CHƯƠNG

III

TẦNG TRANSPORT

Các giao thức tầng transport trên Internet

Tin cậy, truyền theo thứ tự (TCP)

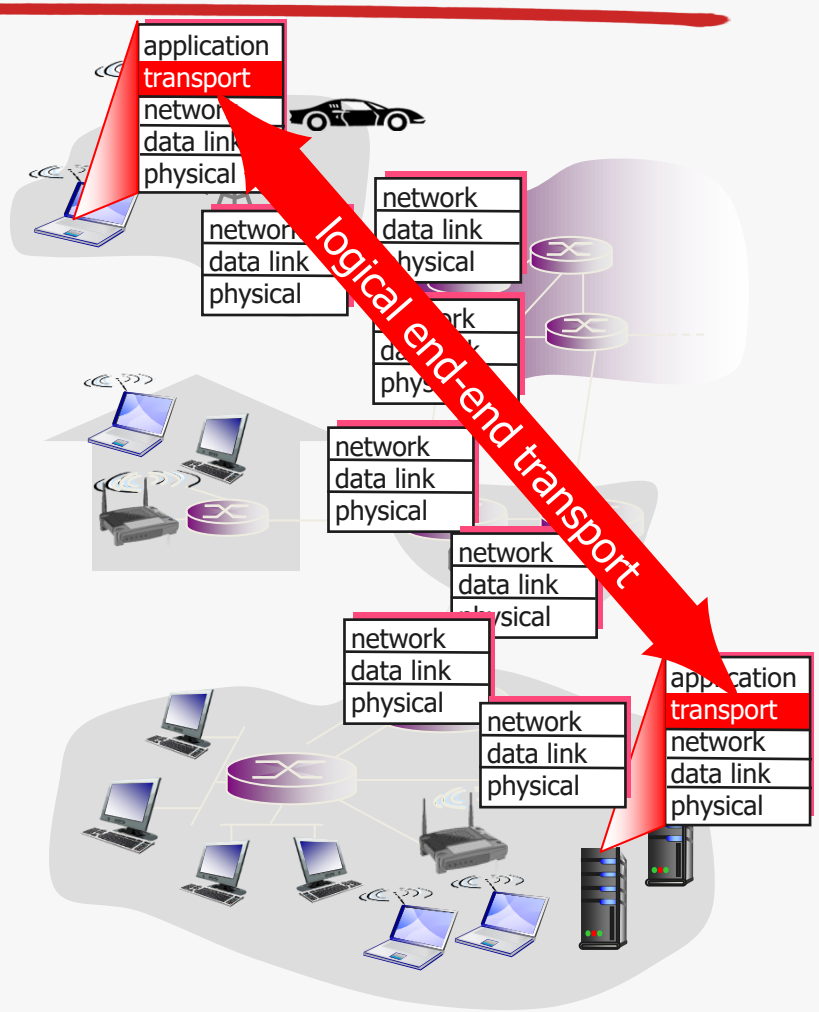
- Điền khiển tắt nghẽn
- Điều khiển luồng
- Thiết lập kết nối

Không tin cậy, truyền không theo thứ tự: UDP

- Không rườm rà, mở rộng "nổ lực tốt nhất" (best-effort) của IP

Không có các dịch vụ:

- Bảo đảm độ trễ
- Bảo đảm băng thông



UDP: User Datagram Protocol [RFC 768]

"đơn giản," "bare bones" Internet transport protocol

Dịch vụ "best effort" ("nổ lực tốt nhất"), các segment UDP segments có thể bị:

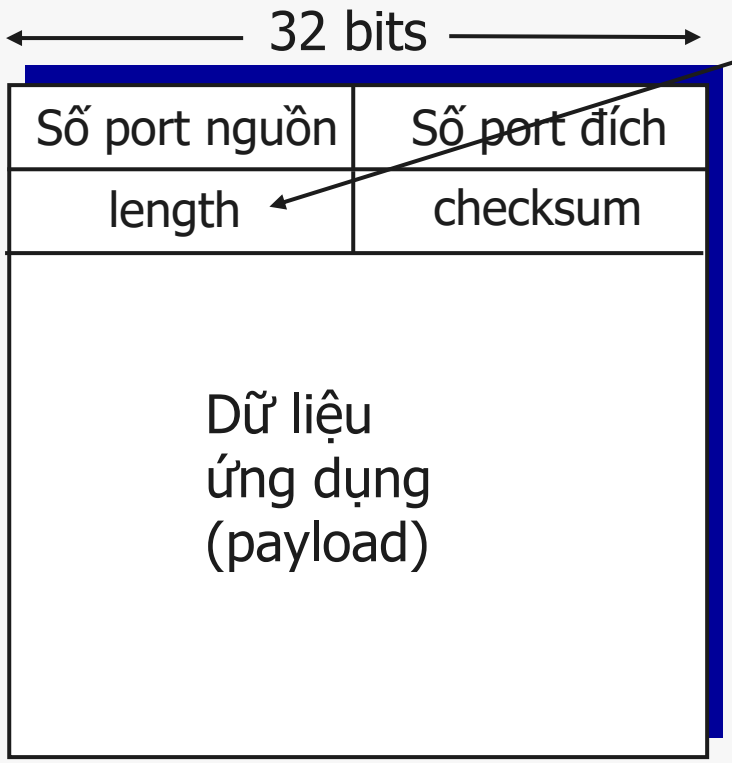
- Mất mát
- Vận chuyển không theo thứ tự đến ứng dụng

Connectionless (phi kết nối):

- Không bắt tay giữa bên nhận và gửi UDP
- Mỗi segment UDP được xử lý độc lập

- Ứng dụng UDP:
  - Các ứng dụng đa phương tiện trực tuyến (chịu mất mát(loss tolerant), (cần tốc độ) (rate sensitive))
  - DNS
  - SNMP
- Truyền tin cậy trên UDP:
  - Thêm độ tin cậy tại tầng application
  - Phục hồi lỗi tại các ứng dụng cụ thể!

UDP: segment header



Định dạng segment UDP

Độ dài được tính bằng byte của segment UDP, bao gồm cả header

Tại sao có UDP?

- Không thiết lập kết nối (cái mà có thể gây ra độ trễ)
- Đơn giản: không trạng thái kết nối tại nơi gửi và nhận
- Kích thước header nhỏ
- Không điều khiển tắt nghẽn: UDP có thể gửi dữ liệu nhanh như mong muốn

UDP checksum

Mục tiêu: dò tìm "các lỗi" (các bit cờ được bật) trong các segment đã được truyền

bên gửi:

Xét nội dung của segment, bao gồm các trường của header, là chuỗi các số nguyên 16-bit

checksum: bổ sung (tổng bù 1) của các nội dung segment

Bên gửi đặt giá trị checksum vào trường checksum UDP

bên nhận:

Tính toán checksum của segment đã nhận

Kiểm tra giá trị trên có bằng với giá trị trong trường checksum hay không:

- NO - có lỗi xãy ra
- YES - không có lỗi. *Nhưng có thể còn lỗi khác nữa không?* Xem phần sau….

Giao thức nào dưới đây không đảm bảo dữ liệu gửi đi có tới máy nhận hoàn chỉnh hay không?

A TCP

B UDP

C ARP

D RARP

Cho 2 số nguyên 16 bit
1 0 0 1 1 0 0 1 0 1 0 0 1 1 1 0
1 1 1 0 0 1 0 0 0 1 1 0 0 1 0 1
Kết quả tính check sum của 2 dãy số nguyên trên là:

A 1000001001001100

B 1000001001001011

C 0111110110110011

D 0111110110110100

Internet checksum: ví dụ

Ví dụ: cộng 2 số nguyên 16 bit

	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
bit dư		1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
tổng	1	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

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

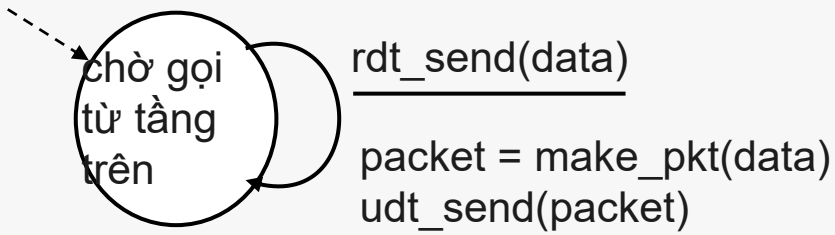
rdt1.0: truyền tin cậy trên 1 kênh tin cậy

Kênh cơ bản tin cậy hoàn toàn (underlying channel perfectly reliable)

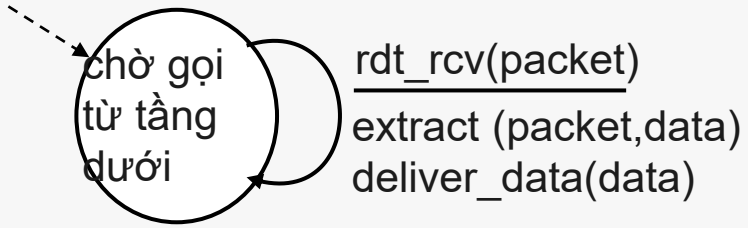
- không có bit lỗi
- không mất mát gói

Các FSMs riêng biệt cho bên gửi và nhận:

- Bên gửi gửi dữ liệu vào kênh cơ bản (underlying channel)
- Bên nhận đọc dữ liệu từ kênh cơ bản (underlying channel)



bên gửi



bên nhận

rdt2.0: kênh với các lỗi

Kênh cơ bản có thể đảo các bit trong packet

- checksum để kiểm tra các lỗi

Câu hỏi: làm sao khôi phục các lỗi:

Làm thế nào để con người phục hồi "lỗi" trong cuộc trò chuyện?

rdt2.0: kênh với các lỗi

Kênh cơ bản có thể đảo các bit trong packet
- checksum để kiểm tra các lỗi

Câu hỏi: làm sao khôi phục các lỗi:
- *acknowledgements (ACKs)*: bên nhận thông báo rõ ràng cho bên gửi rằng packet được nhận thành công (OK)
- *negative acknowledgements (NAKs)*: bên nhận thông báo rõ ràng cho bên gửi rằng packet đã bị lỗi
- Bên gửi truyền lại gói nào được xác nhận là NAK

Các cơ chế mới trong **rdt2.0** (sau **rdt1.0**):
- Phát hiện lỗi
- Phản hồi: các thông điệp điều khiển (ACK,NAK) từ bên nhận đến bên gửi

rdt2.0 có lỗ hỏng nghiêm trọng!

Điều gì xảy ra nếu ACK/NAK bị hỏng?

Bên gửi sẽ không biết điều gì đã xảy ra ở bên nhận!

Không thể đơn phương truyền lại: có thể trùng lặp

Xử lý trùng lặp:

Bên gửi truyền lại packet hiện thời nếu ACK/NAK bị hỏng

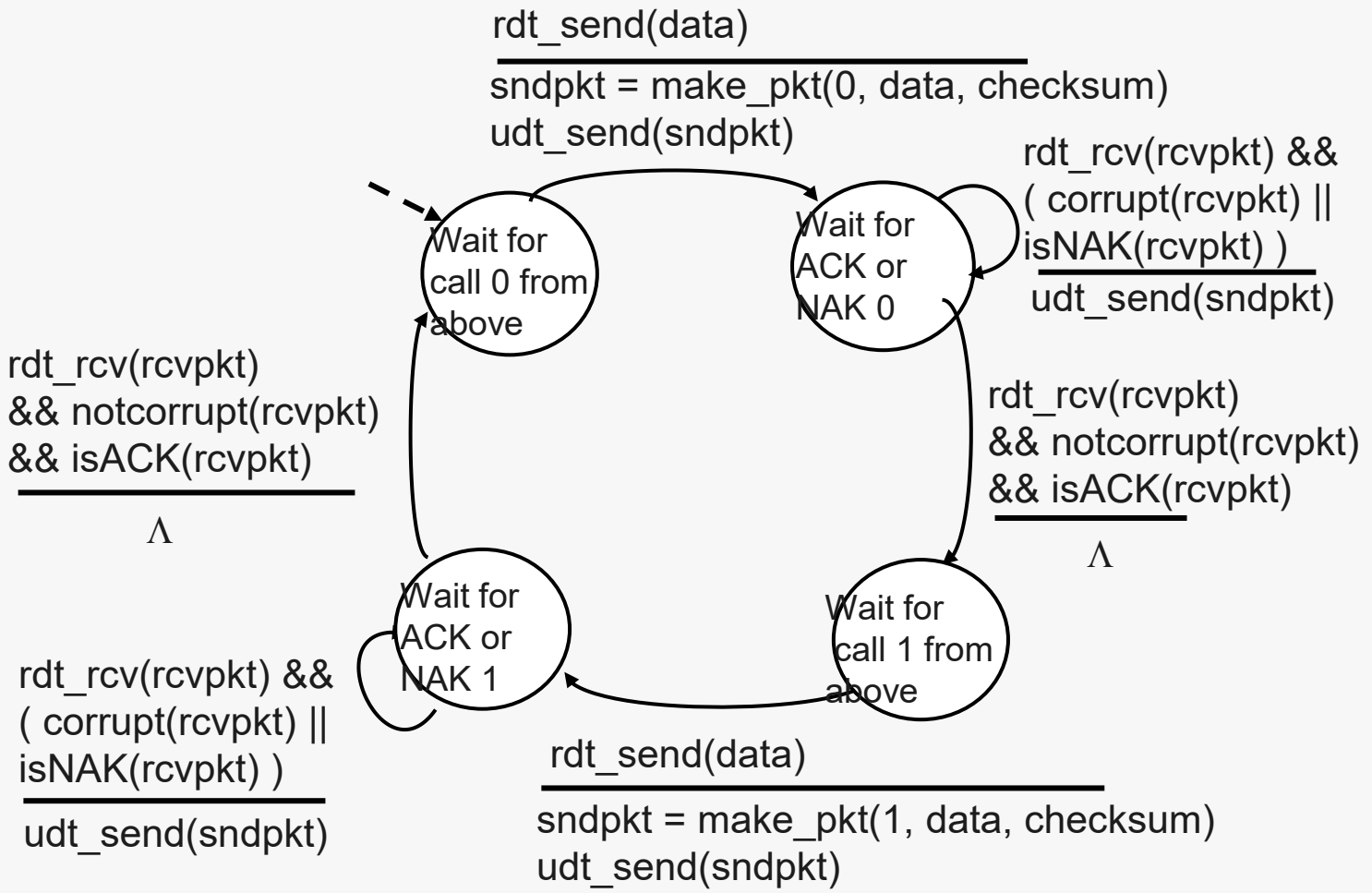
Bên gửi thêm *số thứ tự* vào trong mỗi packet *(sequence number)*

Bên nhận hủy packet bị trùng lặp

dừng và chờ

Bên gửi gửi một packet, sau đó chờ phản hồi từ bên nhận

rdt2.1: bên gửi, xử lý các ACK/NAK bị hỏng



Để thực hiện truyền nhận dữ liệu tin cậy trên đường truyền có lỗi ở cấp độ bit, hai cơ chế được sử dụng là: CheckSum để phát hiện lỗi ở cấp độ bit, và Phản hồi từ máy nhận (ACK: Máy nhận gửi ACK để nói rằng đã nhận gói tin dữ liệu không bị lỗi, NAK: Máy nhận gửi NAK để nói rằng đã nhận gói tin dữ liệu bị lỗi). Hãy nêu vấn đề của giải pháp trên?

A Vấn đề là khi gói tin phải hồi ACK/NAK bị lỗi, máy gởi không biết chính xác dữ liệu đã truyền đến máy nhận có bị lỗi không.

B Vấn đề là khi gói tin phải hồi ACK/NAK bị lỗi, máy gởi không biết chính xác dữ liệu đã truyền đến máy nhận có bị lỗi không.

C Vấn đề là khi gói tin phải hồi ACK/NAK bị lỗi, máy gởi không biết chính xác dữ liệu đã truyền đến máy nhận có bị lỗi không.

D Vấn đề là khi gói tin phải hồi ACK/NAK bị lỗi, máy gởi không biết chính xác dữ liệu đã truyền đến máy nhận có bị lỗi không.

rdt2.2: một giao thức không cần NAK

Chức năng giống như rdt2.1, chỉ dùng các ACK

Thay cho NAK, bên nhận gởi ACK cho gói cuối cùng được nhận thành công

- Bên nhận phải rõ ràng chèn số thứ tự của gói vừa được ACK

ACK bị trùng tại bên gửi dẫn tới kết quả giống như hành động của NAK: *truyền lại gói vừa rồi*

rdt3.0: các kênh với lỗi và mất mát

Giả định mới: kênh ưu tiên cũng có thể làm mất gói (dữ liệu, các ACK)

- checksum, số thứ tự, các ACK, việc truyền lại sẽ hỗ trợ...nhưng không đủ

Cách tiếp cận: bên gửi chờ ACK trong khoảng thời gian "hợp lý"

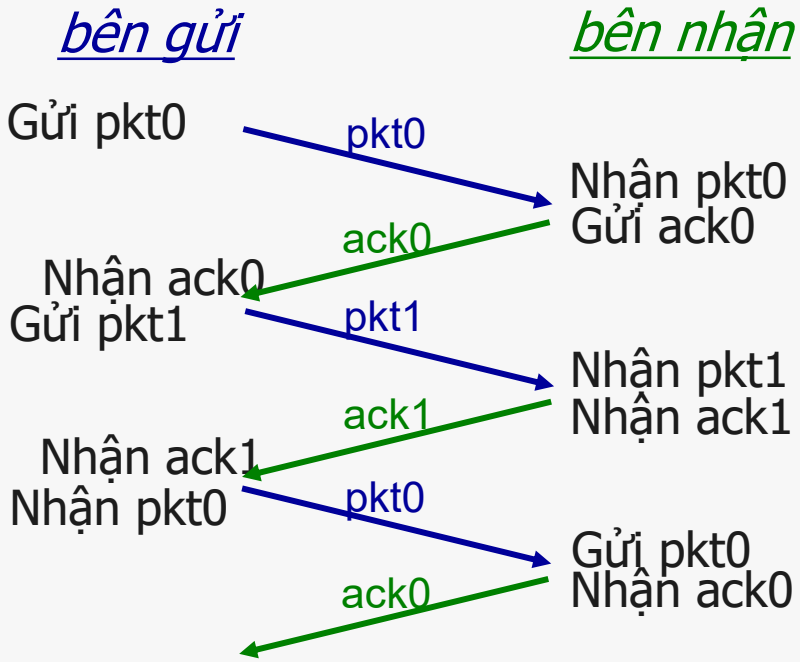
Truyền lại nếu không có ACK được nhận trong khoảng thời gian này

Nếu gói (hoặc ACK) chỉ trễ (không mất):

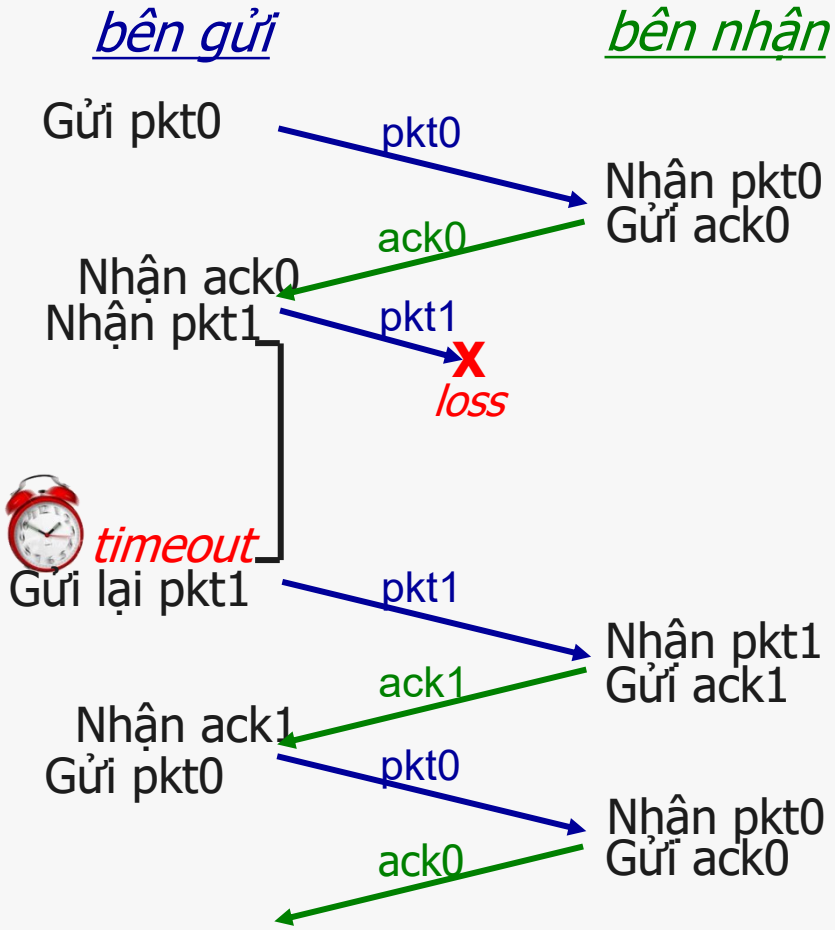
- Việc truyền lại sẽ gây trùng, nhưng số thứ tự đã xử lý trường hợp này
- Bên nhận phải xác định số thứ tự của gói vừa gửi ACK

Yêu cầu bộ định thì đếm lùi

Hành động của rdt3.0

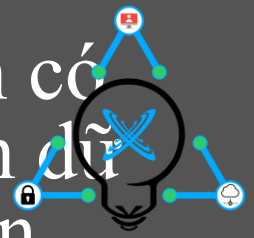


(a) Không mất mát



(b) Mất gói

Để đảm bảo quá trình truyền nhận dữ liệu đúng trên kênh truyền có khả năng có lỗi hoặc mất gói tin xảy ra, sử dụng giao thức truyền dữ liệu tin cậy rtd3.0, các điều kiện và thông số nào sau đây là cần thiết:



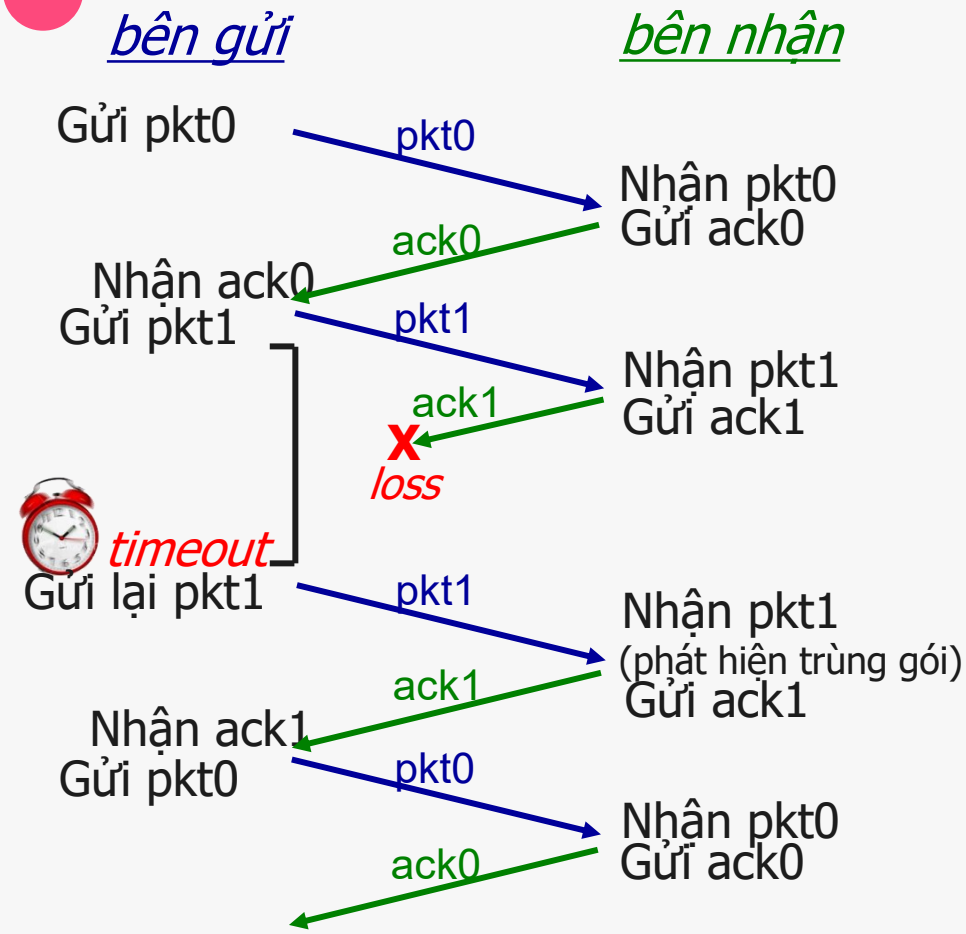
A Checksum,sequential number,ACK,retransmission,timer

B Sidling window,go –back-n, selective repeat

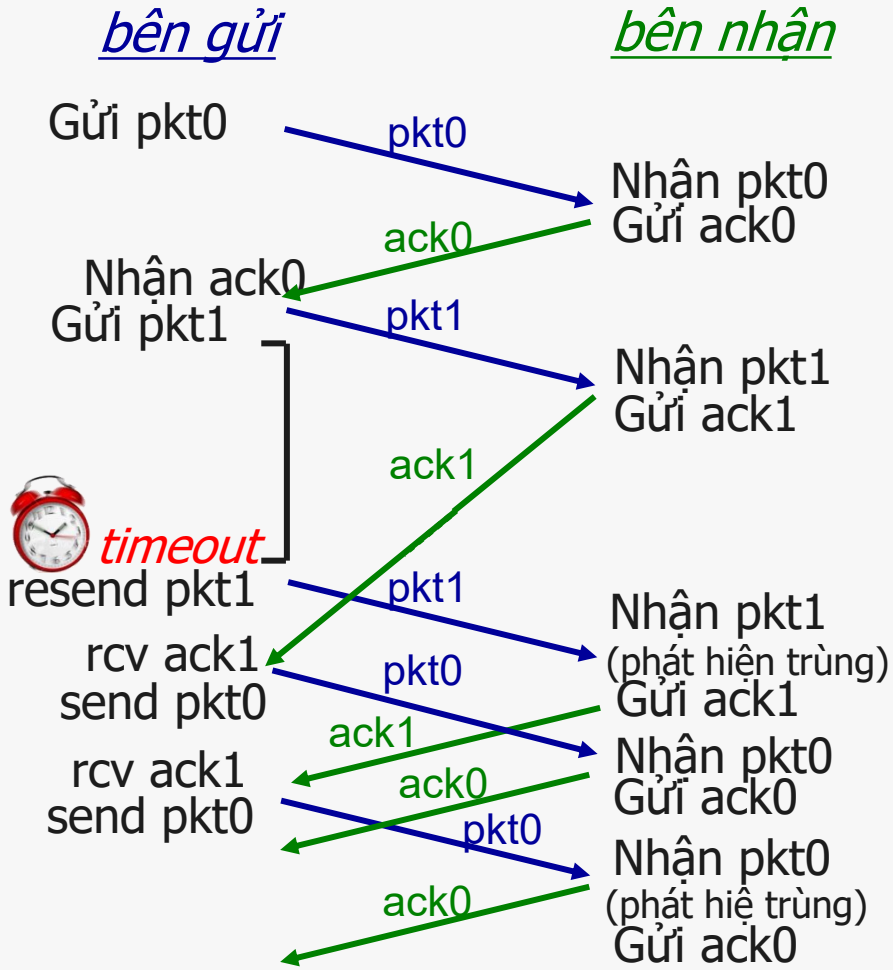
C Checksum,sequential number,ACK,NAK,retransmission

D Slow start, congestion avoidance,fast retransmit,fast recovery

Hành động của rdt3.0

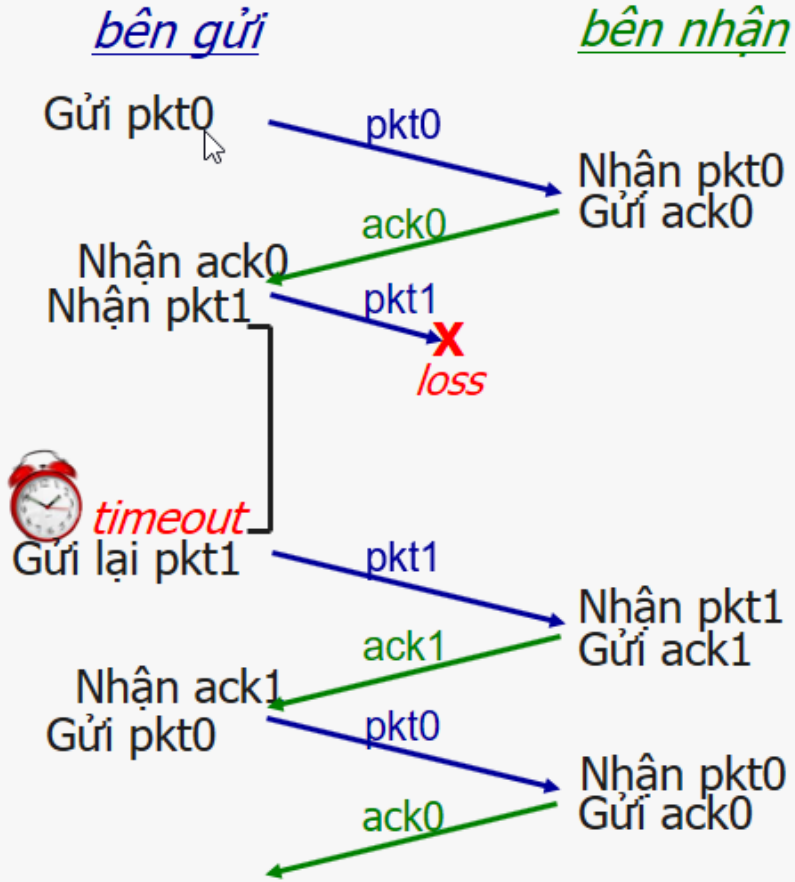


(c) Mất ACK



(d) premature timeout/ delayed ACK

Xem hình và cho biết đây là hành động nào của rdt 3.0?



A Không mất mát

B Mất gói tin dữ liệu

C Mất ACK

D Timeout/delayed ACK

Trong giao thức truyền tin cậy (rtd),giao thức nào sau đây xử lí được trường hợp mất gói tin ACK:

A Rtd2.1

B Rtd3.0

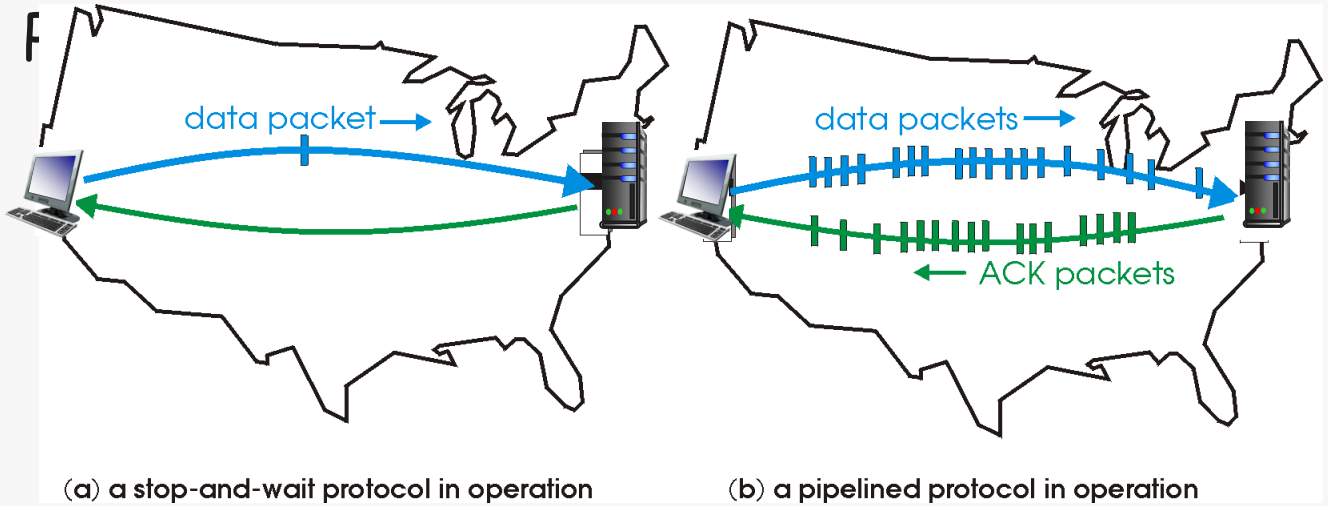
C Rtd2.2

D Tất cả các đáp án trên

Các giao thức Pipelined

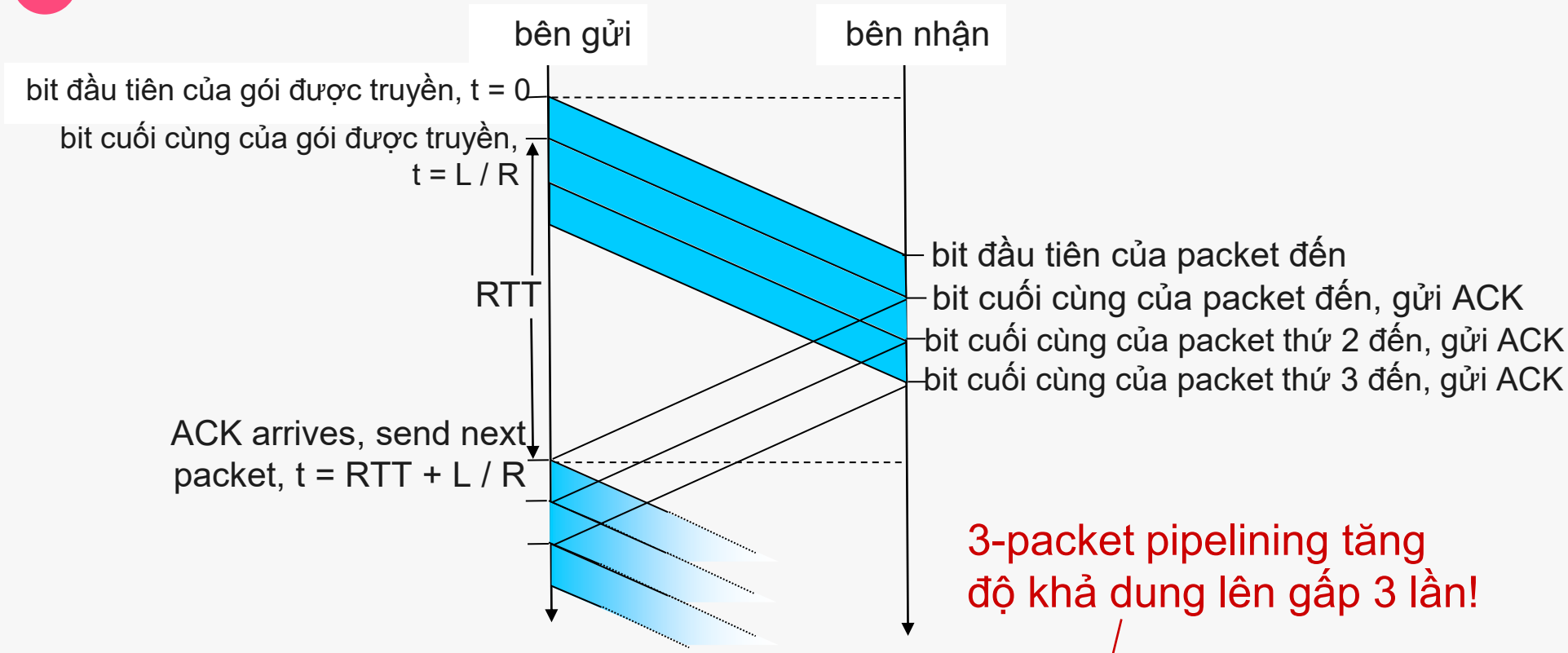
pipelining: bên gửi cho phép gửi nhiều gói đồng thời, không cần chờ báo nhận được

- Nhóm các số thứ tự phải được tăng dần
- F



hai dạng phổ biến của các giao thức pipelined : *go-Back-N, lặp có lựa chọn*

Pipelining: độ khả dụng tăng



3-packet pipelining tăng độ khả dụng lên gấp 3 lần!

$$U_{sender} = \frac{3L / R}{RTT + L / R} = \frac{.0024}{30.008} = 0.00081$$

Pipelined protocols: tổng quan

Go-back-N:

Bên gửi có thể có đến N packet không cần ACK trong đường ống (pipeline)

Bên nhận chỉ gởi *cumulative ack*

- Sẽ không thông báo nhận packet thành công nếu có một gián đoạn

bên gửi có bộ định thì cho packet sớm nhất mà không cần ACK (oldest unacked packet)

- Khi bộ định thì hết, truyền lại thất cả các packet mà không được ACK

Lặp có lựa chọn (Selective Repeat):

Bên gửi có thể có đến N packet không cần ACK trong đường ống (pipeline)

Bên nhân gửi rcvr ack rieeng biệt (*individual ack)* cho mỗi packet

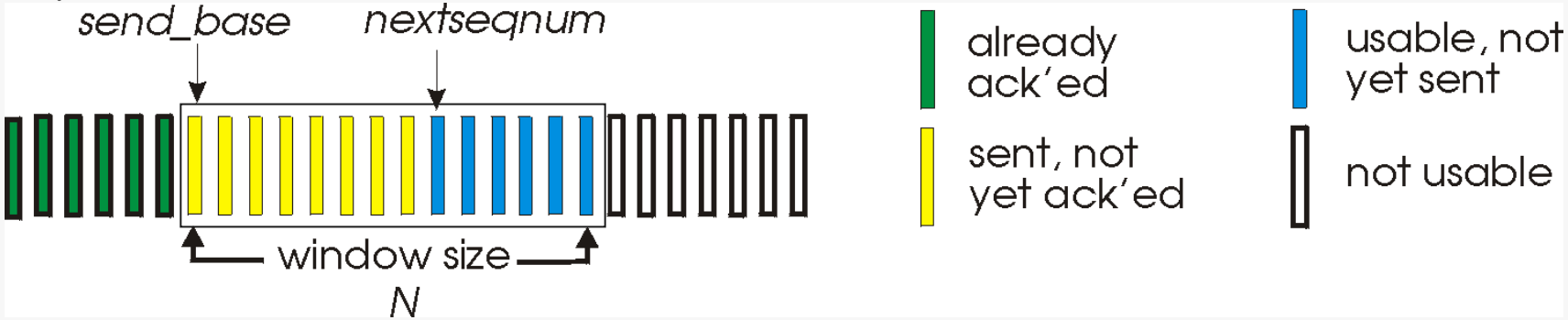
Bên nhận duy trì bộ định thì cho mỗi packet không được ACK

- Khi bộ định thì của packet nào hết hạn, thì chỉ truyền lại packet không được ACK đó

Go-Back-N: bên gửi

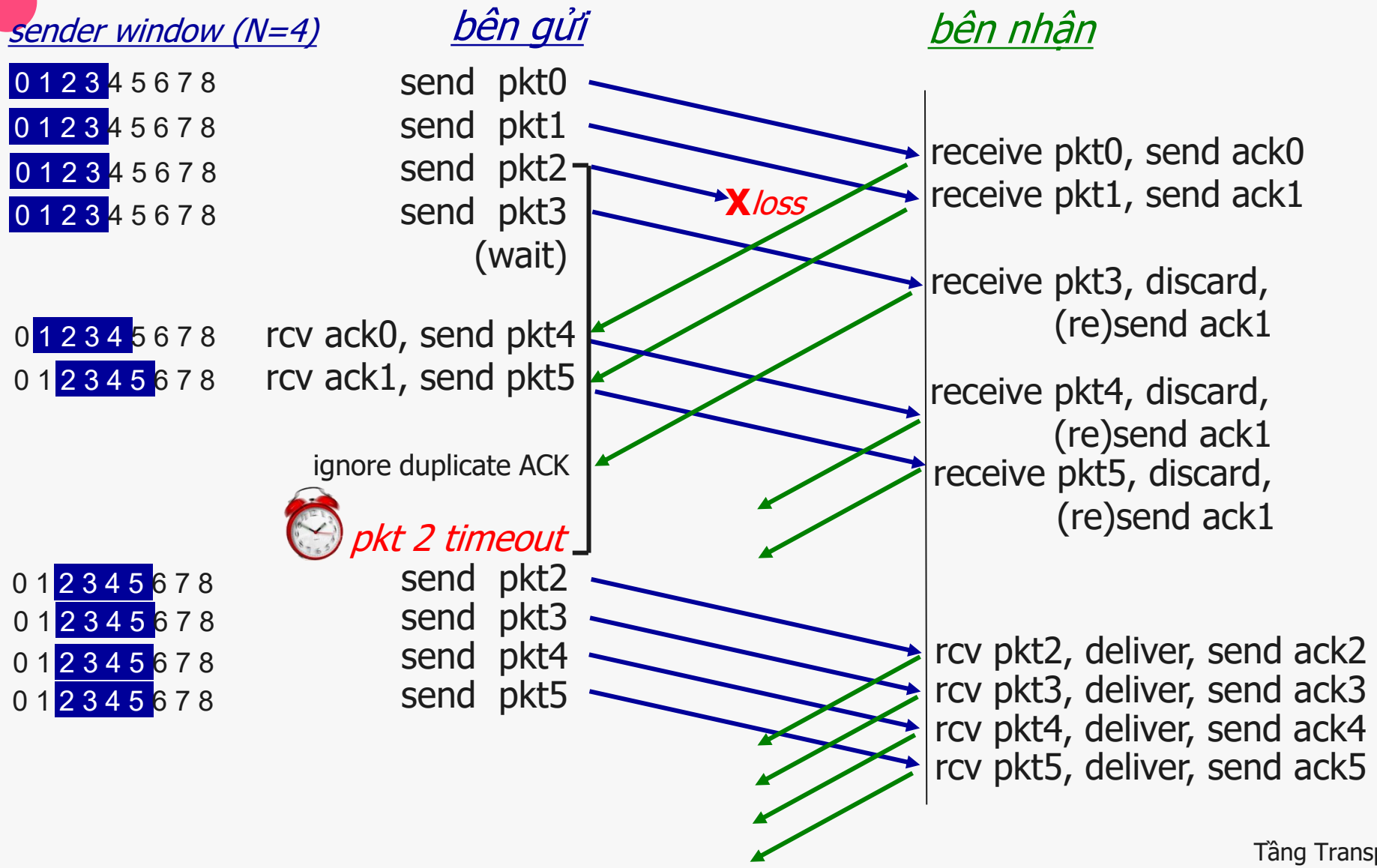
Số thứ tự k-bit trong header của packet

"cửa sổ"("window") lên đến N packet liên tiếp không cần ACK được cho phép



- ACK(n): thông báo nhận tất cả các packet lên đến n, bao gồm n số thứ tự - *"ACK tích lũy"("cumulative ACK")*
  - Có thể nhận ACK trùng (xem bên nhận)
- Định thì cho packet sớm nhất đang trong tiến trình xử lý (oldest in-flight pkt)
- *timeout(n)*: truyền lại packet n và tất cả các packet có số thứ tự cao hơn trong cửa sổ (window)

Hoạt động GBN



Lặp có lựa chọn (Selective repeat)

Bên nhận thông báo đã nhận đúng tất cả từng gói một

- Đệm các gói, khi cần thiết, cho sự vận chuyển trong thứ tự ngẫu nhiên đến tầng cao hơn

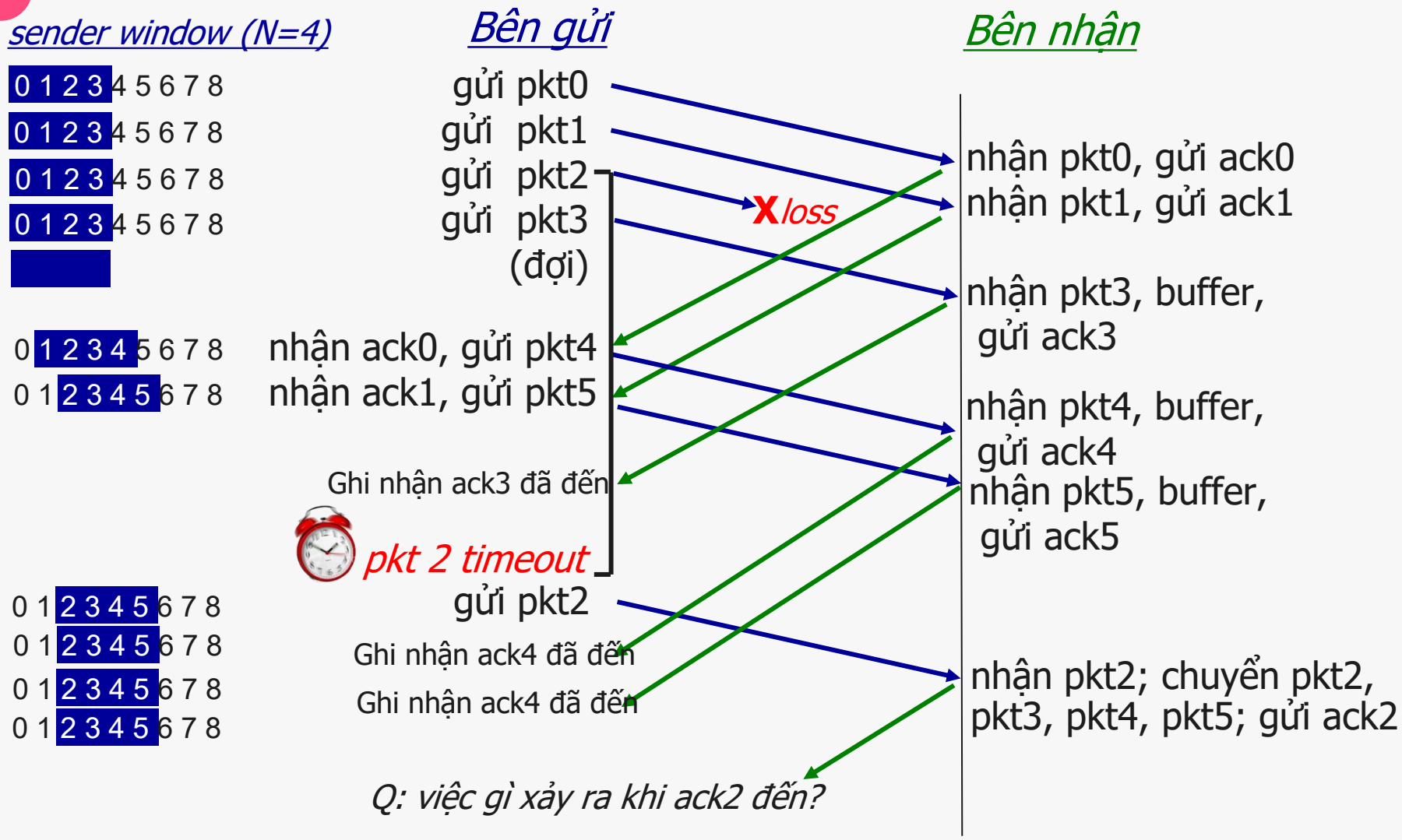
Bên gửi chỉ gửi lại các packet nào mà ACK không được nhận

- Bên gửi định thời cho mỗi packet không có gửi ACK

Cửa sổ bên gửi (sender window)

- N số thứ tự liên tục
- Hạn chế số thứ tự các gói không gửi ACK

Hành động của lặp lại có lựa chọn



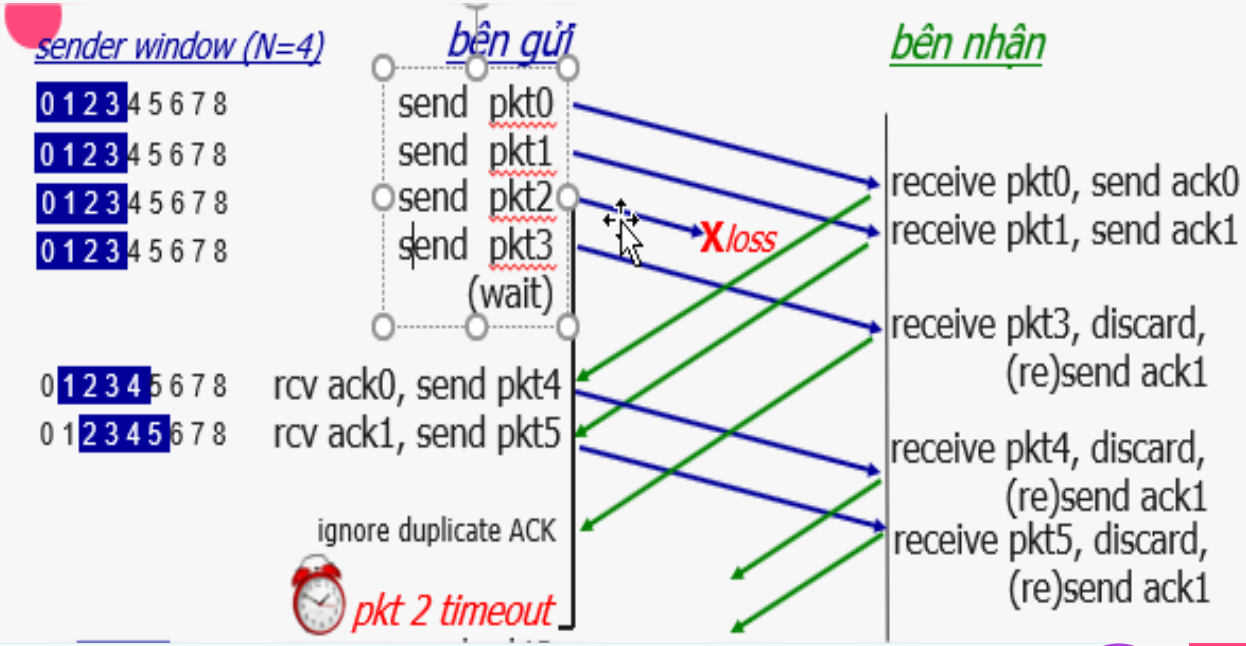
Xem hình mô tả hoạt động của Go-back-N dưới đây, sau thời gian timeout, bên gửi sẽ hành động như thế nào?

A Gửi lại pkt2,pkt3,pkt4,pkt5

B Gởi lại pkt2

C Gửi lại pkt0,pkt1,pkt2,pkt3

D Gửi lại pkt1,pkt2,pkt3,pkt4



TCP: tổng quan RFCs: 793,1122,1323, 2018, 2581

point-to-point:

- Một bên gửi, một bên nhận

Tin cậy, dòng byte theo thứ tự (in-order *byte steam*):

- Không "ranh giới thông điệp" ("message boundaries")

pipelined:

- Điều khiển luồng và tắt nghẽn của TCP thiết lập kích thước cửa sổ (window size)

Dữ liệu full duplex:

- Luồng dữ liệu đi 2 chiều trong cùng 1 kết nối
- MSS: kích thước tối đa của segment (maximum segment size)

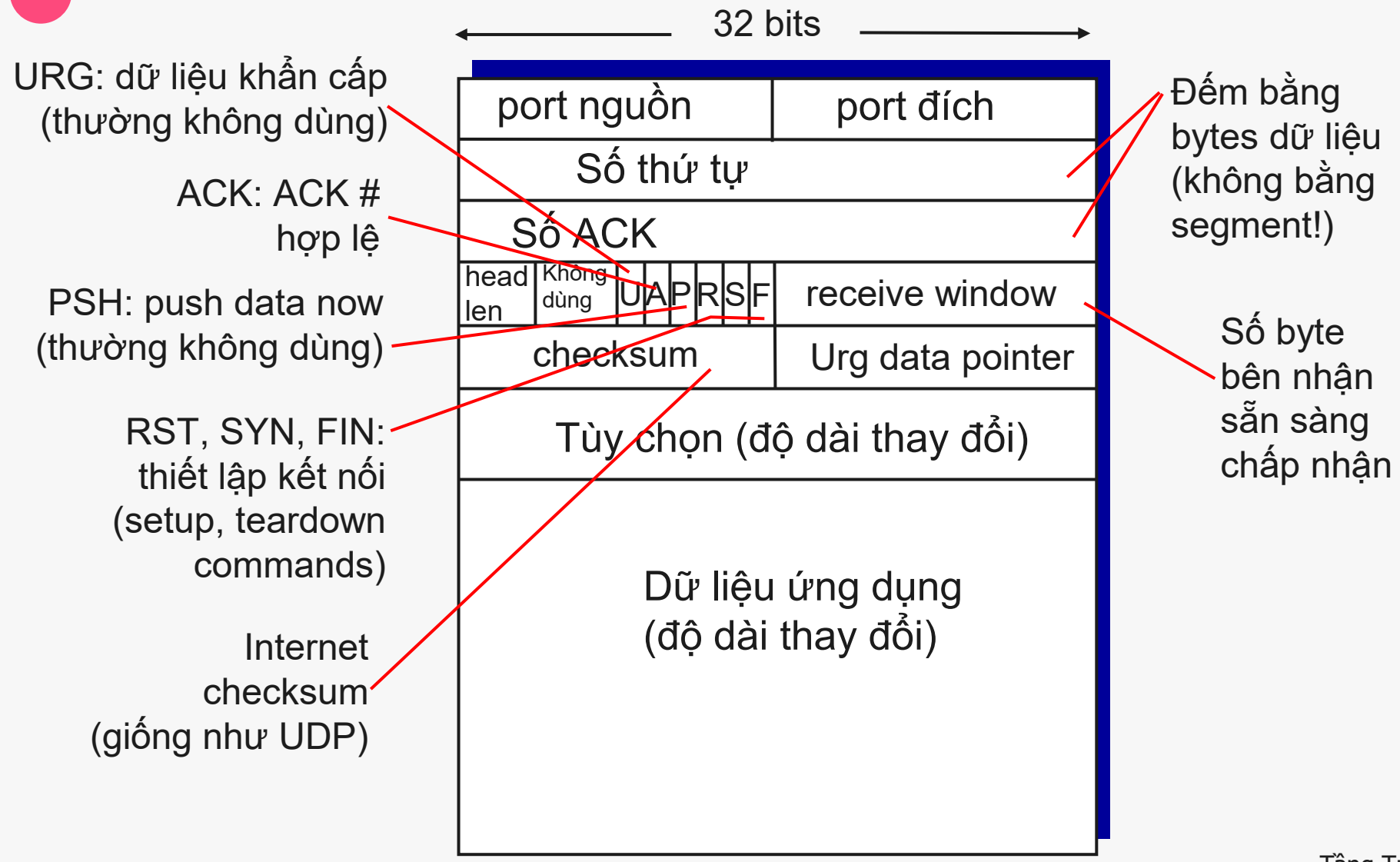
Hướng kết nối:

- Bắt tay (trao đổi các thông điệp điều khiển) khởi tạo trạng thái bên gửi và nhận trước khi trao đổi dữ liệu

Luồng được điều khiển:

- Bên gửi sẽ không áp đảo bên nhận

Cấu trúc segment TCP segment



Trong cấu trúc header của TCP Segment có 6 cờ là:

A SYN, ACK, PSH, RST, FIN, URG

B CON, ACK, PSH,RST, FIN,URG

C SYN,ACK, PSH,DAT,CON,URG

D SYN,DAT,PSH,RST,FIN,URG

Số thứ tự TCP và ACK

Các số thứ tự:

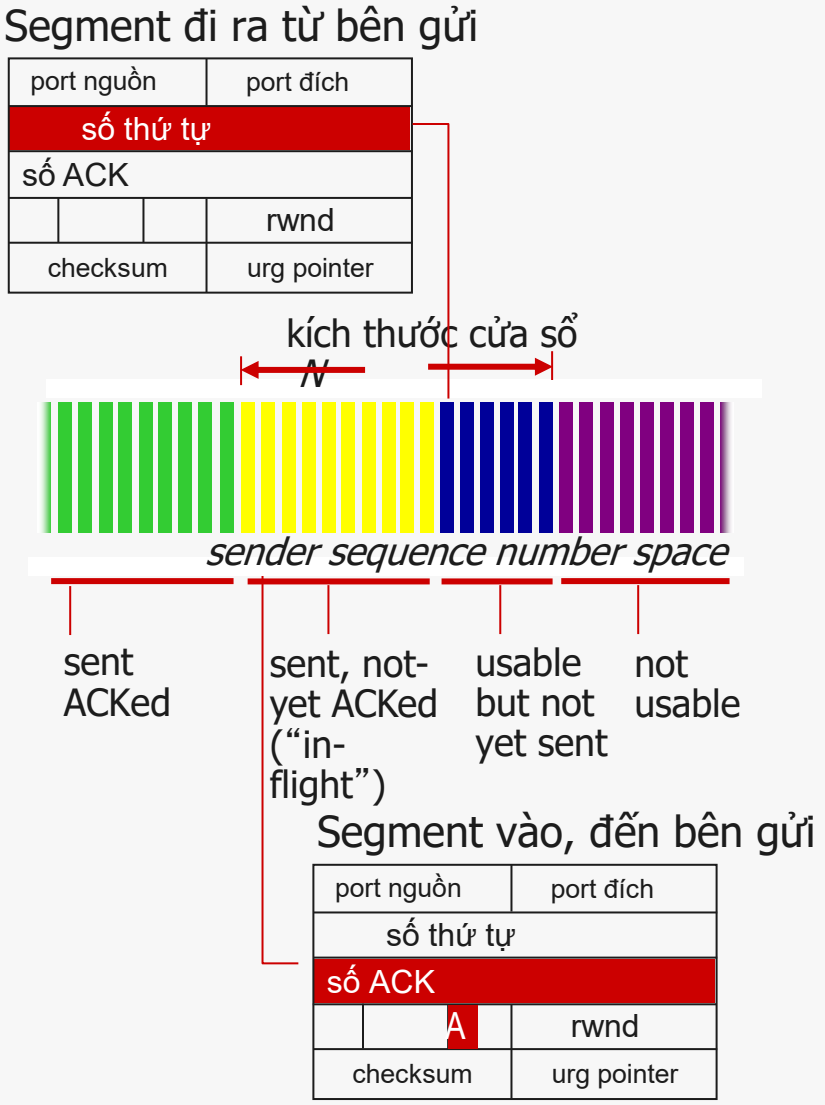
- Dòng byte "đánh số" byte đầu tiên trong dữ liệu của segment

Các ACK:

- số thứ tự của byte kế tiếp được mong đợi từ phía bên kia
- ACK tích lũy

Hỏi: làm thế nào để bên nhận xử lý các segment không theo thứ tự

- Trả lời: TCP không đề cập, tùy thuộc người thực hiện



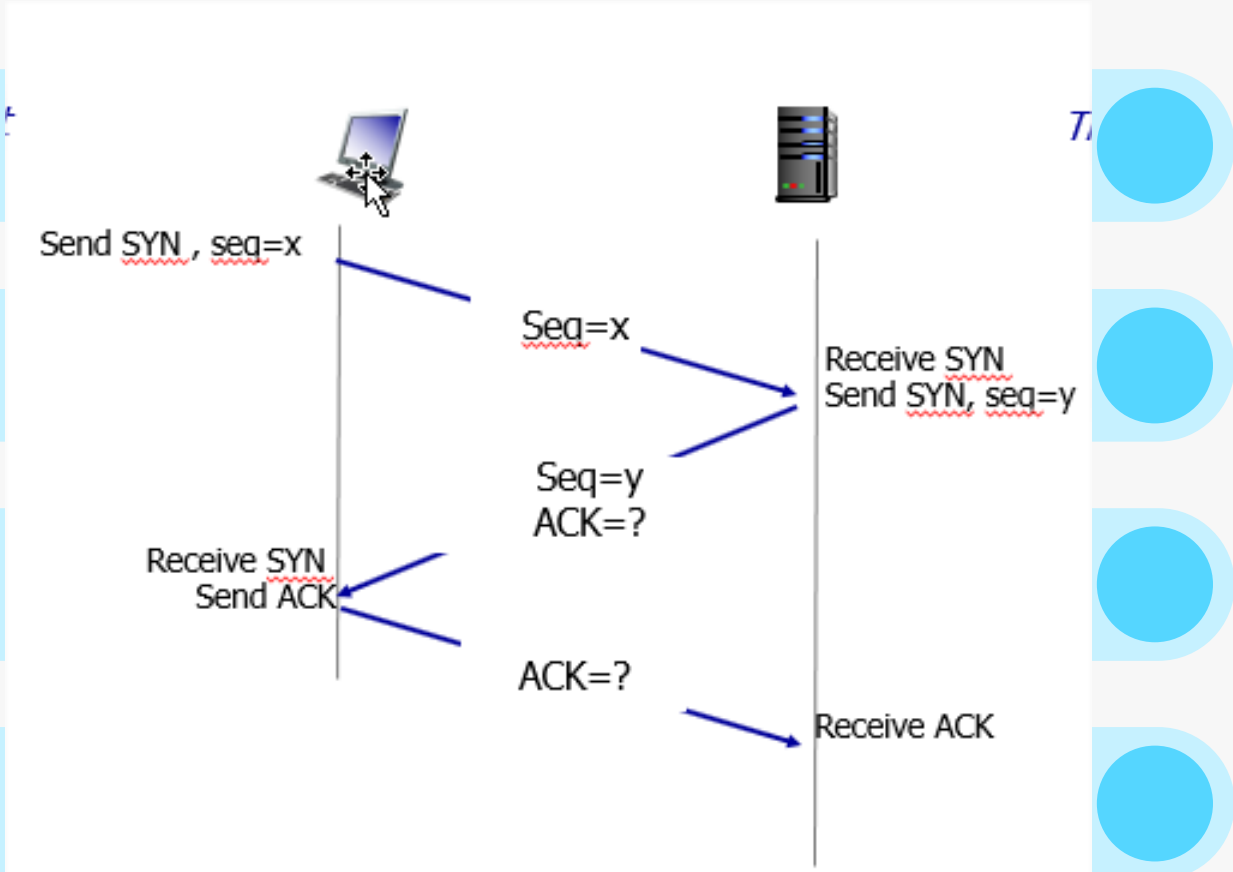
Mô tả quá trình bắt tay 3 bước trong kết nối TCP như hình: Ở bước 2, host B sẽ gửi gói tin sang host A có trường ACK number là bao nhiêu?

A X+1

B Y+1

C 0

D 11



Điều nào sau đây là đúng về bắt tay 3 bước của TCP

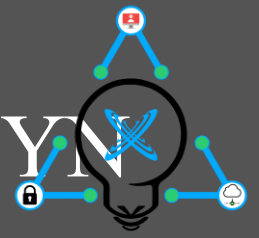
A FIN bit của gói đầu tiên được gán bằng 1

B SYN bit của gói đầu tiên được gán bằng 1

C Số seq của gói SYN đầu tiên luôn luôn là 0

D Gói TCP SYN đầu tiên được gửi ra từ phía server

Trong giao thức TCP, SYN segment sẽ có SEQ và giá trị SYN flag là bao nhiêu?



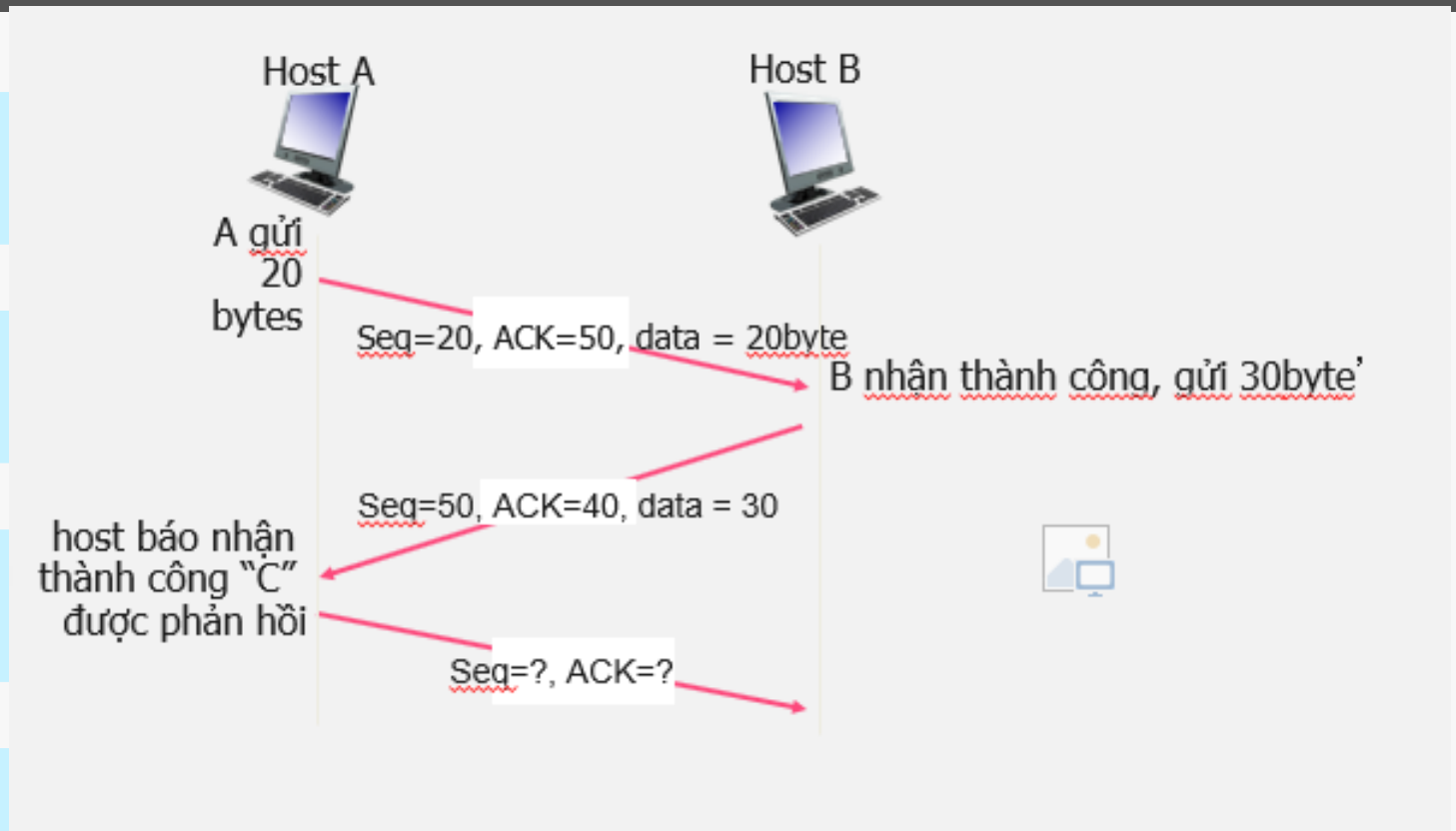
A SEQ=ISN,SYN =1(ISN: initial sequence number)

B SEQ=1,SYN =1

C SEQ=ISN, SYN=0

D SEQ=0,SYN=0

Dựa trên hình dưới đây, giá trị của số thứ tự (SEQ)và ACK trong gói tin cuối cùng là bao nhiêu?



A SEQ=80, ACK=50

B SEQ=40,ACK =80

C SEQ=40, ACK=50

D SEQ=80, ACK=50

TCP truyền dữ liệu tin cậy

TCP tạo dịch vụ rdt trên dịch vụ không tin cậy của IP

- Các segment pipelined
- Các ack tích lũy
- Bộ định thì truyền lại đơn (single retransmission timer)

Việc truyền lại được kích hoạt bởi:

- Sự kiện timeout
- Các ack bị trùng

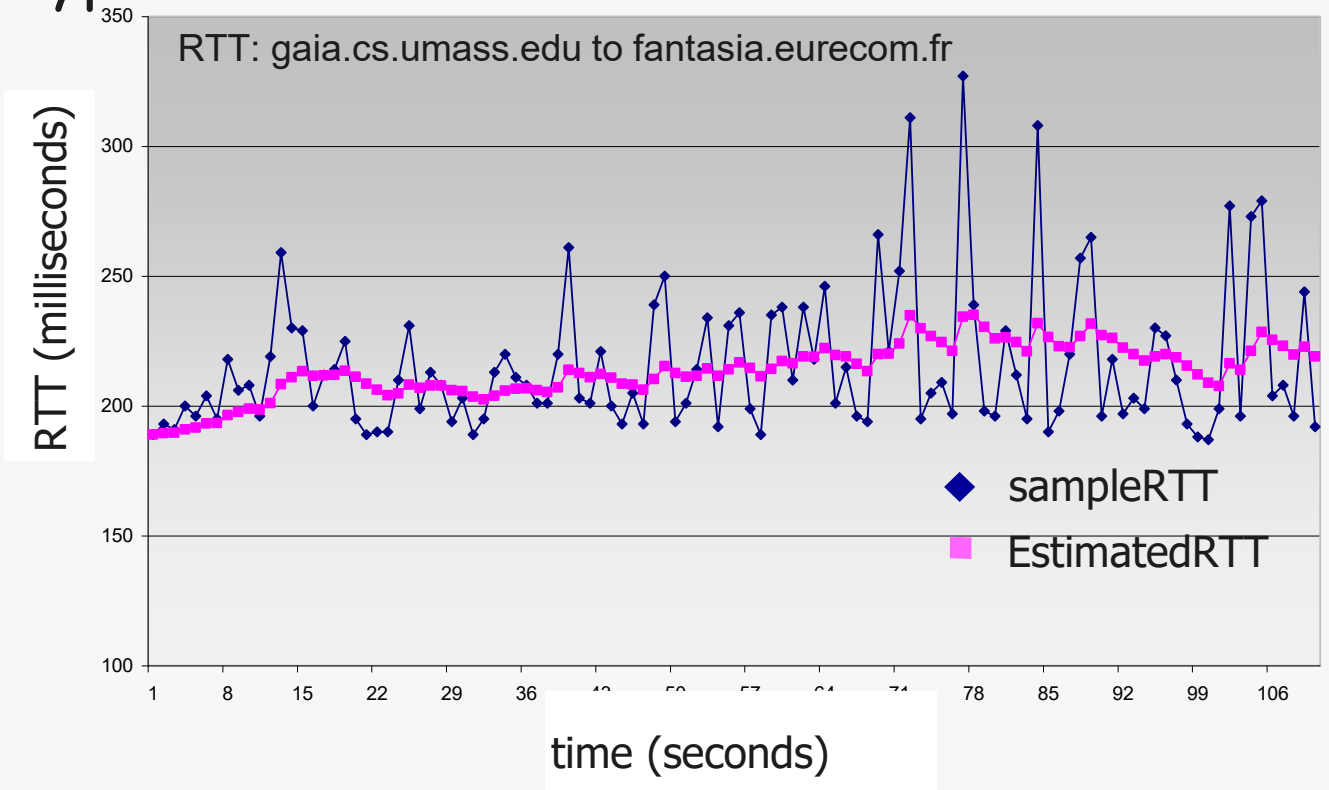
Lúc đầu khảo sát TCP đơn giản ở bên gửi:

- Lờ đi các ack bị trùng
- Lờ đi điều khiển luồng và điều khiển tắt nghẽn

TCP round trip time và timeout

EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT

- Đường trung bình dịch chuyển hàm mũ (exponential weighted moving average)
- ảnh hưởng của mẫu đã xảy ra sẽ làm giảm tốc độ theo cấp số nhân
- typical value: α = 0.125



TCP round trip time và timeout

Khoảng thời gian timeout (timeout interval):

EstimatedRTT cộng với "biên an toàn"

- Sự thay đổi lớn trong **EstimatedRTT** -> an toàn biên lớn hơn

Ước lượng độ lệch SampleRTT từ EstimatedRTT:

$$DevRTT = (1-\beta)*DevRTT + \beta*|SampleRTT-EstimatedRTT|$$

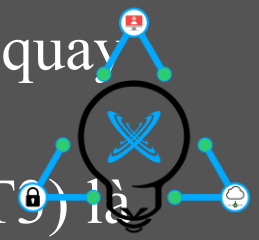
(typically, $\beta = 0.25$)

$$TimeoutInterval = EstimatedRTT + 4*DevRTT$$

estimated RTT

"biên an toàn"

Xem thủ tục ước lượng RTT của TCP: Giả sử một kết nối TCP có 4 segment ACK quay về bên gửi và nhờ đó người ta đo được thời gian đi-về của segment thứ nhất (SampleRTT1) là 90 msec, thứ hai (SampleRTT2) là 110 msec, thứ ba (SampleRTT3) là 114 msec, và thứ tư (SampleRTT4) là 88msec. Giả sử hệ số anpha=0.2.Người ta ước lượng đucợ giá trị EstimatedRTT ngay sau khi ACK thứ tư quay về là:



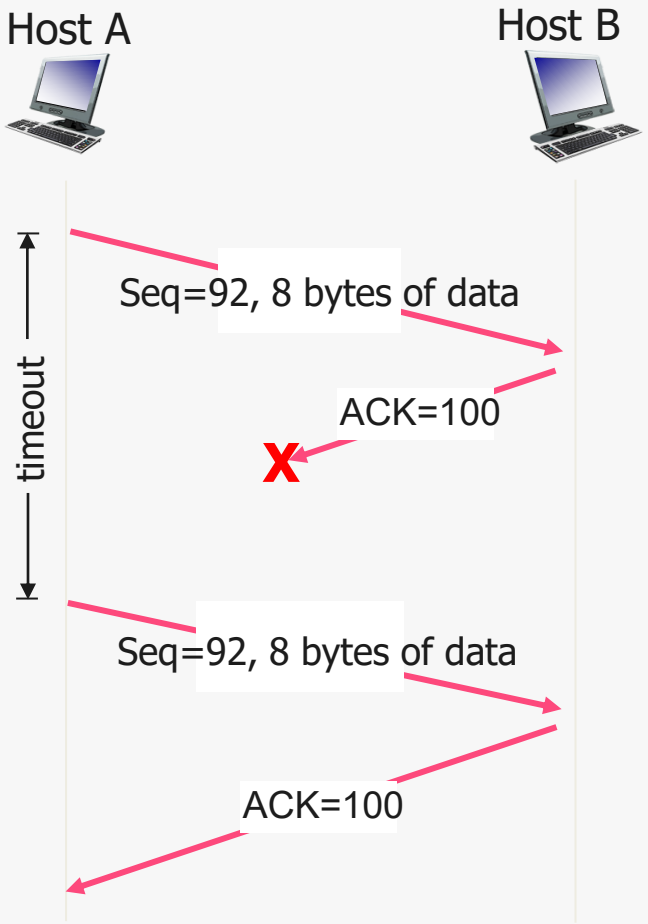
A 92.88 msec

B 96 msec

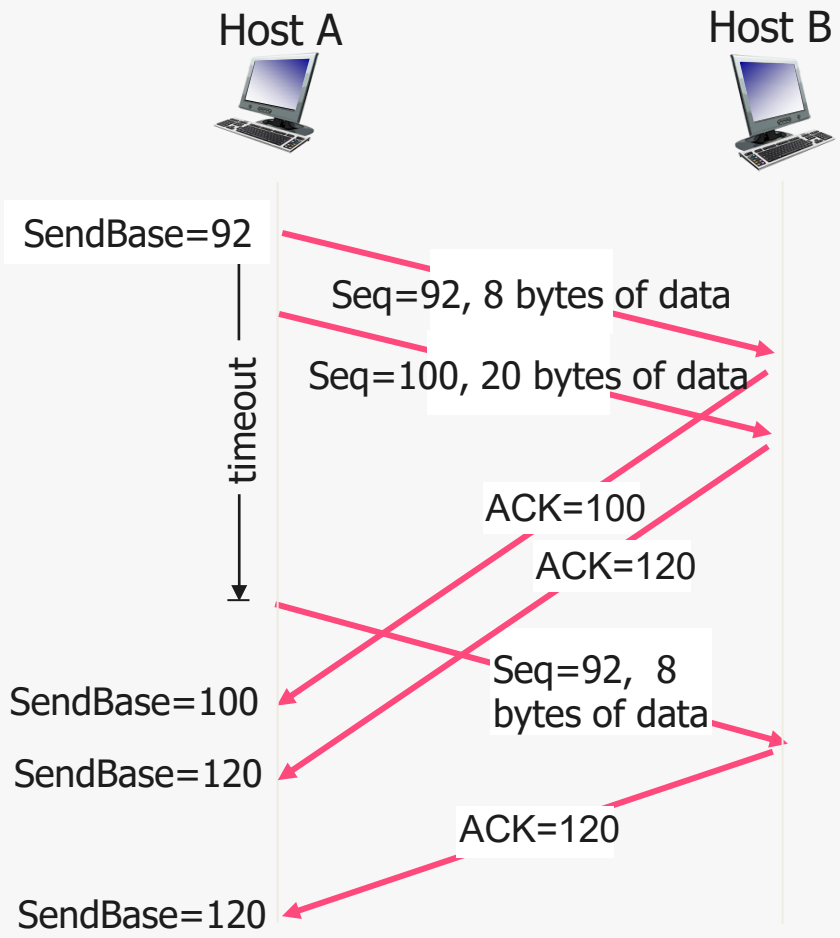
C Các lựa chọn đều sai.

D 100.5 msec

TCP: tình huống truyền lại

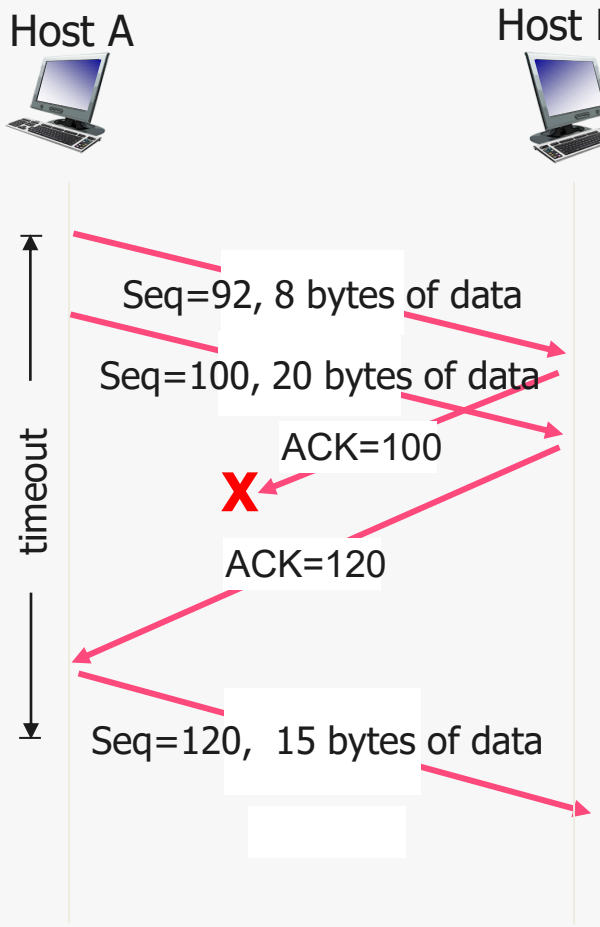


Tình huống mất ACK



Timeout sớm

TCP: tình huống truyền lại



ACK tích lũy

TCP truyền lại nhanh

Chu kỳ time-out thường tương đối dài:

- Độ trễ dài trước khi gởi lại packet bị mất

Phát hiện các segment bị mất thông qua các ACKs trùng.

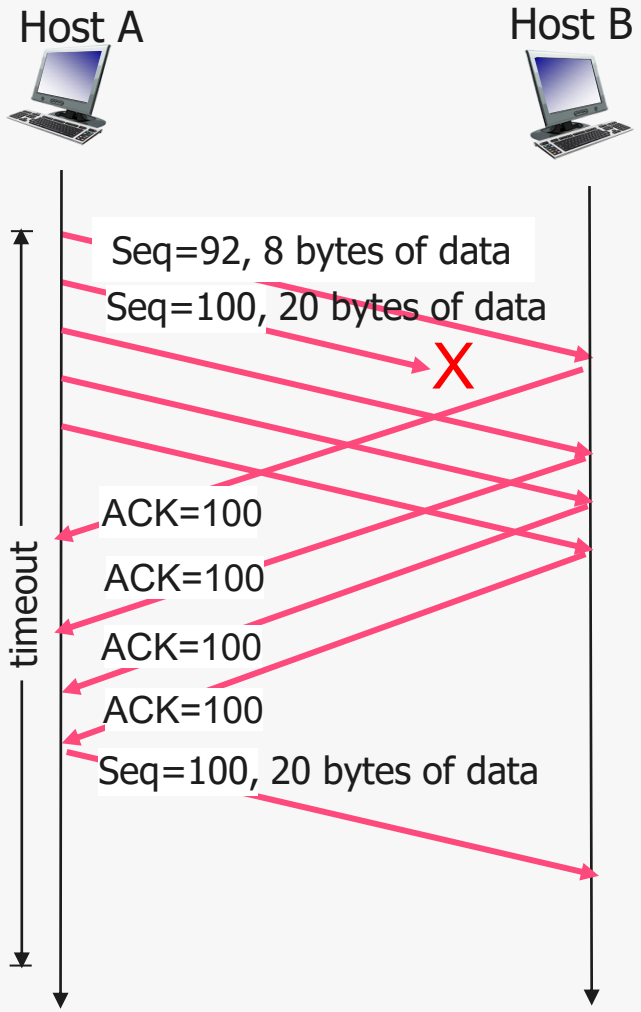
- Bên gửi thường gửi nhiều segment song song
- Nếu segment bị mất, thì sẽ có khả năng có nhiều ACK trùng.

TCP truyền lại nhanh

Nếu bên gửi nhận 3 ACK của cùng 1 dữ liệu ("3 ACK trùng"), thì gửi lại segment chưa được ACK với số thứ tự nhỏ nhất

- Có khả năng segment không được ACK đã bị mất, vì thế không đợi đến thời gian timeout

TCP truyền lại nhanh

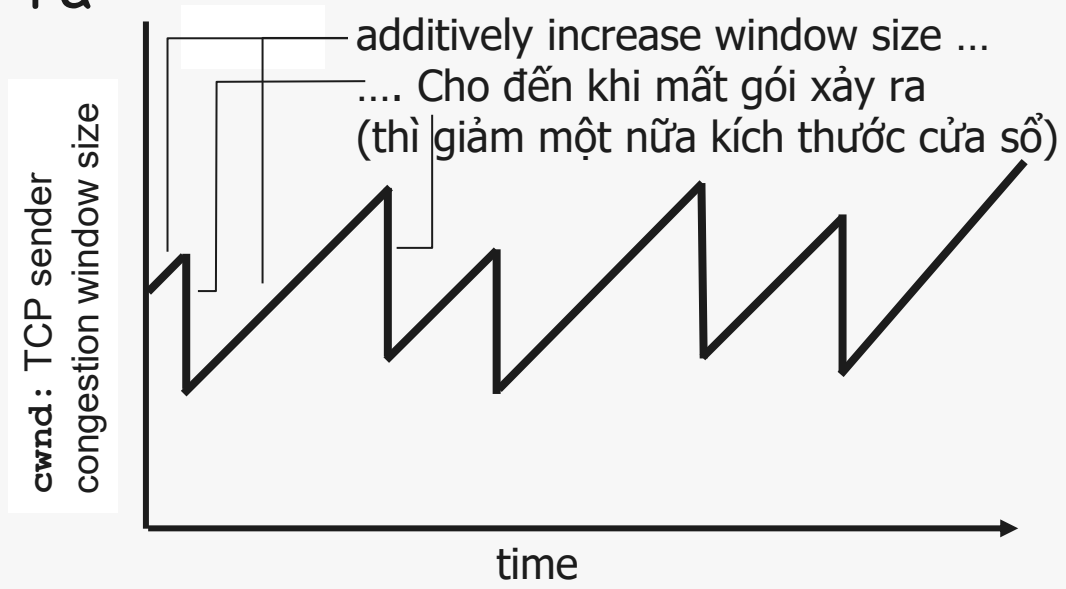


Truyền lại nhanh sau khi
bên gửi nhận 3 lần ACK bị trùng

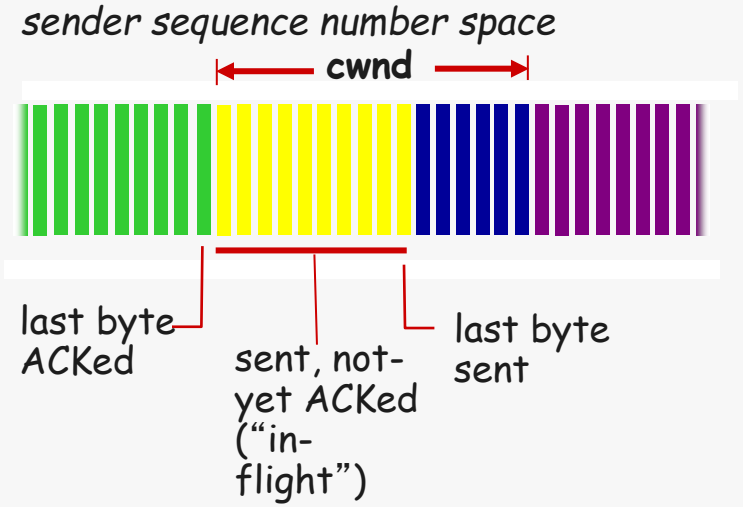
TCP điều khiển tắt nghẽn: additive increase, multiplicative decrease

- *Hướng tiếp cận:* bên gửi tăng tốc độ truyền (kích thước cửa sổ), thăm dò băng thông có thể sử dụng, cho đến khi mất mát gói xảy ra
  - *additive increase:* tăng **cwnd** bởi 1 MSS mỗi RTT cho đến khi mất gói xảy ra
  - *multiplicative decrease:* giảm một nữa **cwnd** sau khi mất gói xảy ra

AIMD saw tooth behavior: thăm dò băng thông



TCP điều khiển tắt nghẽn: chi tiết



Bên gửi giới hạn truyền tải:

$$\text{LastByteSent} - \text{LastByteAcked} \leq \text{cwnd}$$

cwnd thay đổi, chức năng nhận biết tắt nghẽn trên mạng

TCP tốc độ gửi:

Ước lượng: gửi các byte cwnd, đợi ACK trong khoảng thời gian RTT, sau đó gởi thêm các byte

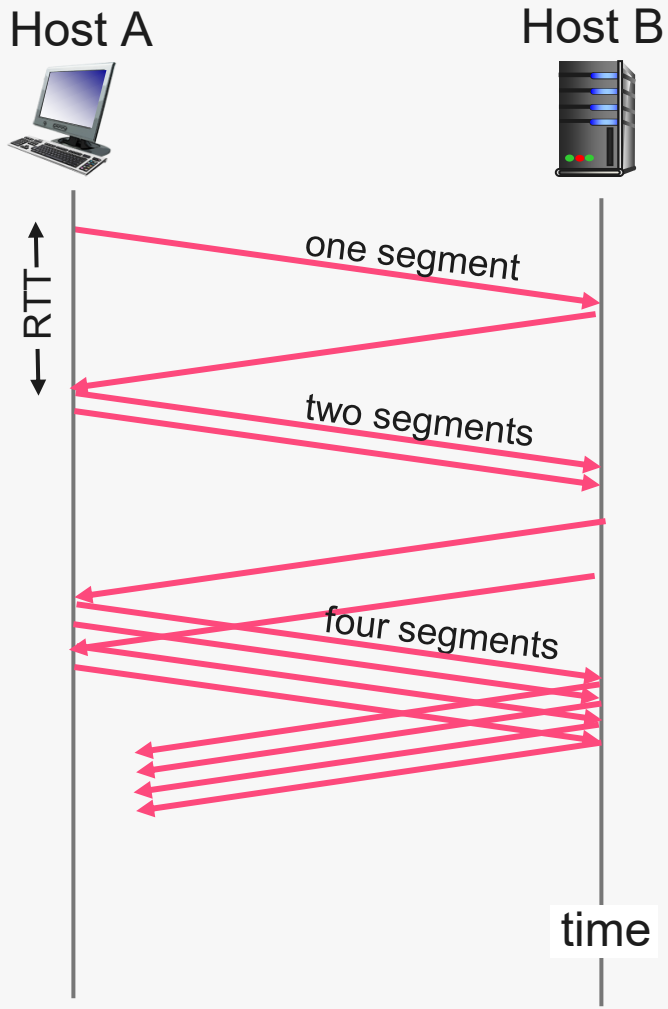
$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

TCP Slow Start

Khi kết nối bắt đầu, tăng tốc độ theo cấp số nhân cho đến sự kiện mất gói đầu tiên xảy ra:

- initially **cwnd** = 1 MSS
- Gấp đôi **cwnd** mỗi RTT
- Được thực hiện bằng cách tăng **cwnd** cho mỗi ACK nhận được

Tóm lại: tốc độ ban đầu chậm, nhưng nó sẽ tăng lên theo cấp số nhân



TCP: phát hiện, phản ứng khi mất gói

Mất gói được chỉ ra bởi timeout:

- **cwnd** được thiết lập 1 MSS;
- Sau đó kích thước cửa sổ sẽ tăng theo cấp số nhân (như trong slow start) đến ngưỡng, sau đó sẽ tăng tuyến tính

Mất gói được xác định bởi 3 ACK trùng nhau: TCP RENO

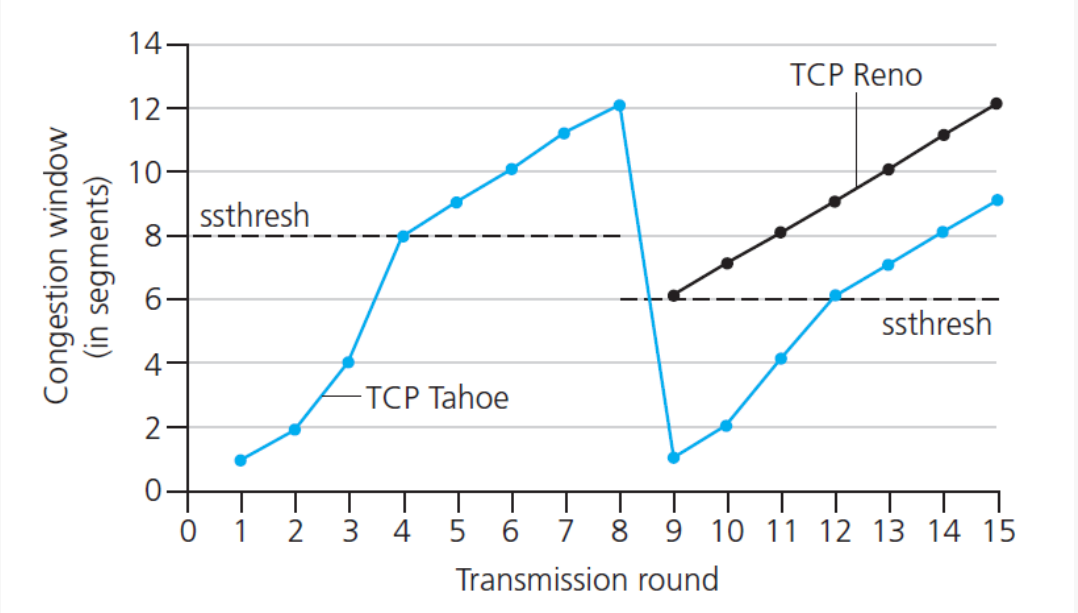
- Các ACK trùng lặp chỉ ra khả năng truyền của mạng
- **cwnd** bị cắt một nữa sau đó tăng theo tuyến tính

TCP Tahoe luôn luôn thiết lập **cwnd** bằng 1(timeout hoặc 3 ack trùng nhau)

TCP: chuyển từ slow start qua CA

Hỏi: khi nào tăng cấp số nhân nên chuyển qua tuyến tính?

Trả lời: khi **cwnd** được 1/2 giá trị của nó trước thời gian timeout.



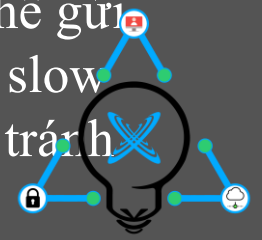
Thực hiện:

ssthresh thay đổi

Khi mất gói, **ssthresh** được thiết lập về chỉ 1/2 của **cwnd** trước khi mất gói

Để quản lí nghẽn(congestion) trong TCP, máy gửi duy trì tham số CWin để chỉ số bytes mà nó có thể gửi trước khi nhận được phản hồi từ máy nhận.Bên cạnh đó, máy gửi còn sử dụng một tham số khác là slow start threshold: SSThershold (đơn vị byte). Khi CWin > SSThreshold thì máy gửi sẽ rất cẩn trọn để tránh gây ra congestion.

Giả định rằng SSThreshold =4000 bytes, CWin =8000 bytes, kích thước của gói tin là 500 bytes.Máy gửi gửi 16 gói tin và nhận được 16 phản hồi. Hỏi giá trị của SSThreshold và CWin sau khi đã nhận được phản hồi là gì ?



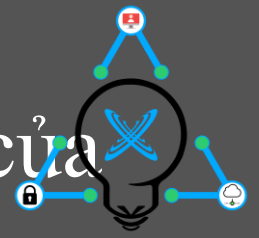
A SSThreshold =4000 bytes, CWin =8000 bytes

B SSThreshold =4000 bytes, CWin =8500 bytes

C SSThresshold=8000 bytes,CWin=8000 bytes

D SSThresshold=8000 bytes,CWin=4000 bytes

Trong TCP RENO, khi gặp 3 ACK trùng nhau, thì giá trị của congestion window được thiết lập lại bao nhiêu?



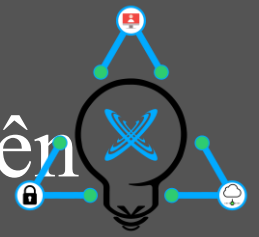
A Bị cắt một nửa

B Vẫn giữ giá trị như trước khi gặp 3 ACK trùng nhau

C 1

D 0

Trong các giao thức giao vận Internet, giao thức nào có liên kết:



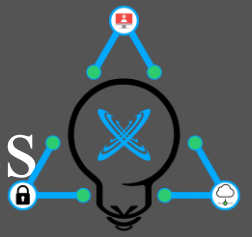
A UDP

B TCP

C TCP và UDP

D Không phải các đáp án trên

Điều gì là đúng đối với các giao thức dạng connectionless (không kết nối)?



A Hoạt động chậm hơn các giao thức dạng connection-oriented

B Các gói dữ liệu có phần header phức tạp hơn so với giao thức dạng connection-oriented

C Cung cấp một dịch vụ phân phát dữ liệu không đáng tin cậy

D Nút gửi phải truyền lại những dữ liệu đã bị mất trên đường truyền.

Gói tin TCP yêu cầu kết nối sẽ có giá trị của các cờ ?

A RST=1, SYN=1

B ACK=1, SYN=1

C ACK=0, SYN=1

D ACK=1, SYN=0

Diễn giải khác biệt chủ yếu giữa TCP và UDP là:

A TCP: truyền tin có bảo đảm. UDP: truyền tin không bảo đảm

B TCP: không có điều khiển luồng. UDP: có điều khiển luồng

C TCP: truyền nhanh. UDP: truyền chậm

D TCP: được sử dụng phổ biến. UDP: ít được sử dụng

Giả sử Host A muốn gửi một file có kích thước 3 triệu bytes đến Host B.Từ Host A đến Host B có 3 đoạn đường truyền nối tiếp nhau, với tốc độ truyền tương ứng là R1=1Mbps,R2=5Mbps,R3=2Mbps.Thời gian truyền file đến host B là:

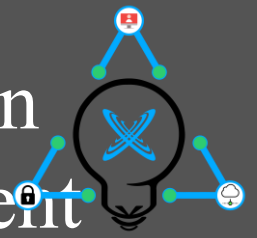
A 24s

B 3s

C 4.8s

D 12s

Bên gửi gửi 1 TCP Segment có Sequence Number =92,và phần Payload (data)=8 bytes.Bên nhận sẽ trả lời với Acknowledgement Number là bao nhiêu để báo nhận thành công TCP Segment này?



A 100

B 93

C 92

D 8

Giả sử Host A muốn gửi một file có kích thước lớn đến Hot B. Từ Host A đến Host B có 3 đoạn truyền được nối tiếp nhau, với tốc độ truyền tương ứng là R1=500 Kbps,R2=2 Mbps, R3=1Mbps.Giả sử không có dữ liệu nào khác được truyền trên mạng. Thông lượng đường truyền (throughput) từ Host A đến Host B là:

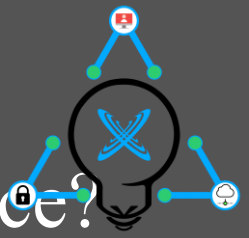
A 500Kbps

B 3500/3 Mbps

C Không có đáp án đúng

D 1 Mbps

Những tính chất nào sau đây không được cung cấp bởi TCP Service?



A Điều khiển luồn (Flow control)

B Đảm bảo hiệu suất tối thiểu (Minimum throught guarantee)

C Truyền tin cậy (Reliable transmission)

D Điều khiển tắc nghẽn (Congestion control)

Đường truyền có băng thông 1 Gbps có nghĩa là:

A 1024 Mbps

B 1000000000 bps

C 1024 x1024 x1024 bps

D 1000000 KBps

Tính hiệu suất(utilization) của giao thức dừng và chờ (stop and wait protocol),Với Giả thuyết: kích thước gói tin là 2KB, đường link (transmission rate) là 100Mbps, độ trễ lan truyền giữa hai đầu cuối (end-end delay) là 15ms. Lưu ý : 1MB làm tròn thành 1000KB và tìm câu trả lời gần với đáp số nhất.

A 0.003

B 0.005

C 0.006

D 0.001

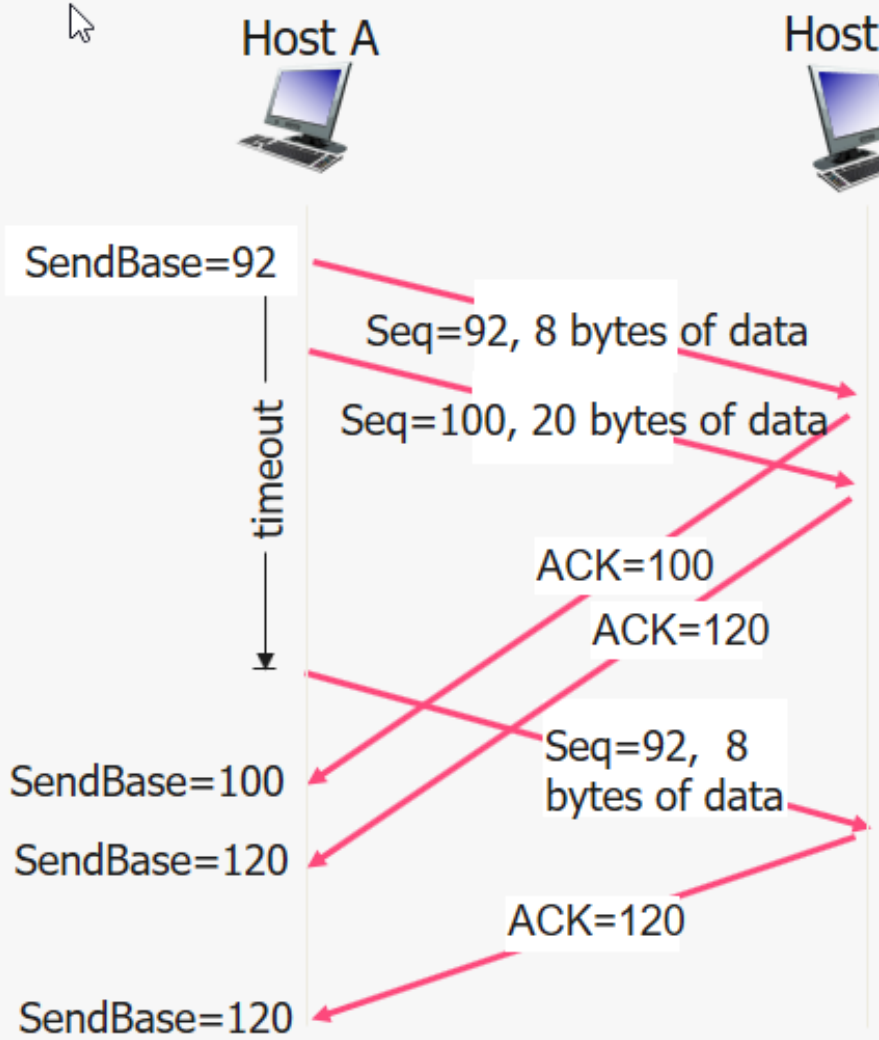
Xem hình vẽ đây là tình huống nào:

A Mất ACK

B Timeout sớm.

C ACK tích lũy

D Truyền lại nhanh



THE END