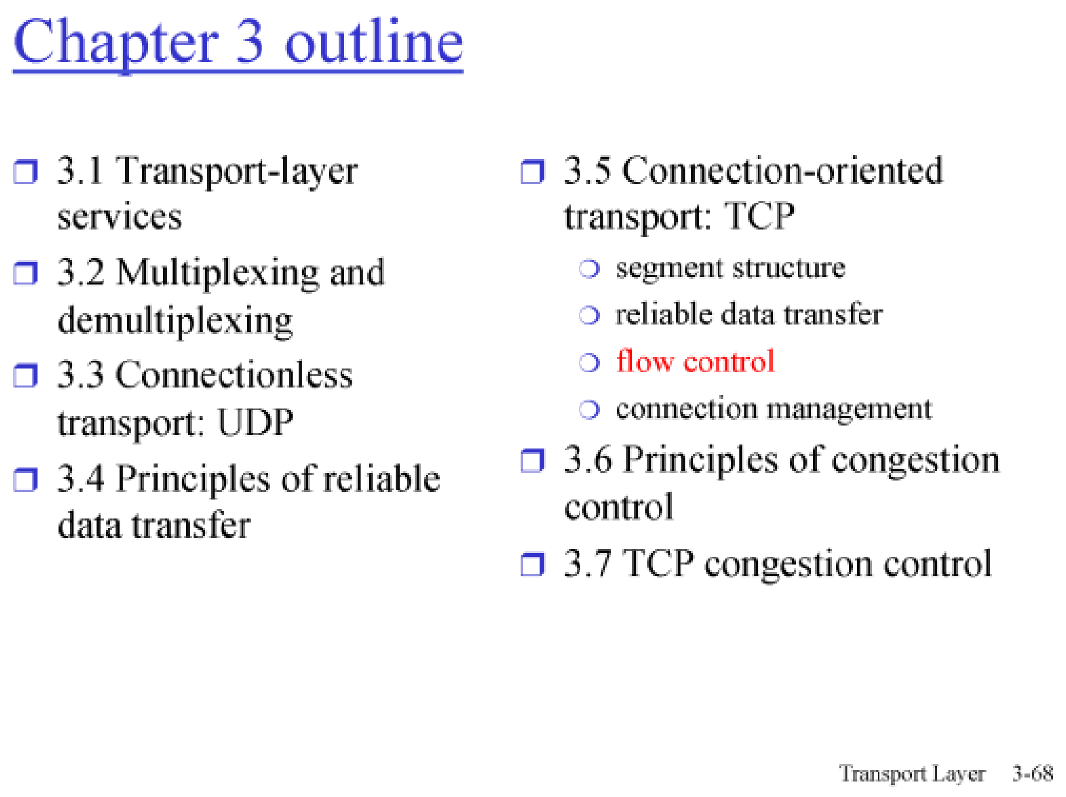
전송계층3

강의날짜	@2022/10/10
작성일시	@2022년 10월 10일 오후 9:19
편집일시	@2022년 10월 10일 오후 11:54
분야	네트워크
공부유형	스터디 그룹
복습	□
태그	



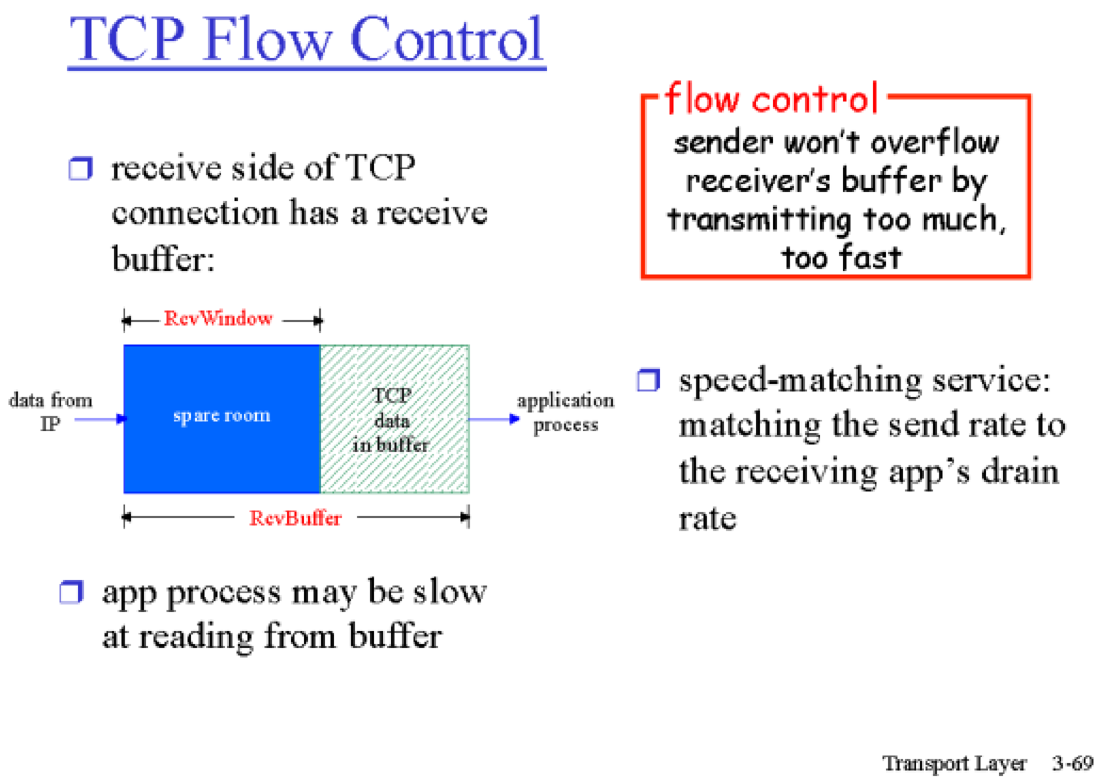
Chapter 3 outline

- 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

Transport Layer 3-68

- flow control : tcp에서 가장 중요한 기능 3가지

- reliable data transfer / flow control / connection management
- flow control 동작 단순 / 직관적
- 리시버의 능력에 맞춰
- A : send buf, recv buf
- B : send buf, recv buf
- flow control : recv buf가 받을 수 있는 능력만큼 send buf가 보내야 의미가 있는것
- recv buf에 얼마를 받을 수 있는지에 대한 정보를 헤더에 담아 지속적으로 전송
- 보내는 속도가 빠르다 : 단위시간 당 보내는 양이 많다
- 양 쪽에 버퍼 두 개 나의 Seq#도 알아야되고, 상대방의 Seq#도 알아야함



TCP Flow Control

❒ receive side of TCP connection has a receive buffer:

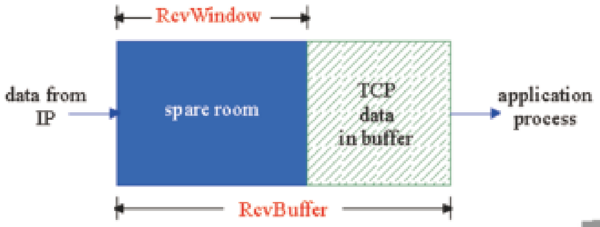
❒ app process may be slow at reading from buffer

flow control
sender won't overflow receiver's buffer by transmitting too much, too fast

❒ speed-matching service: matching the send rate to the receiving app's drain rate

Transport Layer 3-69

TCP Flow control: how it works



(Suppose TCP receiver discards out-of-order segments)

- ❒ spare room in buffer

= `RcvWindow`

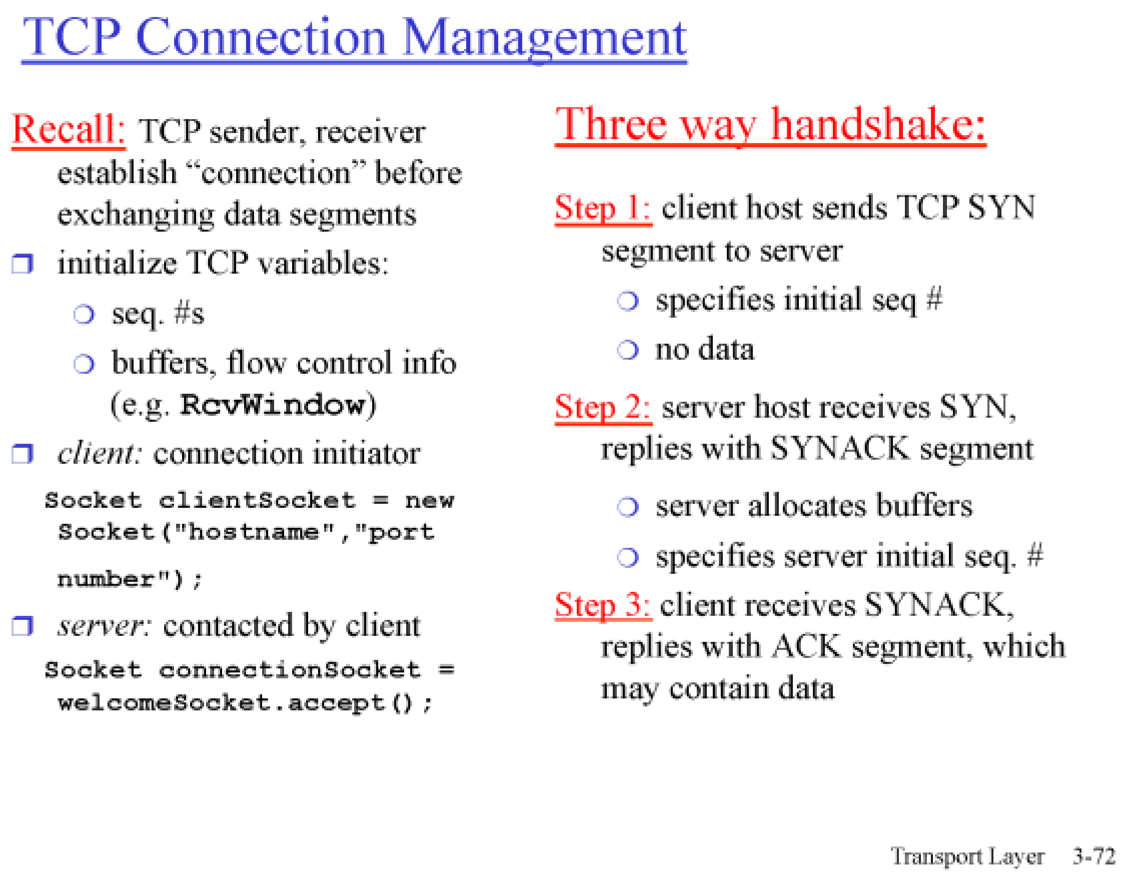
= `RcvBuffer-[LastByteRcvd - LastByteRead]`

- ❒ Rcvr advertises spare room by including value of **`RcvWindow`** in segments

- ❒ Sender limits unACKed data to **`RcvWindow`**
  - ❍ guarantees receive buffer doesn’t overflow

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 Connection Management

Recall: TCP sender, receiver establish "connection" before exchanging data segments

- initialize TCP variables:
  - seq. #s
  - buffers, flow control info (e.g. `RcvWindow`)
- *client:* connection initiator

```
Socket clientSocket = new
 Socket("hostname","port
 number");
```

- *server:* contacted by client

```
Socket connectionSocket =
 welcomeSocket.accept();
```

Three way handshake:

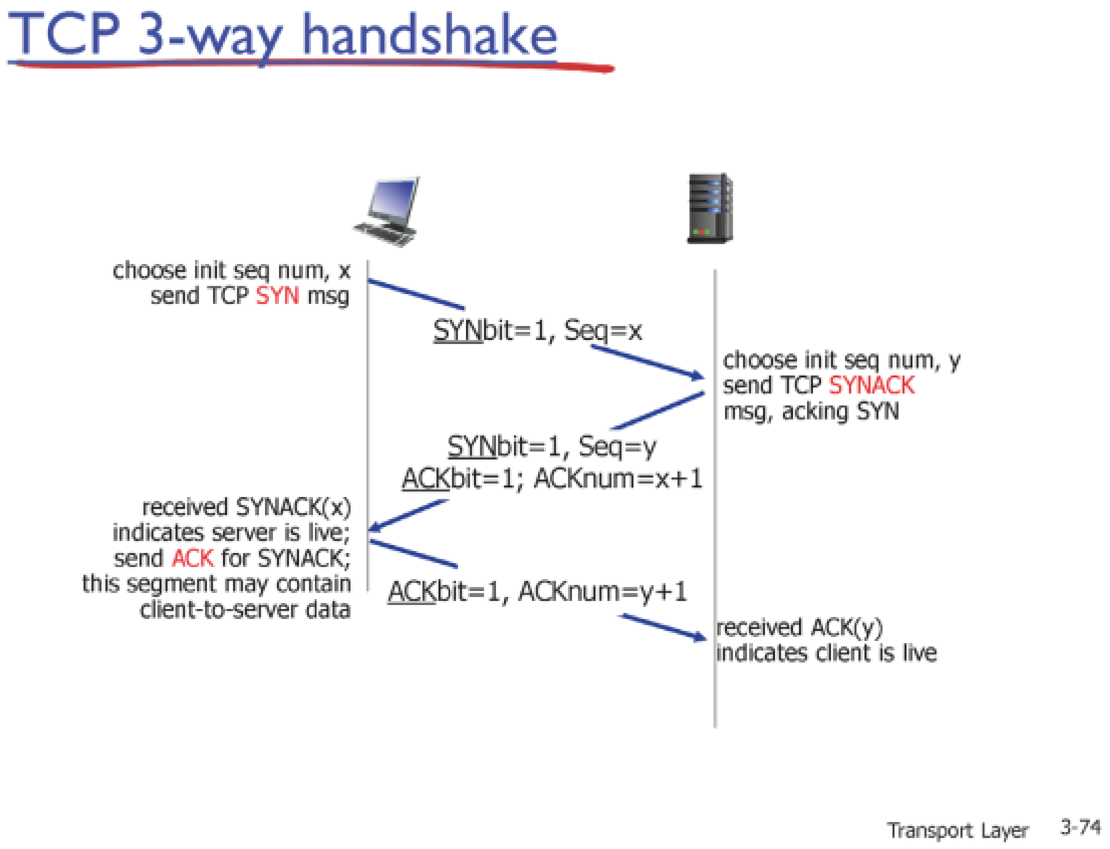
Step 1: client host sends TCP SYN segment to server

- specifies initial seq #
- no data

Step 2: server host receives SYN, replies with SYNACK segment

- server allocates buffers
- specifies server initial seq. #

Step 3: client receives SYNACK, replies with ACK segment, which may contain data



TCP 3-way handshake

choose init seq num, x
send TCP SYN msg

SYNbit=1, Seq=x

choose init seq num, y
send TCP SYNACK
msg, acking SYN

SYNbit=1, Seq=y
ACKbit=1; ACKnum=x+1

received SYNACK(x)
indicates server is live;
send ACK for SYNACK;
this segment may contain
client-to-server data

ACKbit=1, ACKnum=y+1

received ACK(y)
indicates client is live

Transport Layer 3-74

- 항상 세 번 왔다갔다 하는게 정석 핸드쉐이크 교신할 때 많이 사용

1. C→S : TCP SYN
2. S→C : TCP SYNACK
3. C→S : HTTP req
4. S→C : HTTP response

- Sender는 Min(net, recv, sender가 보내는 양)
- network, recv 중에 누가 더 상태가 나쁜가를 계쏙해서 알아봐야함
- recv는 상태를 알 수 있으나 Network 상태는 어떻게 알 수 있는가는 확실하지 않음
- network 는 공공임
- network가 막힌다는 것 : 막히기 때문에 재전송하면 더더욱 막히게 되는 악순환이있음
- TCP가 제대로 돌려면 네트워크가 막히면 안됨
- 네트워크가 막히지 않게하려면, 데이터 속도를 줄여야함
- TCP라는 것은 각각 존재하되 서로를 위해 행동함

Closing TCP Connection

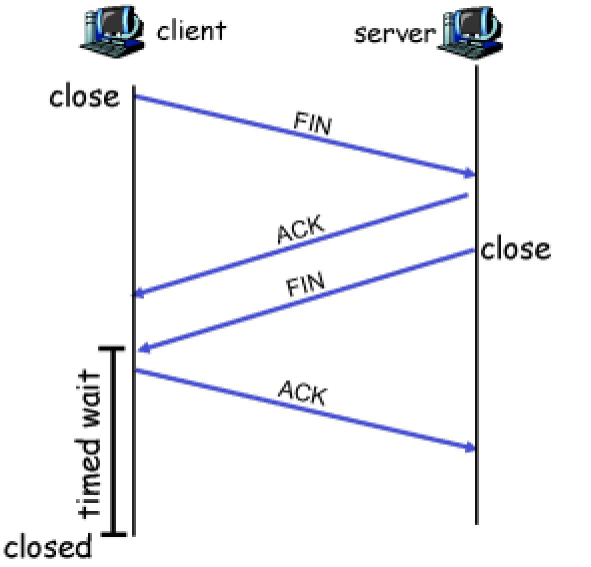
Closing a connection:

client closes socket:

```
clientSocket.close();
```

Step 1: client end system sends TCP FIN control segment to server

Step 2: server receives FIN, replies with ACK. Closes connection, sends FIN.

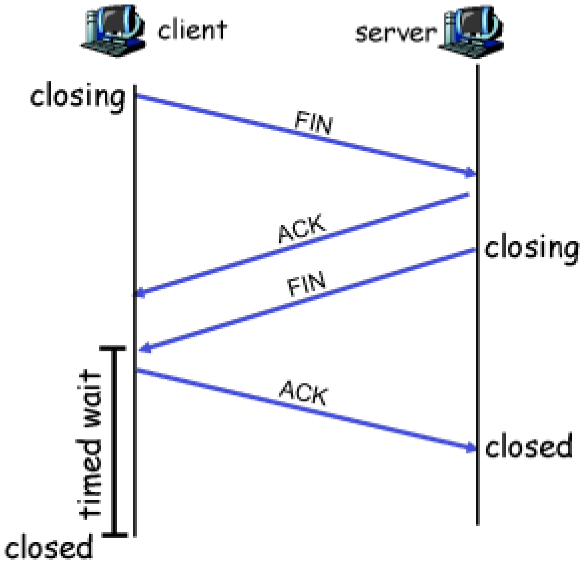


TCP Connection Management (cont.)

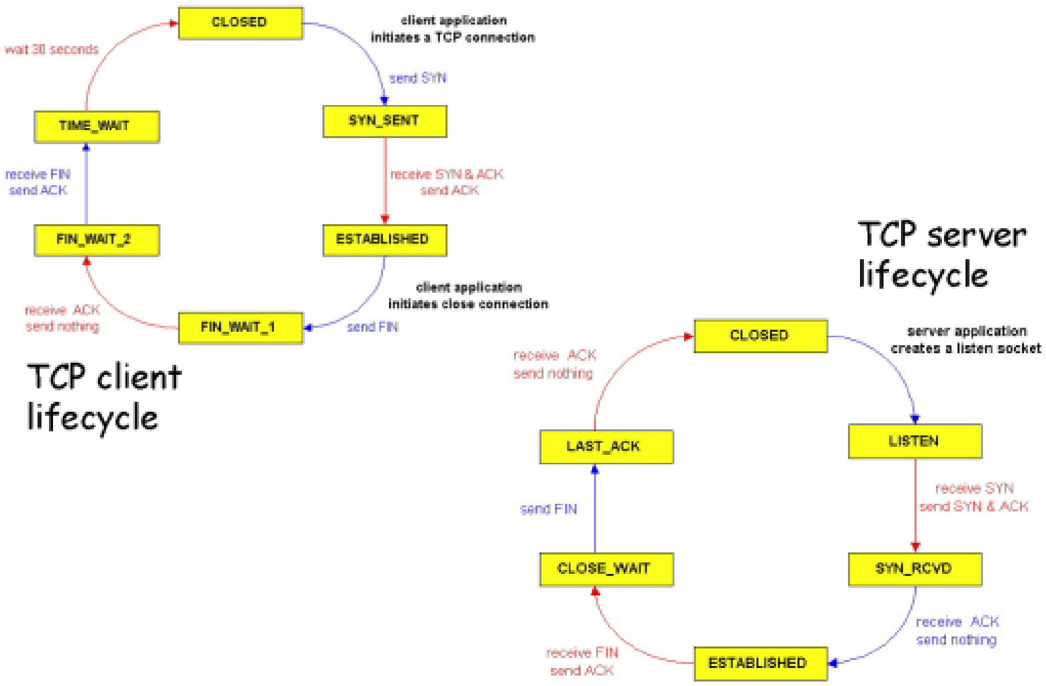
Step 3: client receives FIN, replies with ACK.

- Enters "timed wait" - will respond with ACK to received FINs

Step 4: server, receives ACK. Connection closed.

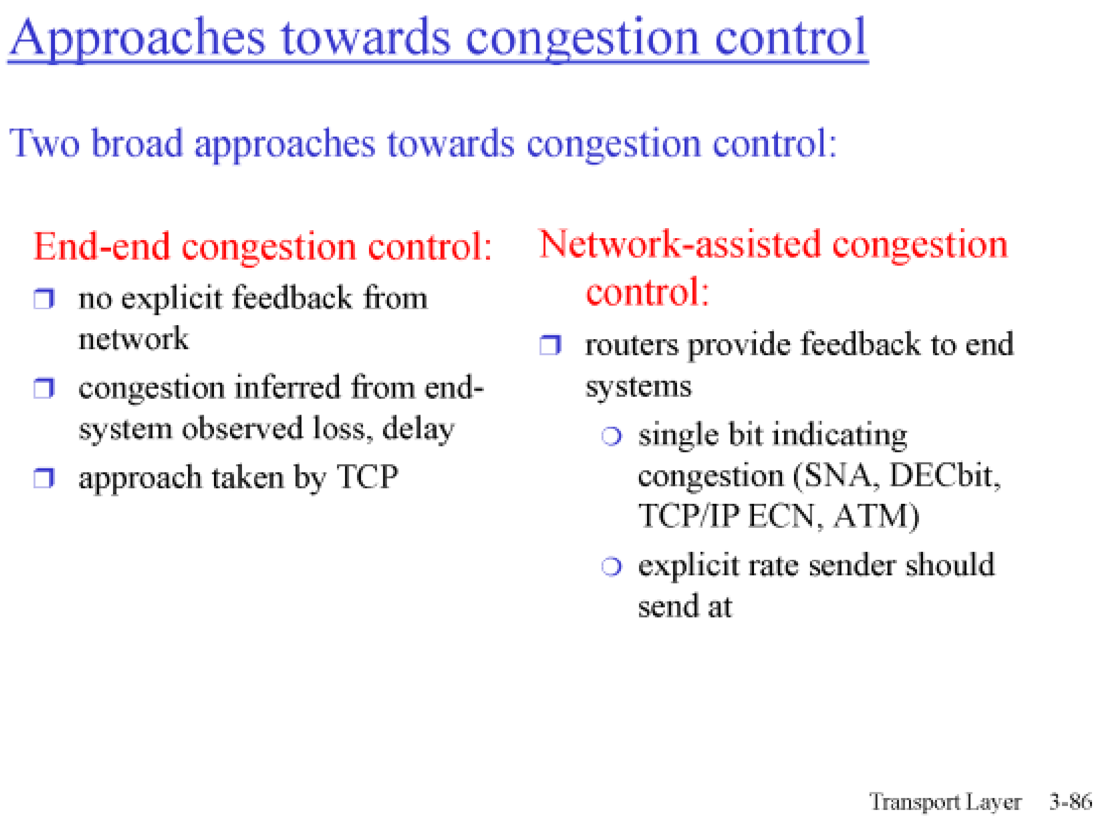


TCP Connection Management (cont)



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



Approaches towards congestion control

Two broad approaches towards congestion control:

End-end congestion control:

- no explicit feedback from network
- congestion inferred from end-system observed loss, delay
- approach taken by TCP

Network-assisted congestion control:

- routers provide feedback to end systems
  - single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
  - explicit rate sender should send at

Transport Layer 3-86

- end-end 가 실제 사용하는 방식
- 아무것도 주지 않지만 각자가 유추해서 행동함 TCP segment로 유추해서 행동함
- segment 쭉 보내는데 ACK가 느리게 오거나 오지 않으면 무엇인가 문제가 생겼다고 판단, 아주 정확하지 않고, 엇비슷함
- 결국에 전송량을 판단하는 것은 send buf, network가 잘되면 전송량을 늘림
- 잘 안되는 것 같으면 전송량을 줄임