Korea University

Computer Networks
컴퓨터네트워크

(L4: Ch. 3. Transport Layer)

Wonjun Lee, Ph.D., IEEE Fellow
Network and Security Research Lab. (NetLab)
http://netlab.korea.ac.kr
http://mobile.korea.ac.kr
Korea University

NetLab
Network and Security Research Lab.

Chapter 3: roadmap

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
- Principles of congestion control
- TCP congestion control
- Evolution of transport-layer functionality

TCP congestion control: AIMD

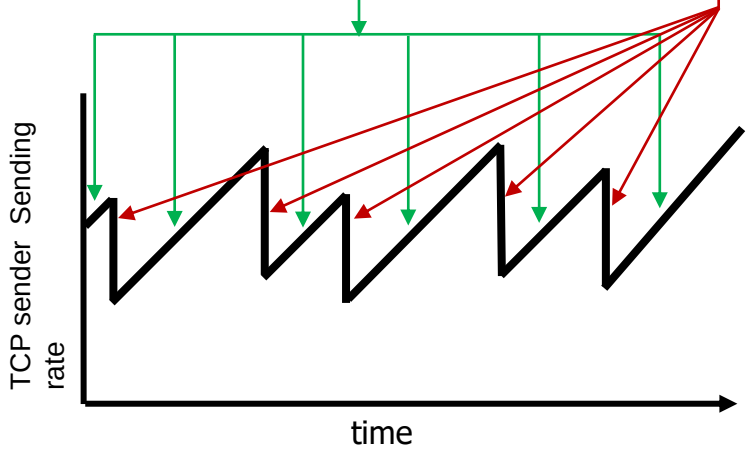
- *approach:* senders can increase sending rate until packet loss (congestion) occurs, then decrease sending rate on loss event

Additive Increase

increase sending rate by 1 maximum segment size every RTT until loss detected

Multiplicative Decrease

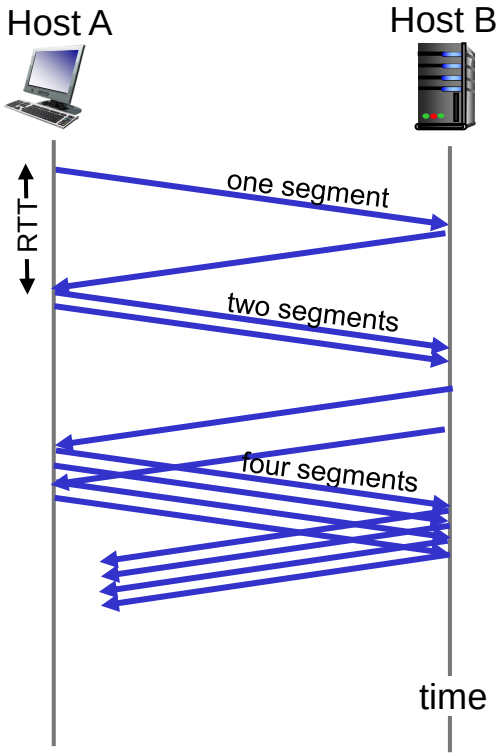
cut sending rate in half at each loss event



AIMD sawtooth behavior: *probing* for bandwidth

TCP slow start

- when connection begins, increase rate exponentially until first loss event:
  - initially **`cwnd`** = 1 MSS
  - double **`cwnd`** every RTT
  - done by incrementing **`cwnd`** for every ACK received
- *summary:* initial rate is slow, but ramps up exponentially fast



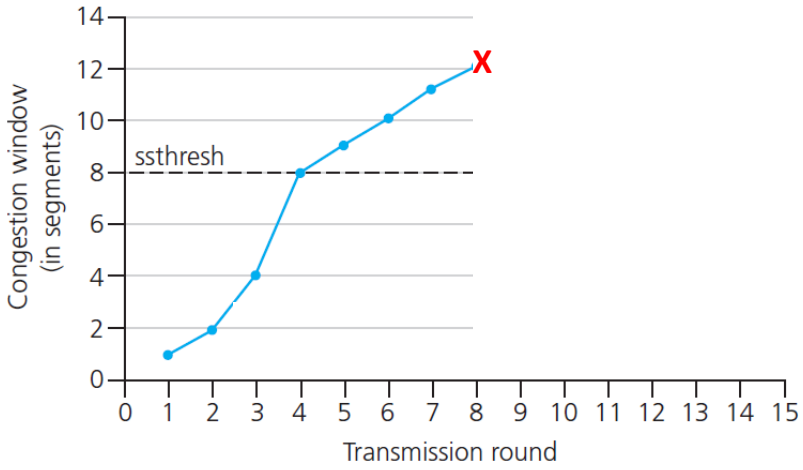
TCP: from slow start to congestion avoidance

Q: when should the exponential increase switch to linear?

A: when **`cwnd`** gets to 1/2 of its value before timeout.

Implementation:

- variable **`ssthresh`**
- on loss event, **`ssthresh`** is set to 1/2 of **`cwnd`** just before loss event



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

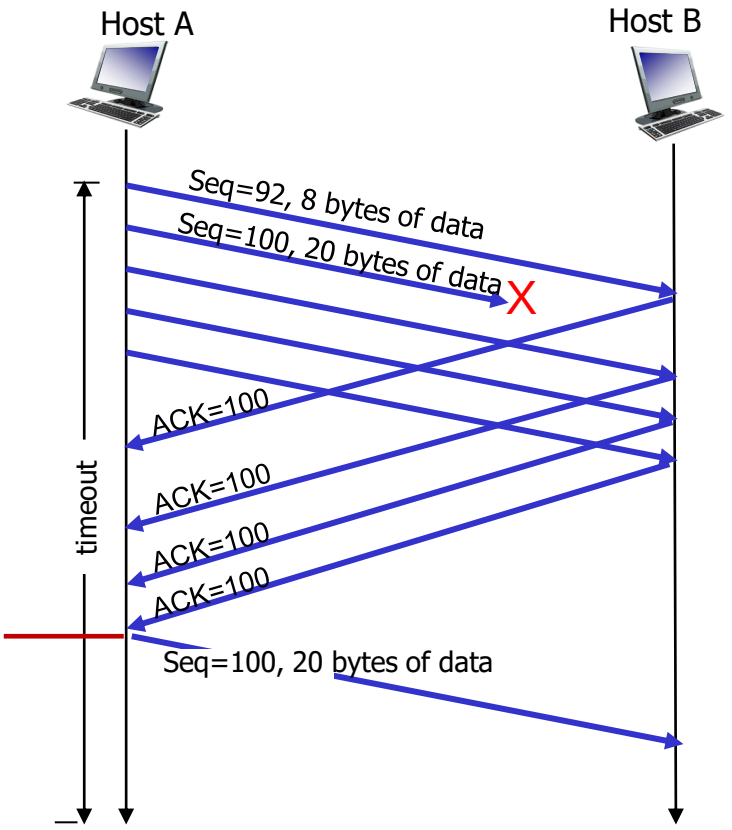
TCP fast retransmit

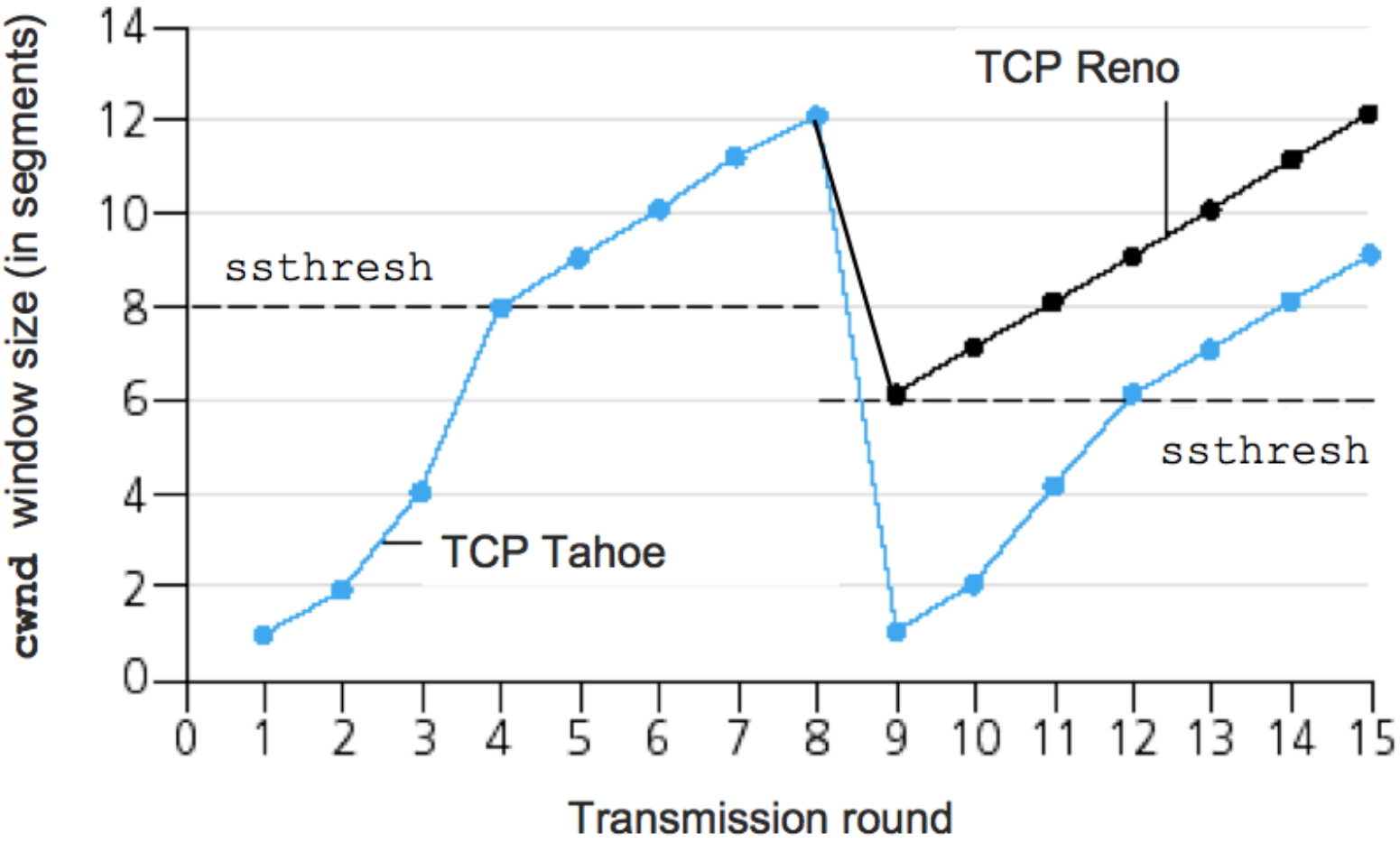
TCP fast retransmit

if sender receives 3 additional ACKs for same data ("triple duplicate ACKs"), resend unACKed segment with smallest seq #

- likely that unACKed segment lost, so don't wait for timeout

Receipt of three duplicate ACKs indicates 3 segments received after a missing segment – lost segment is likely. So retransmit!



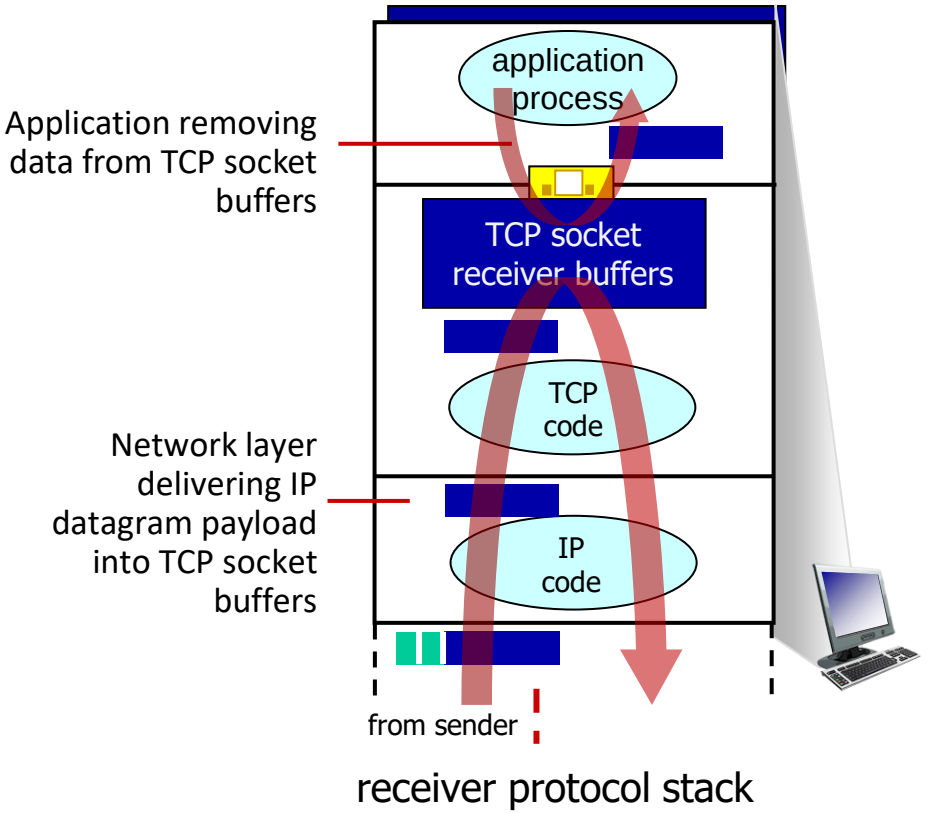

cwnd window size (in segments)
14
12
10
8
6
4
2
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Transmission round
TCP Reno
ssthresh
TCP Tahoe
ssthresh

Chapter 3: roadmap

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- Principles of congestion control
- TCP congestion control

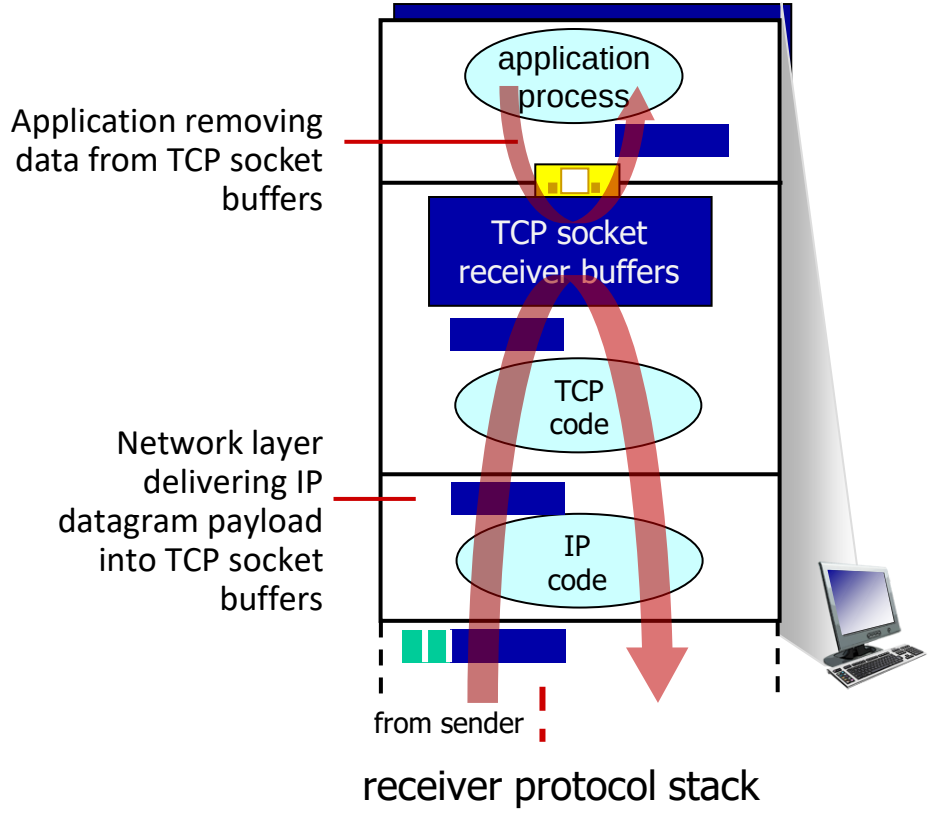
TCP flow control

Q: What happens if network layer delivers data faster than application layer removes data from socket buffers?



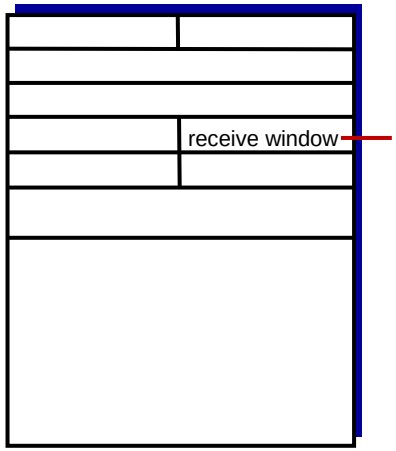
TCP flow control

Q: What happens if network layer delivers data faster than application layer removes data from socket buffers?



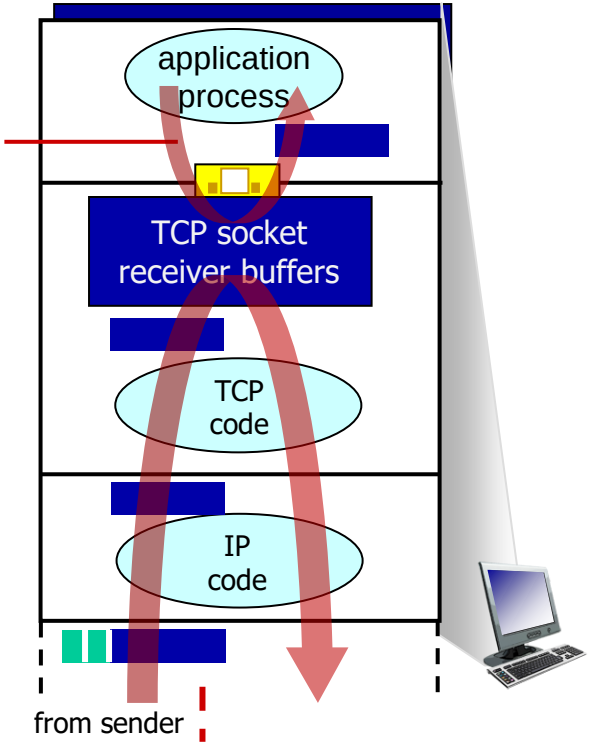
TCP flow control

Q: What happens if network layer delivers data faster than application layer removes data from socket buffers?



flow control: # bytes receiver willing to accept

Application removing data from TCP socket buffers



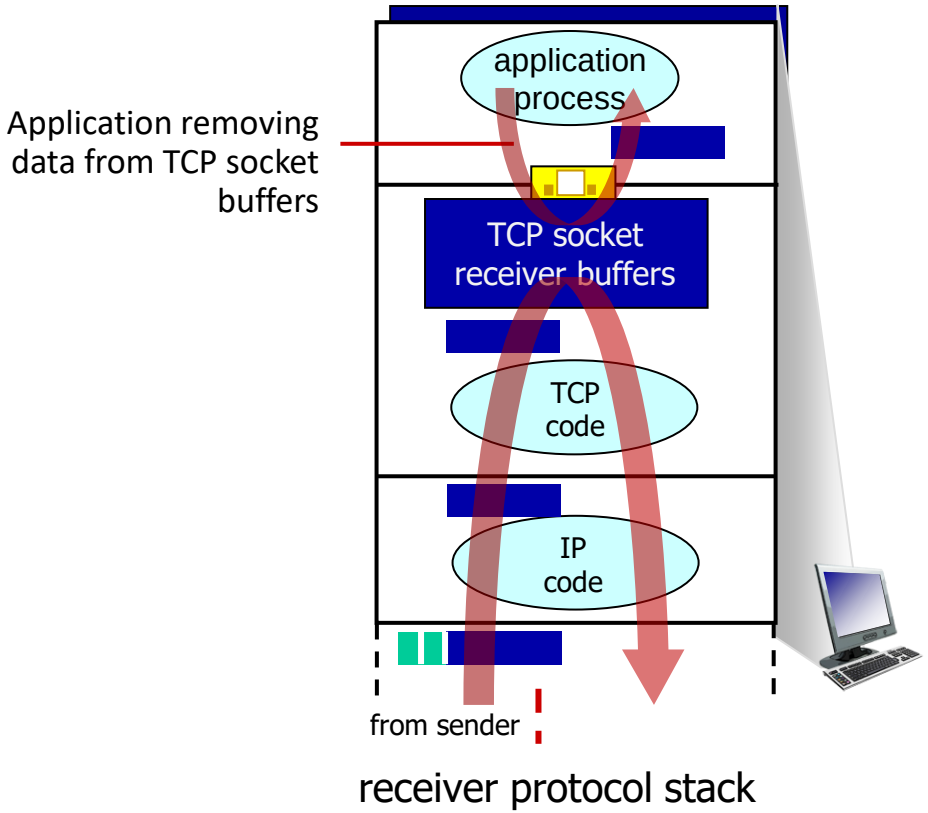
receiver protocol stack

TCP flow control

Q: What happens if network layer delivers data faster than application layer removes data from socket buffers?

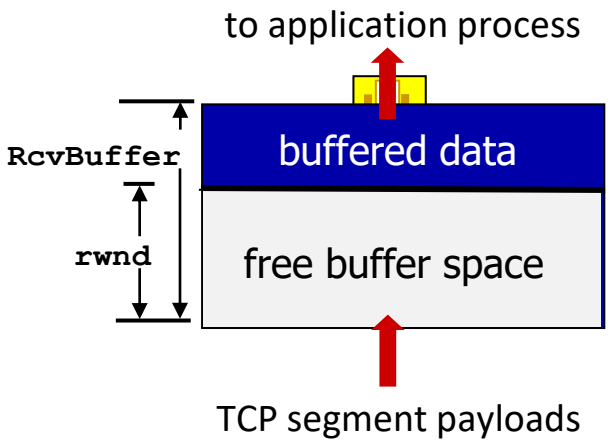
flow control

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



TCP flow control

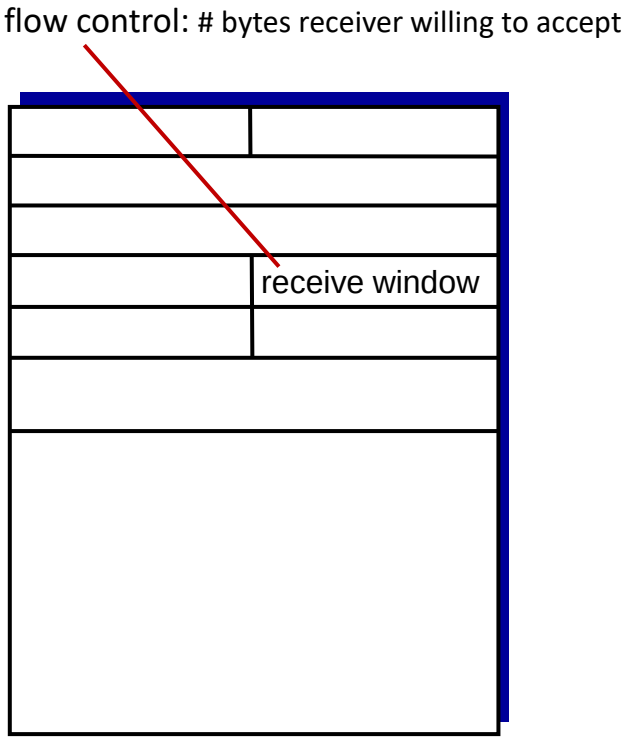
- TCP receiver "advertises" free buffer space in **`rwnd`** field in TCP header
  - **`RcvBuffer`** size set via socket options (typical default is 4096 bytes)
  - many operating systems autoadjust **`RcvBuffer`**
- sender limits amount of unACKed ("in-flight") data to received **`rwnd`**
- guarantees receive buffer will not overflow



TCP receiver-side buffering

TCP flow control

- TCP receiver "advertises" free buffer space in **`rwnd`** field in TCP header
  - **`RcvBuffer`** size set via socket options (typical default is 4096 bytes)
  - many operating systems autoadjust **`RcvBuffer`**
- sender limits amount of unACKed ("in-flight") data to received **`rwnd`**
- guarantees receive buffer will not overflow

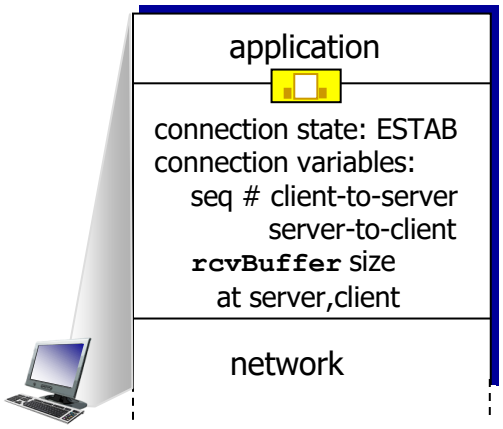


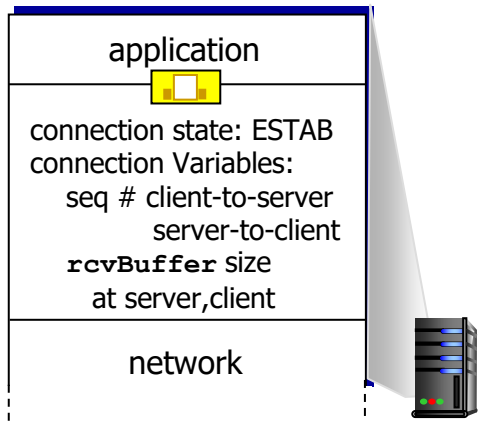
TCP segment format

TCP connection management

before exchanging data, sender/receiver “handshake”:

- agree to establish connection (each knowing the other willing to establish connection)
- agree on connection parameters (e.g., starting seq #s)



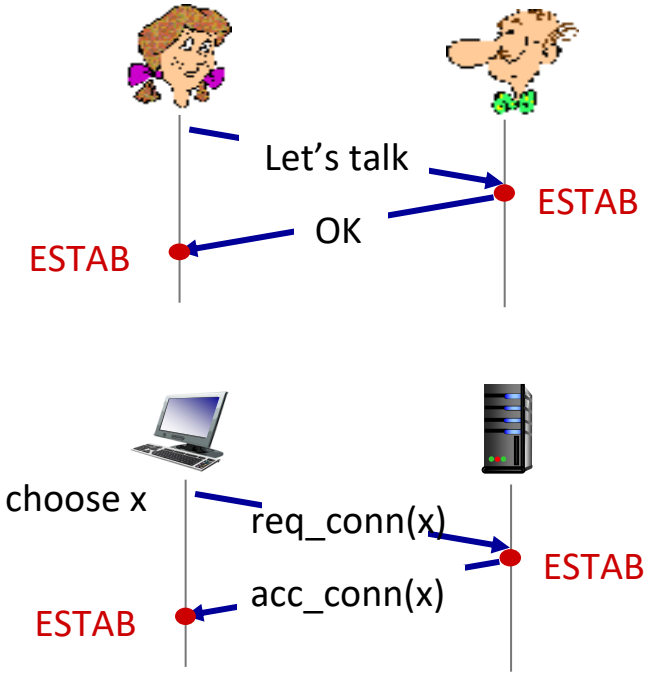



```
Socket clientSocket =
  newSocket("hostname","port number");
```

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

Agreeing to establish a connection

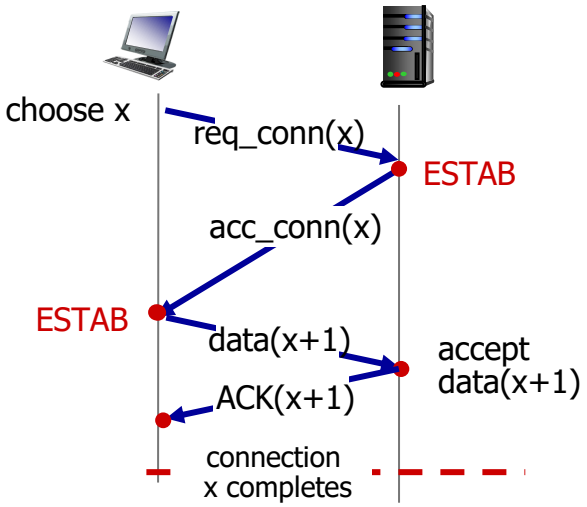
2-way handshake:



Q: will 2-way handshake always work in network?

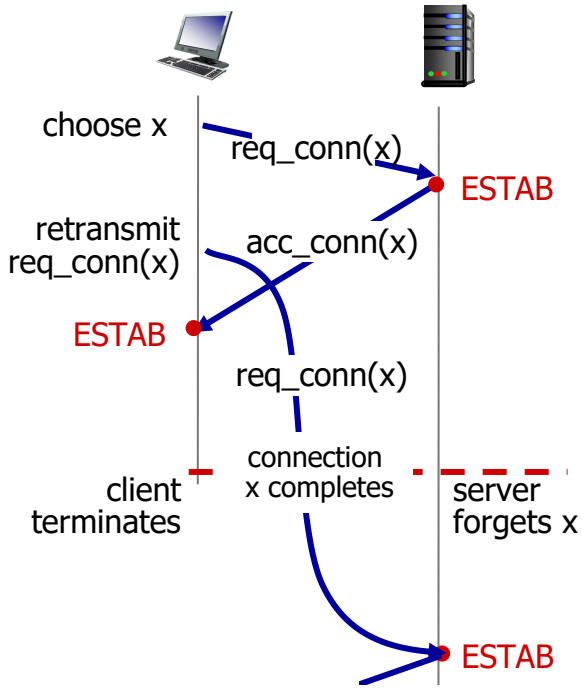
- variable delays
- retransmitted messages (e.g. req_conn(x)) due to message loss
- message reordering
- can't "see" other side

2-way handshake scenarios



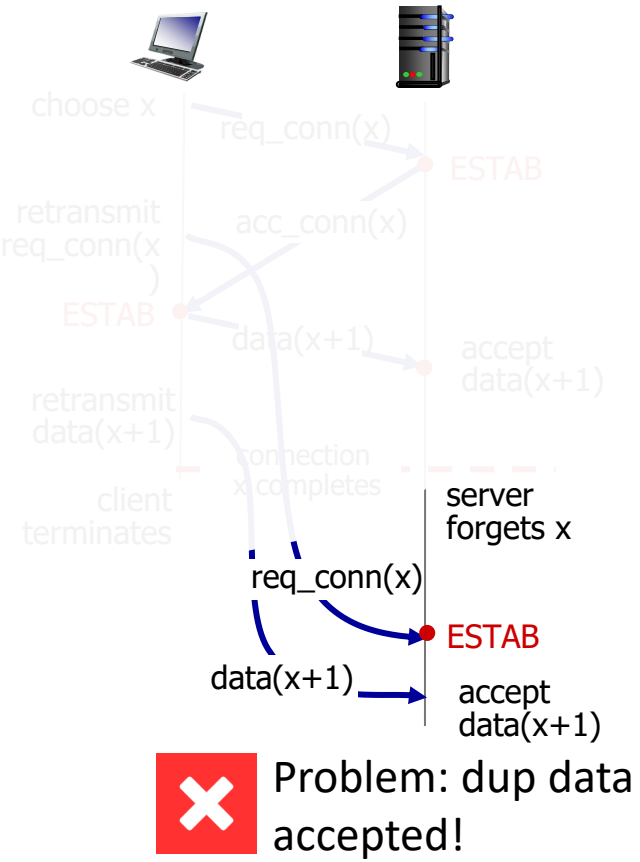
No problem!

2-way handshake scenarios

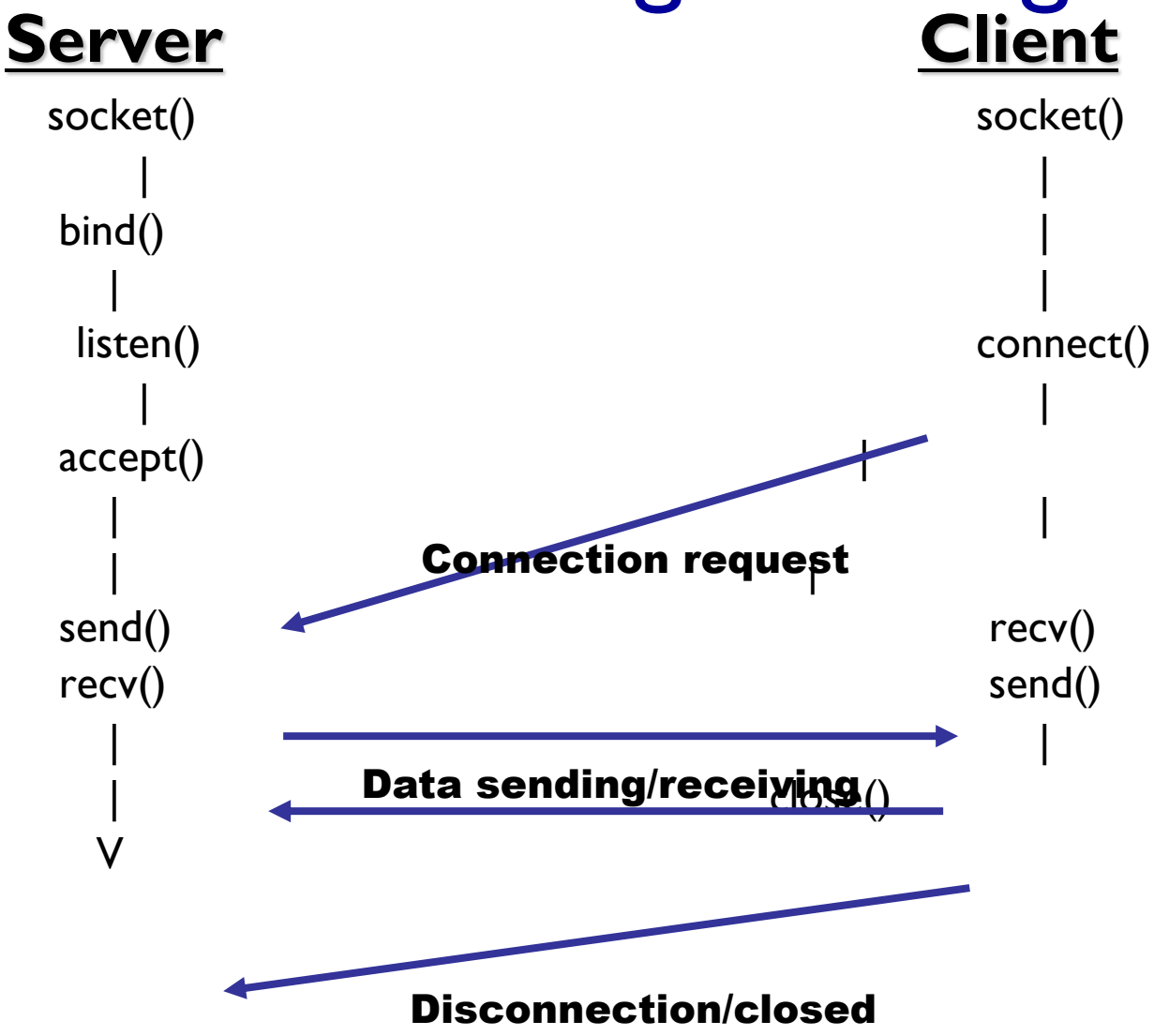


Problem: half open connection! (no client)

2-way handshake scenarios



TCP Socket Programming



TCP 3-way handshake

Client state

```
clientSocket = socket(AF_INET, SOCK_STREAM)
```

LISTEN

```
clientSocket.connect((serverName,serverPort))
```

choose init seq num, x
send TCP SYN msg

SYNSENT

SYNbit=1, Seq=x

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

ESTAB

ACKbit=1, ACKnum=y+1

Server state

```
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
connectionSocket, addr = serverSocket.accept()
```

LISTEN

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

SYN RCVD

received ACK(y)
indicates client is live

ESTAB

Korea University

Computer Networks
컴퓨터네트워크

(Course Wrap-up)

Wonjun Lee, Ph.D., IEEE Fellow
Network and Security Research Lab. (NetLab)
http://netlab.korea.ac.kr
http://mobile.korea.ac.kr
Korea University

NetLab
Network and Security Research Lab.

사이버국방학과 네트워크/통신 과목은 3학년 1, 2학기 통년으로 구성됨

- 3학년 1학기 과목53 **컴퓨터네트워크와실습** (유선 네트워크, 인터넷 아키텍쳐 + 유선 보안);

- 3학년 2학기 과목81 **무선이동통신네트워크** (무선/이동 통신네트워크 + 무선 보안);

➔ 폐지

스마트보안학부 (사이버국방학과 포함)

네트워크/통신 과목은 3학년 1, 2학기 통년으로 구성

- **3학년 1학기:**
  - 과목53 ~~컴퓨터네트워크와실습~~ → **과목 100 컴퓨터네트워크** (유선 네트워크, 인터넷 아키텍쳐);
- **3학년 2학기:**
  - **(신설) 컴퓨터네트워크보안** (네트워크 보안, 무선/모바일 통신네트워크 + 무선 보안);

Shall be open for 스마트보안학부/컴퓨터학과/데이터과학과/전전학부/자유전공학부/…

Cf) 고려대 컴퓨터학과: 데이터통신, 컴퓨터네트워크

서울대 컴퓨터공학부: 데이터통신, 컴퓨터네트워크