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

unreliable network

↓

reliable network ←

흐름제어

TCP Flow Control

sender와 reciever의 data 처리 속도 차이를 제어

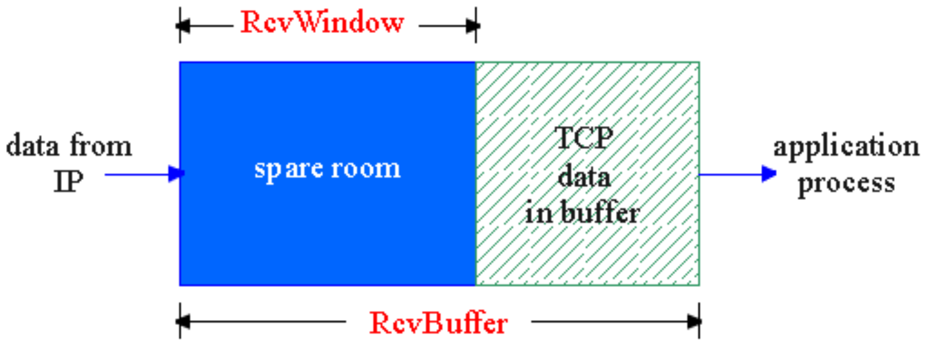
rcv가 너무 많은 pkt을 받지 않게 조절

flow control

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

rcv가 sender에게 자신의 상태를 feedback

- receive side of TCP connection has a receive buffer:



- app process may be slow at reading from buffer

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

데이터 처리 속도 : 수신 > 송신 ok

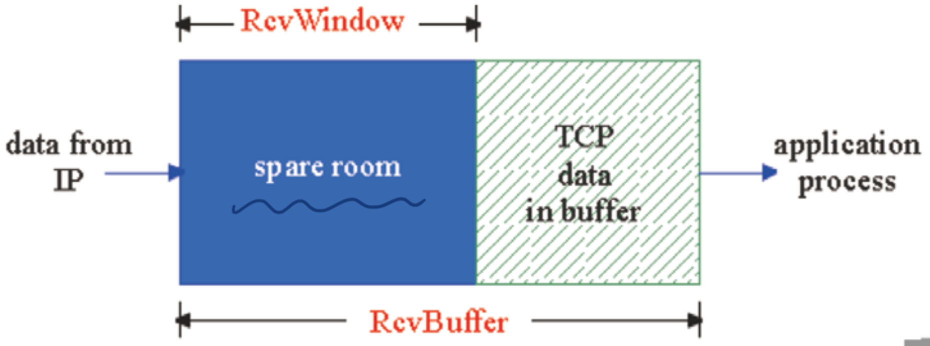
수신 < 송신 Not ok

↓

데이터 손실

∴ 송신측의 데이터 전송량을 수신측에 따라 조절

TCP Flow control: how it works



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

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

- spare room in buffer

= **RcvWindow**

= **RcvBuffer-[LastByteRcvd - LastByteRead]**

- 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:

for. 논리적 접속 establish

Step 1: client host sends TCP SYN segment to server
- specifies initial seq #
- no data client → server

Step 2: server host receives SYN, replies with SYNACK segment
server → client
- server allocates buffers
- specifies server initial seq. #

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

TCP/IP Protocol을 이용하는 응용프로그램이 정확한 데이터전송을 위해 상대 컴퓨터와 사전에 세션을 수립하는 과정.

→ TCP 접속을 성공적으로 하기 위해 중요.

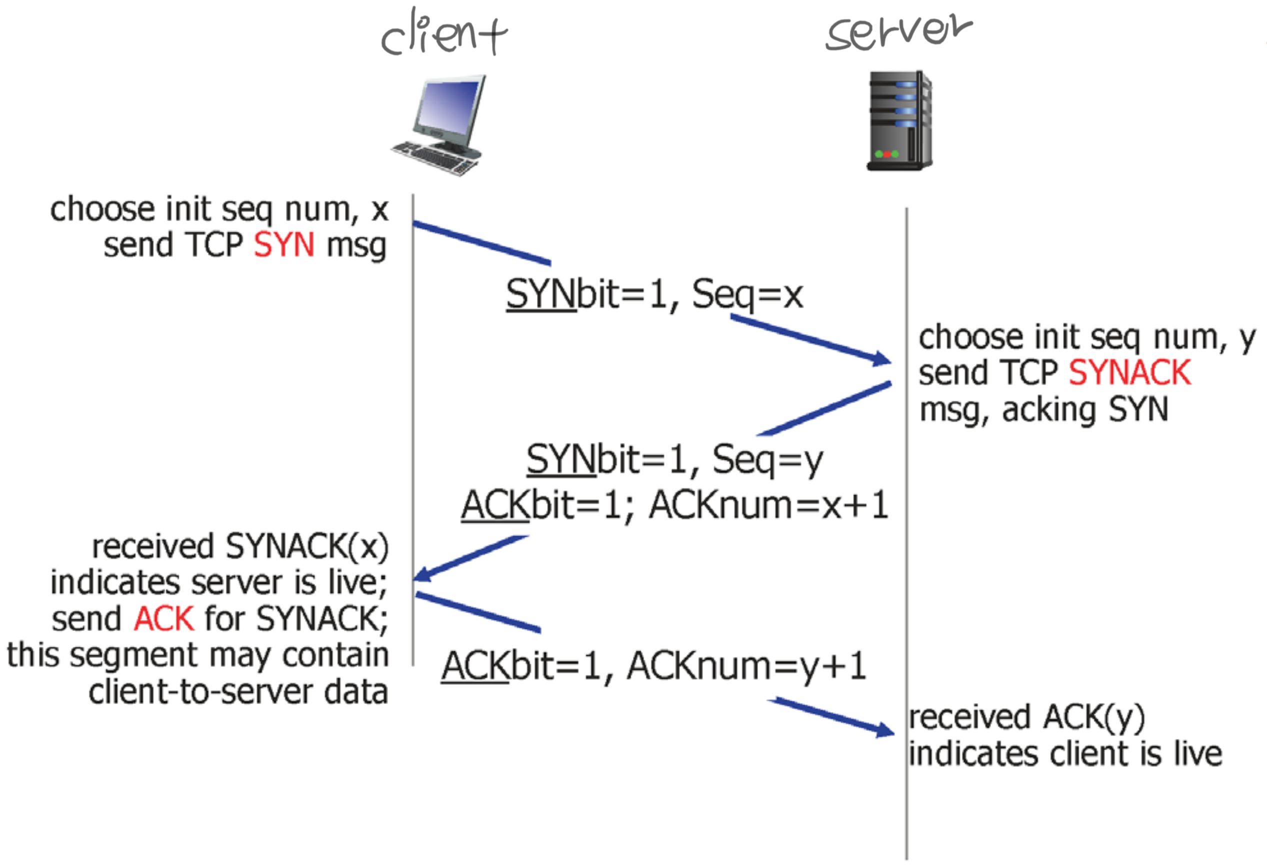
TCP 3-way handshake

양쪽 모두 데이터를 전송할 준비가 되었음을 보장, 서로 준비 완료됨을 알 수 있게 한다.

상대의 초기 순차 일련 번호를 얻을 수 있게한다.

A client는 B 서버에 접속요청 Pck을 보냄.

A클라이언트는 SYN을 보내고 SYN/Ack응답 기다리는 SYN-SENT상태. → B server SYN_RECEIVED상태. → B server ESTABLISHED.



choose init seq num, x
send TCP SYN msg

SYNbit=1, Seq=x

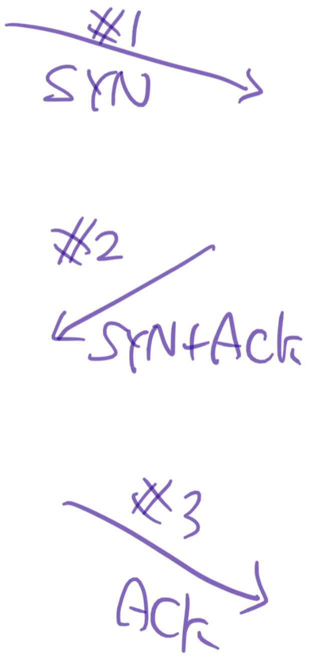
choose init seq num, y
send TCP SYNACK
msg, acking SYN

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



Closing TCP Connection

④.

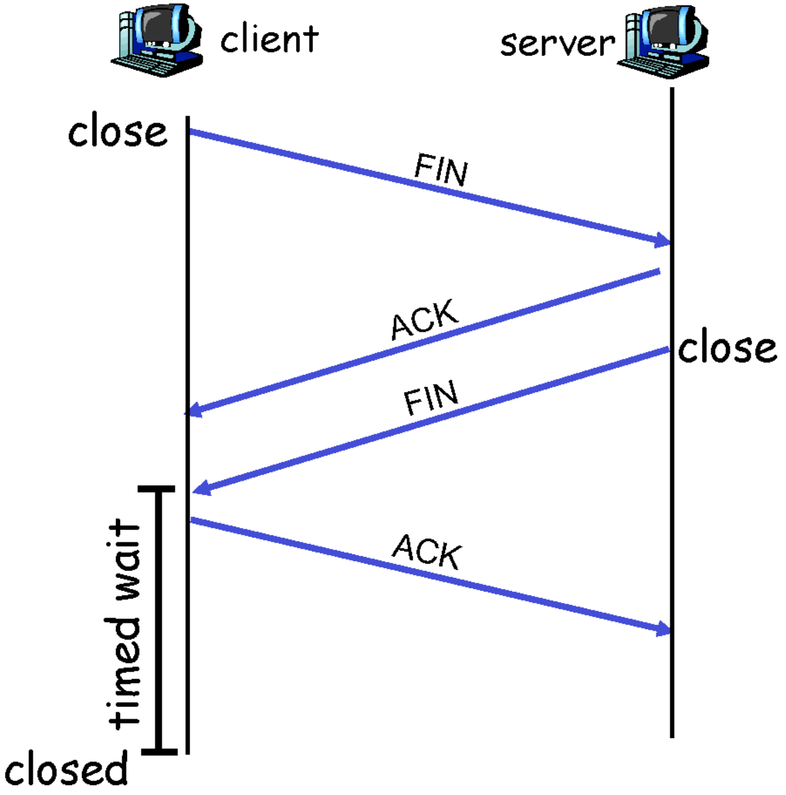
Closing a connection:

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.

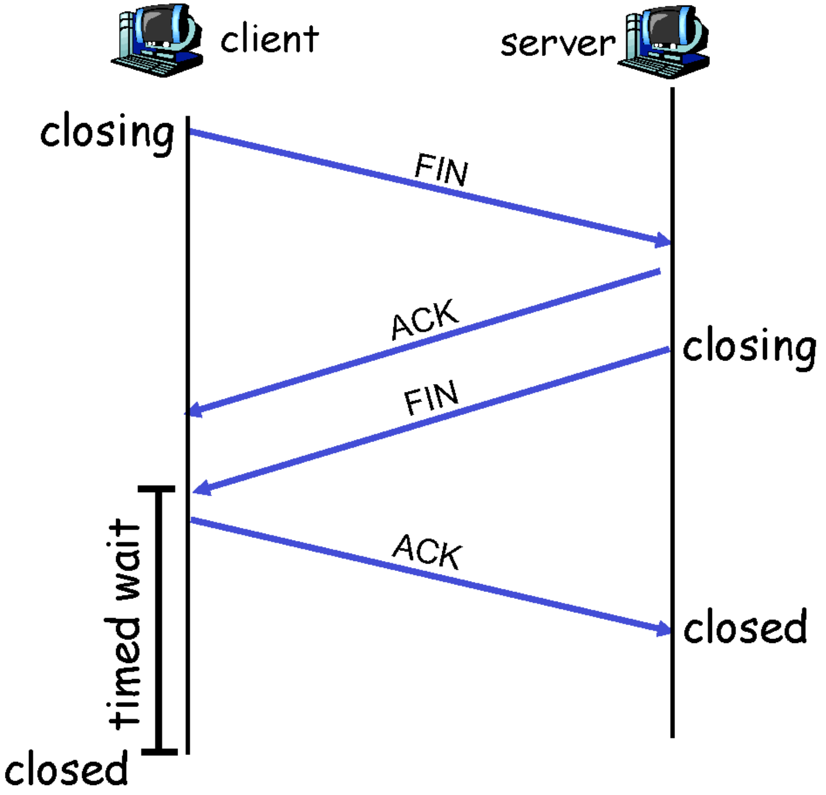


TCP Connection Management (cont.)

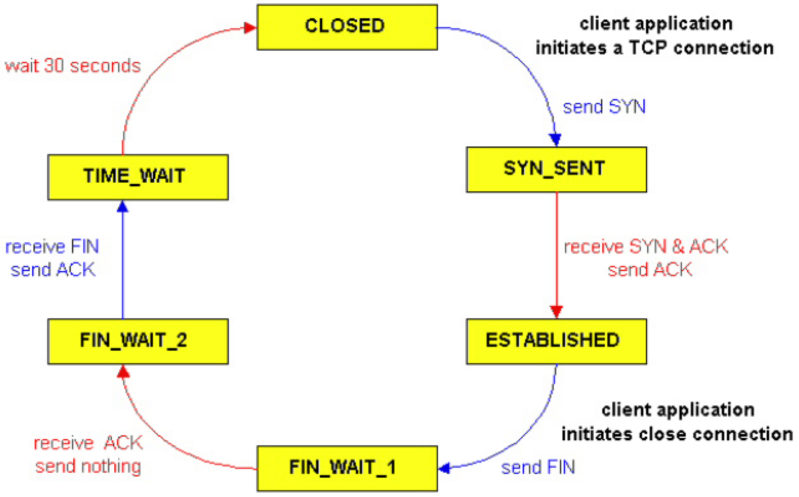
Step 3: client receives FIN, replies with ACK.

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

Step 4: server, receives ACK. Connection closed.



TCP Connection Management (cont)



CLOSED

client application initiates a TCP connection

send SYN

SYN_SENT

receive SYN & ACK
send ACK

ESTABLISHED

client application initiates close connection

send FIN

FIN_WAIT_1

receive ACK
send nothing

FIN_WAIT_2

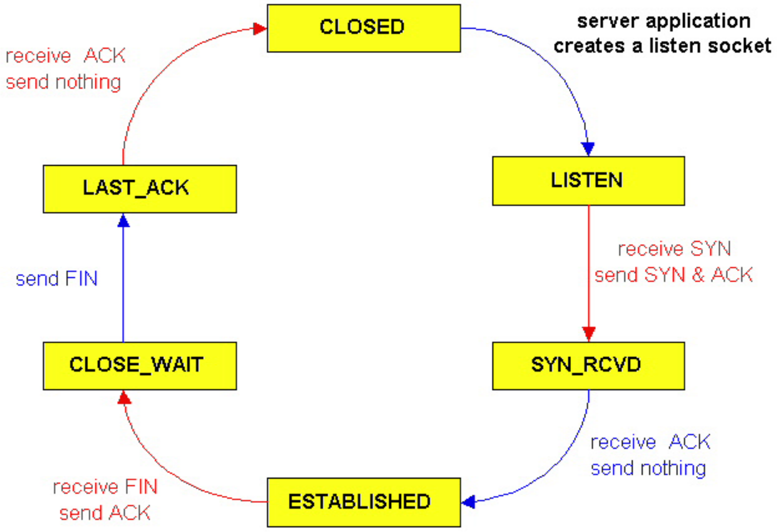
receive FIN
send ACK

TIME_WAIT

wait 30 seconds

TCP client lifecycle

TCP server lifecycle



CLOSED

server application creates a listen socket

LISTEN

receive SYN
send SYN & ACK

SYN_RCVD

receive ACK
send nothing

ESTABLISHED

receive FIN
send ACK

CLOSE_WAIT

send FIN

LAST_ACK

receive ACK
send nothing

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