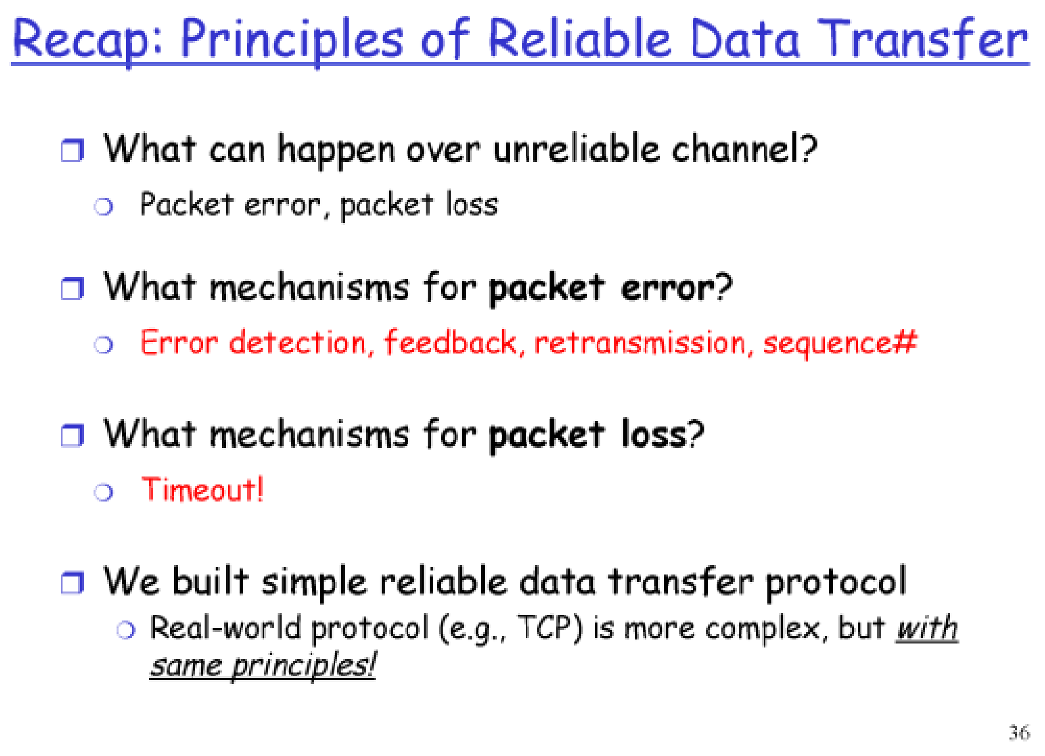
전송계층1

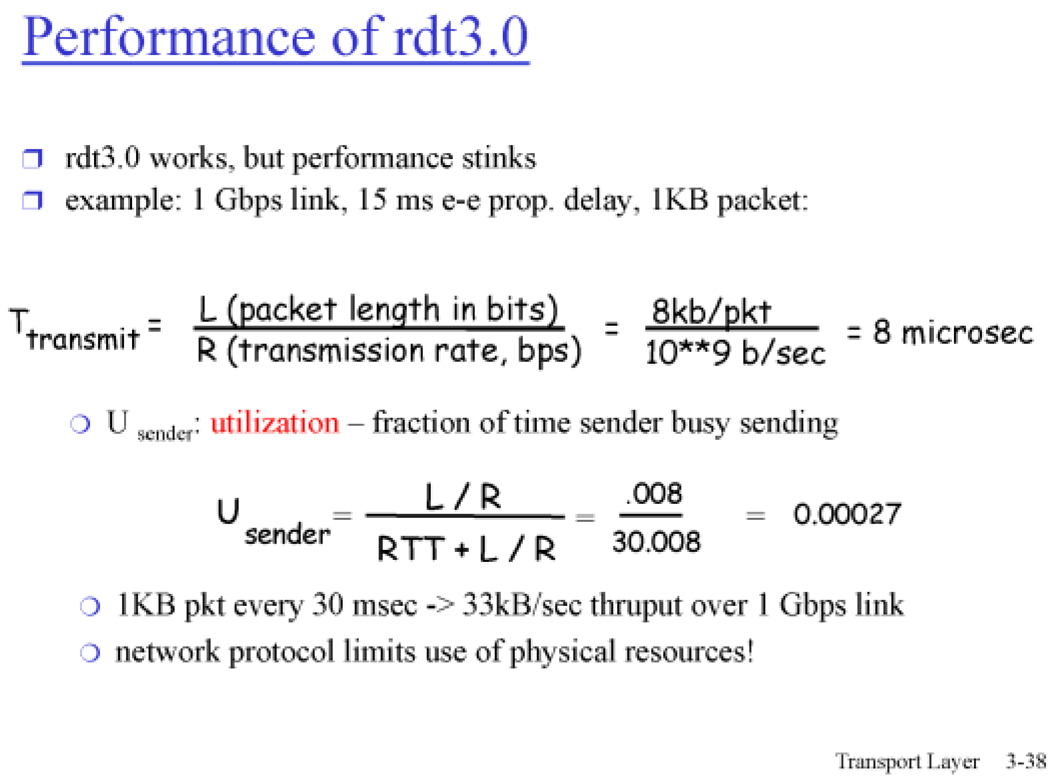
강의날짜	@2022/10/03
작성일시	@2022년 10월 3일 오후 1:35
편집일시	@2022년 10월 3일 오후 9:19
분야	네트워크
공부유형	스터디 그룹
복습	□
태그	



Recap: Principles of Reliable Data Transfer

- What can happen over unreliable channel?
  - Packet error, packet loss
- What mechanisms for **packet error**?
  - Error detection, feedback, retransmission, sequence#
- What mechanisms for **packet loss**?
  - Timeout!
- We built simple reliable data transfer protocol
  - Real-world protocol (e.g., TCP) is more complex, but *with same principles!*

36



Performance of rdt3.0

- rdt3.0 works, but performance stinks
- example: 1 Gbps link, 15 ms e-e prop. delay, 1KB packet:

$$T_{transmit} = \frac{\text{L (packet length in bits)}}{\text{R (transmission rate, bps)}} = \frac{\text{8kb/pkt}}{\text{10**9 b/sec}} = \text{8 microsec}$$

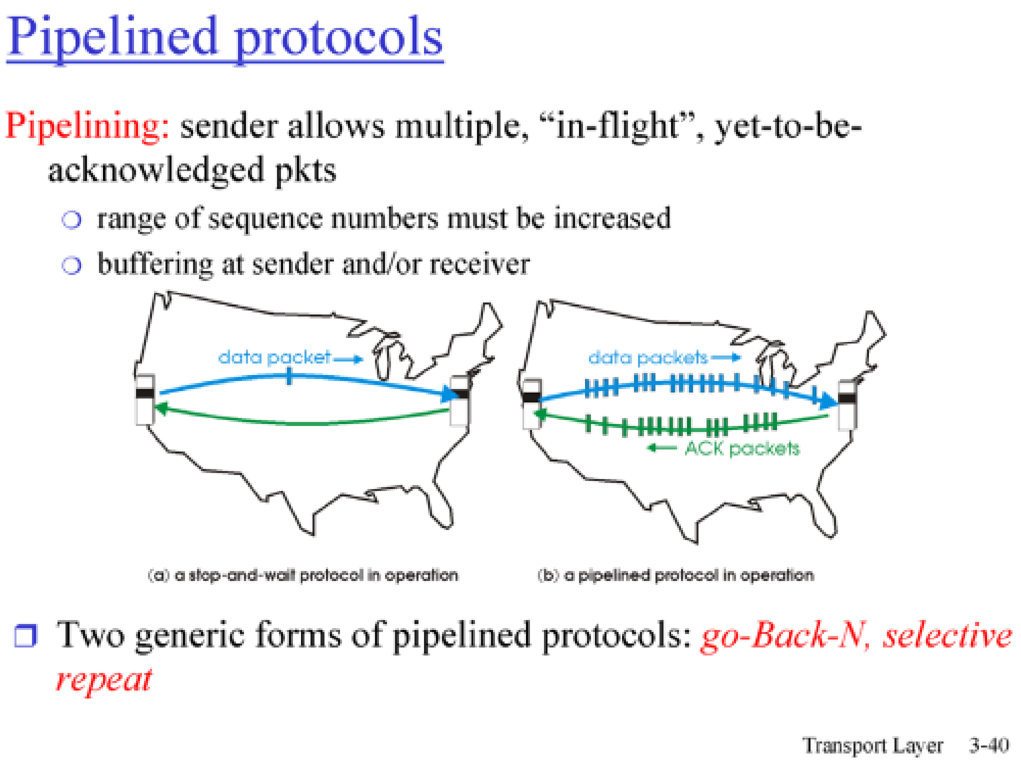
  - U_{sender}: utilization – fraction of time sender busy sending

$$U_{sender} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

  - 1KB pkt every 30 msec -> 33kB/sec thruput over 1 Gbps link
  - network protocol limits use of physical resources!

Transport Layer 3-38

- 효율이 얼마나 좋은가
- 센더가 리시버한테 패킷하나를 보내는데 걸리는 시간
- 너무나 비효율적
- RDT 프로토콜은 신뢰성이 좋지만 성능은 매우 안 좋다
- 한번에 하나씩 보내기 때문에



Pipelined protocols

Pipelining: sender allows multiple, "in-flight", yet-to-be-acknowledged pkts

- range of sequence numbers must be increased
- buffering at sender and/or receiver

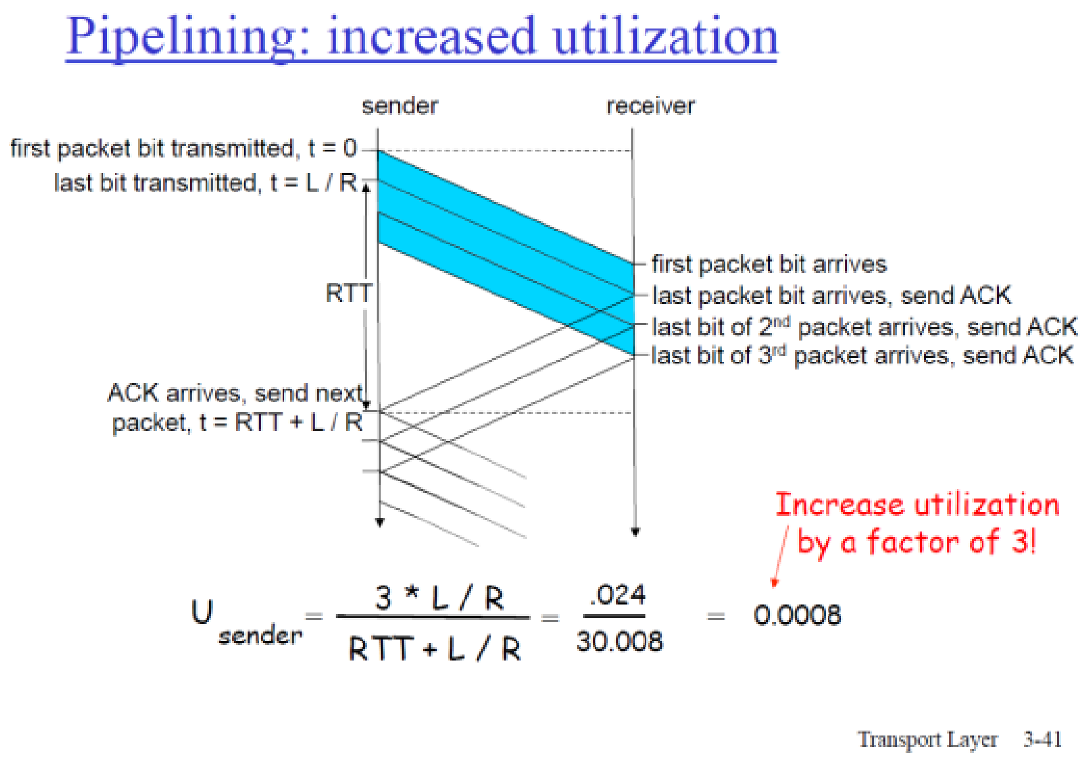
(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

- Two generic forms of pipelined protocols: *go-Back-N, selective repeat*

Transport Layer 3-40

- 한번에 쏟아부어야함
- 파이프라인 프로토콜 : go-back-N, selective repeat
- 현실세계에서 존재하는 것은 아님



Pipelining: increased utilization

sender

receiver

first packet bit transmitted, t = 0

last bit transmitted, t = L / R

RTT

ACK arrives, send next packet, t = RTT + L / R

first packet bit arrives

last packet bit arrives, send ACK

last bit of 2nd packet arrives, send ACK

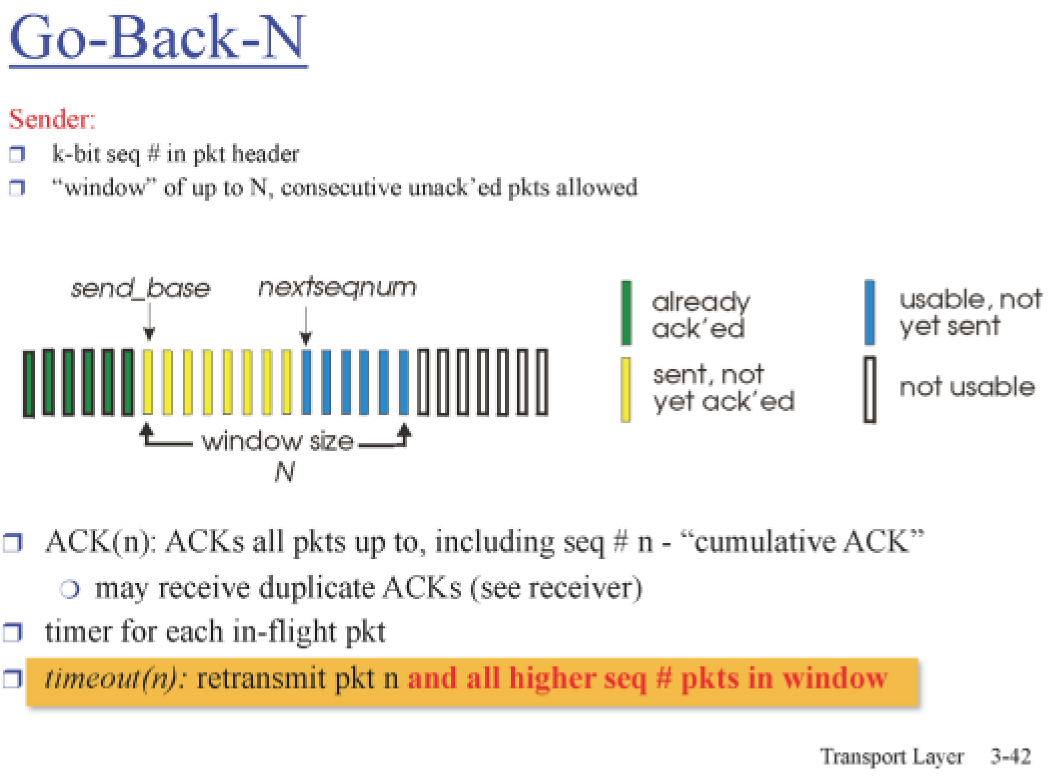
last bit of 3rd packet arrives, send ACK

Increase utilization by a factor of 3!

$$U_{sender} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$

Transport Layer 3-41

- 3배가 늘어남
- 한번에 피드백 없이 보내는 양이 많을 수록 유실량이 많아지긴함



Go-Back-N

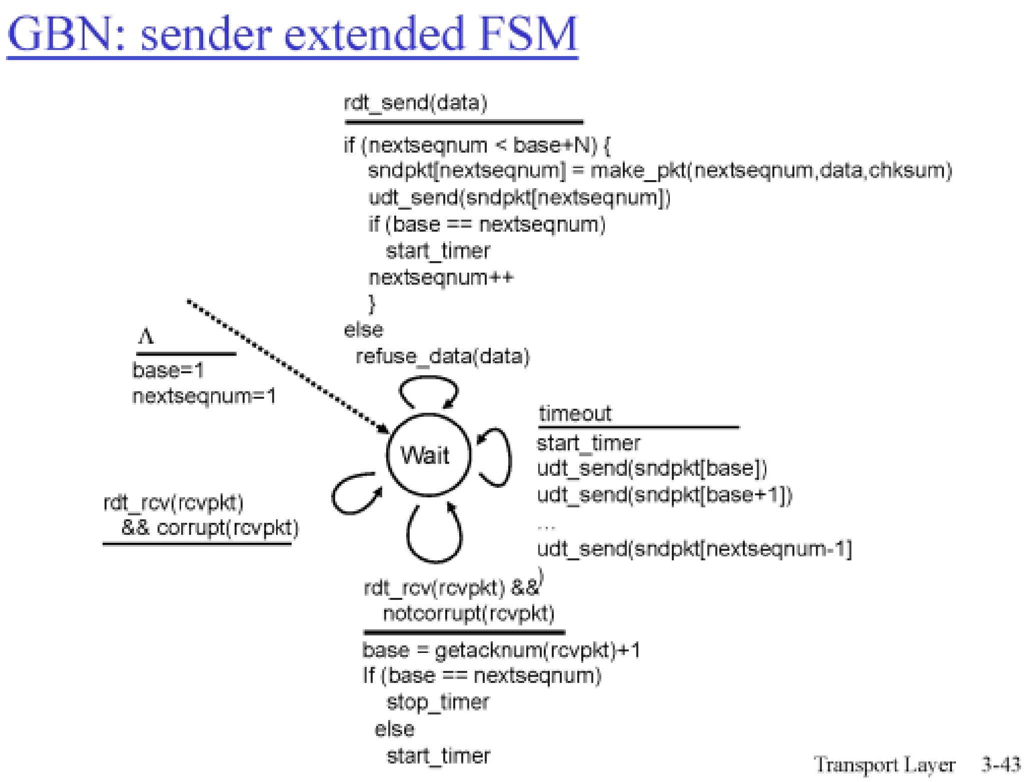
Sender:

- k-bit seq # in pkt header
- "window" of up to N, consecutive unack'ed pkts allowed

- ACK(n): ACKs all pkts up to, including seq # n - "cumulative ACK"
  - may receive duplicate ACKs (see receiver)
- timer for each in-flight pkt
- *timeout(n):* retransmit pkt n **and all higher seq # pkts in window**

Transport Layer 3-42

- 한꺼번에 많은 패킷을 쏟아부음
- window size 만큼은 피드백 받지 않고 쏟아부음
- window size = N / N만큼은 쏟아부을 수 있음
- cumulative ACK
  - ACK 11 : 11번까지는 완벽하게 잘 받았으니까 12번째를 기다린다



GBN: sender extended FSM

```
rdt_send(data)
_____________________
if (nextseqnum < base+N) {
   sndpkt[nextseqnum] = make_pkt(nextseqnum,data,chksum)
   udt_send(sndpkt[nextseqnum])
   if (base == nextseqnum)
      start_timer
   nextseqnum++
   }
else
 refuse_data(data)
```

```
Λ
_____________
base=1
nextseqnum=1
```

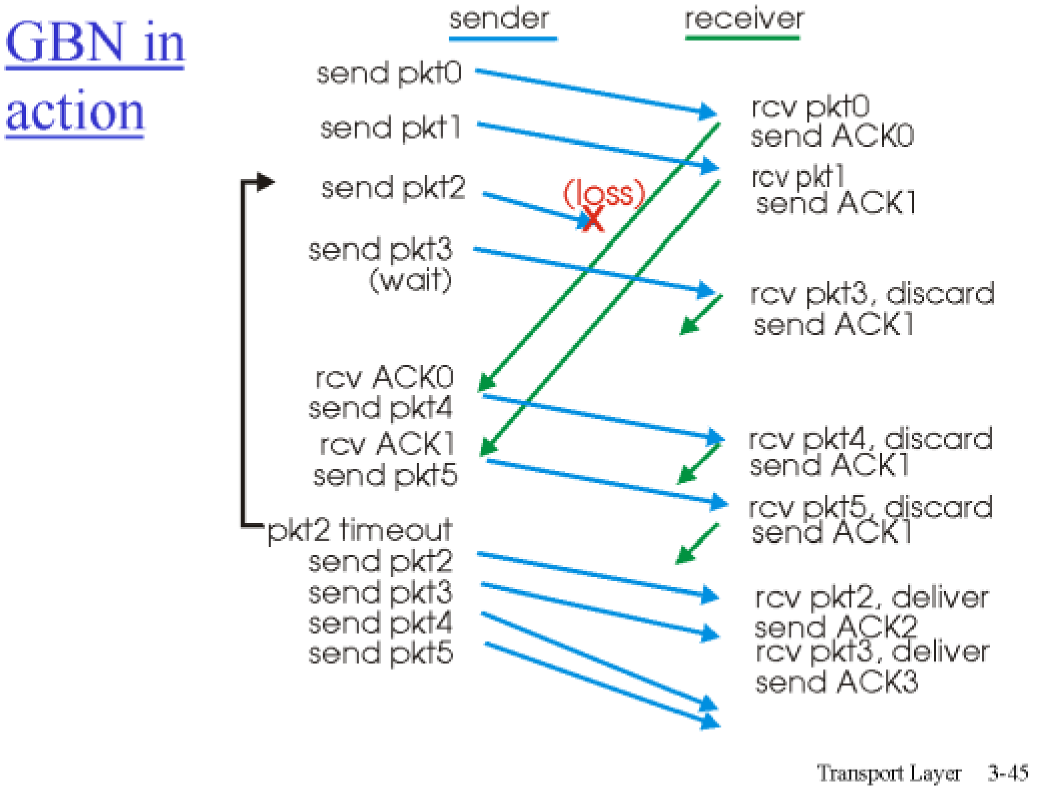
```
timeout
_____________________
start_timer
udt_send(sndpkt[base])
udt_send(sndpkt[base+1])
…
udt_send(sndpkt[nextseqnum-1])
```

```
rdt_rcv(rcvpkt)
  && corrupt(rcvpkt)
_____________________
```

```
rdt_rcv(rcvpkt) &&
  notcorrupt(rcvpkt)
_____________________
base = getacknum(rcvpkt)+1
If (base == nextseqnum)
   stop_timer
 else
   start_timer
```

Transport Layer 3-43

- 리시버가 단순함
- PKT0 PKT2 PKT1 순서대로 오면 1번을 계속기다리느라 2번을 유실함



GBN in action

sender
receiver

send pkt0
send pkt1
send pkt2
send pkt3
(wait)

(loss)

rcv pkt0
send ACK0
rcv pkt1
send ACK1

rcv pkt3, discard
send ACK1

rcv ACK0
send pkt4
rcv ACK1
send pkt5

rcv pkt4, discard
send ACK1
rcv pkt5, discard
send ACK1

pkt2 timeout
send pkt2
send pkt3
send pkt4
send pkt5

rcv pkt2, deliver
send ACK2
rcv pkt3, deliver
send ACK3

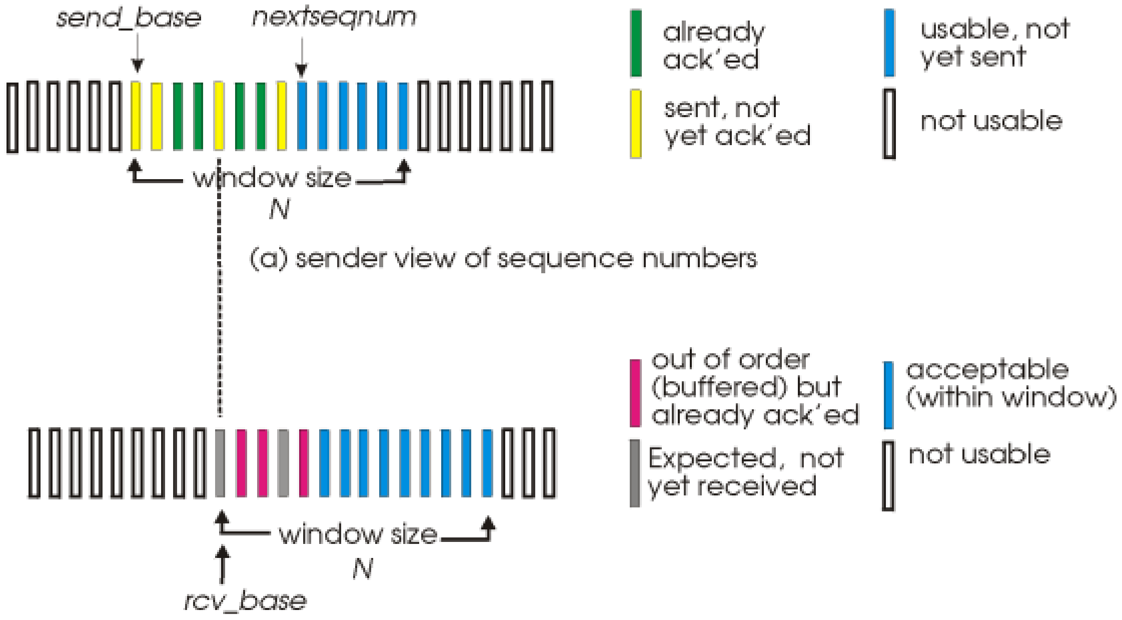
Transport Layer 3-45

- window 안에 있는 애들은 아직 불확실함
- 유실됐을 경우에 유실된 패킷 포함해서 n개를 다 재전송

Selective Repeat

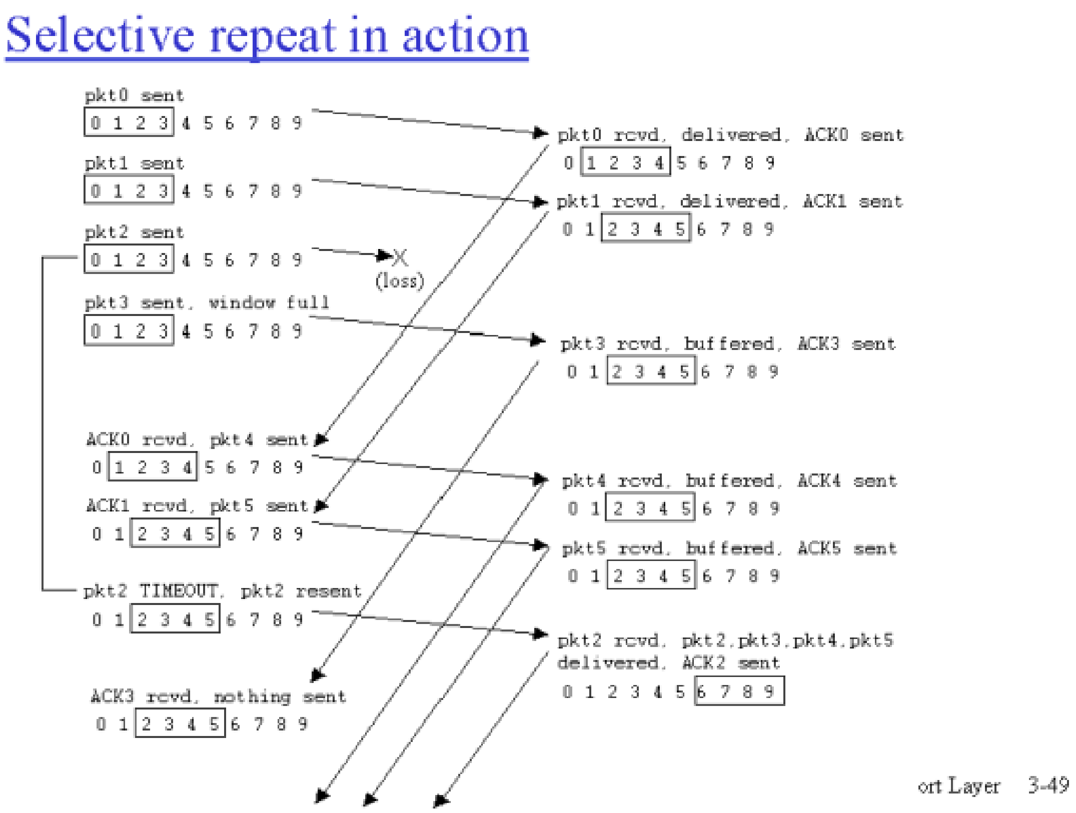
- receiver *individually* acknowledges all correctly received pkts
  - buffers pkts, as needed, for eventual in-order delivery to upper layer
- sender only resends pkts for which ACK not received
  - sender timer for each unACKed pkt
- sender window
  - N consecutive seq #'s
  - again limits seq #s of sent, unACKed pkts

Selective repeat: sender, receiver windows



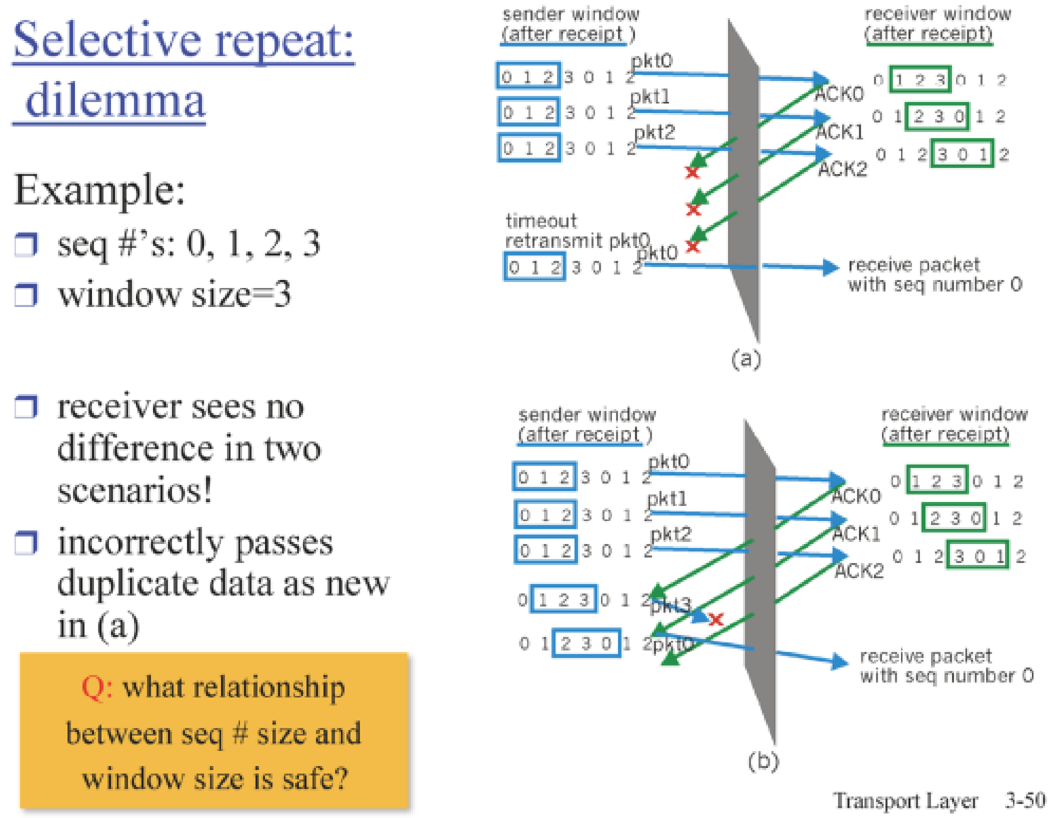
(a) sender view of sequence numbers

(b) receiver view of sequence numbers



Selective repeat in action

- 유실된 패킷만 재전송
- 대신 리시버가 일을 좀 더 해야함



Selective repeat: dilemma

Example:

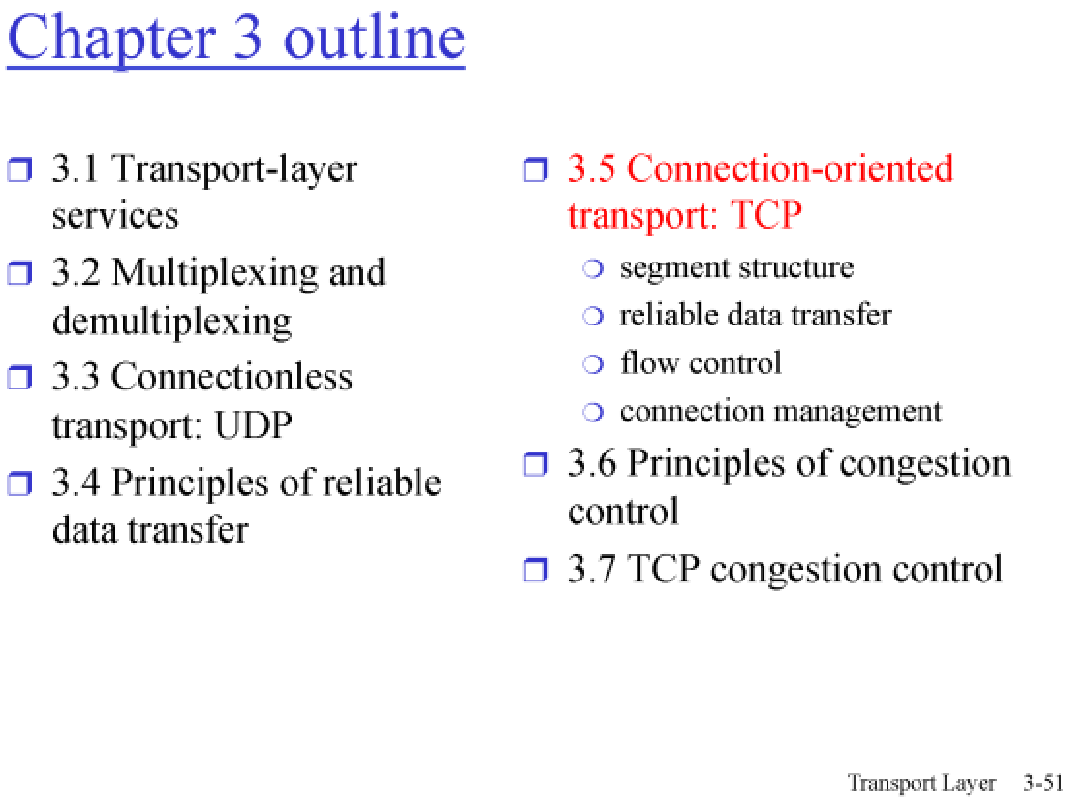
- seq #'s: 0, 1, 2, 3
- window size=3

- receiver sees no difference in two scenarios!
- incorrectly passes duplicate data as new in (a)

Q: what relationship between seq # size and window size is safe?

Transport Layer 3-50

- win = 3
- 필요한 최소한의 seq은 2배..?



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 3-51

- tcp 에는 window timer까지 달면 난리남
- window를 대표하는 타이머 하나만 담