

KHOA CÔNG NGHỆ THÔNG TIN

Môn học: Nhập môn mạng máy tính

Giảng viên: Lê Viết Thanh

Mã số: 7480102

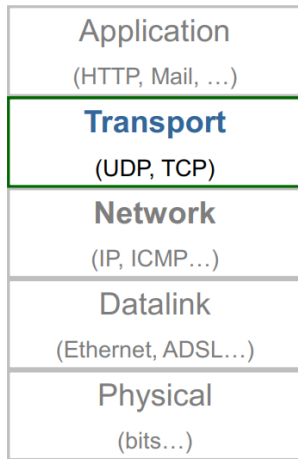
Ngày 22 tháng 9 năm 2023

NỘI DUNG TRÌNH BÀY

- 1 Tổng quan về tầng giao vận
- 2 UDP (User Datagram Protocol)
- 3 TCP (TransmissionControl Protocol)
 - Khái niệm về truyền thông tin cậy

Kiến trúc phân tầng

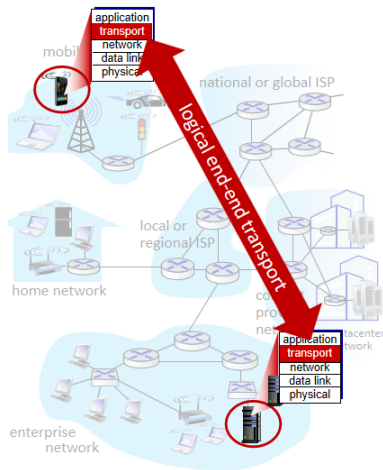
- Hỗ trợ các ứng dụng trên mạng
- Điều khiển truyền dữ liệu giữa các tiến trình của tầng ứng dụng
- Chọn đường và chuyển tiếp gói tin giữa các máy, các mạng
- Hỗ trợ việc truyền thông cho các thành phần kết tiếp trên cùng 1 mạng
- Truyền và nhận dòng bit trên đường truyền vật lý



Hình: Mô hình TCP/IP

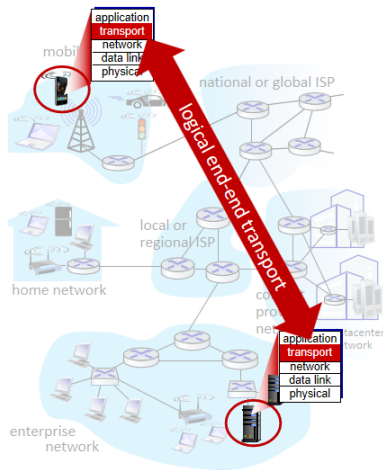
Tổng quan về tầng giao vận

- 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



Hình: Logical end-end transport

- 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
- Đơn vị truyền: datagram (UDP), segment (TCP)



Hình: Logical end-end transport

Tại sao cần 2 loại dịch vụ TCP và UDP?

- 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



Hình: Variety Services

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 | TCP or UDP |
| Internet telephony | giao thức riêng | TCP or UDP |

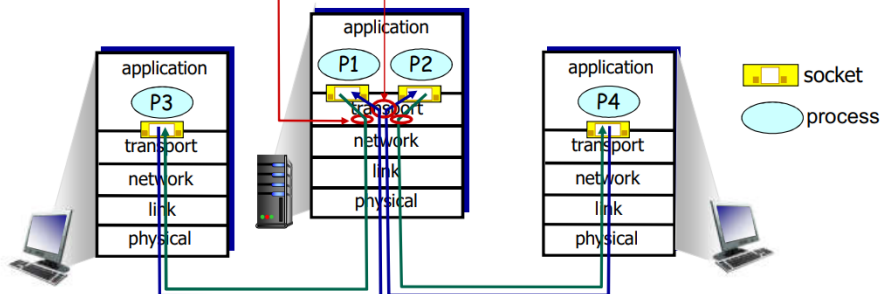
Dồn kênh/phân kênh - Mux/Demux

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

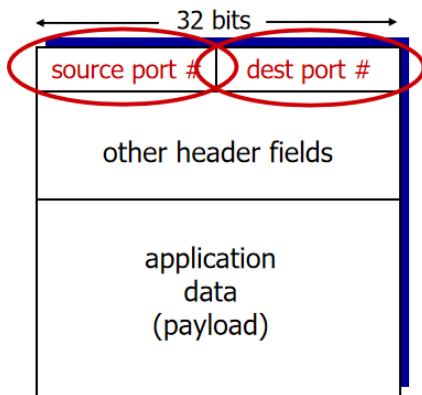
Sử dụng thông tin trên tiêu đề gói tin để gửi dữ liệu tới đúng socket



Hình: Dồn kênh và phân kênh

Cách hoạt động Mux/Demux

- 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



Hình: Dồn kênh và phân kênh

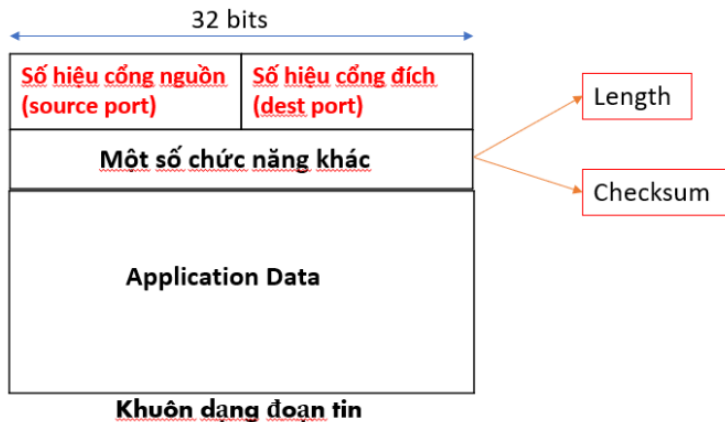
Tổng quan Khuôn dạng gói tin UDP

Đặc điểm chung

- Giao thức hướng không kết nối (connectionless)
- Không sử dụng báo nhận:
 - ▶ Phía nguồn gửi dữ liệu nhanh nhất, nhiều nhất có thể
- 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
- UDP có những chức năng cơ bản gì?
 - ▶ Dồn kênh/phân kênh
 - ▶ Phát hiện lỗi bit bằng checksum

Khuôn dạng thư tín (datagram)

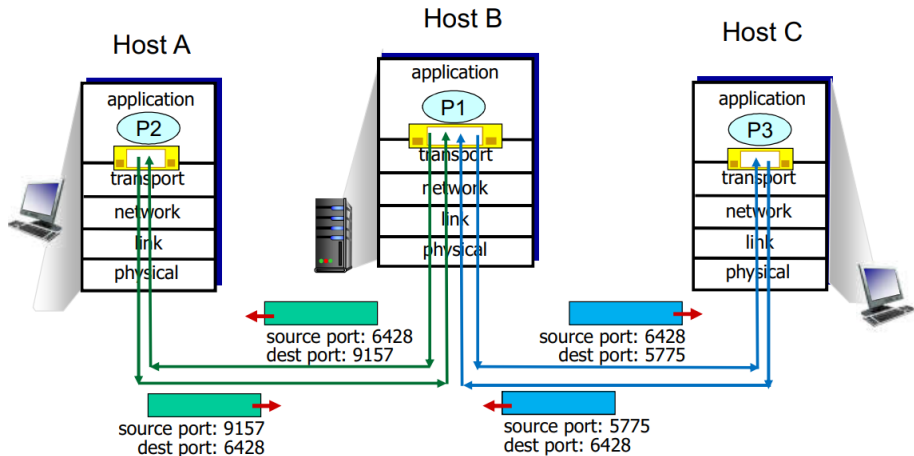
- UDP sử dụng đơn vị dữ liệu gọi là – datagram



Hình: Khuôn dạng thư tín

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



Hình: Quá trình trao đổi dữ liệu

Các vấn đề của UDP

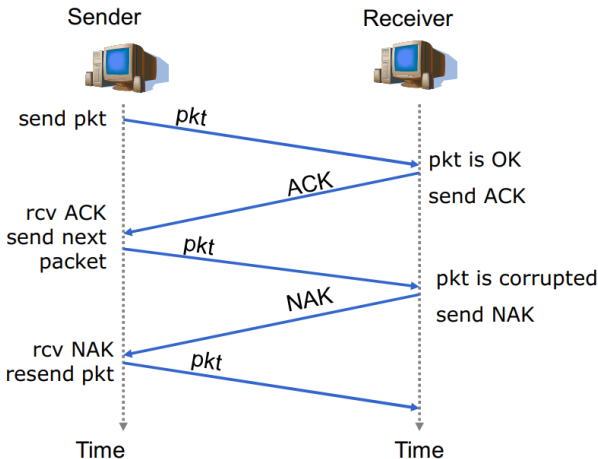
- Không có kiểm soát tắc nghẽn
 - ▶ Làm Internet bị quá tải
- Không bảo đảm được độ tin cậy
 - ▶ Các ứng dụng phải cài đặt cơ chế tự kiểm soát độ tin cậy
 - ▶ Việc phát triển ứng dụng sẽ phức tạp hơn

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): gói tin được nhận thành công
 - ▶ NAK (negative acknowledgements): gói tin bị lỗi
- Phản ứng của bên gửi?
 - ▶ Truyền lại nếu là NAK

Hoạt động



Hình: Hoạt động của TCP

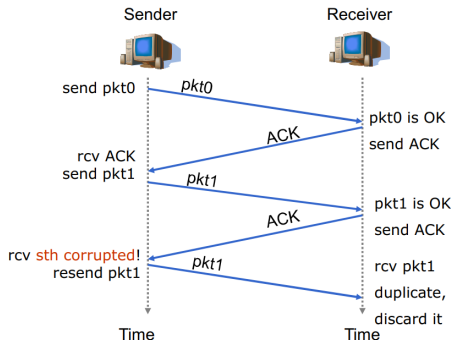
Lỗi ACK/NAK

what happens if ACK/NAK corrupted?

- sender doesn't know what happened at receiver!
- can't just retransmit: possible duplicate

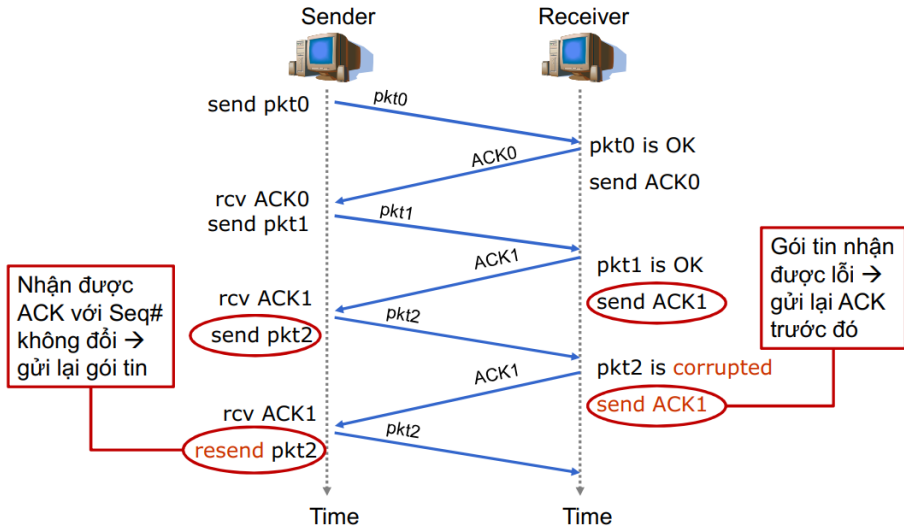
handling duplicates:

- sender retransmits current pkt if ACK/NAK
- corrupted sender adds sequence number to each pkt
- receiver discards (doesn't deliver up) duplicate pkt



Hình: Xử lý lỗi ACK/NAK

Giải pháp không dùng NAK

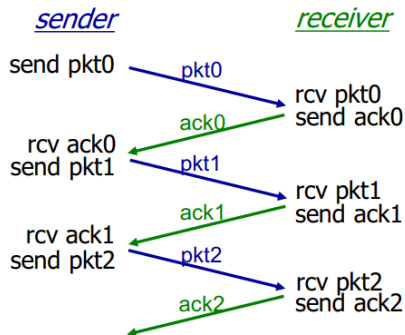


Hình: Không dùng NAK

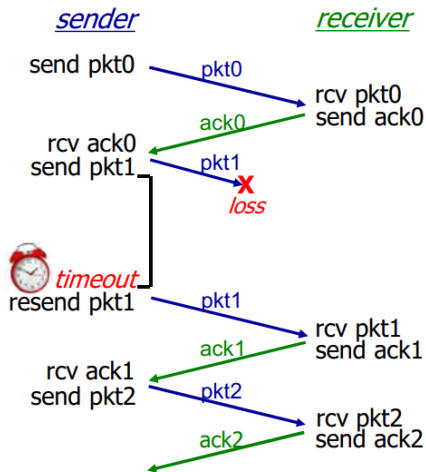
Kênh có lỗi bit và mất gói tin

- 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

Ví dụ



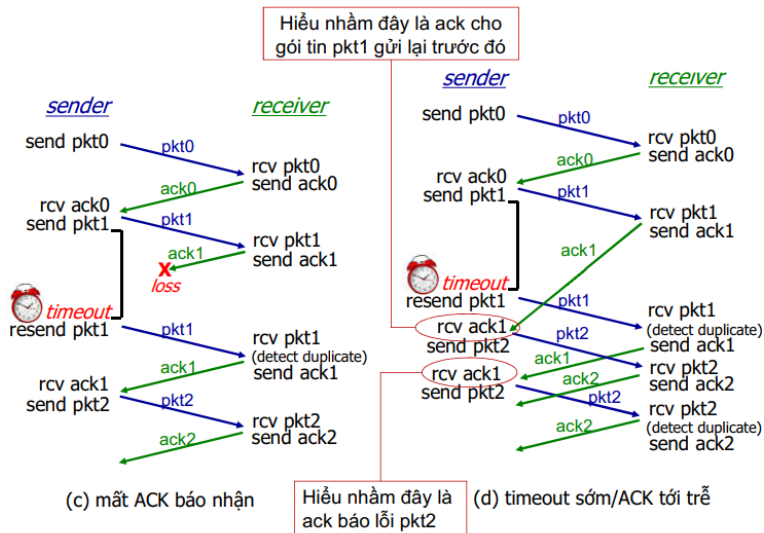
(a) Không có mất gói tin



(b) mất gói tin gửi đi

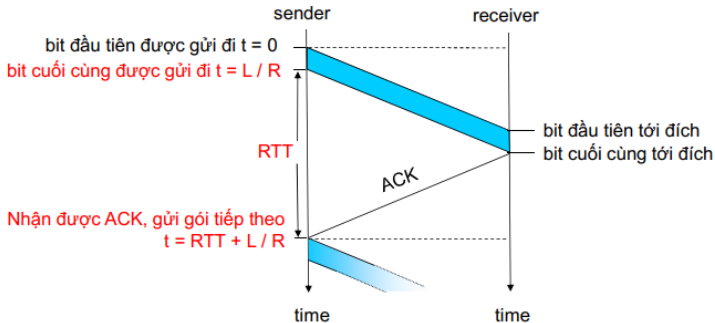
Hình: Kênh có lỗi bit và mất gói tin

Ví dụ



Hình: Kênh có lỗi bit và mất gói tin

Hiệu năng của stop-and-wait

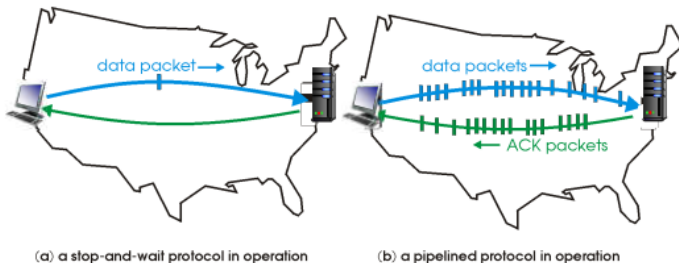


L: Kích thước gói tin
R: Băng thông
RTT: Round trip time

$$performance = \frac{L/R}{RTT + L/R}$$

Hình: Hiệu năng

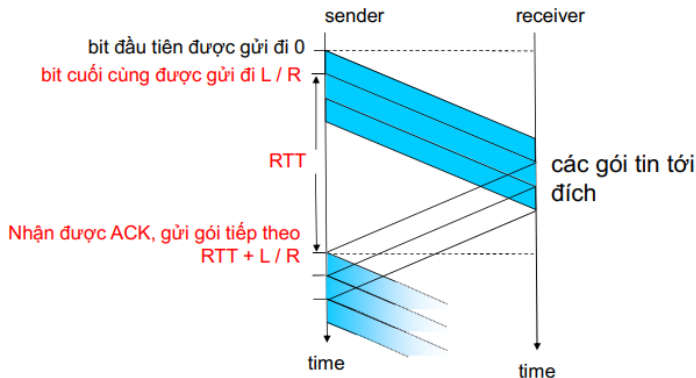
Pipeline



Hình: 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

Hiệu năng của pipeline



- L: Kích thước gói tin
R: Băng thông
RTT: Round trip time
n: Số gói tin gửi liên tục

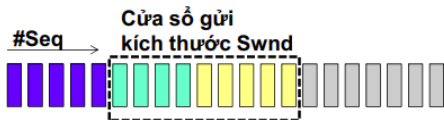
$$performance = \frac{n * L/R}{RTT + L/R}$$

Hình: Hiệu năng Pipeline

Go-back-N

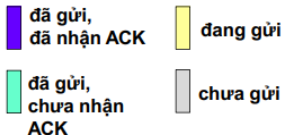
Bên gửi

- 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



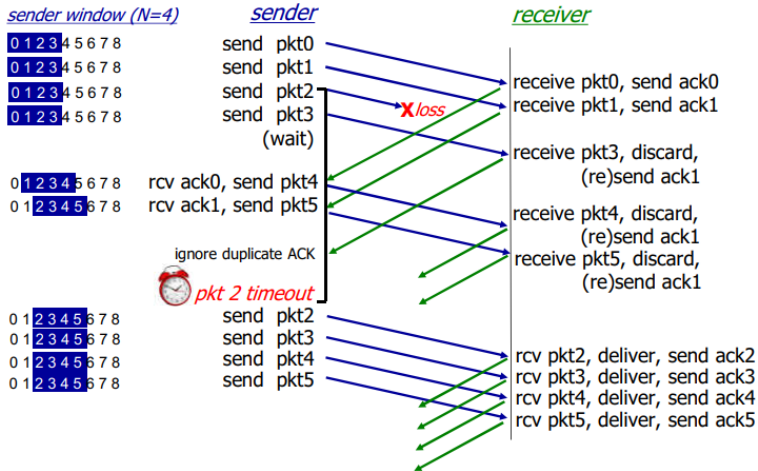
Bên nhận

- 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



Hình: Go back N

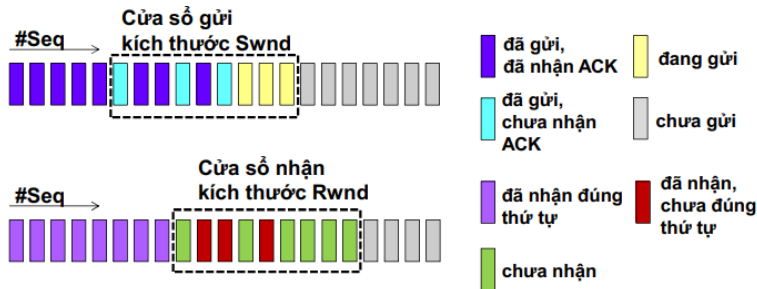
Go back N



Hình: Go back N

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



Hình: Selective Repeat

Selective Repeat

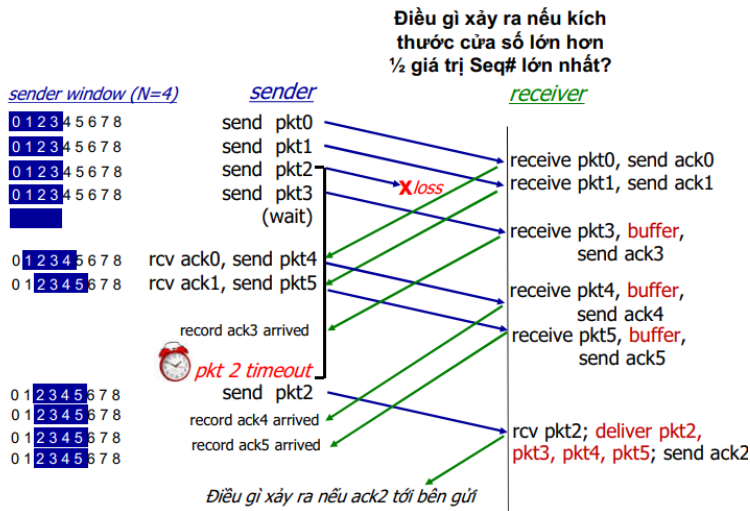
Bên gửi

- 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_j :
 - ▶ 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

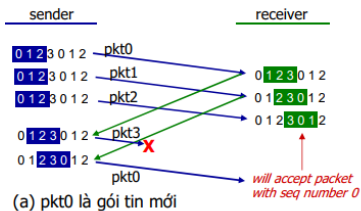
- Chỉ nhận gói tin trong cửa sổ nhận
- Nhận pkt_i :
 - ▶ Gửi lại ACK_i
 - ▶ Không đúng thứ tự: đưa vào bộ đệm
 - ▶ Đúng thứ tự: chuyển cho tầng ứng dụng cùng với các gói tin trong bộ đệm đã trở thành đúng thứ tự sau khi nhận pkt_i

Selective Repeat

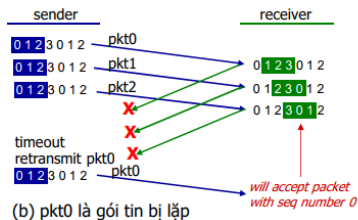


Hình: Selective Repeat

Kích thước cửa sổ quá lớn



- Giả sử Seq# = 0, 1, 2, 3
- Kích thước cửa sổ: 3
- Phía nhận không phân biệt được 2 trường hợp
- Trong trường hợp b, gói tin pkt0 gửi lại được bên nhận coi như gói tin mới, đưa vào bộ đệm chờ xử lý



Hình: Selective Repeat