2018.2.

컴퓨터 네트워킹

TCP(Transmission Control Protocol) ①

III. 트랜스포트 계층

5. 연결지향형 트랜스포트: TCP

(1) TCP 연결

(2) TCP 세그먼트 구조

(4) 신뢰적 데이터 전달

(3) 왕복시간(RTT) 예측과 타임아웃

JEJU NATIONAL UNIVERSITY

김예찬 (think.computer@jejunu.ac.kr)

• TCP 연결

Goal :

• TCP 연결이 갖는 특징을 UDP와 비교하여 설명할 수 있다.

- TCP는 데이터를 주고 받기에 앞서,
  송/수신 프로세스 간 핸드셰이크(Handshake)를 먼저 하므로 연결 지향형

- 신뢰적 데이터 전달 프로토콜. (in-order, no loss or corrupt)

- TCP 프로토콜은 오직 종단 시스템에서만 동작하며,
  중간의 네트워크 요소(라우터 등)에서는 동작하지 않는다.
  (연결 상태가 두 종단 시스템에만 존재)

- 전이중(Full-duplex) 서비스를 제공한다.
  - 전이중 : 동시에 양방향으로 데이터를 송수신할 수 있는 통신 방식.

- TCP 연결은 항상 점대점(Point-to-Point)이다.
  (단일 송신자 – 단일 수신자)
  # 멀티캐스팅 등은 TCP에서 지원 X.

- 3-way Handshake
  (1) Client가 특별한 TCP 세그먼트를 서버에게 보낸다.
  (2) Server가 이에 대한 확인 응답 목적의 TCP 세그먼트로 답한다.
  (3) Client가 마지막으로 세번째 특별한 세그먼트로 다시 응답한다.
  * 처음 2개의 세그먼트에는 Payload가 분명히 없다.
  * TCP 연결이 설정되면, 두 프로세스는 데이터 통신 가능.

- TCP는 초기 3-way Handshaking 동안
  준비된 버퍼의 하나인 연결의 송신 버퍼로 데이터를 보낸다.
  - RFC에는 TCP는 언제 버퍼된 데이터를 전송해야 하는지를 기술하지 않는다.
  - 송신 버퍼에서 데이터 묶음을 만들어서 네트워크로 보내는 것이 일반적이다.

- Pipelined Transmission 방식 (GBN, SR의 혼합)
  - 흐름 제어(Flow control)와 혼잡 제어(Congestion Control) 지원.
  - MSS(Maximum Segment Size) : 최대 세그먼트 크기
    - 실제로는 세그먼트의 데이터(Payload)의 최대 크기를 의미한다.

[+a]

- TCP는 메시지를 하나의 단일 가상 경로를 이용하여 전송한다.
  (재전송, 확인응답 처리의 용이함)

- 주목해야 할 점 :
  최선형 전달 서비스인 IP 서비스를 이용하여
  TCP가 연결 지향을 어떻게 지원하나?

• TCP 세그먼트 구조

Goal :

• TCP 세그먼트의 형식(Format)과 필드(Field)를 설명할 수 있다.

TCP 세그먼트 구조

PORT# : 다중화, 역다중화 목적

Data Offset으로서 기능한다. (어디서부터 데이터인지 수신자에게 알려주어야)

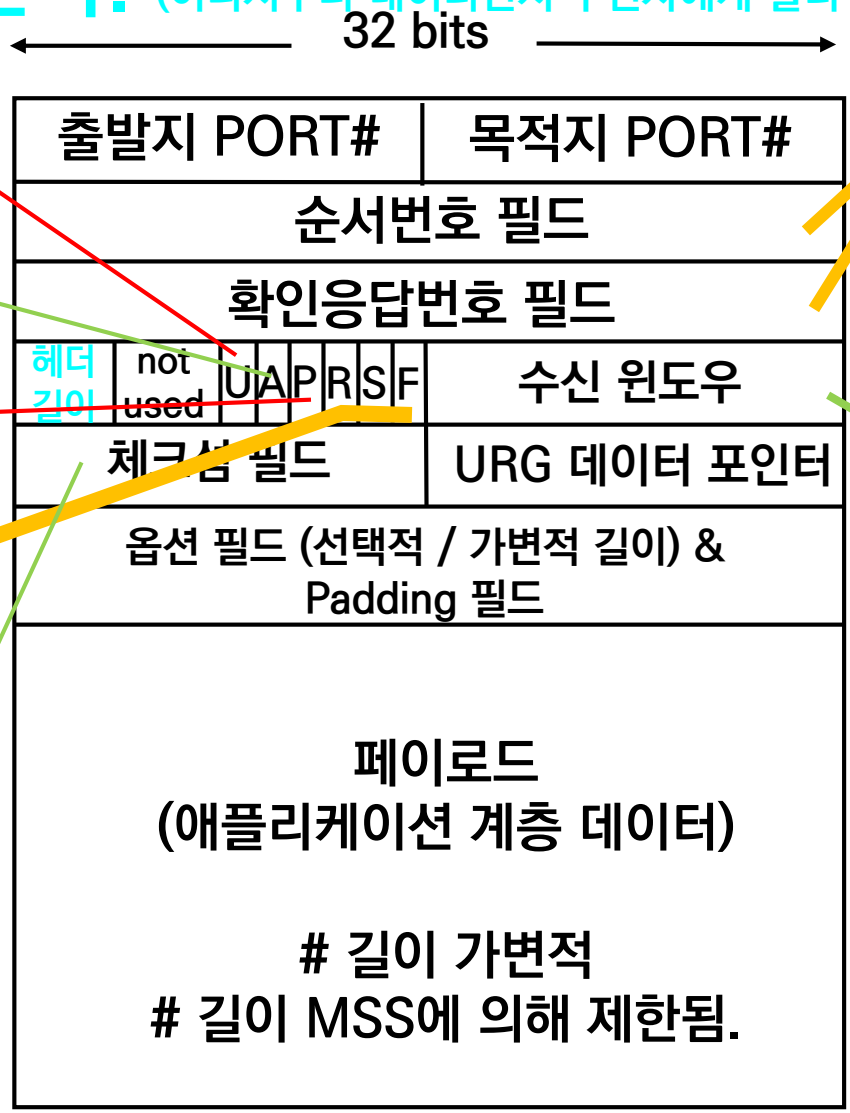
URG(urgent) : 긴급 데이터 여부 필드
(일반적으로 사용되지 않음.)

ACK : 확인응답 패킷 여부 필드
(1 : valid, 0 : invalid)

PSH (push) : 즉시 상위 계층으로 데이터
전달해야 하는지 여부 필드
(일반적으로 사용되지 않음)

RST, SYN, FIN:
연결 설정, 해제를 위해 사용됨.
(핸드셰이킹)

Checksum :
오류 검출 목적



Segment의
시작 바이트 스트림
크기로 기재.
(Segment 단위 X)

흐름 제어를
목적으로
존재.

수신자가
받아들이려는
바이트의 크기.

헤더 필드

데이터 필드

순서번호, 확인응답번호 필드

- 신뢰적 데이터 전달 서비스를 위해 필수적인 필드 2가지.

- 순서번호 필드 :

  세그먼트의 첫 바이트의
  바이트-스트림 번호

- 확인응답번호 필드 :

  TCP는 누적확인응답을 제공한다.

  단, 수신자가 기대하는 다음 바이트의 순서 번호를 확인응답번호를 쓴다.

  예. ACK536 = 0~535까지 제대로 받았다.

  536번째 바이트부터 시작되는 데이터를 송신해줄 것을 기대한다.

순서번호, 확인응답번호 필드

- 확인응답 관련 :

  Q. 순서가 틀린 세그먼트가 도착하면 무조건 폐기(discard)?

  A. 구현을 어떻게 하느냐에 달린 문제.

  - case 1. 무조건 폐기
  - case 2. 수신자는 순서가 틀린 데이터를 버퍼링한다.

    (대역폭 관점에서 이것이 효율적. 실제 많이 취하는 방식.)

순서번호, 확인응답번호 필드

- 순서번호 관련 :

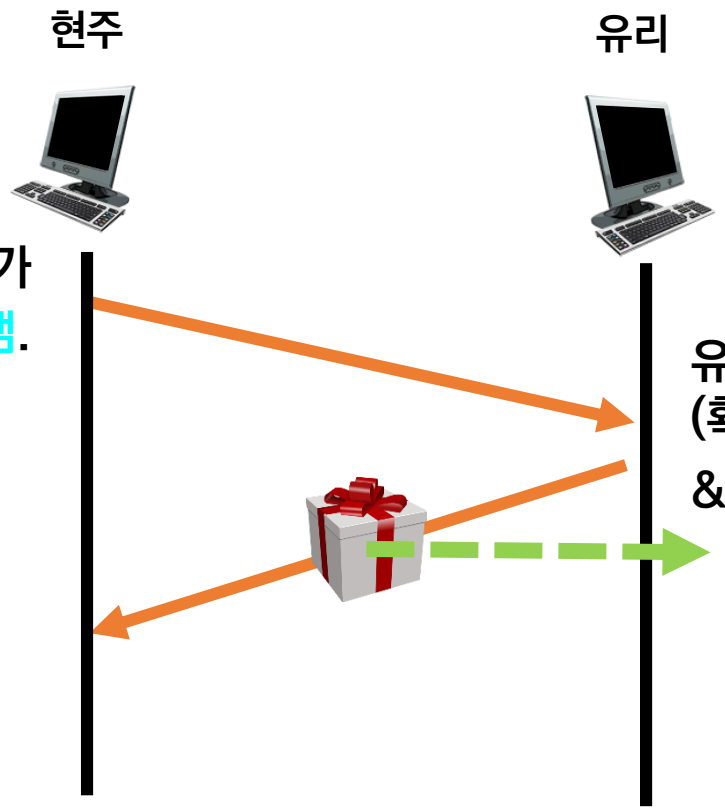
  실제로 TCP 연결의 양쪽 시작 순서번호를 각각 임의로 선택한다.

  (* 항상 0으로 시작하지 않는다!)

  => 이유 : 이것은 두 호스트 사이에 이미 종료된 연결로부터
  아직 네트워크에 남아 있는 세그먼트가
  같은 두 호스트 간의 나중 연결(port#도 동일한 경우)에서
  유효한 세그먼트로 오인될 확률을 줄이기 위해서!

순서번호, 확인응답번호 필드

- GBN, SR은 '전이중(Full-duplex)'를 고려하지 않음.
  - 실제로 한 호스트는 특정 시점에 Sender이자 Receiver 역할을 동시에 수행.
  - (이전처럼) 데이터 패킷과 확인응답(ACK) 패킷을 완전히 서로 다른 것처럼 간주하지 말고, 이 둘을 통합하여 사용하는 것이 효율적이다.



현주가 '제안보고서'라는 데이터를 보냄.

유리가 현주가 보낸 것을 받았음을 확인응답
(확인응답 = 누적 의미 & '패킷 순서번호')
& 현주가 요청한 '신의탑 만화' 데이터를 보냄.

확인응답 패킷이자
데이터 패킷!

★ 데이터를 운반하는
세그먼트 안에서 확인응답이 전달된다.

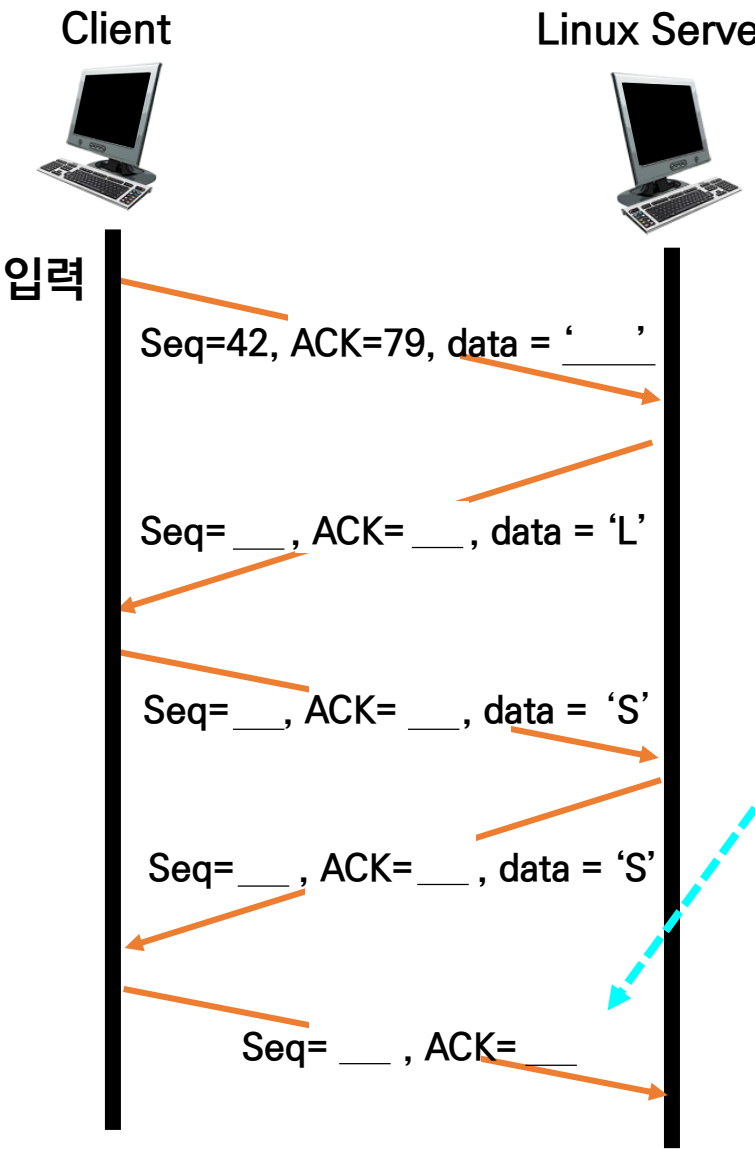
Quiz.

- TCP 연결이 5,000바이트의 파일을 전송한다고 가정하자. 첫 번째 바이트는 10,001의 번호를 가지고 있다. 만일 각각이 1,000바이트를 가지는 5개의 세그먼트에 의해서 데이터가 전달된다면, 각 세그먼트의 순서 번호는 어떻게 되는가?

Segment 1	→	Sequence Number:	10,001	Range:	10,001	to	11,000
Segment 2	→	Sequence Number:	11,001	Range:	11,001	to	12,000
Segment 3	→	Sequence Number:	12,001	Range:	12,001	to	13,000
Segment 4	→	Sequence Number:	13,001	Range:	13,001	to	14,000
Segment 5	→	Sequence Number:	14,001	Range:	14,001	to	15,000

Quiz.

- 아래 빈칸을 채우시오.



빈 데이터 필드를 가진
세그먼트
(확인응답 목적으로만 쓰임.)

단, 1글자는 1byte.

우측은 텔넷 애플리케이션에 관한
타이밍 다이어그램.

서버와 클라이언트는
1byte의 데이터를 주고받는다고 가정.

L
Seq = 79, ACK = 43
Seq = 43, ACK = 80
Seq = 80, ACK = 44
Seq = 44, ACK = 81

- **신뢰적 데이터 전달 : TCP**

Quiz.

O/X로 답하시오.

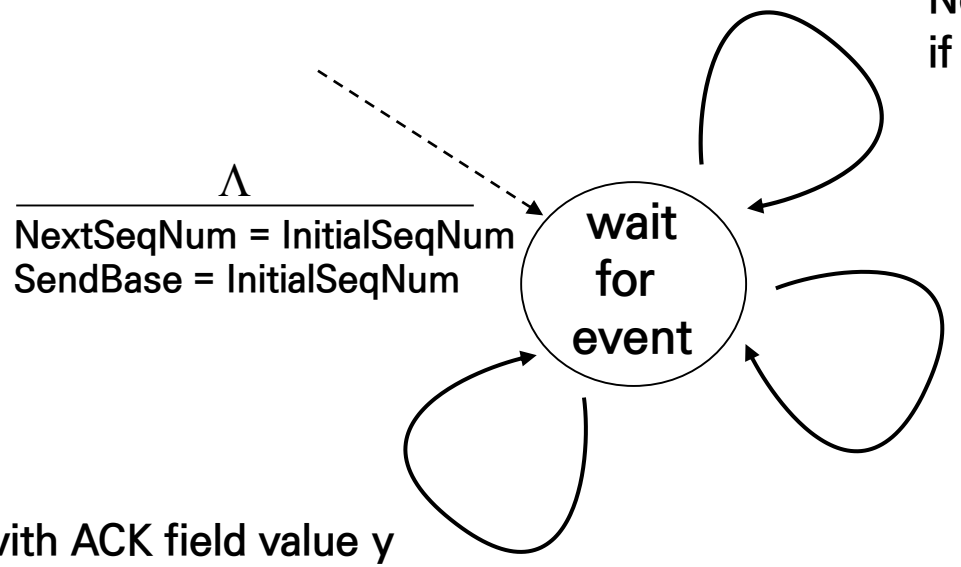
- TCP 수신자는 만약 중복된 패킷을 수신하면, 그것을 폐기한다.
- TCP 수신자는 만약 중복된 패킷을 수신하면, 송신자에게 이에 대한 ACK를 송신한다.
- TCP는 일반적으로 out-of-order segment를 폐기한다.
- 일반적으로 TCP 수신 버퍼는 혼잡 윈도우 크기보다 훨씬 크다.

O O X O

TCP가 제공하는 신뢰적 데이터 전달 서비스

- 파이프라이닝
- 누적 확인응답(ACK)
- 단일 재전송 타이머
- Out-of-order 패킷에 대한 버퍼링 고려.

TCP 송신자 action (간소화)



data received from application above

create segment, seq. #: NextSeqNum
pass segment to IP (i.e., “send”)
NextSeqNum = NextSeqNum + length(data)
if (timer currently not running)
    start timer

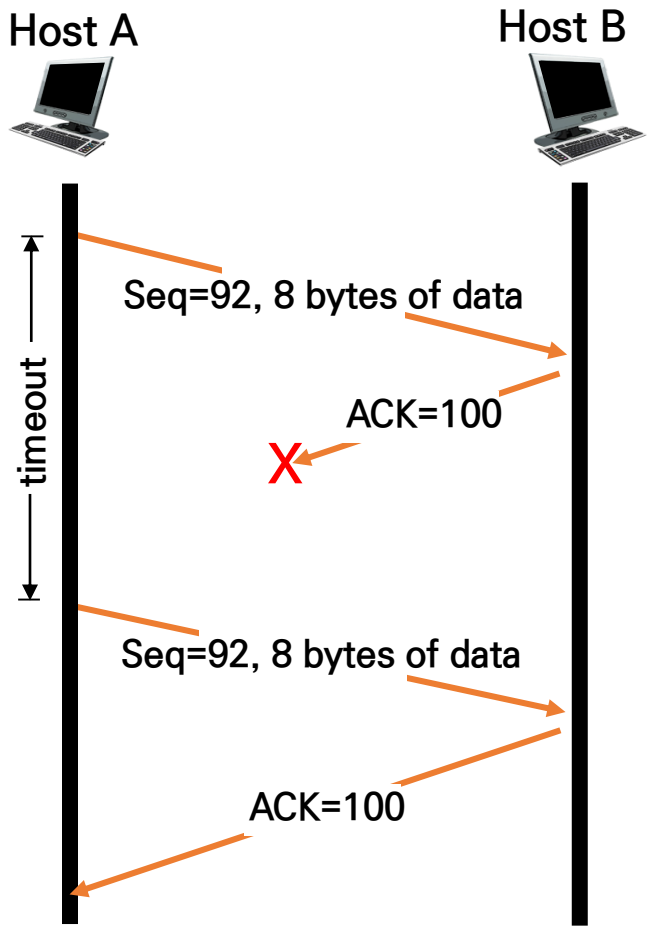
timeout

retransmit not-yet-acked segment
        with smallest seq. #
start timer

ACK received, with ACK field value y

```
if (y > SendBase) {
    SendBase = y
    /* SendBase-1: last cumulatively ACKed byte */
    if (there are currently not-yet-acked segments)
        start timer
    else
        stop timer
}
```

TCP 시나리오 1



Quiz

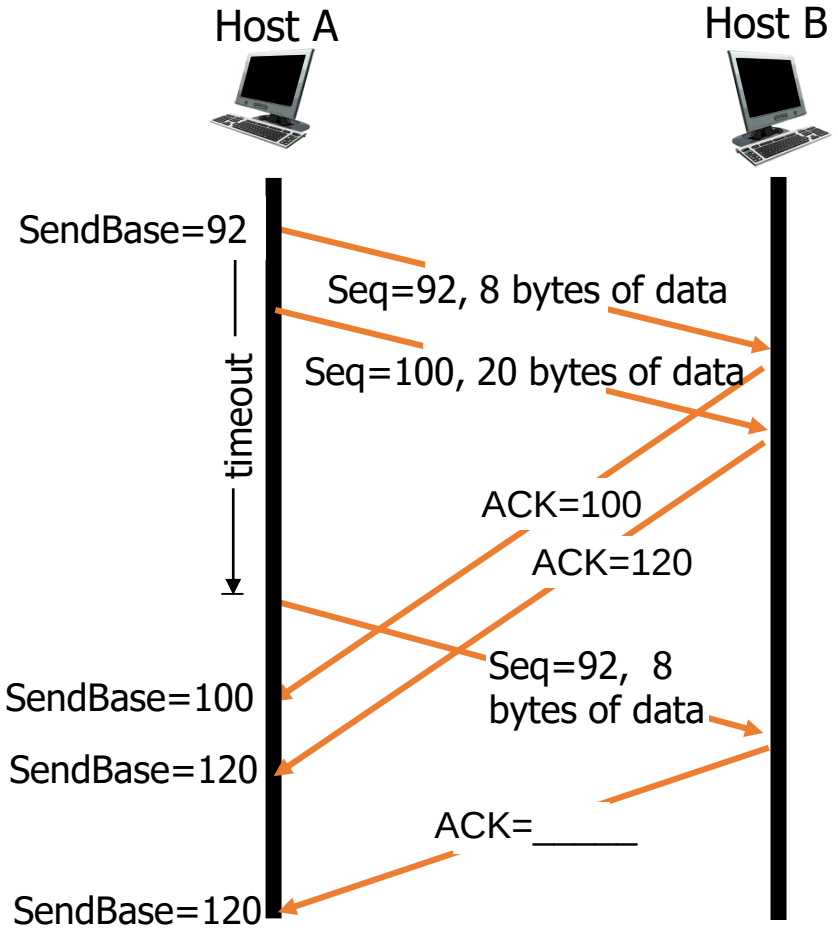
- Host B는 중복 데이터 패킷을 수신한다. (seq=92, 8 bytes) 이때 Host B는 이를 폐기하는가?

- Host A는 자신이 보낸 패킷(seq=92)이 손실되었다고 어떻게 유추하였는가?

Yes

Time-out

TCP 시나리오 2



Quiz

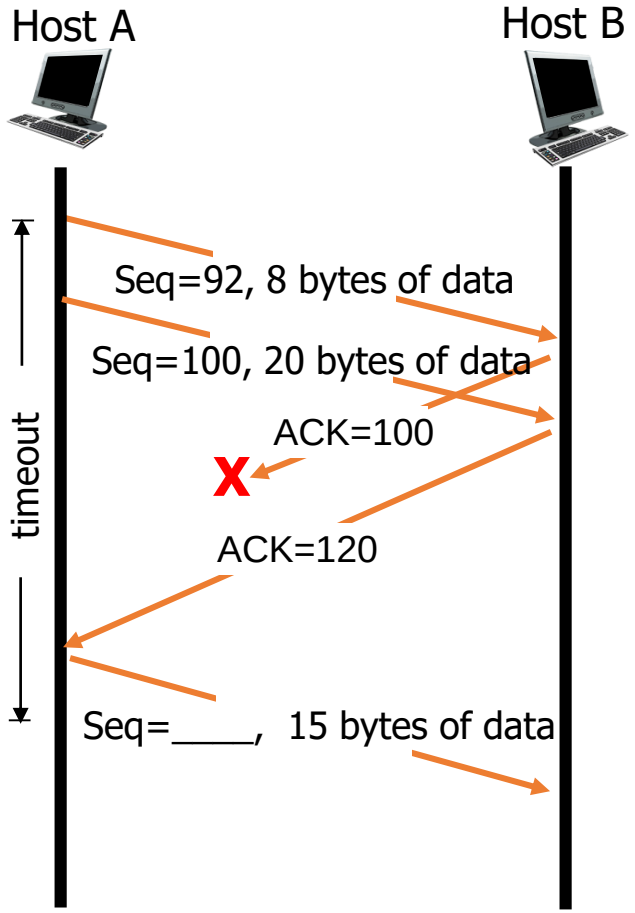
- 왼쪽 빈칸에 들어갈 ACK 값으로 적절한 것을 쓰시오.

- 왼쪽에서 seq 92 패킷에 대한 재전송이 수행되는 이유는 무엇인가?

120

성급한 타임아웃 시간

TCP 시나리오 3



Quiz

- 왼쪽 빈칸에 들어갈 Seq 값으로 적절한 것을 쓰시오.

- 왼쪽에서 ACK=100이 Loss되었다. 송신자는 seq=100 패킷을 재전송해야 하는가? 그 이유를 쓰시오.

120

No. ACK=120을 수신함으로써
누적적으로 [100~119] 패킷도 제대로 수신하였음을
송신자가 알 수 있음.

타임아웃 주기의 2배 설정

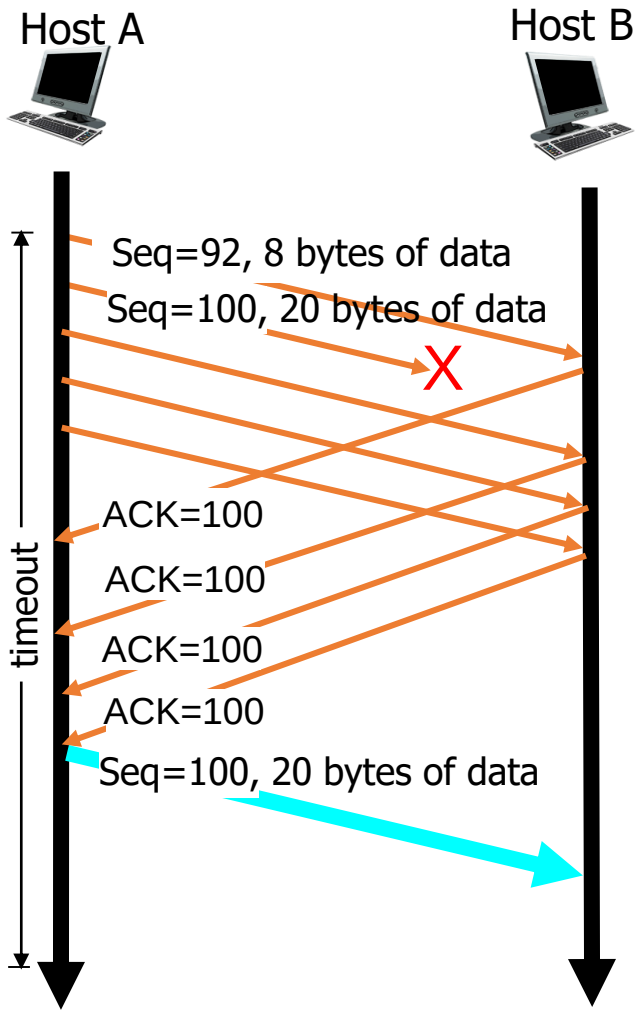
- TCP는 아직 확인응답이 안 된 가장 작은 순서번호를 가진 세그먼트를 재전송한다. (Timeout 시)
  - 재전송 시 Timeout 값 = 직전 Timeout 값 X 2
  - 다만, 상위 계층으로부터의 데이터 수신 사건 + ACK 수신 사건 발생 후 타이머가 시작될 때는 Timeout 값 = EstimatedRTT와 DevRTT의 가장 최근의 값에서 계산함. (*뒤에서 다룰 것.*)

- 생각해볼 점
  - Timeout은 주로 네트워크에서의 혼잡에 의해 발생한다.
  - 이때, 혼잡할 때 재전송을 고집한다면, 혼잡이 오히려 악화될 것이다.
  - 대신에 TCP는 송신자가 더 긴 간격으로 재전송하도록 한다. (일종의 혼잡제어!)

빠른 재전송(fast-retransmit)

- Timeout-triggered Retransmission의 문제점은?
  - Timeout 주기가 때때로 비교적 길다.
  - 긴 Timeout으로 인한 종단간 지연 증가 우려.

- 중복 ACK(Duplicate ACK)에 의한 재전송 정책을 수행하자.
  - 3번 연속 중복된 ACK를 수신하면, 패킷 손실(Loss)로 간주하자.
  - 즉, 재전송(retransmit)하자.



fast retransmit after sender receipt of triple duplicate ACK

[RFC 5681] TCP ACK 생성 권고

수신 측 Event	*이벤트에 따른 Action*
arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
arrival of in-order segment with expected seq #. One other segment has ACK pending	immediately send single cumulative ACK, ACKing both in-order segments
arrival of out-of-order segment higher-than-expect seq. # . Gap detected	immediately send *duplicate ACK*, indicating seq. # of next expected byte
arrival of segment that partially or completely fills gap	immediate send ACK, provided that segment starts at lower end of gap

TCP: GBN이냐? SR이냐?

- 일반적으로 올바르게 수신되었지만 순서가 바뀐 세그먼트를 버퍼링한다.
  - SR
- 기껏해야 세그먼트 n 하나만을 재전송한다.
  - SR
- 세그먼트 n에 대한 타임아웃 전에
  세그먼트 n+1에 대한 긍정 확인응답이 도착하면
  세그먼트를 재전송하지 않는다.
  - GBN
- 누적확인응답
  - GBN

Goal :

- 타임아웃 시간은 무조건 RTT보다 크게 설정하여야 재전송 오버헤드를 막을 수 있다는 것을 안다.
- RTT 시간을 고려한 적절한 타임아웃 시간을 설정하는 원리를 설명할 수 있다.

TCP 타임아웃 값 설정하기

- RTT보다는 커야 한다. (-> 아니면 불필요한 재전송 …)
  - 이는 두 호스트 간 연결이 수립(Establish)되기 전까지는 알 수 없다.
  - RTT 값은 거리에 비례한다. (미국-한국 vs 서귀포-제주)
  - RTT 값은 가변적이다. (네트워크 혼잡)

- 타임아웃 값 설정하기

  예 : 측정된 RTT 값들의 집합 = { 3, 7, 3, 7, 9, 1 }

  $$m = \frac{x_1 + x_2 + \cdots + x_N}{N}$$

  $$V = \frac{(x_1 - m)^2 + (x_2 - m)^2 + \cdots + (x_N - m)^2}{N}$$

  $$\sigma = \sqrt{\frac{(x_1 - m)^2 + (x_2 - m)^2 + \cdots + (x_N - m)^2}{N}}$$

  - 평균 = 5 ; 분산 = 8 ; 표준편차 = 2.8284… = 3 ;
  - 만약 EstimatedRTT(예측 RTT)가 5라고 할 때, 이 값에 DevRTT(표준편차)를 더하면?
    - 5 + 3 = 8
  - EstimatedRTT + 4 X DevRTT = 17 → Safety Margin
    - Timeout 값을 17로 하면, 거의 99.9% 확률로 조급(Premature)하지 않다 말할 수 있다.