

NỘI DUNG

- Tổng quan
- Chức năng
- **UDP**
- Vấn đề vận chuyển dữ liệu an toàn
- TCP

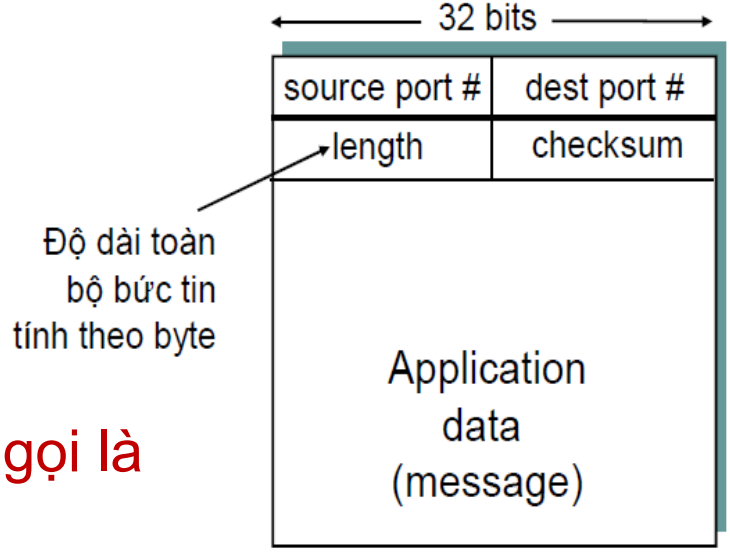
UDP – User Datagram Protocol

Đặc điểm chung

- Giao thức hướng không kết nối (connectionless)
- Truyền tin “best-effort”: chỉ gửi 1 lần, không phát lại
- Vì sao cần UDP?
  - Không cần thiết lập liên kết (giảm 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 – User Datagram Protocol

- UDP có những chức năng cơ bản gì?
  - Dồn kênh/phân kênh
  - Phát hiện lỗi bit bằng checksum



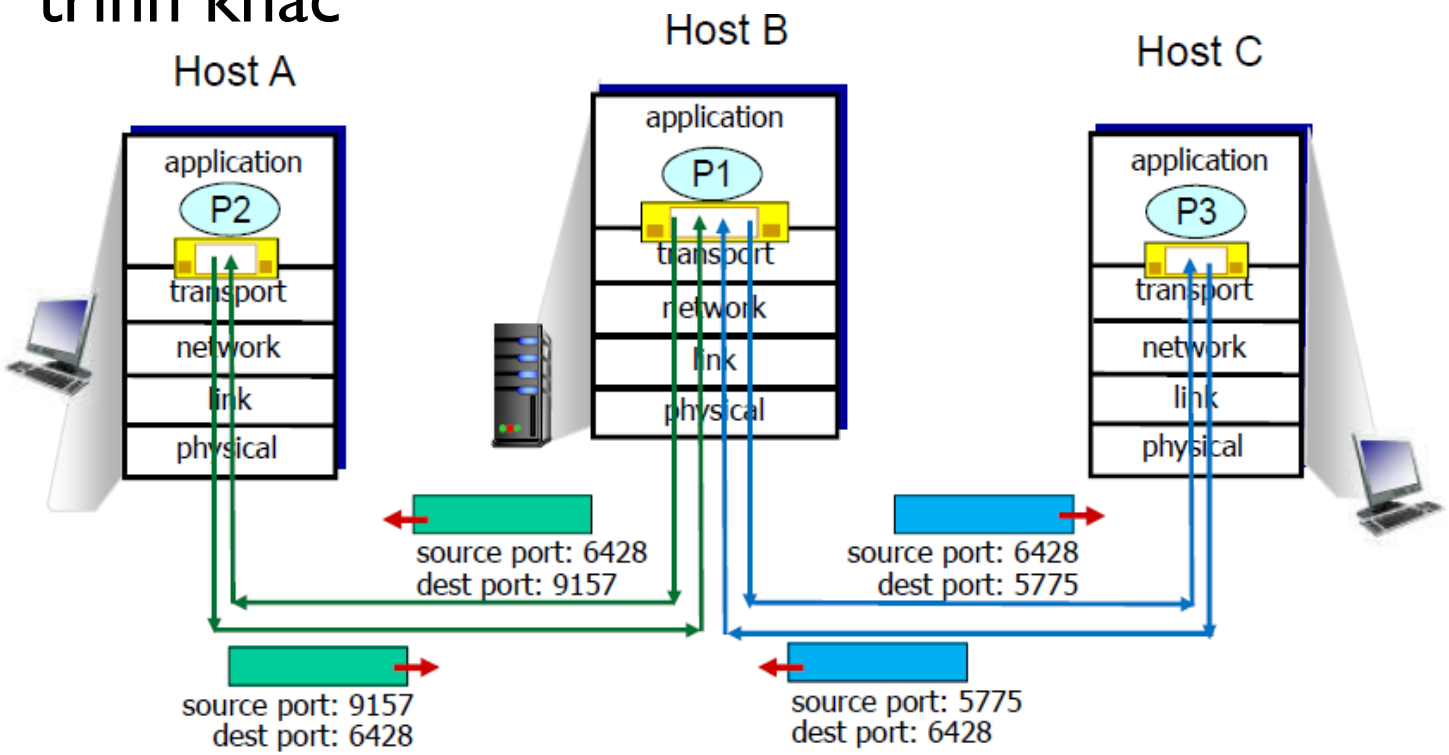
Khuôn dạng đơn vị dữ liệu của UDP

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

UDP – User Datagram Protocol

mux/demux trên ứng dụng UDP

- Mỗi tiến trình chỉ cần sử dụng một socket duy nhất để trao đổi dữ liệu với các tiến trình khác



UDP – User Datagram Protocol

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

NỘI DUNG

- Tổng quan
- Chức năng
- UDP
- **Vấn đề vận chuyển dữ liệu an toàn**
- TCP

Vấn đề truyền dữ liệu sao an toàn?

Vận chuyển dữ liệu an toàn

1. Khi kênh có lỗi bit, không bị mất tin

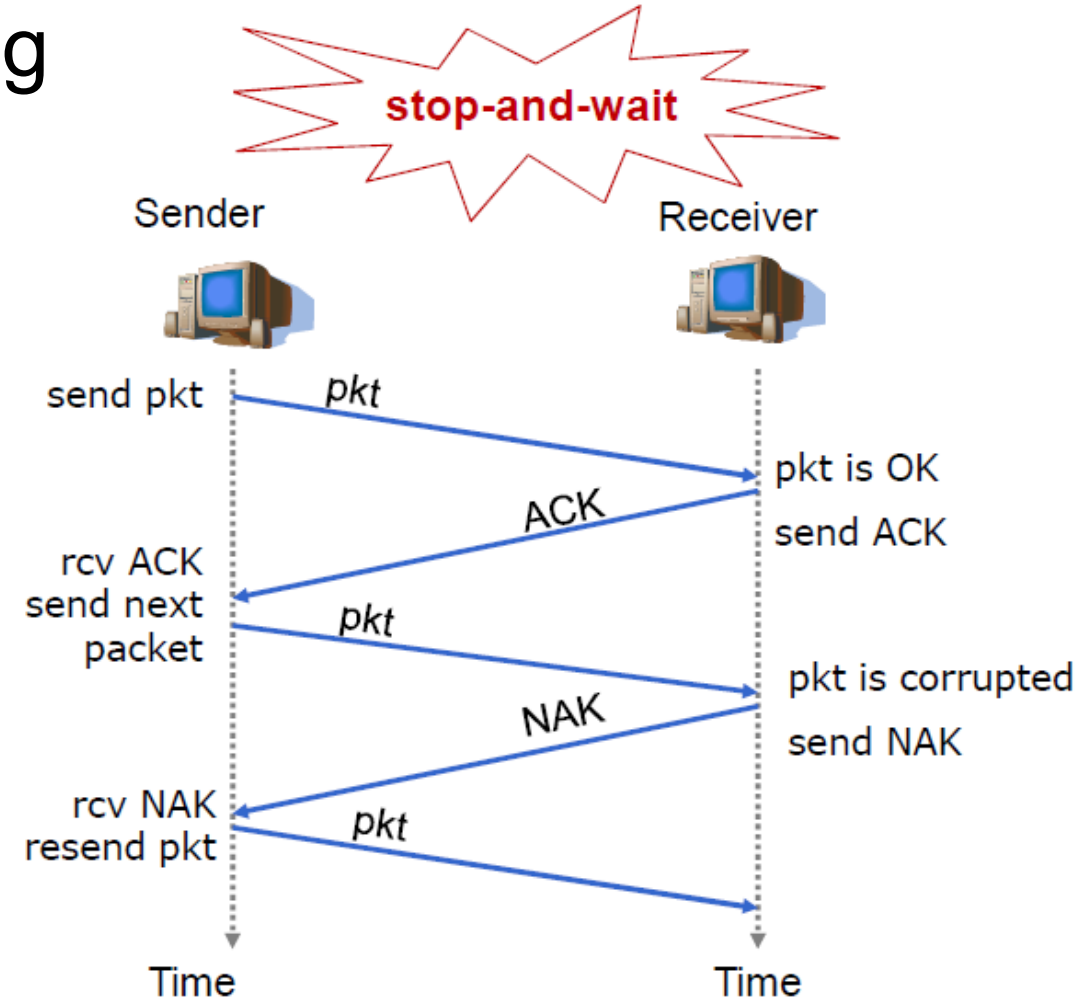
- Bên gởi
  - Gởi kèm theo thông tin kiểm tra lỗi
  - Sử dụng các phương pháp kiểm tra lỗi
    - Checksum, parity checkbit, CRC,..
- Bên nhận
  - Kiểm tra có xảy ra lỗi bit?
  - Hành động khi xảy ra lỗi bit?
    - Báo về bên gởi

Vận chuyển dữ liệu an toàn

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

Vận chuyển dữ liệu an toàn

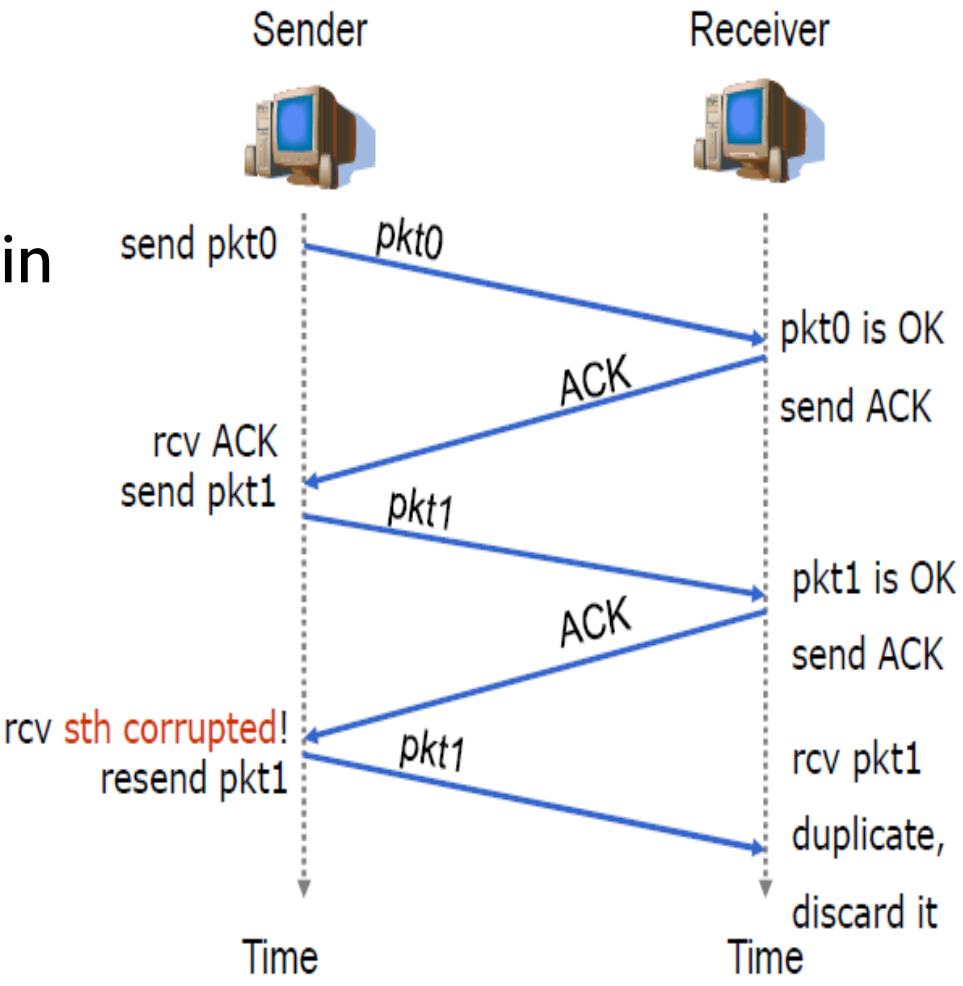
- Hoạt động



Vận chuyển dữ liệu an toàn

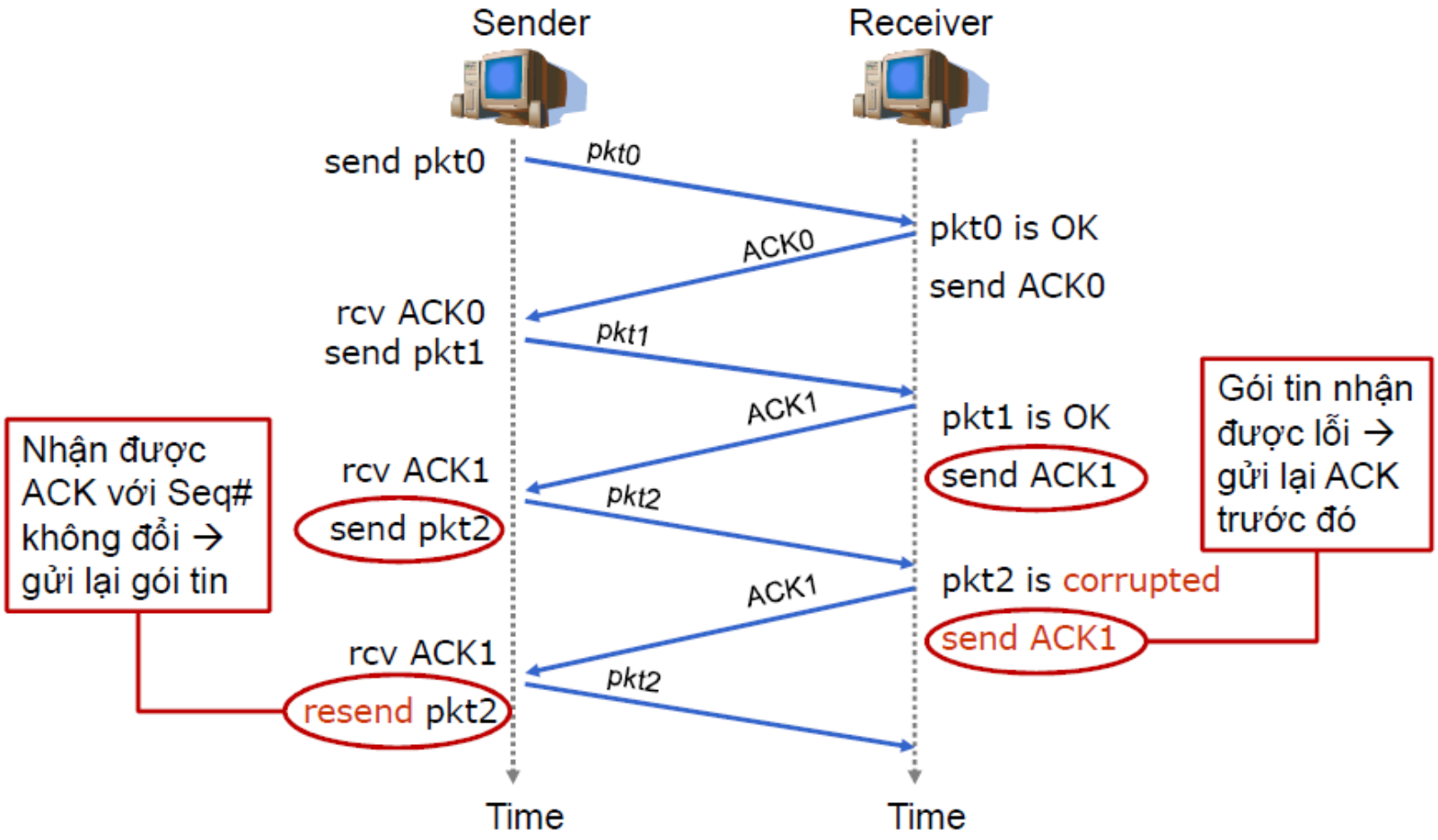
2. Lỗi ACK/NAK

- Cần truyền lại
- Xử lý việc lặp gói tin như thế nào?
- Thêm Seq



Vận chuyển dữ liệu an toàn

Giải pháp không dùng NAK



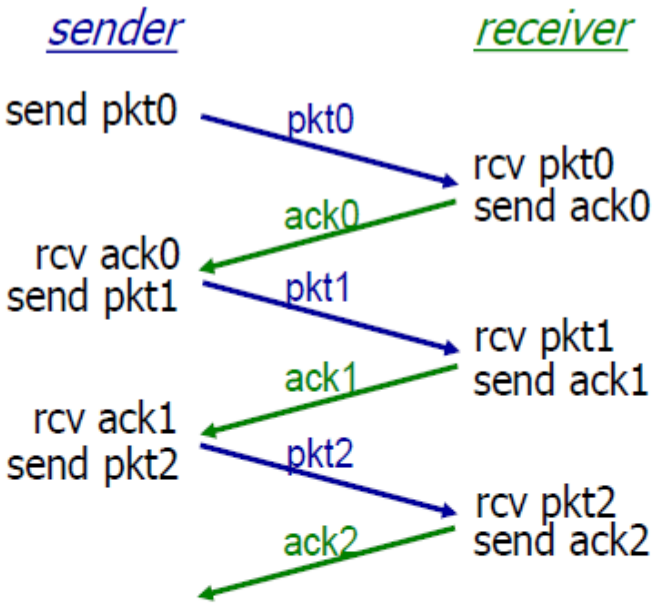
Vận chuyển dữ liệu an toàn

3. Kênh có lỗi bit và mất gói tin

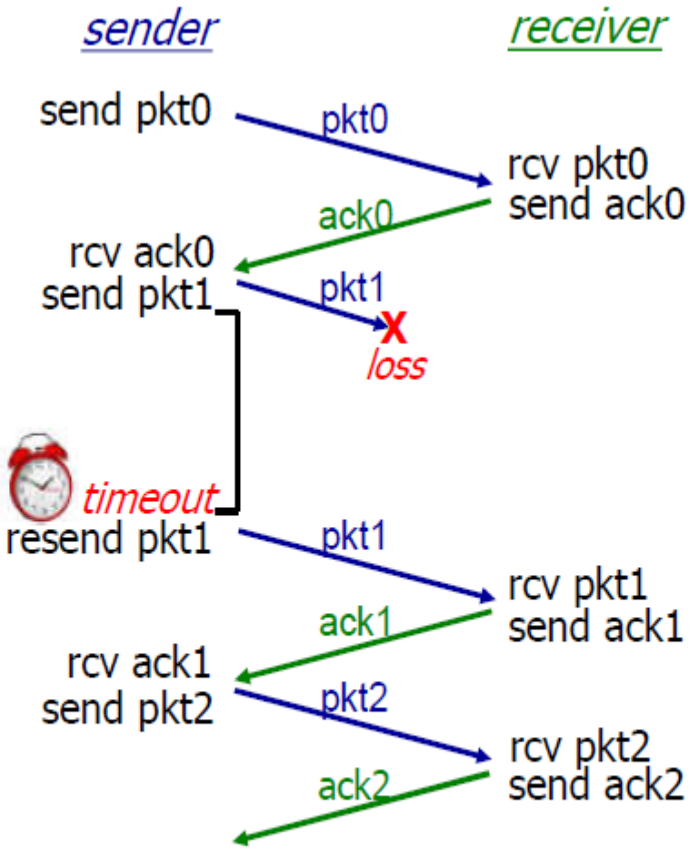
- Dữ liệu và ACK có thể bị mất
- Nếu không nhận được ACK?
  - Truyền lại như thế nào?
  - Timeout!
- Thời gian chờ là bao lâu?
  - Ít nhất là 1 RTT (Round Trip Time)
  - Mỗi gói tin gửi đi cần 1 timer
- Nếu gói tin vẫn đến đích và ACK bị mất?
  - Dùng số hiệu gói tin

Vận chuyển dữ liệu an toàn

Minh họa



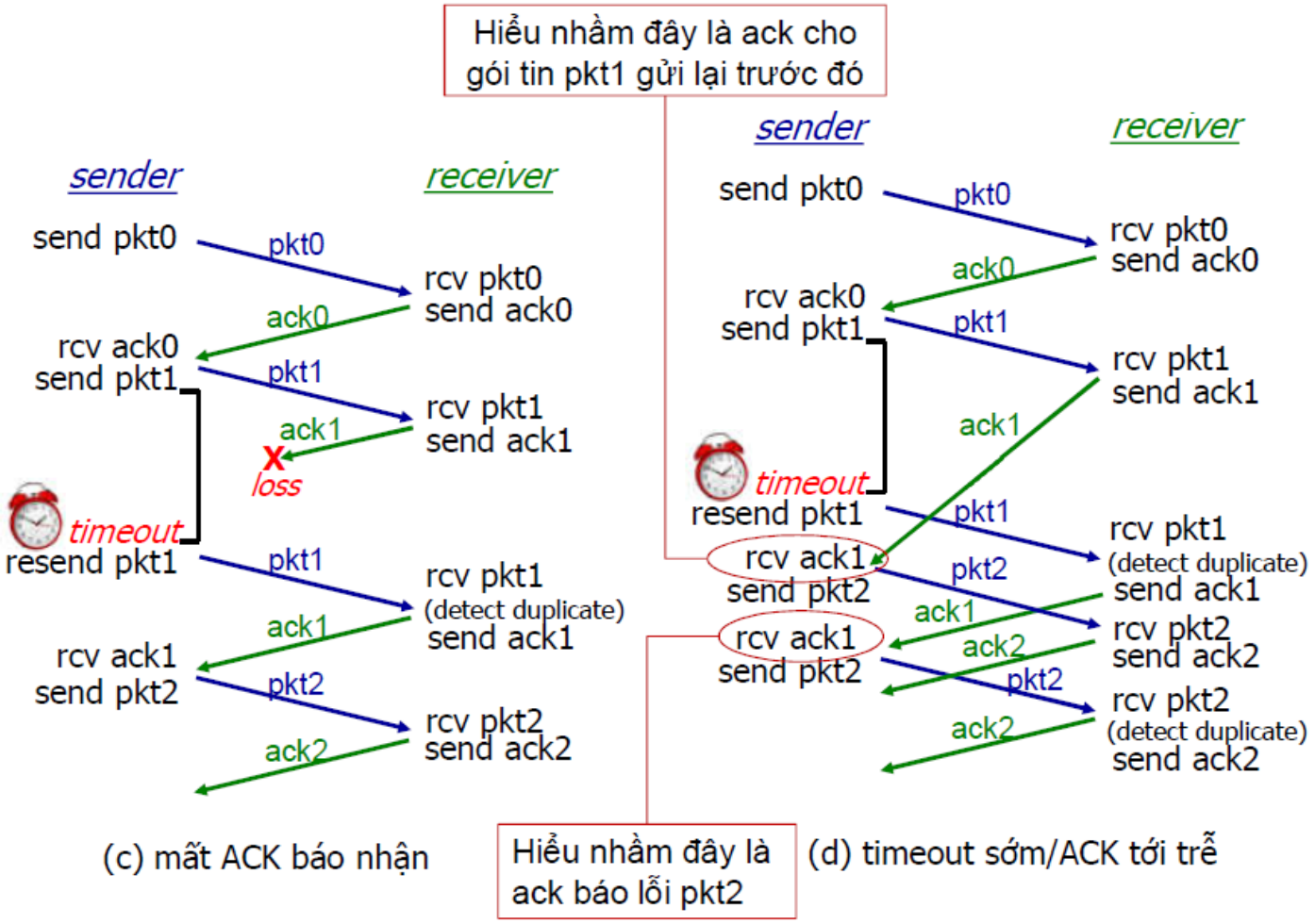
(a) Không có mất gói tin



(b) mất gói tin gửi đi

Vận chuyển dữ liệu an toàn

Minh họa



Vận chuyển dữ liệu an toàn

- Để tránh các tình huống mất dữ liệu, người ta sử dụng các cách:
  - Giao thức RDT (Reliable Data Transfer)
  - Nguyên lý Pipeline
  - Go-back n
  - Selective Repeat

Vận chuyển dữ liệu an toàn

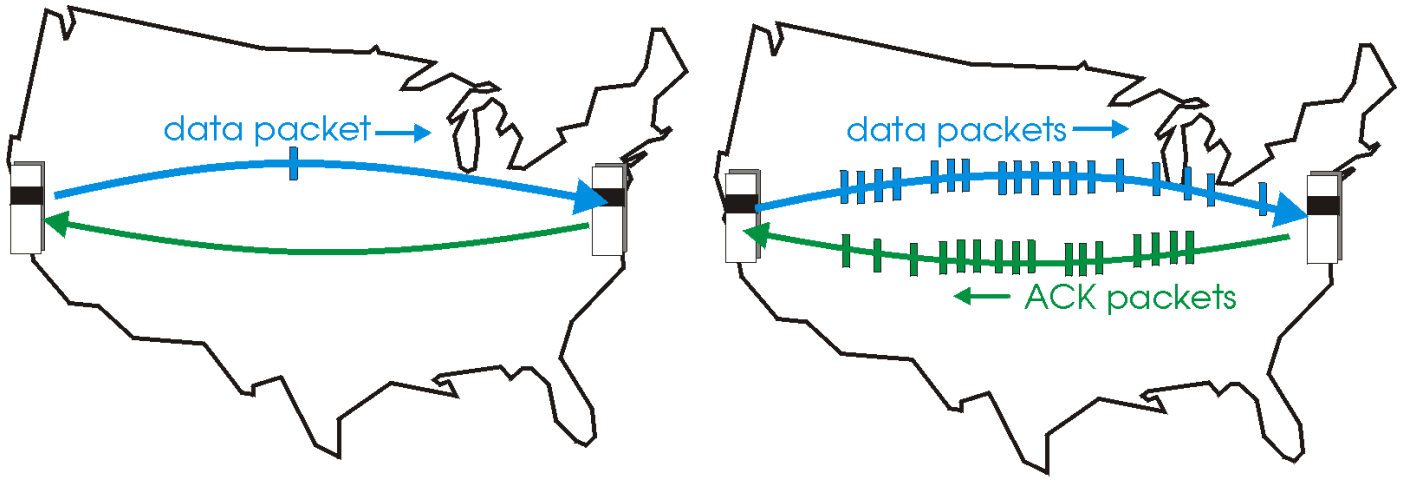
RDT = Reliable Data Transfer

- Nguyên tắc: dừng và chờ
- Bên gởi
  - Gởi gói tin kèm theo thông tin kiểm tra lỗi
  - **Dừng và chờ** đến khi nào gói tin vừa gởi đến được bên nhận **an toàn:** nhận được gói tin ACK
  - Gởi lại khi có lỗi xảy ra: lỗi bit, mất gói
- Bên nhận:
  - Kiểm tra lỗi, trùng lắp dữ liệu
  - Gởi gói tin phản hồi

Vận chuyển dữ liệu an toàn

Pipeline

- Gửi liên tục một lượng hữu hạn các gói tin mà không cần chờ ACK
  - Số thứ tự các gói tin phải tăng dần
  - Dữ liệu gửi đi chờ sẵn ở bộ đệm gửi
  - Dữ liệu tới đích chờ ở bộ đệm nhận

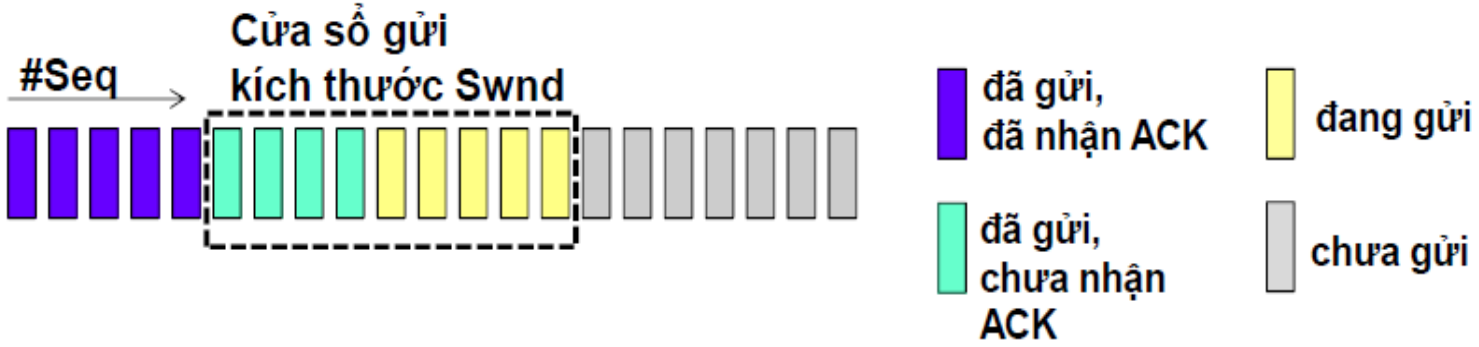


(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

Vận chuyển dữ liệu an toàn

Go-back-N



Bên gửi

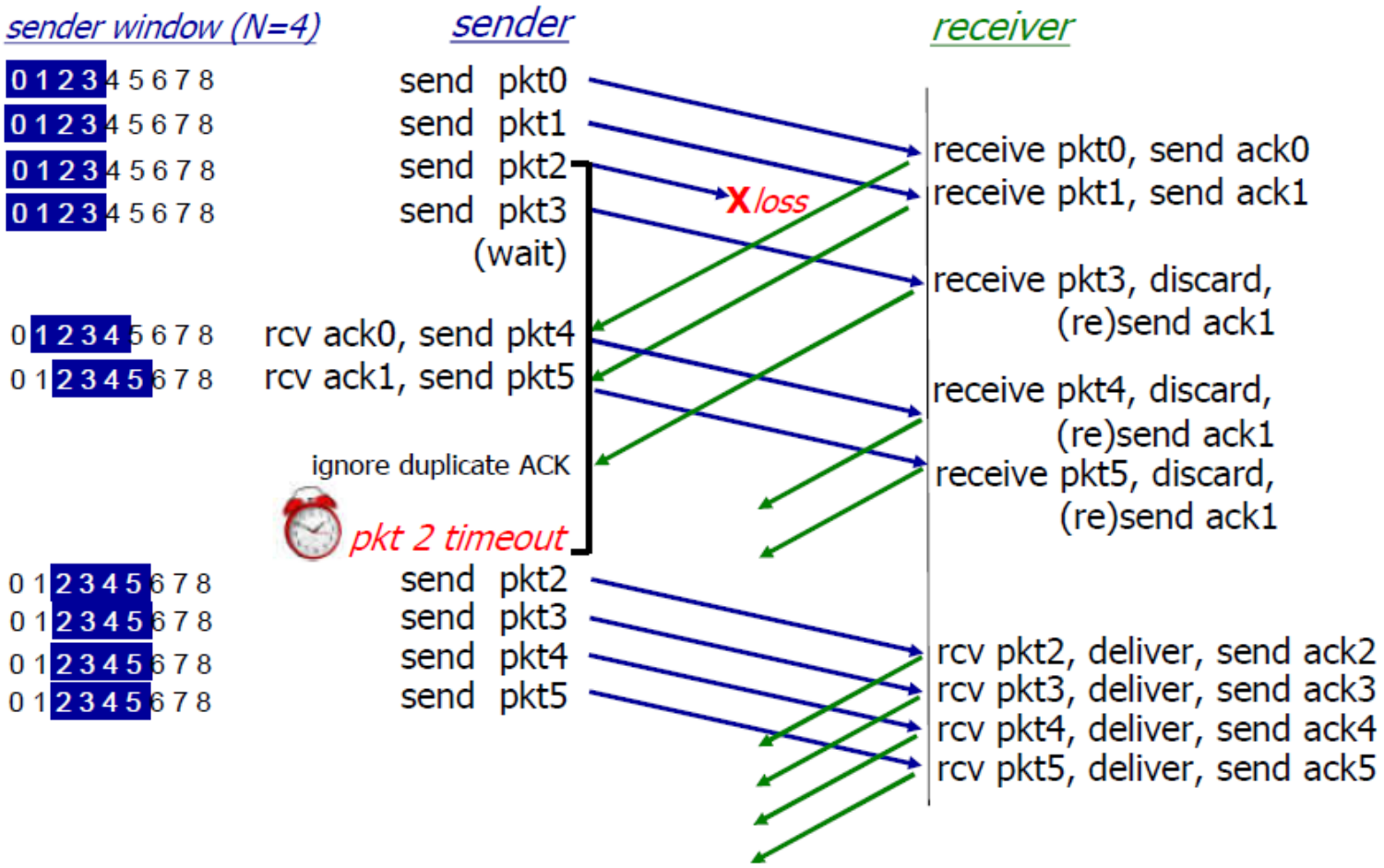
- Chỉ gửi gói tin trong cửa sổ. Chỉ dùng 1 bộ đếm (timer) cho gói tin đầu tiên trong cửa sổ
- Nếu nhận được ACK_i, dịch cửa sổ sang vị trí (i+1). Đặt lại timer
- Nếu timeout cho gói tin pkt_i gửi lại tất cả gói tin trong cửa sổ

Bên nhận

- Gửi ACK_i cho gói tin pkt_i đã nhận được theo thứ tự
- Gói tin đến không theo thứ tự: hủy gói tin và gửi lại ACK của gói tin gần nhất còn đúng thứ tự

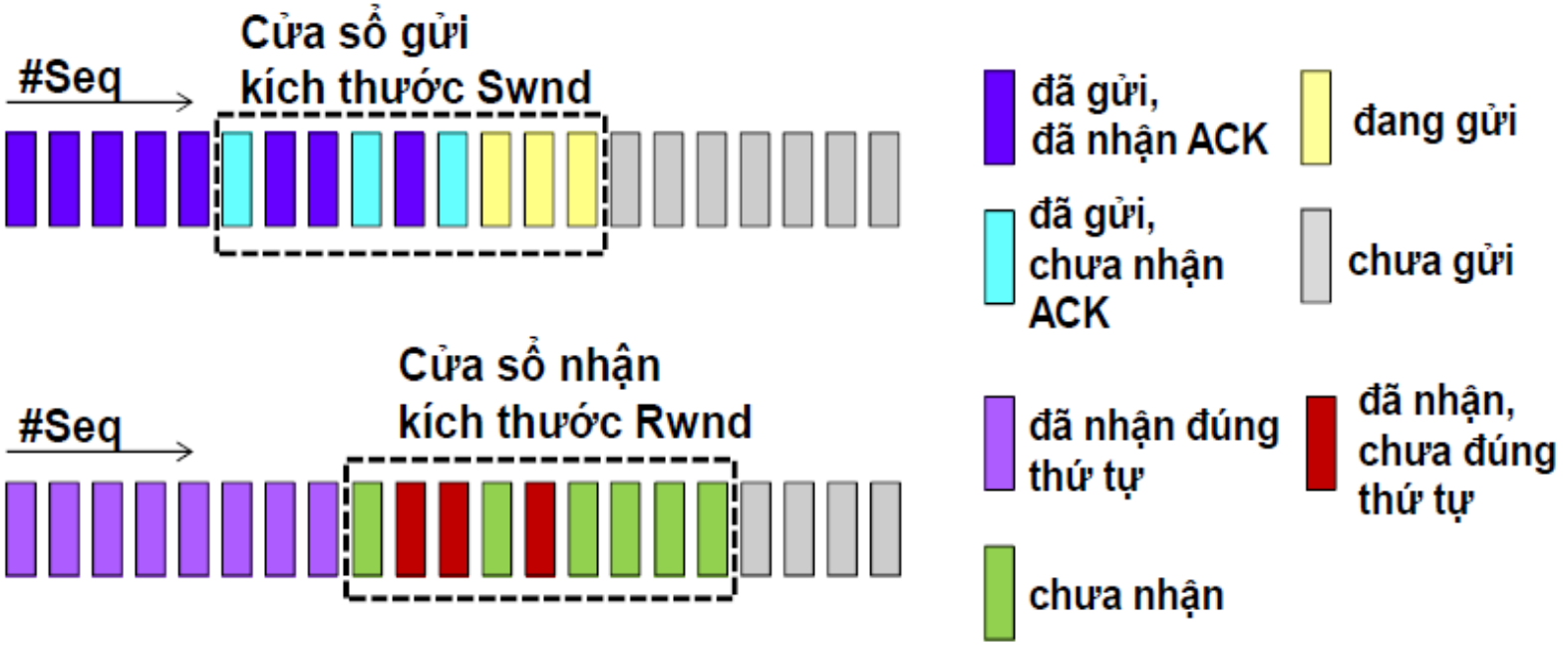
Vận chuyển dữ liệu an toàn

Go-back-N



Vận chuyển dữ liệu an toàn

Selective Repeat



- Gửi: chỉ gửi gói tin trong cửa sổ gửi
- Nhận: chỉ nhận gói tin trong cửa sổ nhận
  - Sử dụng bộ đệm để lưu tạm thời các gói tin tới chưa đúng thứ tự

Vận chuyển dữ liệu an toàn

Selective Repeat

Bên gửi

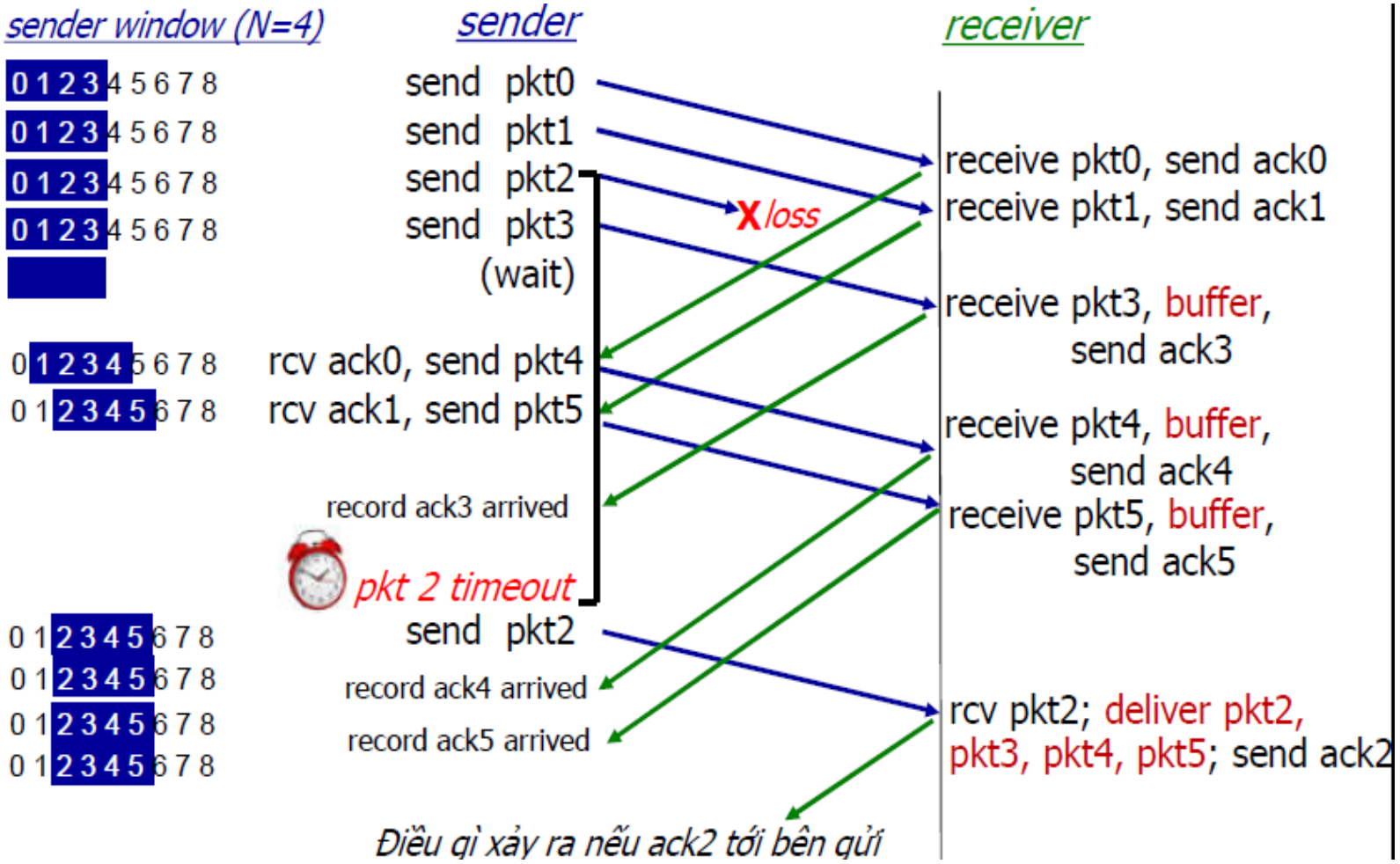
- Chỉ gửi gói tin trong cửa sổ gửi
- Dùng 1 timer cho mỗi gói tin trong cửa sổ
- Nếu timeout cho gói tin pkt_i chỉ gửi lại pkt_i
- Nhận được ACK_i:
  - Đánh dấu pkt_i đã có ACK
  - Nếu i là giá trị nhỏ nhất trong các gói tin chưa nhận ACK, dịch cửa sổ sang vị trí gói tin tiếp theo chưa nhận ACK

Bên nhận

- Chỉ nhận gói tin trong cửa sổ nhận
- Nhận pkt_i:
  - Gửi lại ACK_i
  - Không đúng thứ tự: đưa vào bộ đệm
  - Đúng thứ tự: chuyển cho tầng ứng dụng cùng với các gói tin trong bộ đệm đã trở thành đúng thứ tự sau khi nhận pkt_i

Vận chuyển dữ liệu an toàn

Selective Repeat



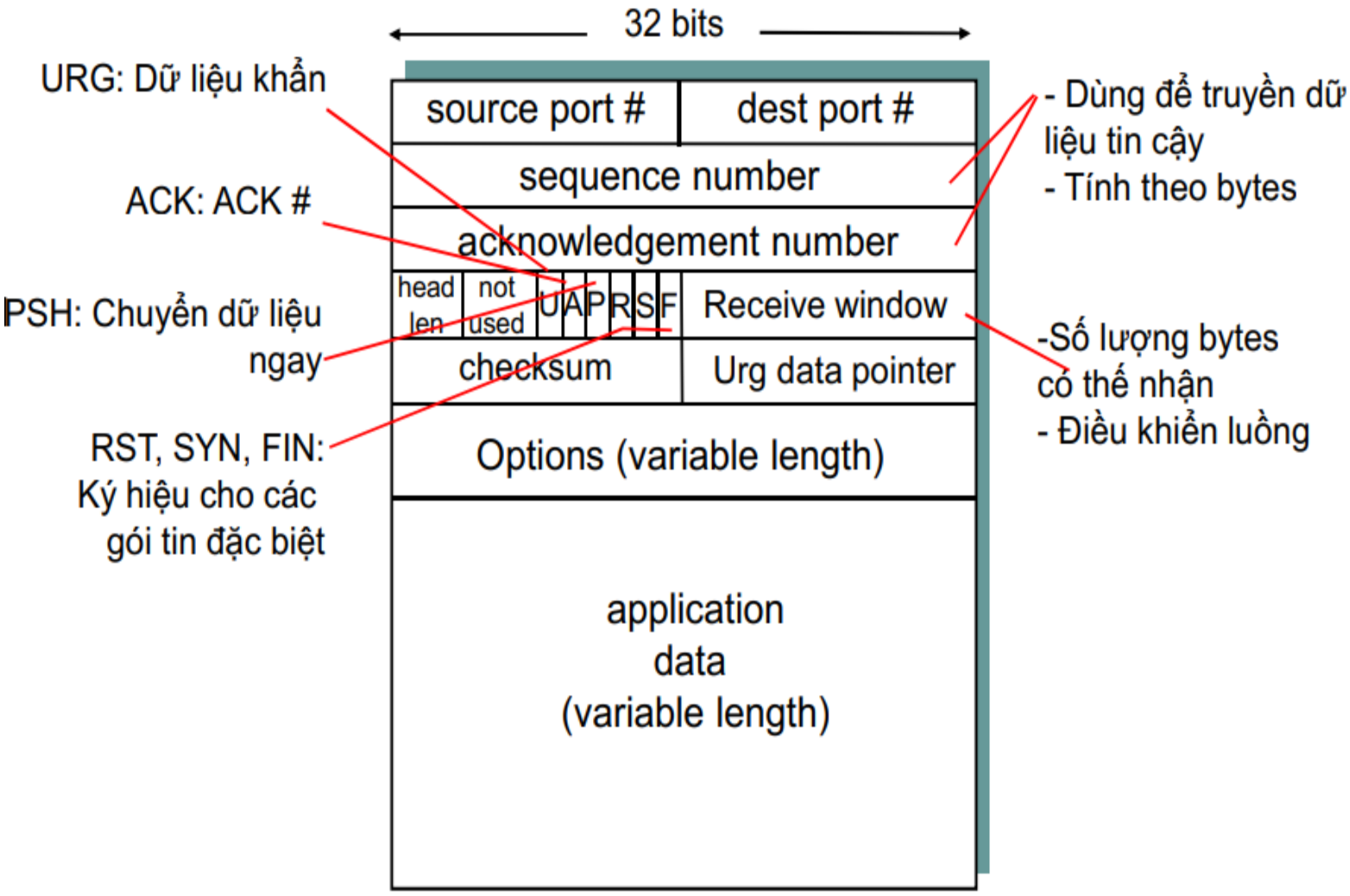
NỘI DUNG

- Tổng quan
- Chức năng
- UDP
- Vấn đề vận chuyển dữ liệu an toàn
- **TCP**

Tổng quan về TCP

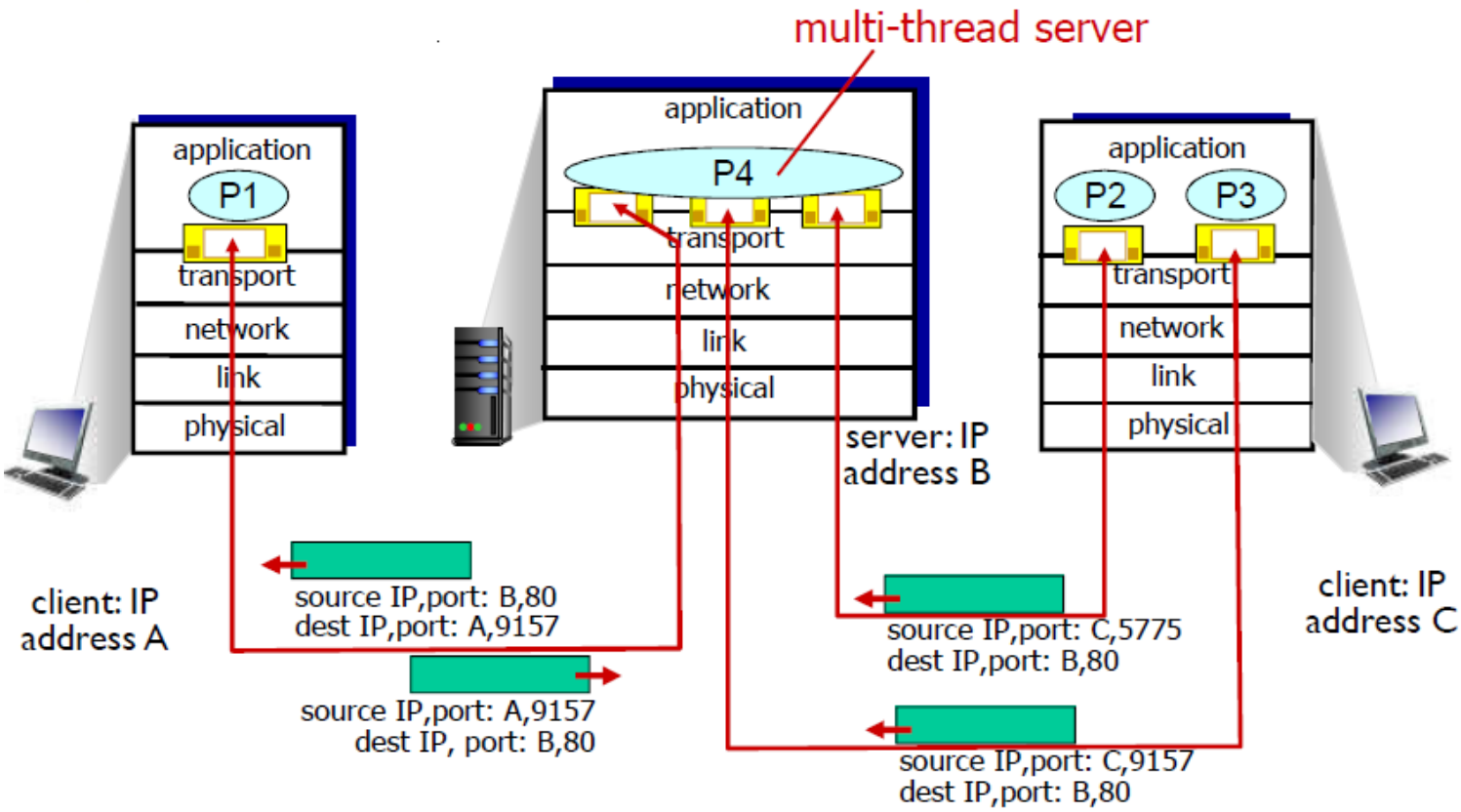
- Giao thức hướng liên kết
  - Bắt tay ba bước
- Giao thức truyền dữ liệu theo dòng byte, tin cậy
  - Sử dụng vùng đệm
- Truyền theo kiểu pipeline
  - Tăng hiệu quả
- Kiểm soát luồng
  - Bên gửi không làm quá tải bên nhận

Khuôn dạng đoạn tin – TCP segment



mux/demux trên ứng dụng TCP

- Sử dụng socket khác nhau để trao đổi với các tiến trình khác nhau



Thông số của liên kết TCP

- Mỗi một liên kết TCP giữa hai tiến trình được xác định bởi bộ 4 thông số (4-tuple):

Địa chỉ IP nguồn
Địa chỉ IP đích
} Tầng mạng

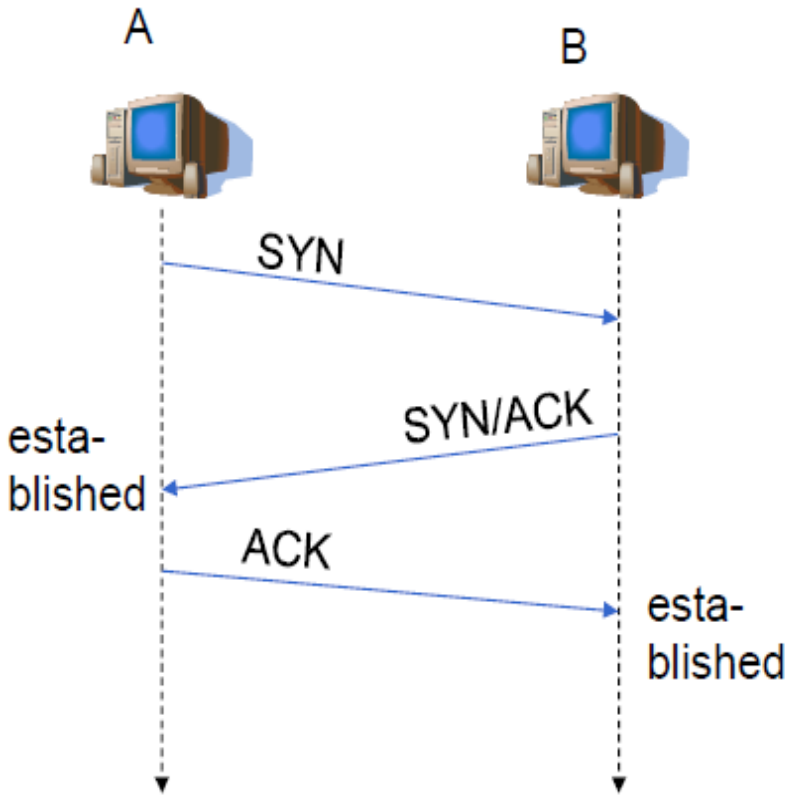
Số hiệu cổng nguồn
Số hiệu cổng đích
} Tầng giao vận

TCP cung cấp dịch vụ tin cậy ntn?

- Kiểm soát lỗi dữ liệu: checksum
- Kiểm soát mất gói tin: phát lại khi có time-out
- 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: có thể thực hiện đồng thời (duplex) trên liên kết
  - Đóng liên kết

Thiết lập liên kết TCP

Giao thực 3 bước

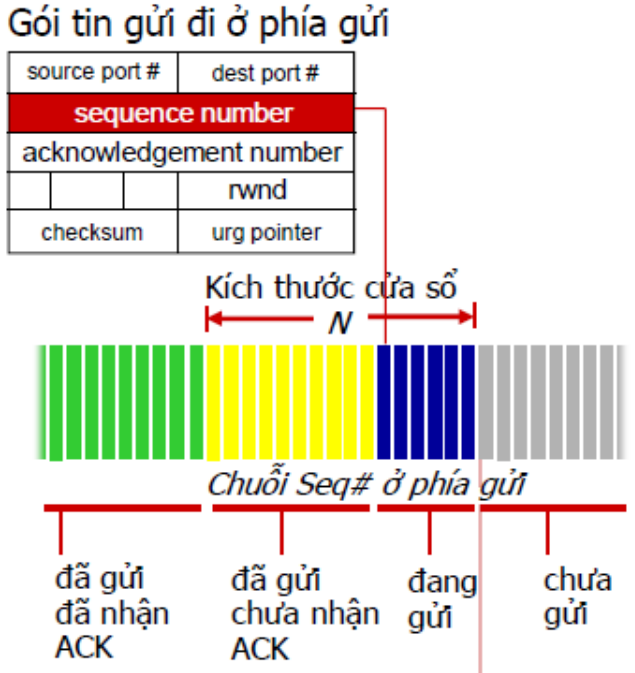


- Bước 1: A gửi SYN cho B
  - chỉ ra giá trị khởi tạo seq # của A
  - không có dữ liệu
- Bước 2: B nhận SYN, trả lời bằng SYN/ACK
  - B khởi tạo vùng đệm
  - chỉ ra giá trị khởi tạo seq. # của B
- Bước 3: A nhận SYNACK, trả lời ACK, có thể kèm theo dữ liệu

Cơ chế báo nhận trong TCP

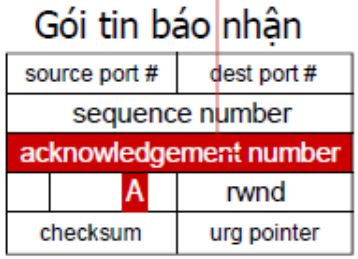
sequence numbers:

- Vị trí của byte đầu tiên trên payload của gói tin (segment) trong luồng dữ liệu



acknowledgements:

- Vị trí của byte tiếp theo muốn nhận trong luồng dữ liệu



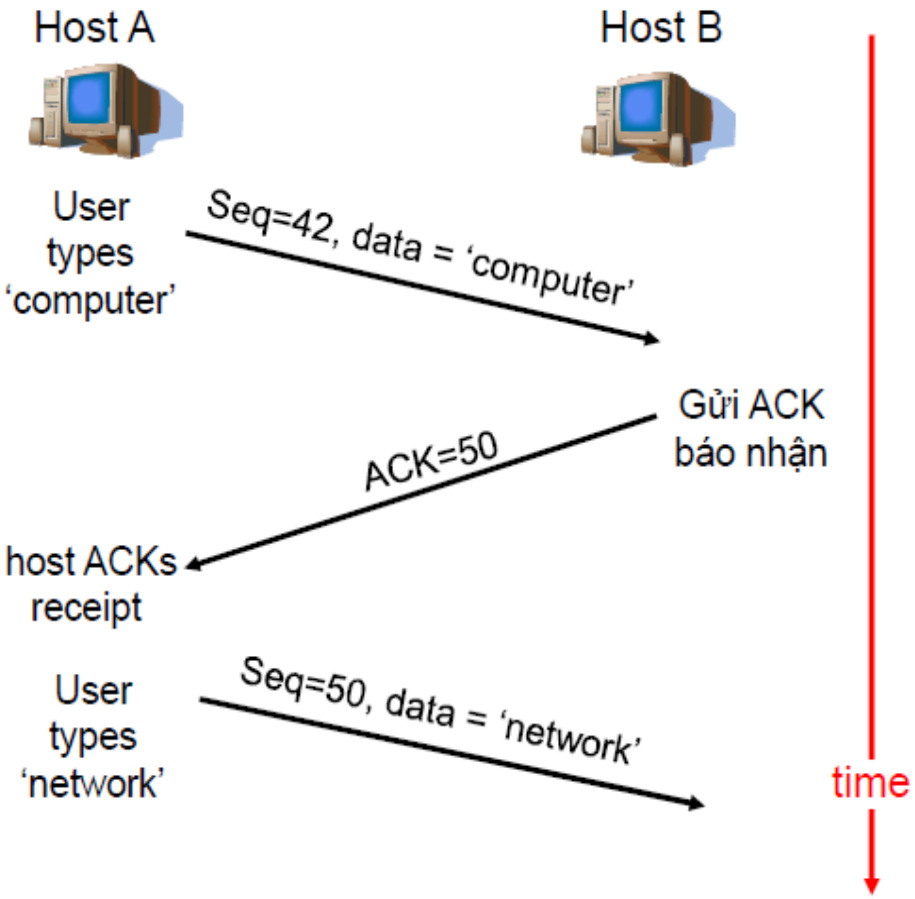
Cơ chế báo nhận trong TCP

Seq. #:

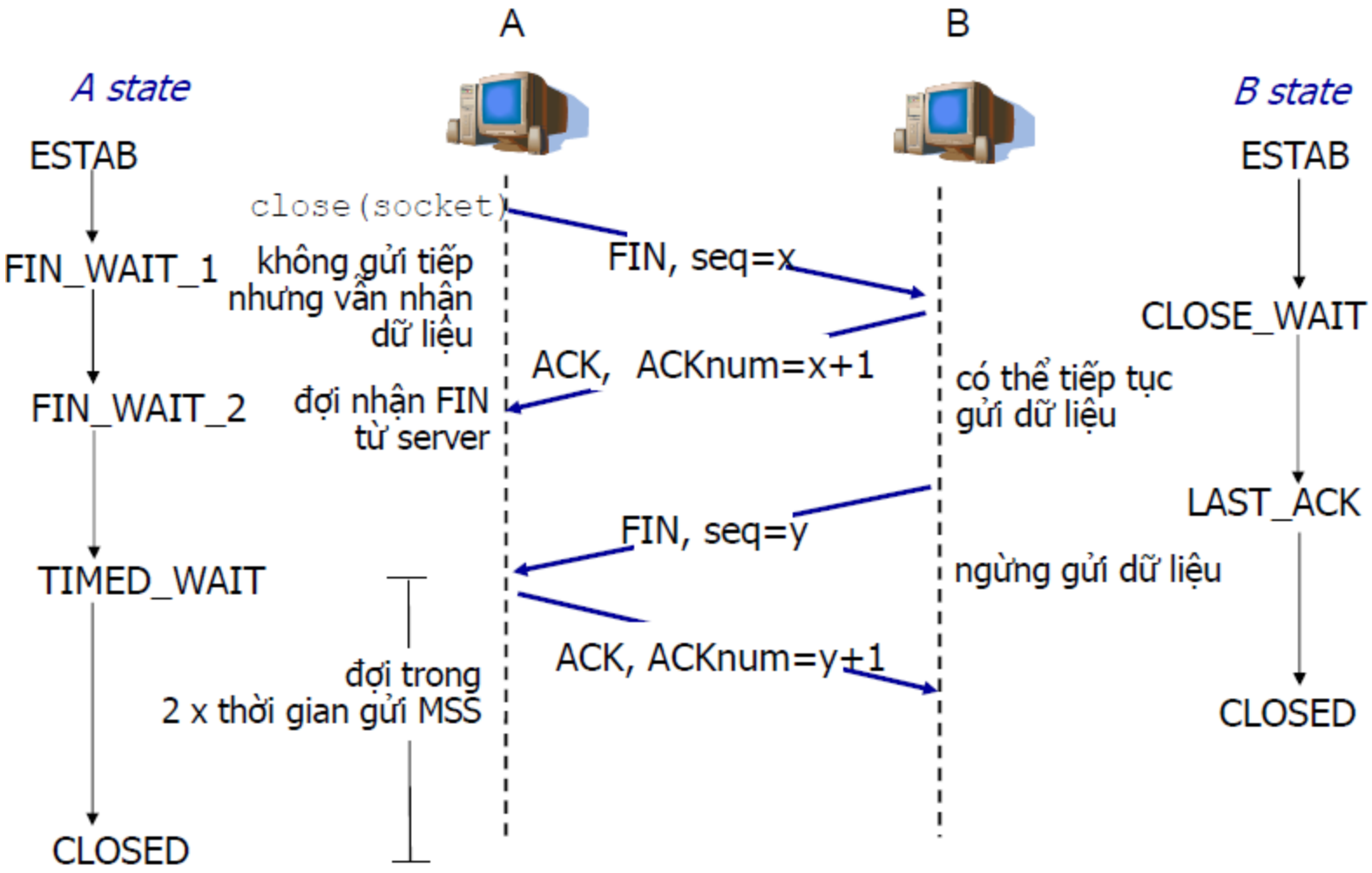
- Số hiệu của byte đầu tiên của đoạn tin trong dòng dữ liệu

ACK:

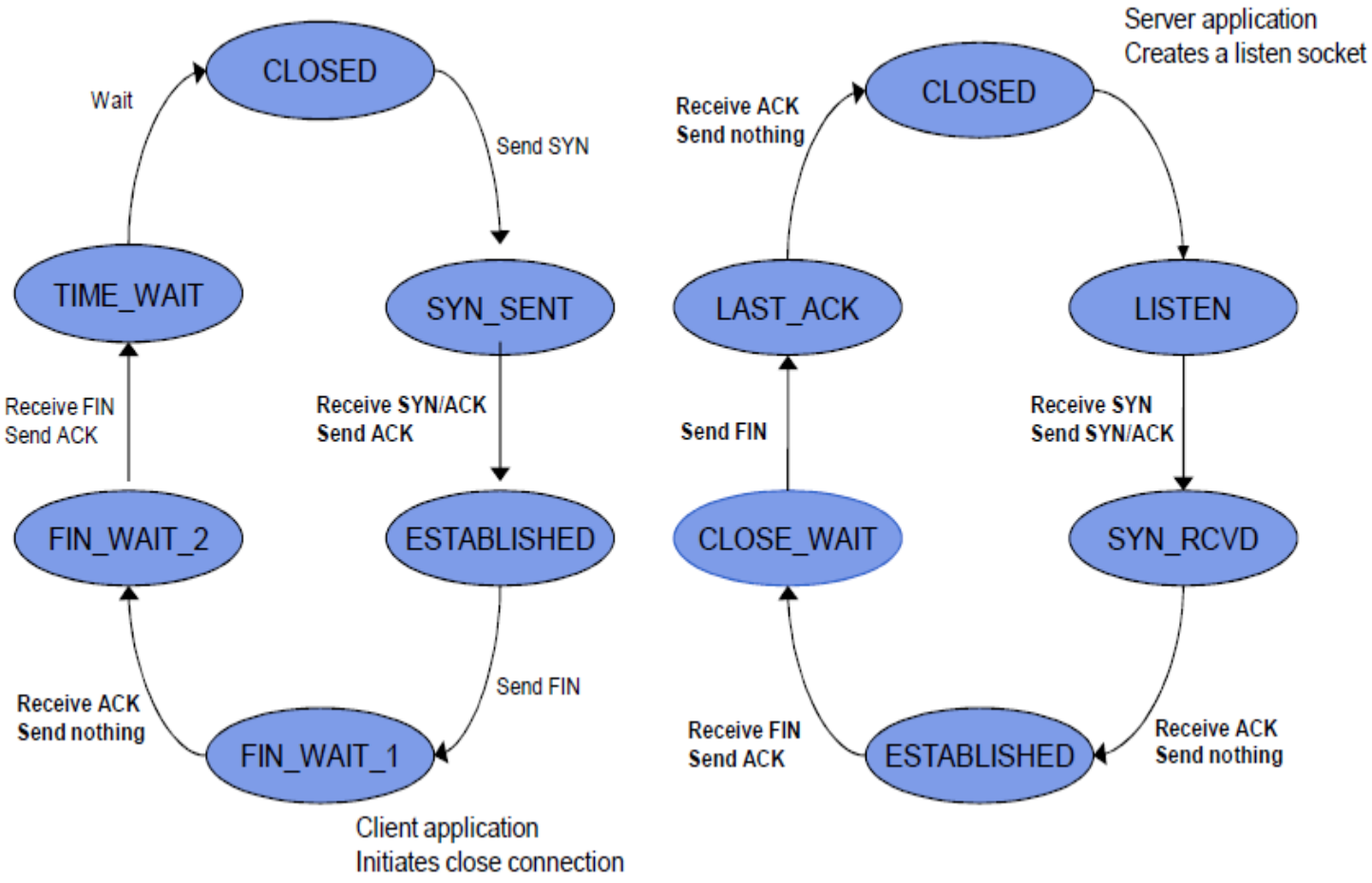
- Số hiệu byte đầu tiên mong muốn nhận từ đối tác



Đóng liên kết



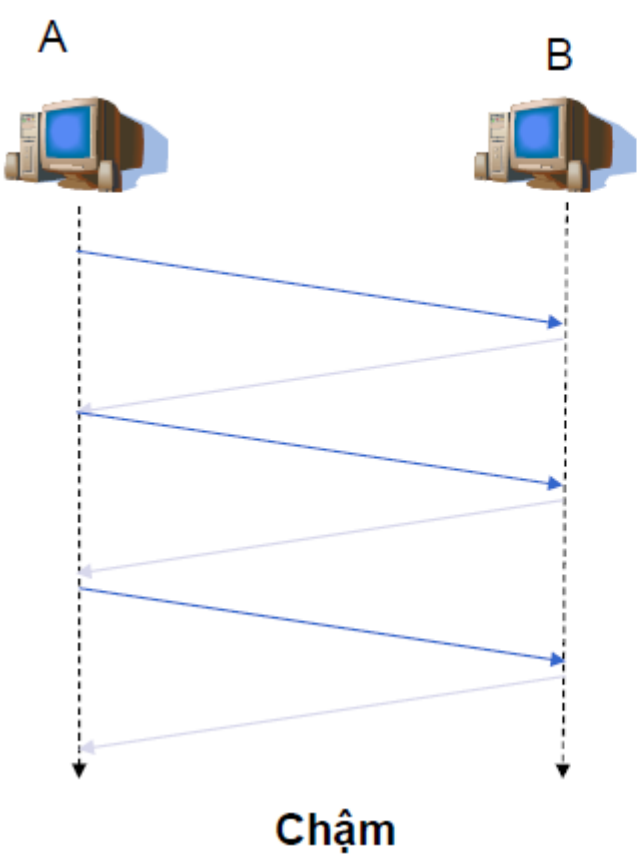
Chu trình sống của TCP (Đơn giản hóa)

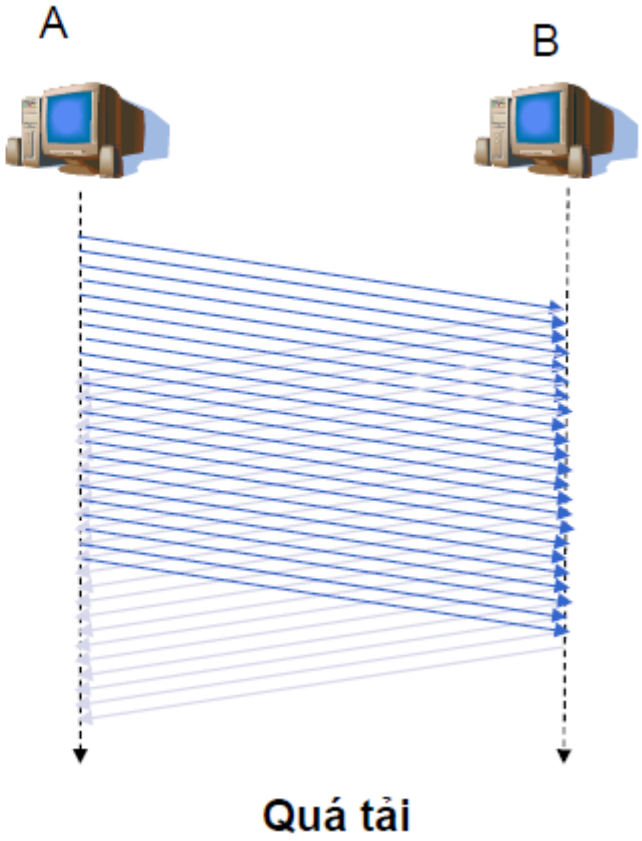


Kiểm soát luồng

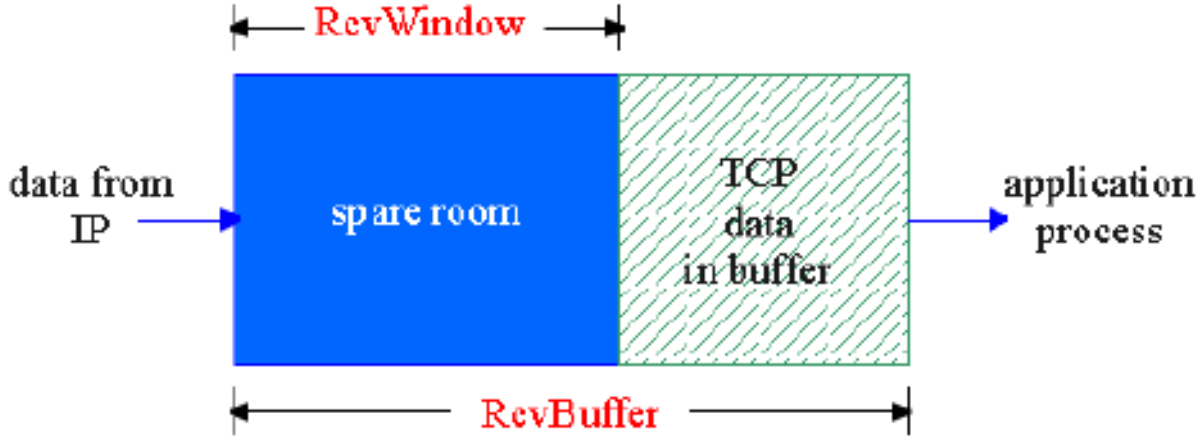
- Điều khiển lượng dữ liệu được gửi đi
  - Bảo đảm rằng hiệu quả là tốt
  - Không làm quá tải các bên
- Các bên sẽ có cửa sổ kiểm soát
  - Rwnd: Cửa sổ nhận
  - Cwnd: Cửa sổ kiểm soát tắc nghẽn
- Lượng dữ liệu gửi đi phải nhỏ hơn min(Rwnd, Cwnd)

Kiểm soát luồng





Kiểm soát luồng trong TCP



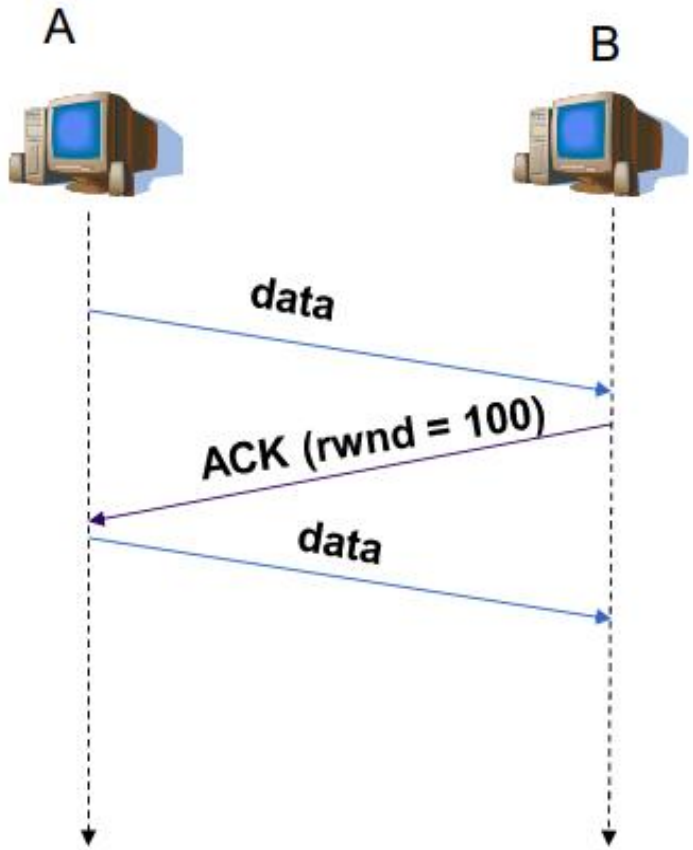
- Kích thước vùng đệm trống

= Rwnd

= `RcvBuffer-[LastByteRcvd - LastByteRead]`

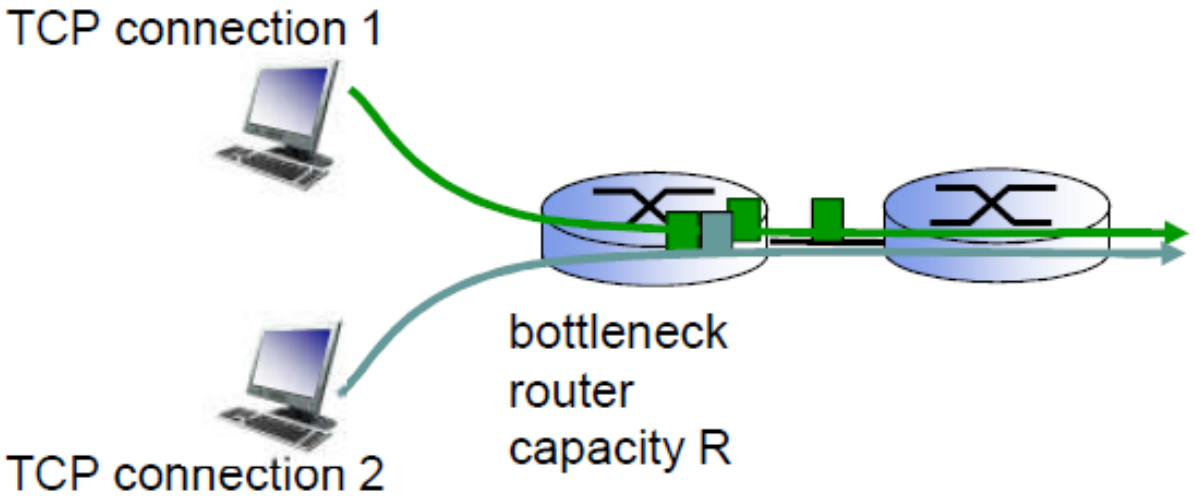
Trao đổi thông tin Rwnd

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



Tính công bằng trong TCP

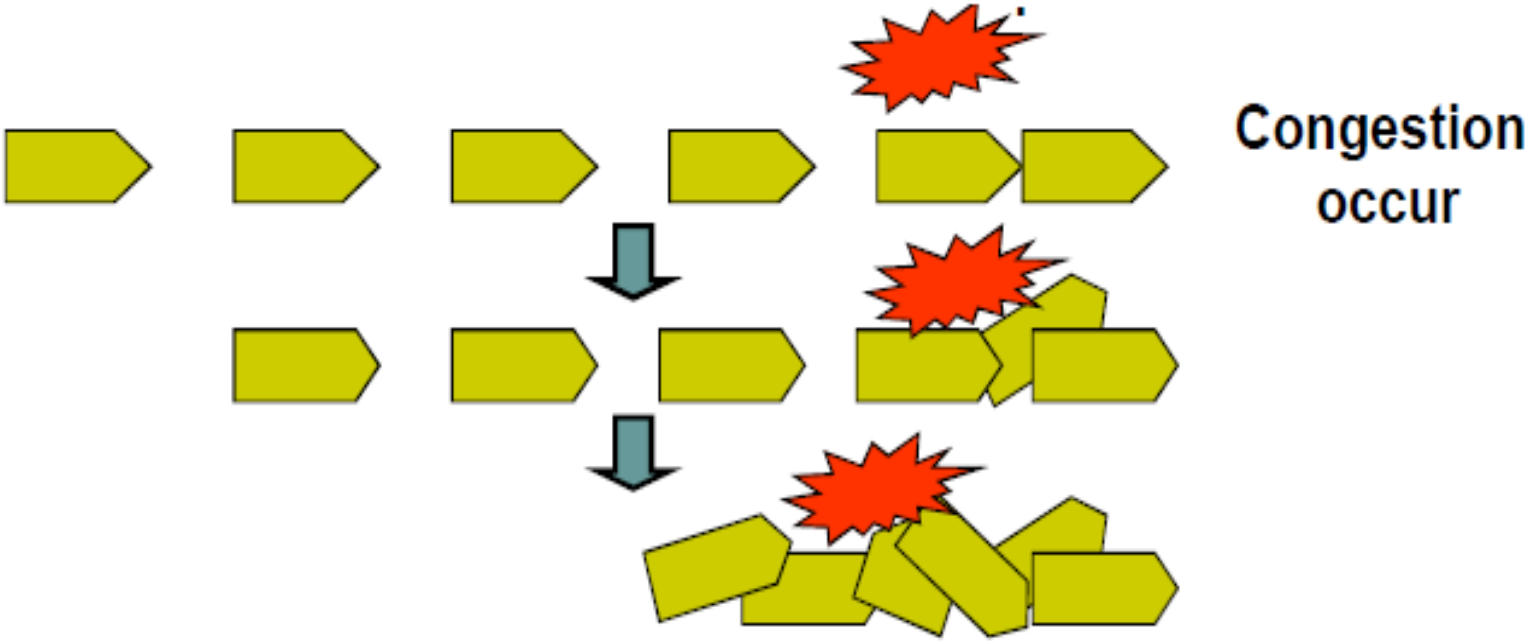
- Nếu có K kết nối TCP chia sẻ đường truyền có băng thông R thì mỗi kết nối có tốc độ truyền trung bình là R/K



Điều kiển tắc nghẽn trong TCP

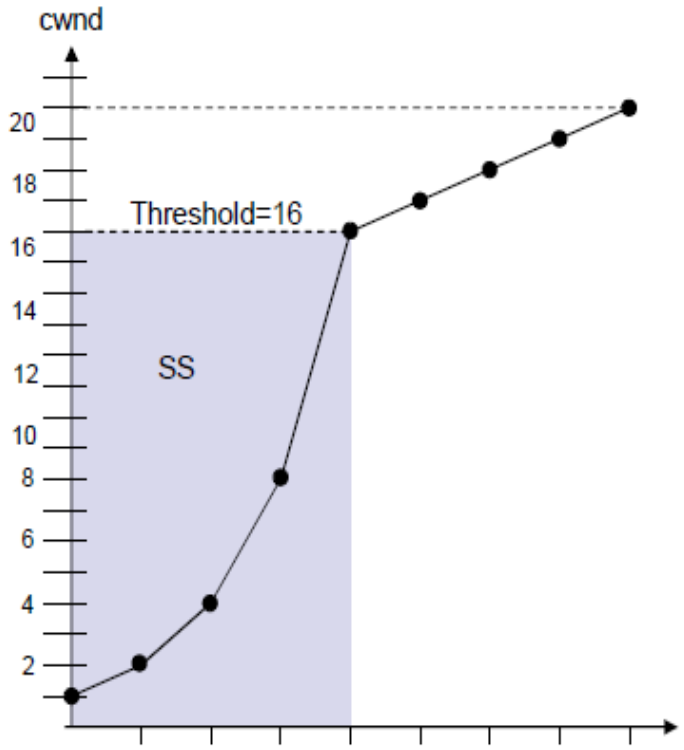
- Khi nào tắc nghẽn xảy ra?
  - Quá nhiều cặp gửi-nhận trên mạng
  - Truyền quá nhiều làm cho mạng quá tải
- Hậu quả của việc nghẽn mạng
  - Mất gói tin
  - Thông lượng giảm, độ trễ tăng
  - Tình trạng của mạng sẽ trở nên tồi tệ hơn.

Điều kiển tắc nghẽn trong TCP



Nguyên lý tắc nghẽn trong

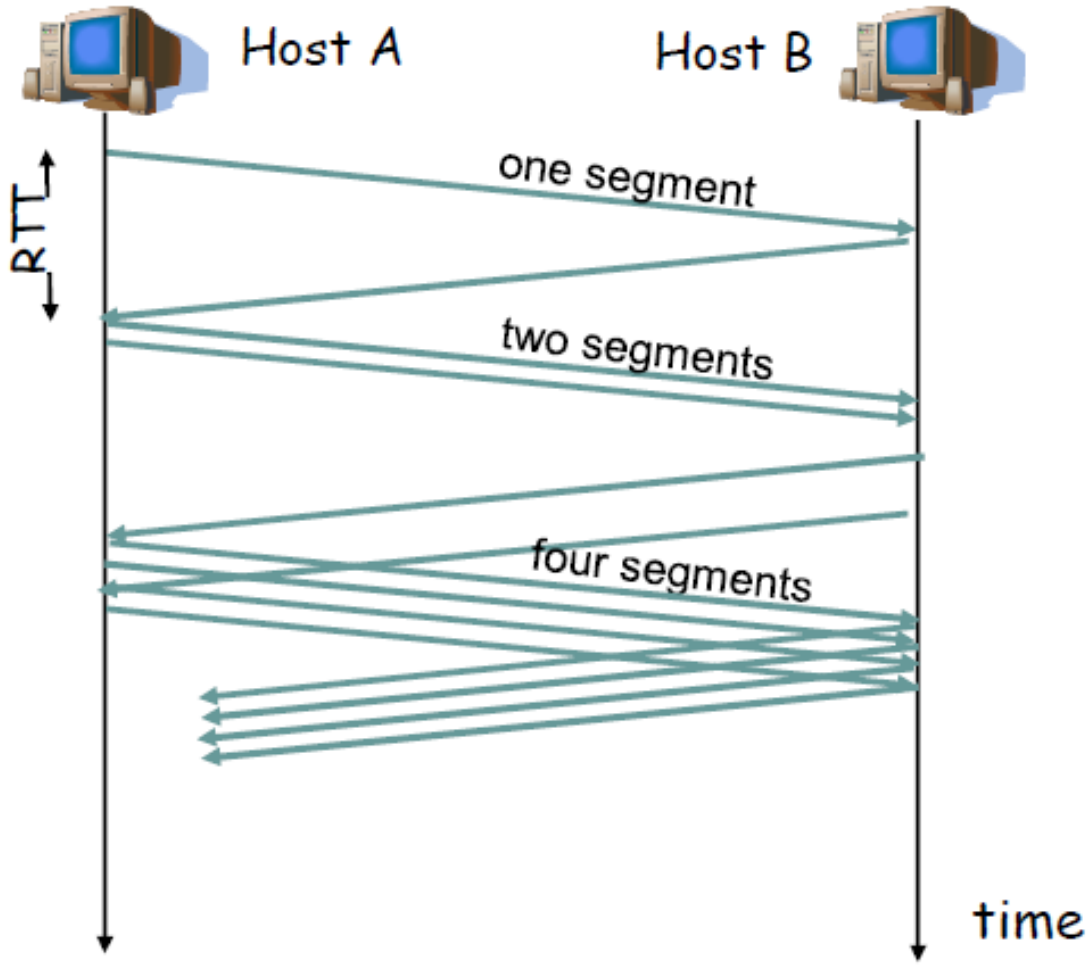
- Slow-start
  - Tăng tốc độ theo hàm số mũ
  - Tiếp tục tăng đến một ngưỡng nào đó
- Tránh tắc nghẽn
  - Tăng dẫn tốc độ theo hàm tuyến tính cho đến khi phát hiện tắc nghẽn
- Phát hiện tắc nghẽn
  - Gói tin bị mất



TCP Slow-start

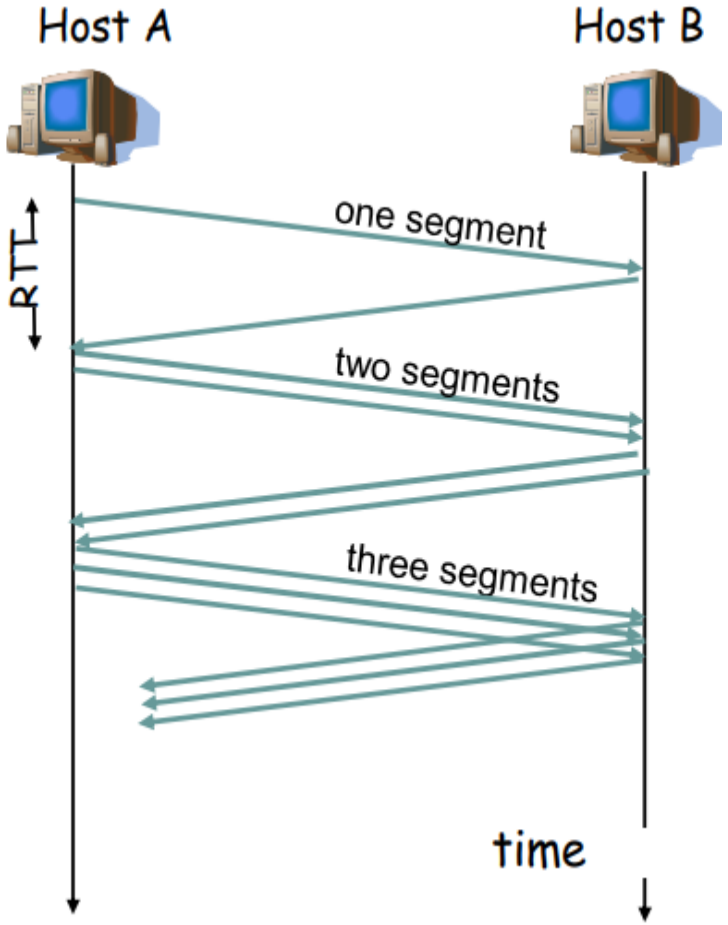
- Ý tưởng cơ bản
  - Đặt cwnd bằng 1 MSS (Maximum segment size)
  - 1460 bytes (giá trị này có thể được thỏa thuận trong quá trình thiết lập liên kết)
- Tăng cwnd lên gấp đôi
  - Khi nhận được ACK
- Bắt đầu chậm, nhưng tăng theo hàm mũ
- Tăng cho đến một ngưỡng: ssthresh
  - Sau đó, TCP chuyển sang trạng thái tránh tắc nghẽn

TCP Slow-start



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 1MSS



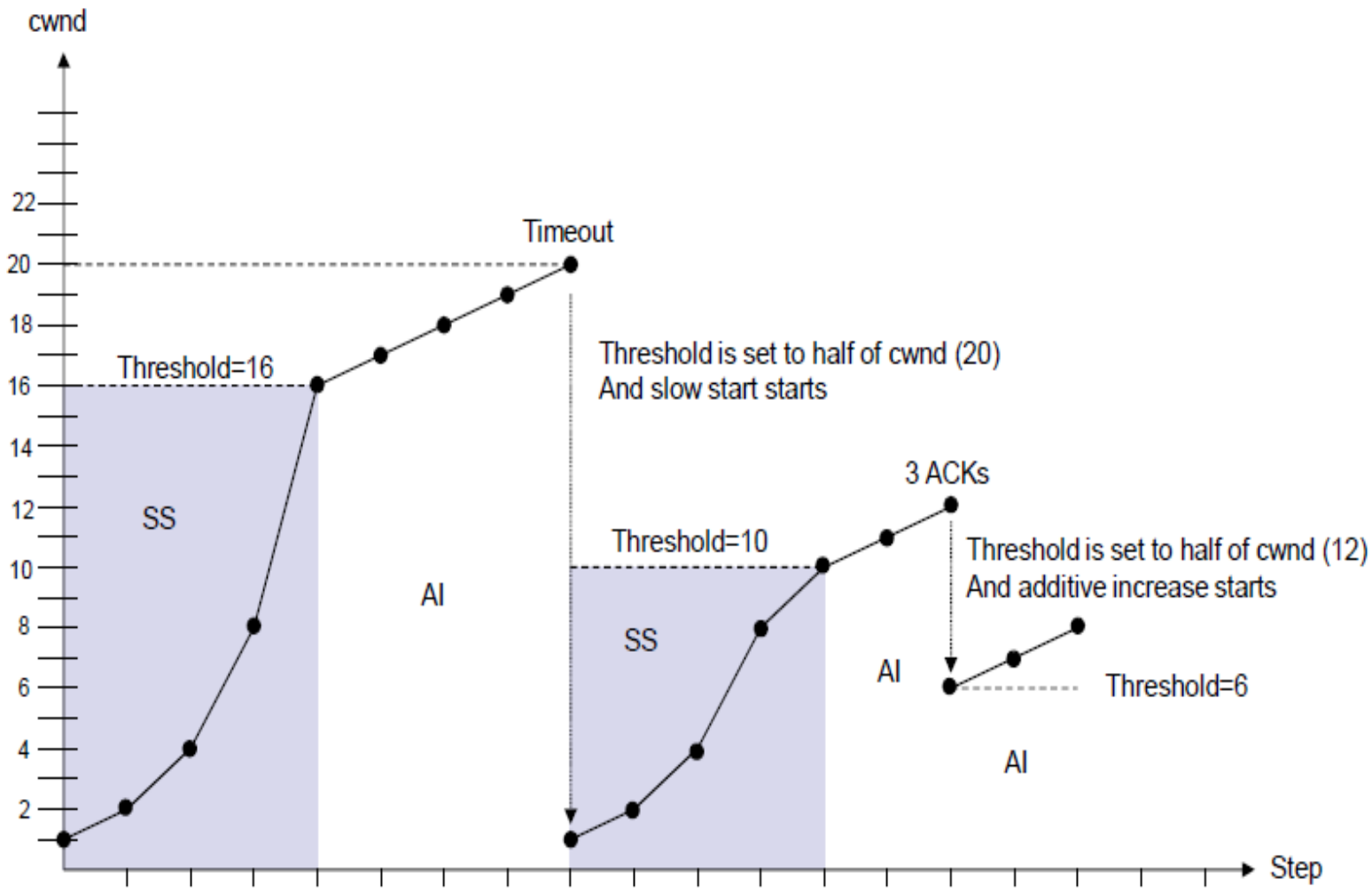
Phản ứng của TCP

- Giảm tốc độ gửi
- Phát hiện tắc nghẽn?
  - Nếu như phải truyền lại
    - Có thể đoán mạng đang có "tắc nghẽn"
- Khi nào thì phải truyền lại?
  - Timeout!
  - Nhận được nhiều ACK cùng ACK#

Phản ứng của TCP

- 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
- Hồi phục nhanh:
  - Nút nhận: nhận được 1 gói tin không đúng thứ tự thì gửi liên tiếp 3 ACK giống nhau.
  - Nút gửi: nhận được 3 ACK giống nhau
    - TCP đặt ngưỡng ssthresh xuống còn một nửa giá trị hiện tại của cwnd
    - TCP đặt cwnd về giá trị hiện tại của ngưỡng mới
    - TCP chuyển trạng thái “congestion avoidance”

Phản ứng của TCP



Q & A