TCP

데이터의 순차전송

TCP란

Transmission Control Protocol의 약자로
데이터의 전송을 제어하는 프로토콜.
TCP는 데이터의 전송을 신뢰할 수 있는 연결을 지향

데이터 전송

라우팅-데이터를 일정 경로를 통해 목적지로 보내는 것
데이터를 효율적으로 보내고, 한 목적지로 향하는 여러 경로로 보내기 위해 데이터를 패킷이라는 단위로 나누어서 전송,
(TCP는 OSI 4계층이기 때문에 바로 밑의 3계층에서 사용하는 패킷이라는 단어를 사용한다)

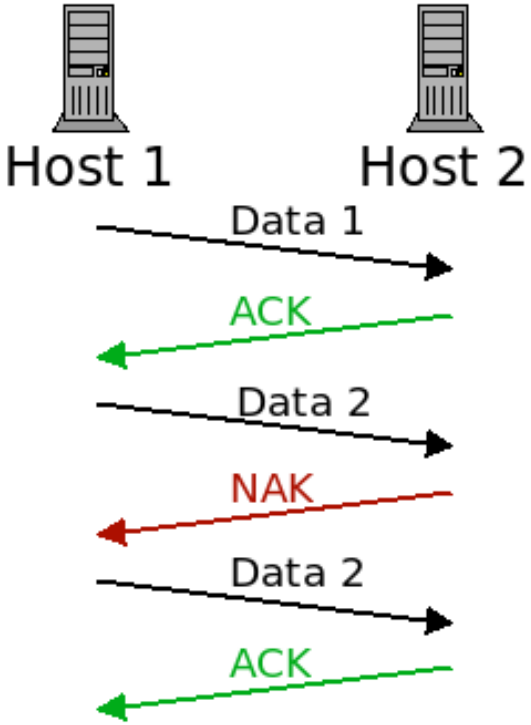
패킷은 받는 입장(수신측)에서 다시 재조립되어야 함.

조립될 때는 원본의 순서도 정확히 알아야 함

-> 보낼 때는 ABC의 순서로 보냈지만 도착하는 순서는 BCA일 수도 있음. 그래서 1-A, 2-B, 3-C 로 보내야, 2-B, 3-C, 1-A 순서로 받더라도 다시 ABC로 만들 수 있음

ACK, NAK

ACK – Acknowledgement 의 약자
데이터 수신측에서 패킷을
올바르게 받았다는 뜻으로,
다음으로 필요한 패킷의 번호
(받은 패킷 번호 + 1)와 함께 보냄



NAK or NACK
– Negative Acknowledgement 의 약자
받은 패킷에 오류가 있다거나 잘못된 순서로 받았다는 뜻으로, 문제가 있는 패킷의 번호와 함께 보냄.

sequential number(#)에 대해서

SEQUENCE NUMBER

TCP 세그먼트안의 데이터의 송신 바이트 흐름의 위치를 가리킴.

다른 호스트로 전달되는 패킷은 여러개의 서로 다른 경로를 거치면서 전달되다 보니 패킷의 순서가 뒤바뀔 수 있음

-> 수신 측에서는 Sequence Number 를 이용해서 재조립

-sequence number는 application data의 양 만큼 증가

-만약 하나의 패킷(segment)의 크기가 8bytes 라면, sequence nubmer는 1, 9, 17, 25, ... 으로 증가

TCP가 NAK을 사용하지 않을 수 있는 이유
(TCP가 ACK을 처리하는 방법)

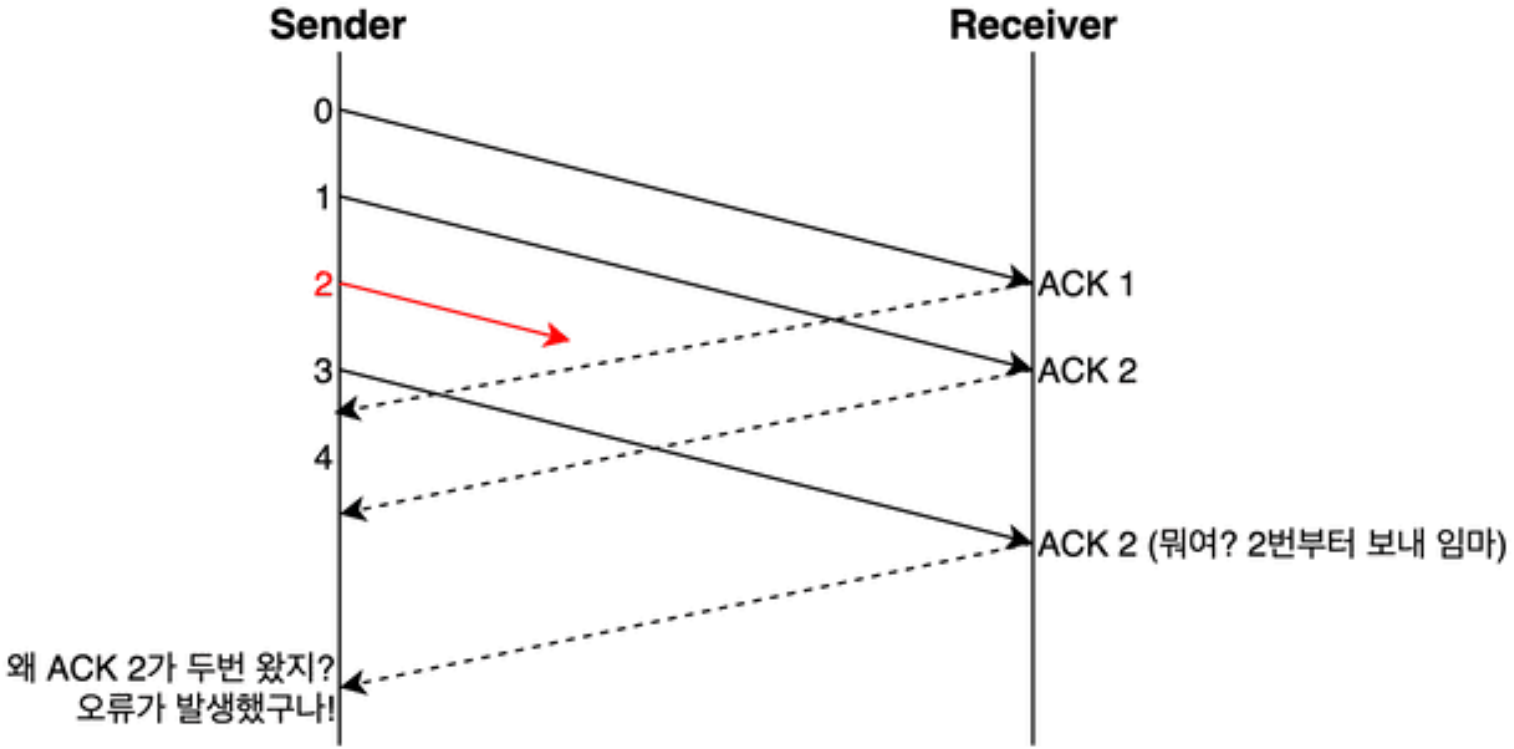
TCP를 사용하는 송수신 측이 오류를 파악하는 방법은 크게 두 가지로 나뉨
-수신 측이 송신 측에게 명시적으로 NACK(부정응답)을 보내는 방법
-송신 측에게 ACK(긍정응답)가 오지 않거나, 중복된 ACK가 계속 해서 오면 오류가 발생했다고 추정하는 방법

NACK를 사용하는 방법이 더 명확하고 간단할 것 같지만, NACK를 사용하게되면 수신 측이 상대방에게 ACK를 보낼 지 NACK를 보낼 지 선택해야하는 로직이 추가적으로 필요하기 때문에, 일반적으로는 ACK만을 사용해서 오류를 추정하는 방법이 주로 사용.

이때 타임아웃은 말 그대로 송신 측이 보낸 데이터가 중간에 유실되어, 수신 측이 아예 데이터를 받지 못해 ACK를 보내지도 않았거나, 수신 측은 제대로 응답했지만 해당 ACK 패킷이 유실되는 경우에 발생.

=> 두 경우 모두 송신 측은 데이터를 전송했는데 수신 측이 응답하지 않고 일정 시간이 경과한 경우

두 번째 방법인 송신 측이 중복된 ACK를 받는 경우 오류라고 판별하는 방법



=> 송신 측은 이미 SEQ 2 데이터를 보낸 상황인데
수신 측이 계속 이번에 2번 보내줄 차례야 라고 말하는 상황인 것.
그럼 송신 측은 자신이 보낸 2번 데이터에 뭔가 문제가 발생했음을 알 수 있다.

TCP에서 패킷이 손실되는 두 가지 유형,
Duplicated ACK과 Packet loss (timeout)현상

세그먼트가 무질서하게 수신될 경우 TCP 는 즉시 duplicate ACK 를 발생시킴

이 때의 ACK 는 가장 최근에 정상적으로 받은 패킷에 대한 ACK 와 동일하기 때문에 sender 의 입장에서 duplicate ACK 로 인식,

보통 TCP sender 는 3 개의 duplicate ACK 가 수신되면 패킷 손실로 간주하고 time out 전에 패킷을 재전송함.

Packe loss의 원인

1. congestion : 네트워크에 트래픽이 너무 많아 더 이상 받을 수 없는 상황, queueing error라고도 함.(큐가 넘쳐나는 경우)

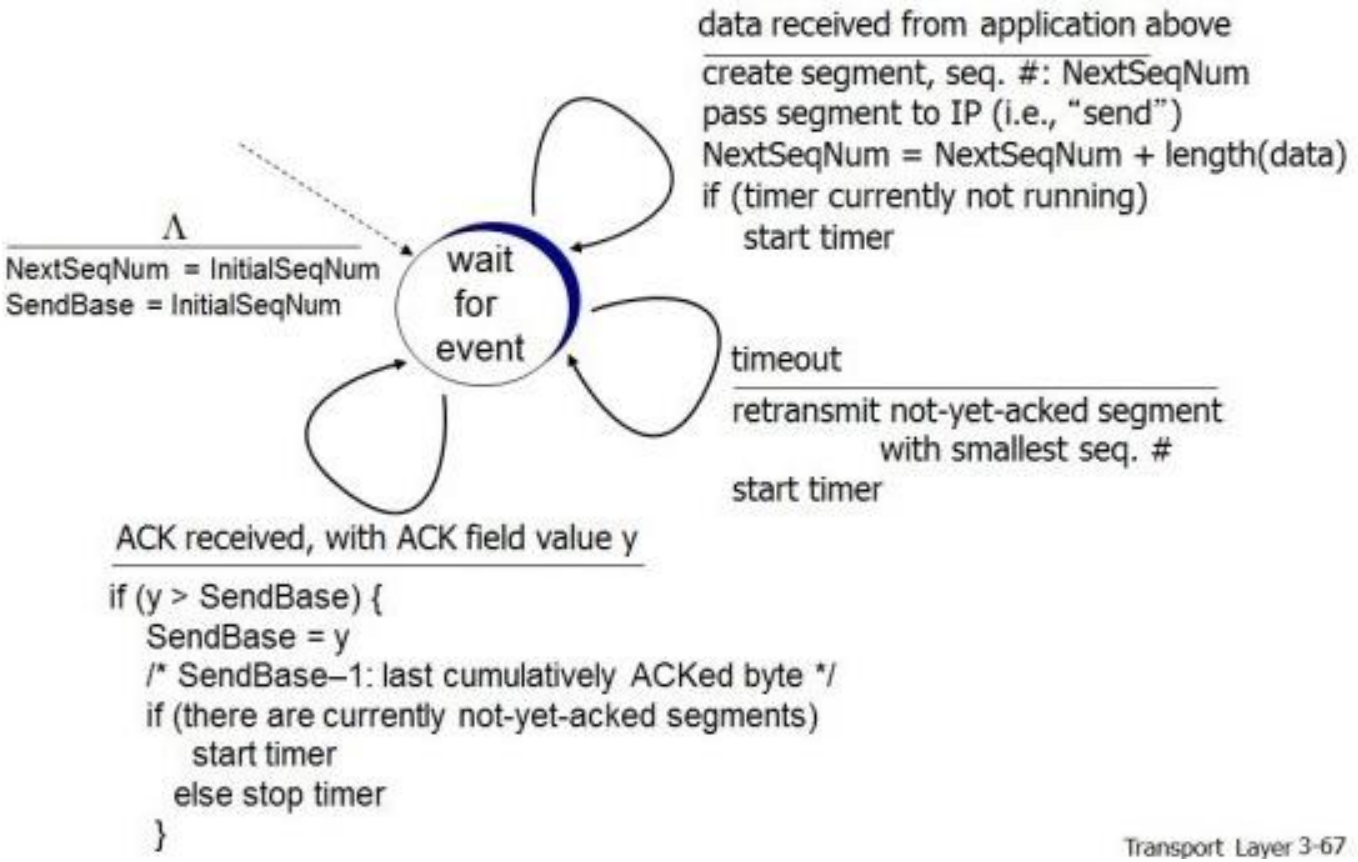
2. 데이터가 noise를 입었을 때 checksum이 틀려서 데이터가 손실되었다고 판단하고 drop시킴. (Transmission error)

3. Routing loop가 발생할 때, 데이터는 TTL값을 가지고 있어서 LOOP돌다가 소실됨.

out of order 현상 : 데이터가 뒤죽 박죽 섞여서 목적지에 도착하는 현상

TCP 재전송 기준
=>RTO(타임 아웃이 발생할 경우)
=>Dup ACK이 세번 발생할때(Fast Retransmition)

TCP sender (simplified)



맨 처음 NextSeqNum 에 처음 데이터가 전송되는 주소를 할당하고 sendBase에도 할당.

sendBase란,

ACK가 어디까지 왔나.. 즉 어디까지 전송이 완료 되었는지를 알 수 있게 함

맨 처음 어플리케이션으로부터 데이터를 받았을 때 action

1. 세그먼트를 생성하고 시퀀스 넘버를 할당하고, 다음 시퀀스 넘버는 현재 + 데이터의 길이(byte)
2. IP 계층 (network계층)으로 세그먼트를 넘김.
3. 만약 타이머가 안 돌아가면 타이머를 설정.

만약 타임아웃이 되었다면,

-가장 작은 seq.#에 대한 ACK를 보낸다. (ACK는 그것을 달라고 요청하는 기능).
다시 타이머 동작.

만약 ACK를 받았다면,

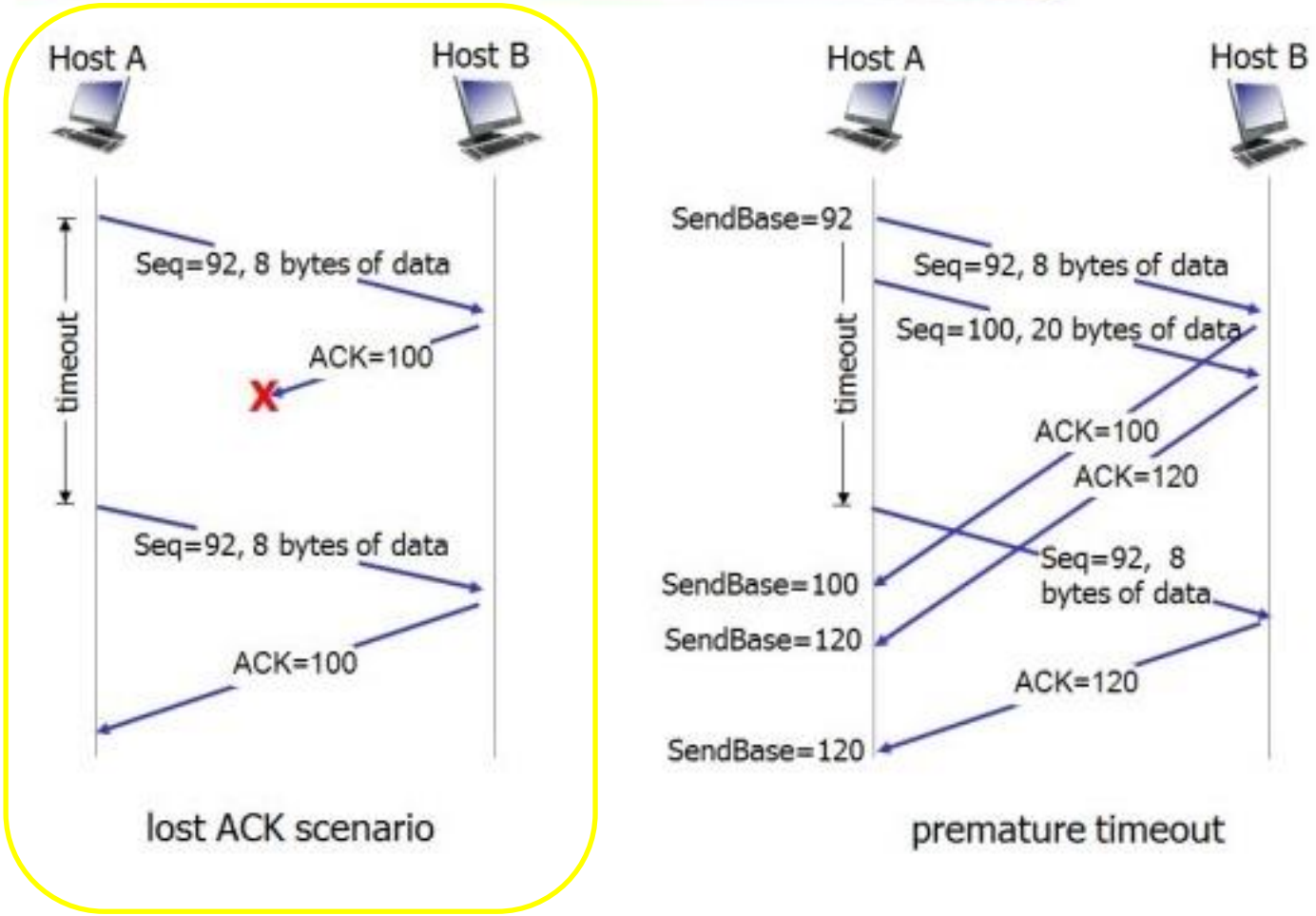
-근데 ACK 값이 sendbase값보다 크면 sendbase값을 그 ACK값으로 갱신.

만약 같거나 작으면 discard, 버리고 아무것도 안함

아직 ack를 못받은 segment가 있으면 타이머를 동작시키고 없다면 타이머를 멈춤

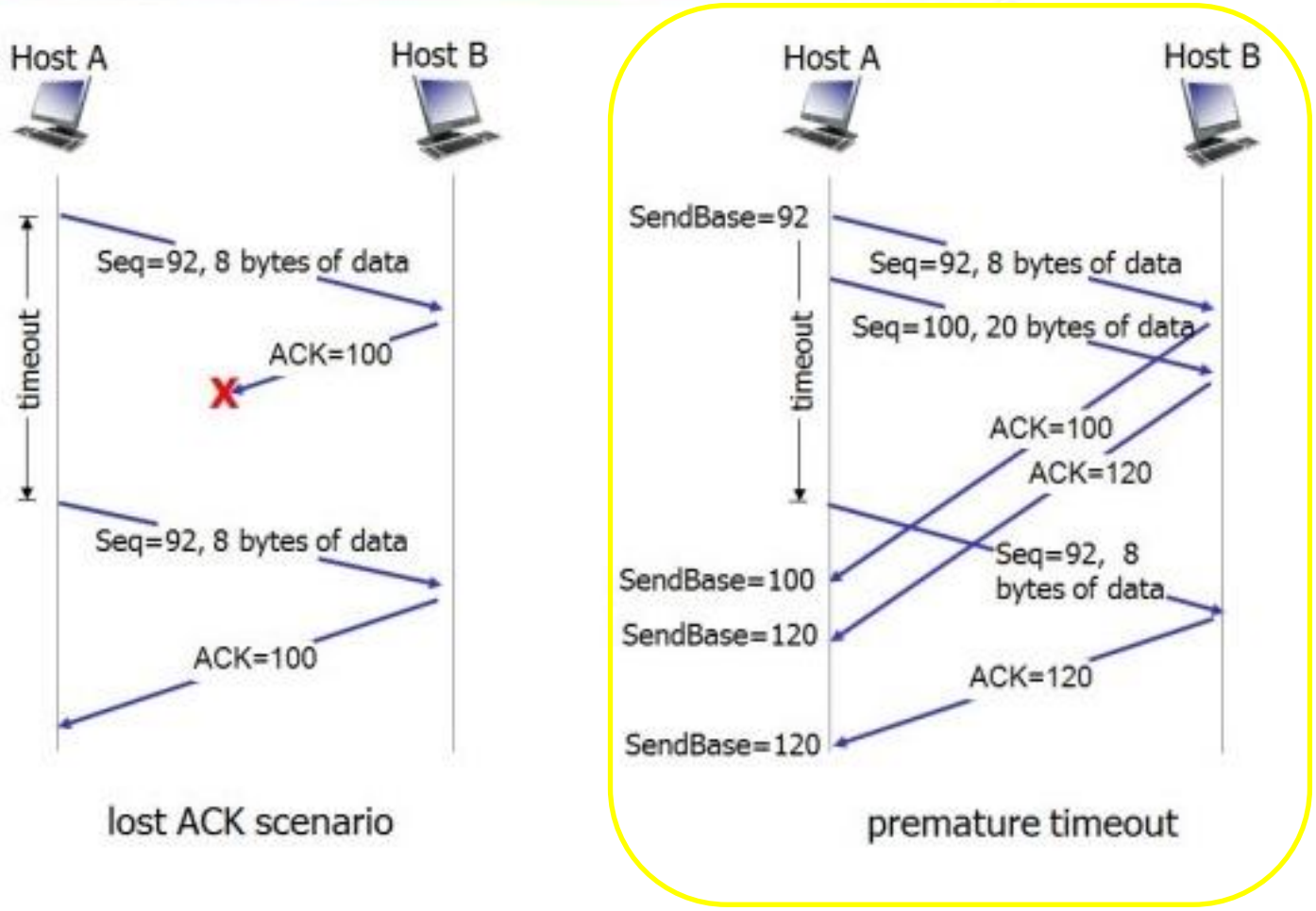
위 과정을 무한루프

TCP: retransmission scenarios



-> ACK 100 이 loss 되서 다시 보내고 ACK를 보냄

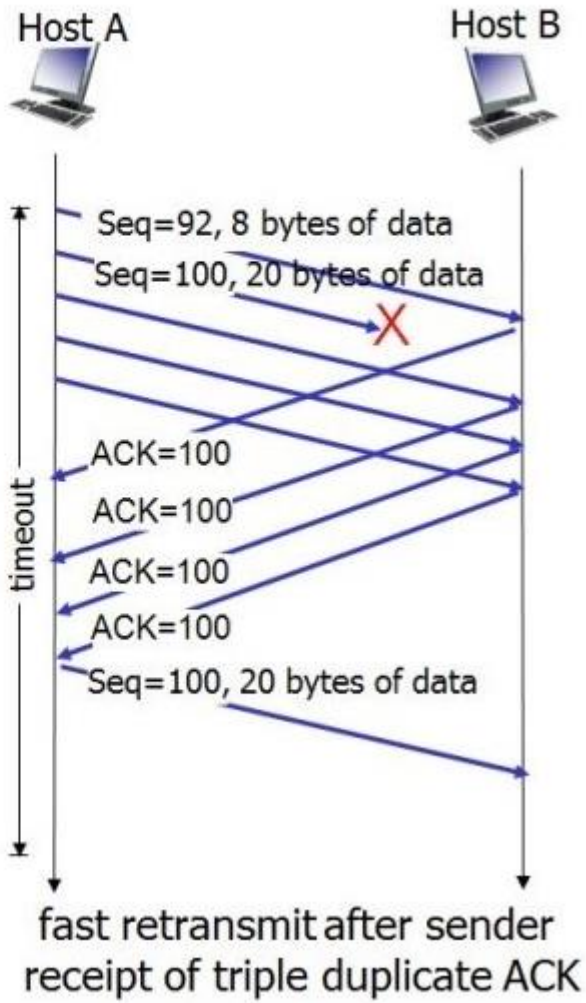
TCP: retransmission scenarios



-> seq 92와 100을 보내고 타임아웃이 되고 seq 92를 보냄,
그 이후 ack 100, 120이 왔다. 그래서 sendBase가 120으로 갱신 됨.

TCP fast retransmit

빠른 재전송(fast retransmit)



timeout 재전송의 문제
-타임아웃주기가 때때로 생각보다 비교적 길다는 것.
->segment를 잃어버렸을 때, 긴 타임아웃 주기는 지연을 증가시킨다

ACK가 100이 계속 오지만 수신자는 타임아웃을 기다림(중복적으로 오는 것이 이미 문제가 생김을 알 수 있음) -> ack가 중복으로 오면 타임아웃 전에 재전송!