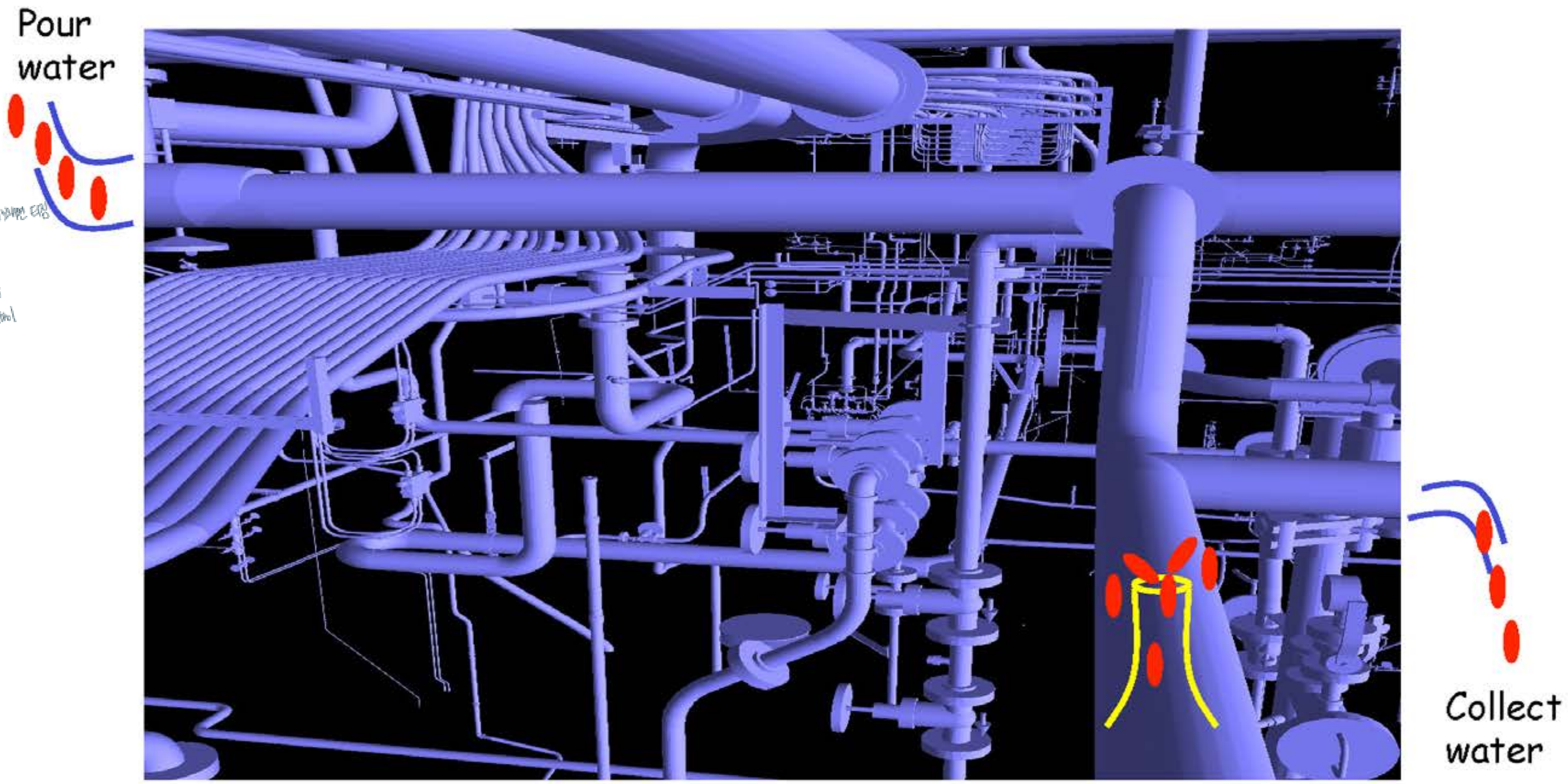
Chapter 3 outline

- 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

The TCP Intuition

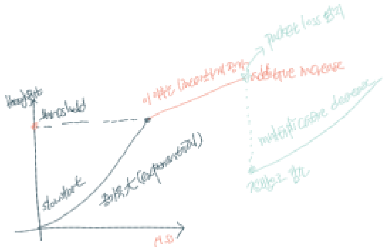
너무 많이 보내면 안됨!
~ 가장 얇은 파이프보다 많이 보내면 터짐
[illegible]
∴ 조금씩 늘려가면서 보내봄
⇒ TCP의 congestion control



TCP Congestion Control

- 3 main phases

MSS (Maximum Segment Size) 약자
= segment의 최대 사이즈



1. Slow Start 시작할 땐 조금씩 보내봄 → 빠르게 증가

   처음) window size = 1MSS 만큼 전송가능
   2MSS
   4MSS
   증가

2. Additive increase threshold 이후로는 천천히 증가

3. Multiplicative decrease packet loss시 반으로 줄임

TCP Congestion Control

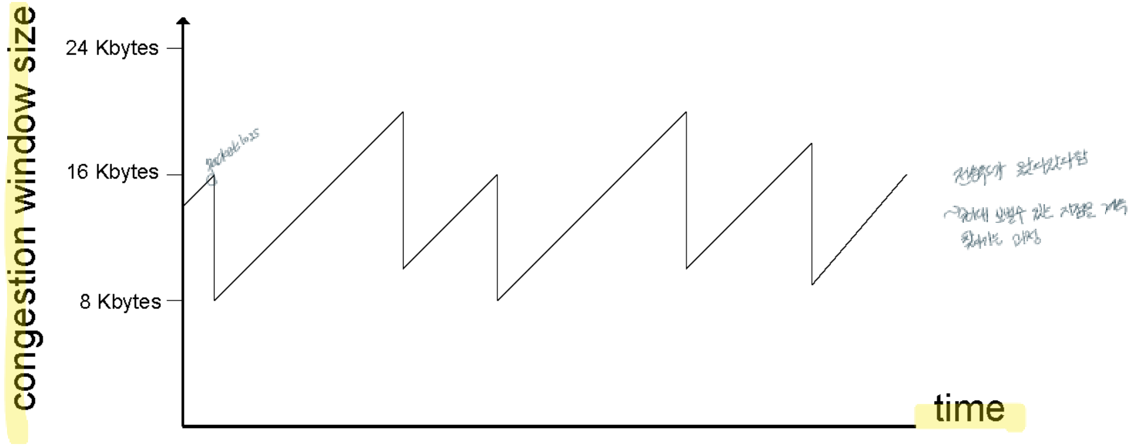
- 3 main phases

1. Slow Start: Do not know bottleneck bandwidth
   So start from zero and quickly ramp up
2. Additive increase: Hey, we are getting close to capacity
   Let's be conservative and increase slow
3. Multiplicative decrease: Oops! Packet drop
   Start over from slow start (from scratch)
   Hmm! many ACKs coming, start midway

TCP congestion control: additive increase, multiplicative decrease

- *Approach:* increase transmission rate (window size), probing for usable bandwidth, until loss occurs
  - *additive increase:* increase **CongWin** by 1 MSS every RTT until loss detected
  - *multiplicative decrease*: cut **CongWin** in half after loss

Saw tooth behavior: probing for bandwidth



TCP Congestion Control: details

- sender limits transmission:

  `LastByteSent-LastByteAcked`
  `≤ CongWin`

- Roughly,

  전송속도

  $$\boxed{\text{rate} = \frac{\text{CongWin}}{\text{RTT}} \ \text{Bytes/sec}}$$

- **`CongWin`** is dynamic, function of perceived network congestion

How does sender perceive congestion?

- loss event = timeout *or* 3 duplicate acks
- TCP sender reduces rate (**`CongWin`**) after loss event

three mechanisms:

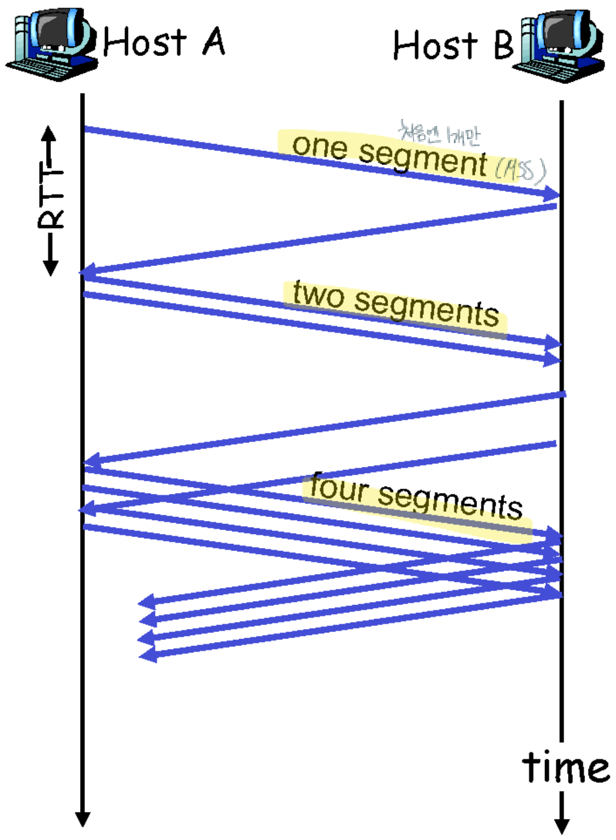
- AIMD
- slow start
- conservative after timeout events

TCP Slow Start

- When connection begins, `CongWin` = 1 MSS
  - Example: MSS = 500 bytes & RTT = 200 msec
  - initial rate = 20 kbps
- available bandwidth may be >> MSS/RTT
  - desirable to quickly ramp up to respectable rate
- When connection begins, increase rate exponentially fast until first loss event

TCP Slow Start (more)

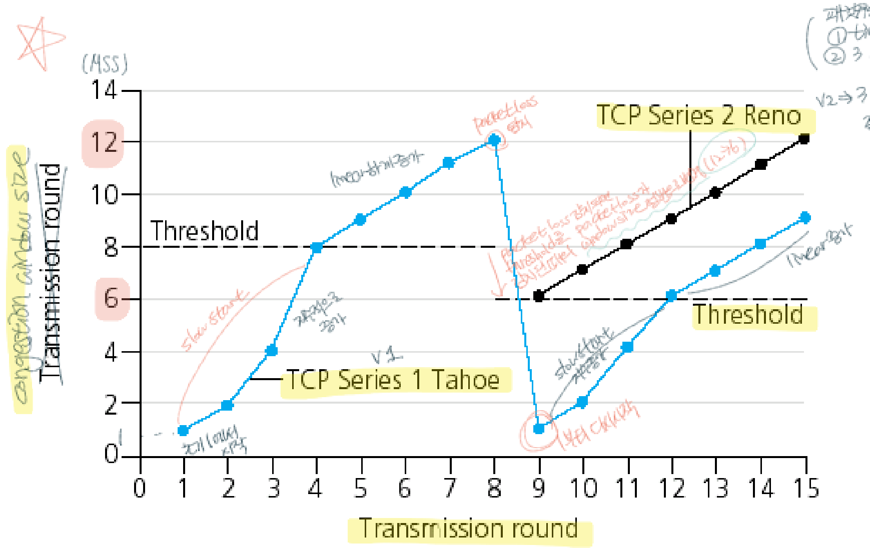
- When connection begins, increase rate exponentially until first loss event:
  - double `CongWin` every RTT
  - done by incrementing `CongWin` for every ACK received
- Summary: initial rate is slow but ramps up exponentially fast



Refinement

Q: When should the exponential increase switch to linear?

A: When **CongWin** gets to 1/2 of its value before timeout.



패킷손실 탐지조건
① timeout → 여러개가 문제 V 심각함
② 3 dup ACK를 받았을때 → 하나만 문제

V2 ⇒ 3 dup ACK인 경우는
절반으로만 window size 조절.
timeout이면 V1처럼 1부터 시작

V1 → 2가지 상황을 똑같이 대응
V2 → 〃 다르게 대응

Implementation:

- Variable Threshold
- At loss event, Threshold is set to 1/2 of CongWin just before loss event

Summary: TCP Congestion Control

- When **CongWin** is below **Threshold**, sender in slow-start phase, window grows exponentially.
- When **CongWin** is above **Threshold**, sender is in congestion-avoidance phase, window grows linearly.
- When a triple duplicate ACK occurs, **Threshold** set to **CongWin/2** and **CongWin** set to **Threshold**.
- When timeout occurs, **Threshold** set to **CongWin/2** and **CongWin** is set to 1 MSS.

TCP Fairness

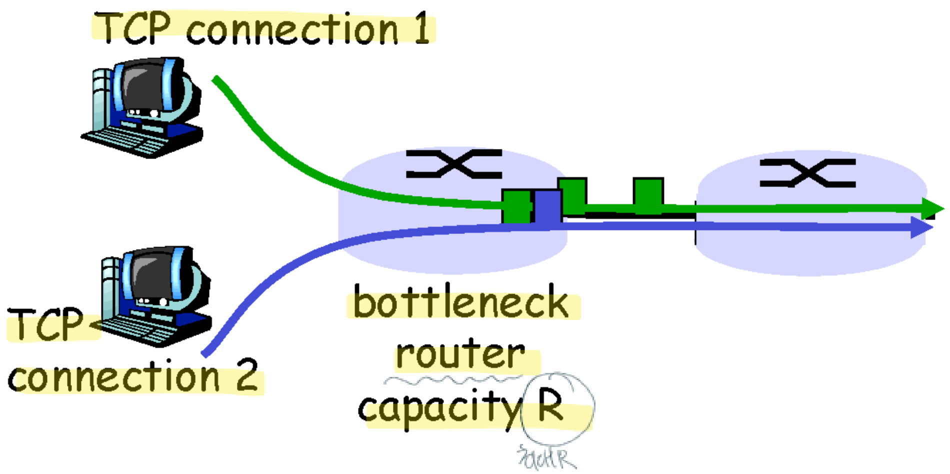
Fairness goal: if K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/K

각각
TCP congestion control.

→ 과연 R/2만큼씩 가질까!

→ TCP가 fair할까?

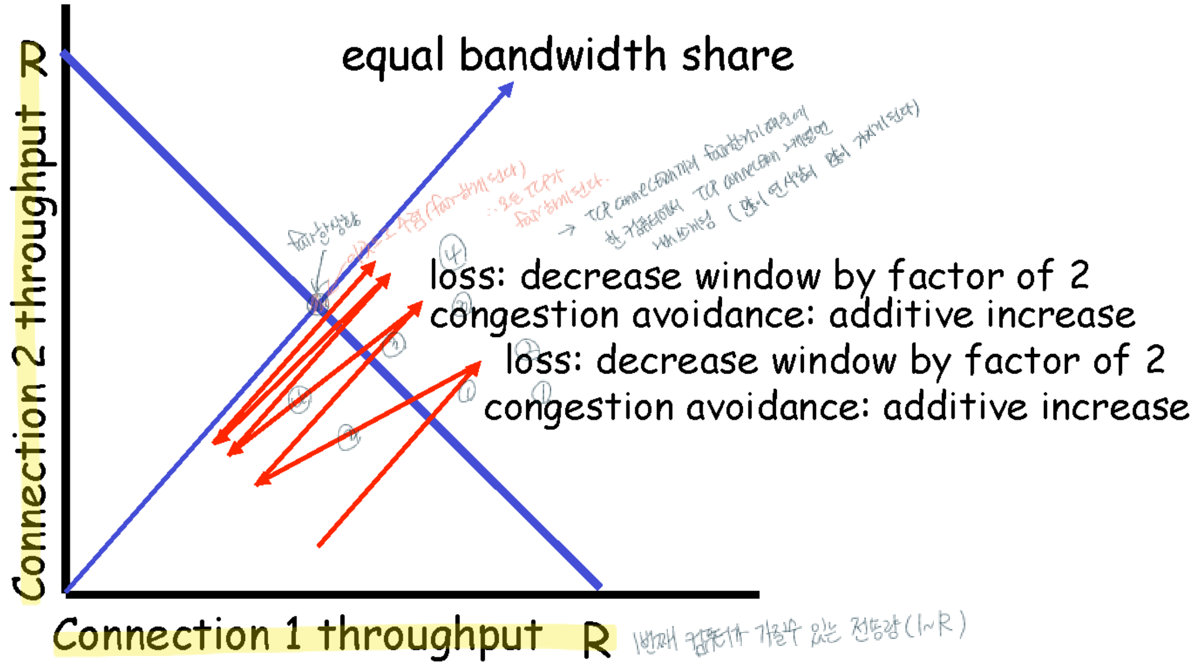
→ fair하다!



Why is TCP fair?

Two competing sessions:

- Additive increase gives slope of 1, as throughout increases
- multiplicative decrease decreases throughput proportionally



Chapter 3: Summary

- principles behind transport layer services:
  - multiplexing, demultiplexing
  - reliable data transfer
  - flow control
  - congestion control
- instantiation and implementation in the Internet
  - UDP
  - TCP

Next:

- leaving the network "edge" (application, transport layers)
- into the network "core"