



# 전송계층2

📅 강의날짜	@2022/10/03
🕒 작성일시	@2022년 10월 3일 오후 9:19
🕒 편집일시	@2022년 10월 3일 오후 11:05
▼ 분야	네트워크
▼ 공부유형	스터디 그룹
☑ 복습	<input type="checkbox"/>
☰ 태그	

## Chapter 3 outline

- ❑ 3.1 Transport-layer services
- ❑ 3.2 Multiplexing and demultiplexing
- ❑ 3.3 Connectionless transport: UDP
- ❑ 3.4 Principles of reliable data transfer
- ❑ 3.5 Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- ❑ 3.6 Principles of congestion control
- ❑ 3.7 TCP congestion control

# TCP: Overview

RFCs: 793, 1122, 1323, 2018, 2581

## point-to-point:

- one sender, one receiver

## reliable, in-order byte stream:

- no “message boundaries”

## pipelined:

- TCP congestion and flow control set window size

## send & receive buffers



## full duplex data:

- bi-directional data flow in same connection
- MSS: maximum segment size

## connection-oriented:

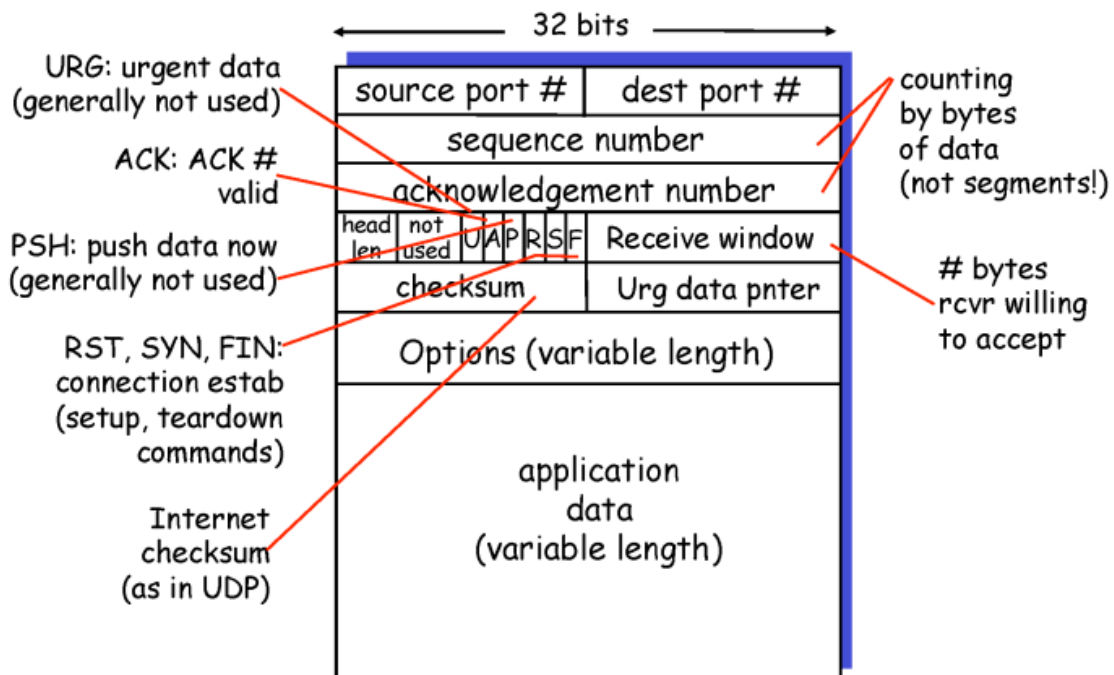
- handshaking (exchange of control msgs) init's sender, receiver state before data exchange

## flow controlled:

- sender will not overwhelm receiver

Transport Layer 3-52

# TCP segment structure



Transport Layer 3-53

- 전송단위 : APP - message / TCP - segment(header, DATA) / IP - packet(header, DATA) / Link - FRMAE(header, DATA)

## TCP seq. #'s and ACKs

### Seq. #'s:

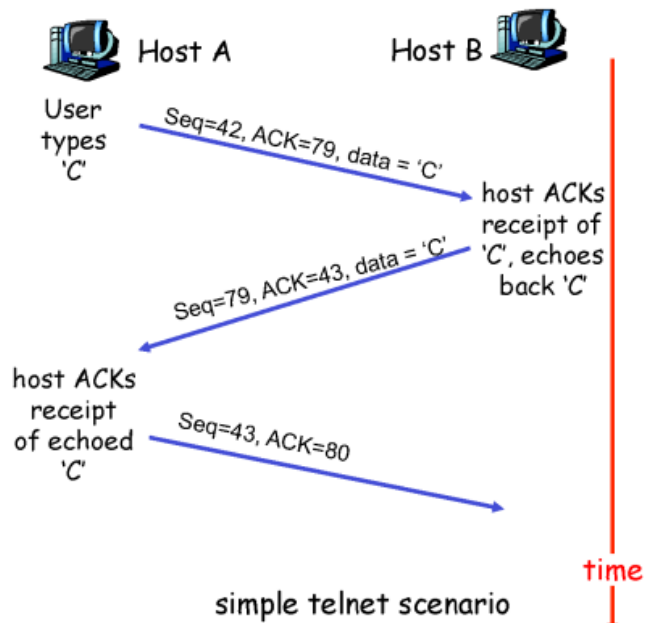
- byte stream "number" of first byte in segment's data

### ACKs:

- seq # of next byte expected from other side
- cumulative ACK

**Q:** how receiver handles out-of-order segments

- A: TCP spec doesn't say, - up to implementor



Transport Layer 3-54

- GBN에서는 ACK10이라면 10번까지 잘 받았다
- TCP에서는 ACK10이라면 9번까지 잘 받았다 10번 내놔라

## Timeout -- function of RTT

**Q:** how to set TCP timeout value?

- ❑ longer than RTT
  - but RTT varies
- ❑ too short: premature timeout
  - unnecessary retransmissions
- ❑ too long: slow reaction to segment loss

**Q:** how to estimate RTT?

- ❑ **SampleRTT**: measured time from segment transmission until ACK receipt
- ❑ **SampleRTT** will vary, want estimated RTT “smoother”
  - average several recent measurements, not just current **SampleRTT**

- RTT : 세그먼트가 갔다가 ACK가 돌아올때까지 시간
- RTT보다 더 걸리면 timeout
  - timeout = RTT

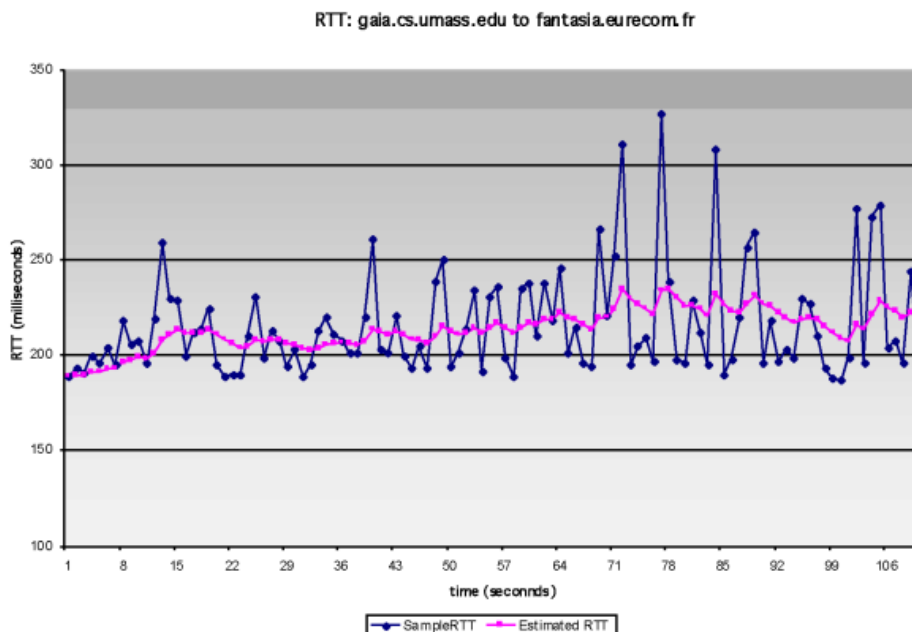
## TCP Round Trip Time and Timeout

$$\text{EstimatedRTT} = (1 - \alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$$

- Exponential weighted moving average
- influence of past sample decreases exponentially fast
- typical value:  $\alpha = 0.125$

- weighted average

### Example RTT estimation:



- 가중평균을 이용한 estimated RTT 값을 사용

## TCP Round Trip Time and Timeout

### Setting the timeout

- **EstimatedRTT** plus “safety margin”
  - large variation in **EstimatedRTT** -> larger safety margin
- first estimate of how much SampleRTT deviates from EstimatedRTT:

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

(typically,  $\beta = 0.25$ )

Then set timeout interval:

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$$

- timeout을 잡되 한번 왔다갔다 하는 시간에 margin을 잡아준다

## Chapter 3 outline

- ❑ 3.1 Transport-layer services
- ❑ 3.2 Multiplexing and demultiplexing
- ❑ 3.3 Connectionless transport: UDP
- ❑ 3.4 Principles of reliable data transfer
- ❑ 3.5 Connection-oriented transport: TCP
  - segment structure
  - **reliable data transfer**
  - flow control
  - connection management
- ❑ 3.6 Principles of congestion control
- ❑ 3.7 TCP congestion control

Transport Layer 3-59

- 파이프라인 방식
- TCP는 타이머 하나 쓴다, 유실된 세그먼트 하나만 재전송
- GBN 또한 타이머 하나, 윈도우 전체 재전

## TCP reliable data transfer

- ❑ TCP creates rdt service on top of IP's unreliable service
- ❑ Pipelined segments
- ❑ Cumulative acks
- ❑ TCP uses single retransmission timer
- ❑ Retransmissions are triggered by:
  - timeout events
  - duplicate acks
- ❑ Initially consider simplified TCP sender:
  - ignore duplicate acks
  - ignore flow control, congestion control

Transport Layer 3-60

## TCP sender events:

### data rcvd from app:

- ❑ Create segment with seq #
- ❑ seq # is byte-stream number of first data byte in segment
- ❑ start timer if not already running (**think of timer as for oldest unacked segment**)
- ❑ expiration interval: TimeoutInterval

### timeout:

- ❑ retransmit segment that caused timeout
- ❑ restart timer

### Ack rcvd:

- ❑ If acknowledges previously unacked segments
  - update what is known to be acked
  - start timer if there are outstanding segments

Transport Layer 3-61



```

NextSeqNum = InitialSeqNum
SendBase = InitialSeqNum

```

```

loop (forever) {
  switch(event)

```

```

    event: data received from application above
           create TCP segment with sequence number NextSeqNum
           if (timer currently not running)
               start timer
           pass segment to IP
           NextSeqNum = NextSeqNum + length(data)

```

```

    event: timer timeout
           retransmit not-yet-acknowledged segment with
             smallest sequence number
           start timer

```

```

    event: ACK received, with ACK field value of y
           if (y > SendBase) {
               SendBase = y
               if (there are currently not-yet-acknowledged segments)
                   start timer
           }

```

```

} /* end of loop forever */

```

## TCP sender (simplified)

Comment:

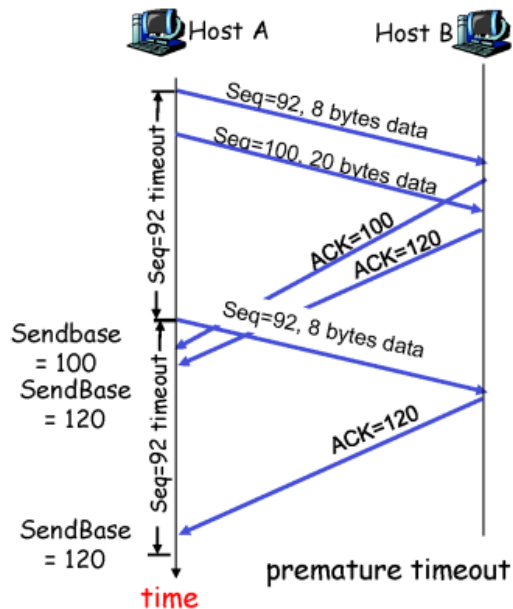
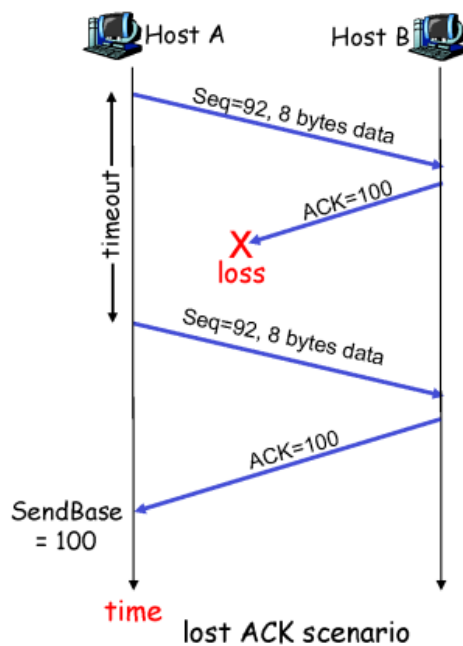
- SendBase-1: last cumulatively ack'd byte

Example:

- SendBase-1 = 71; y = 73, so the rcvr wants 73+ ; y > SendBase, so that new data is acked

Transport Layer 3-62

## TCP: retransmission scenarios

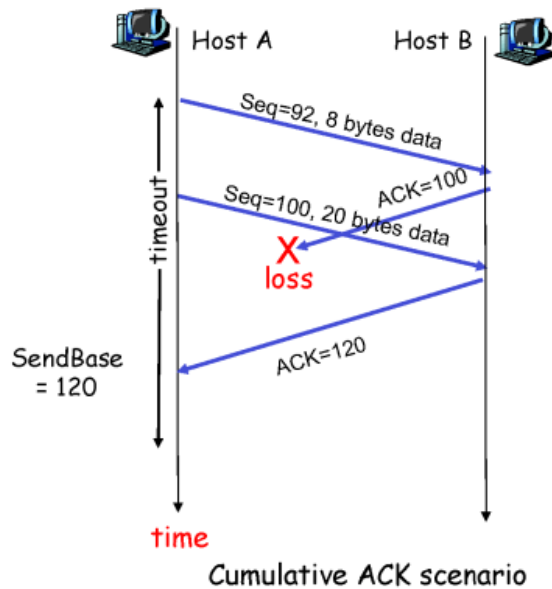


Transport Layer 3-63

- 타이머가 터지기전에 패킷유실 판단 가능

- fast retransmit : 같은 ACK10을 세 번 받으면 재전송하라고 권고
  - ACK10을 duplicate를 3개 받았다는 소리는 총 ACK10가 4개라는 소리

## TCP retransmission scenarios (more)



Transport Layer 3-64