BM 402 Bilgisayar Ağları
(Computer Networks)

M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

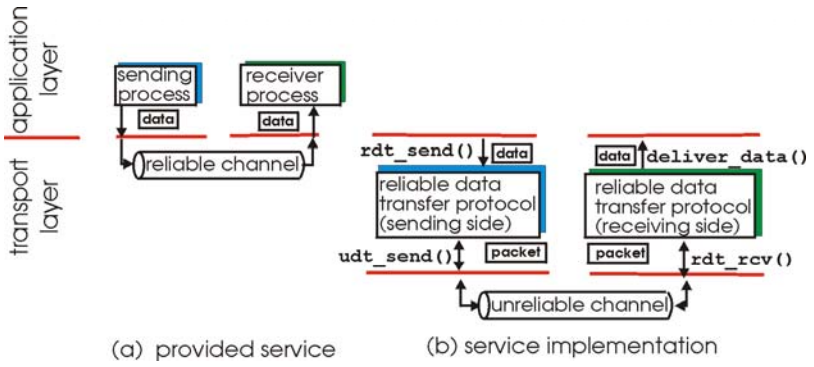
Not: Bu dersin sunumları, ders kitabının yazarları James F. Kurose ve Keith W. Ross tarafından sağlanan sunumlar üzerinde değişiklik yapılarak hazırlanmıştır.

Ders konuları

- Reliable Data Transferin Prensipleri
  - Reliable data transferin performansı
  - Pipeline kullanan protokoller
  - Go-Back-N
  - Selective repeat

Reliable data transferin prensipleri

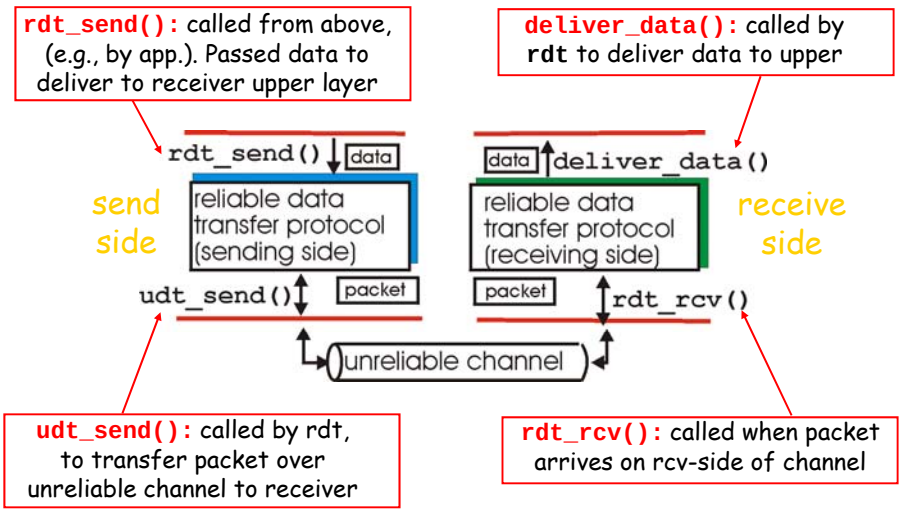
- Application, transport ve data link layer'da önemlidir
- Ağlarda en öncelikli 10 konunun içindedir.



- Reliable data transfer (rdt) protokol daha karmaşıktır.

3/101

Reliable data transfer

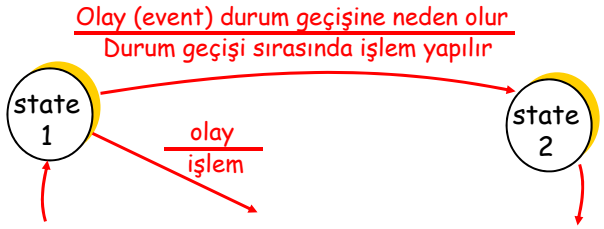


4/101

Reliable data transfer

- reliable data transfer (rdt) protokolün gönderici ve alıcı taraflarını geliştirelim
- Data transferin tek yönlü (unidirectional) olduğunu düşünürsek
  - Ancak kontrol bilgisi iki yönlü gitmektedir
- Sonlu durum makineleriyle (finite state machines-FSM) modellenebilir

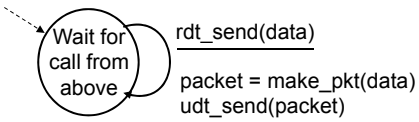
state: bir sonraki durum bu durumdayken oluşan sonraki olayla belirlenir



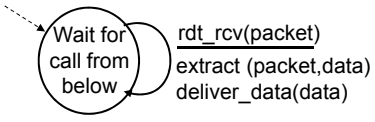
5/101

Rdt 1.0: reliable kanal kullanarak reliable transfer

- Altyapıdaki kanal tümüyle güvenilirdir
  - Bit hatası yoktur
  - Kayıp paket yoktur
- Gönderici ve alıcı için FSM:
  - Gönderici kanala veriyi gönderir
  - Alıcı kanaldan gelen veriyi okur



sender



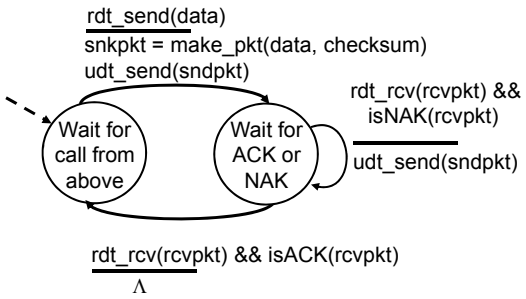
receiver

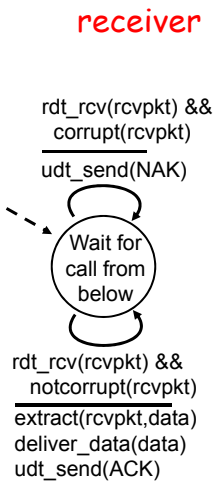
6/101

rdt 2.0: bit hatası olan kanal ile çalışma

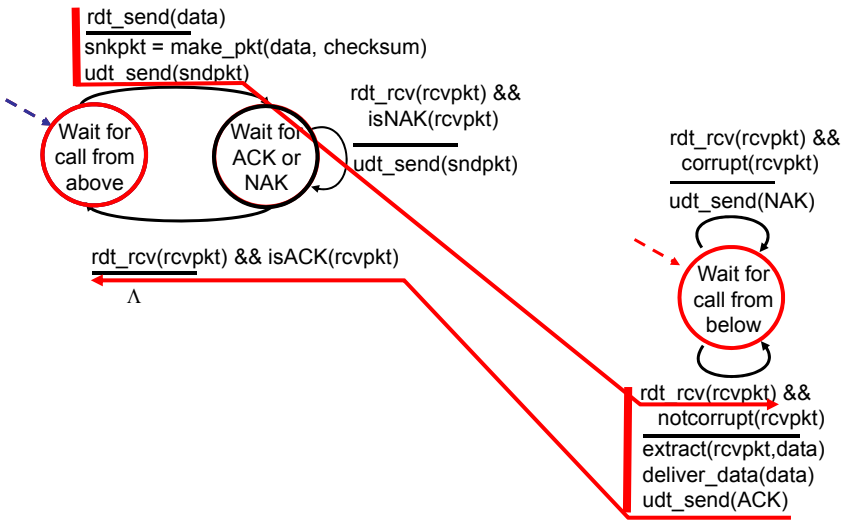
- Kanalda paket içindeki bitlerde bozulma olabilir
  - Bit hatalarını kontrol etmek için checksum kullanılır
- Hatalar nasıl düzeltilir ?
  - *acknowledgements (ACKs):* alıcı göndericiye aldığı paketin hatasız olduğunu iletir.
  - *negative acknowledgements (NAKs):* alıcı göndericiye aldığı paketin hatalı olduğunu bildirir
  - Gönderici NAK ile bildirilen paketi tekar gönderir
- **`rdt2.0`** daki yenilikler(**`rdt1.0 a göre`**):
  - Hata denetimi
  - Alıcı geri bildirimi: kontrol mesajları (ACK, NAK) alıcı->gönderici

rdt2.0: FSM özellikleri



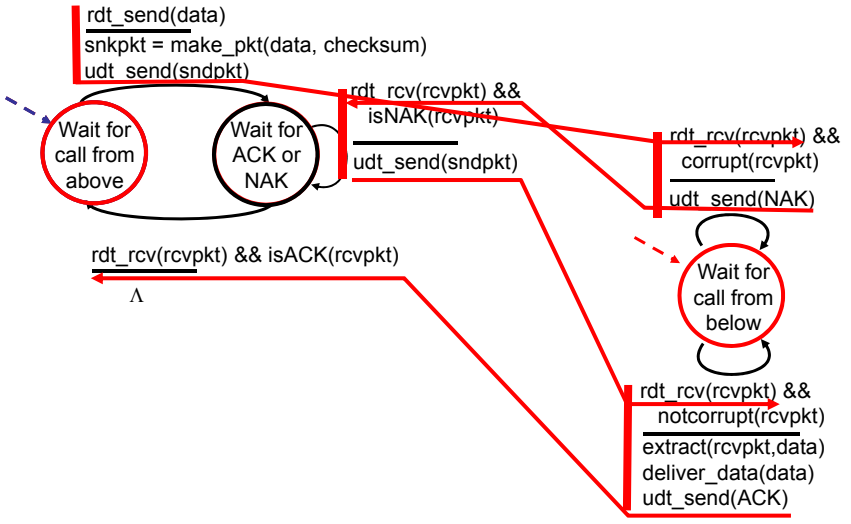


rdt2.0: hata olmadığı zaman çalışma



9/101

rdt2.0: hata durumu



10/101

rdt2.0 da karşılaşılan problemler

ACK/NAK bozulursa ?

- Gönderici alıcıda ne olduğunu bilemez
- Retransmit yapılmaz: duplicate olabilir

Duplicate'lerin seçilmesi:

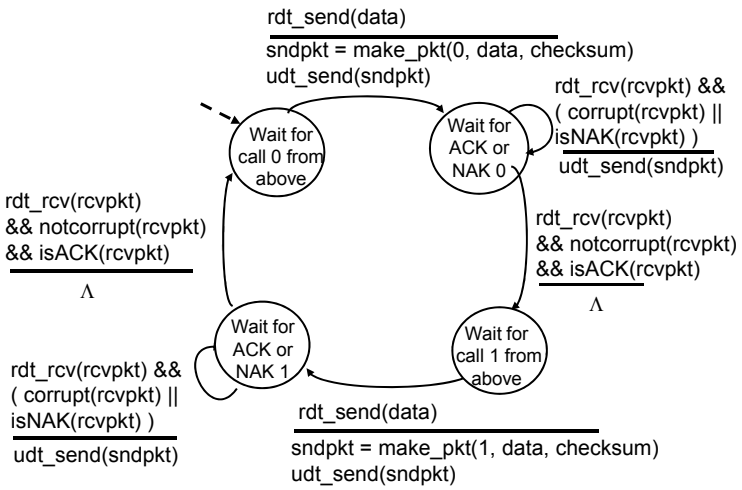
- Gönderici her pakete *sequence number* ekler
- Gönderici mevcut paketi retransmit yapar ACK/NAK bozulursa
- alıcı duplicate paketleri atar

stop and wait

Gönderici bir paket gönderir,
Alıcıdan cevap bekler

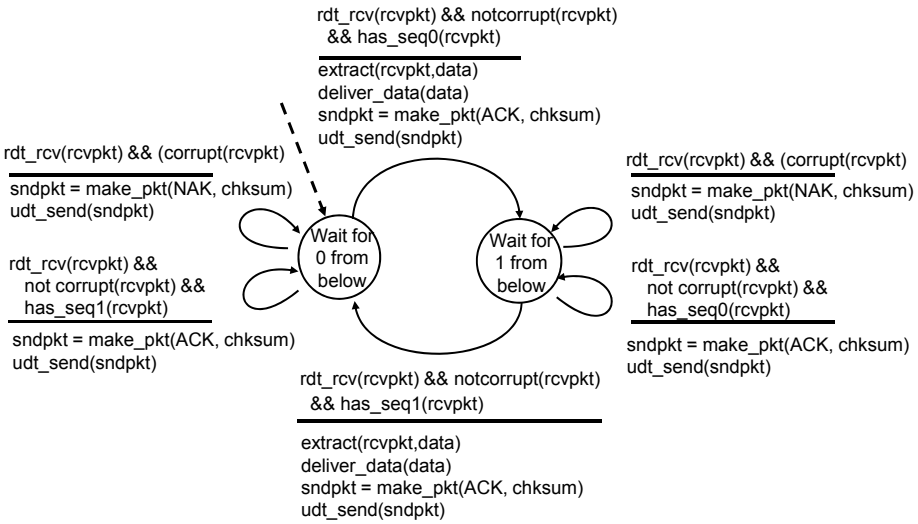
11/101

rdt2.1: gönderici bozulan ACK/NAK ları belirler



12/101

rdt2.1: gönderici bozulan ACK/NAK ları belirler



13/101

rdt2.1: değerlendirme

Gönderici:

- seq # pakete eklenir
- İki seq. no (0,1) yeterlidir.
- alınan ACK/NAK paketin bozuk olup olmadığı kontrol edilir

Alıcı:

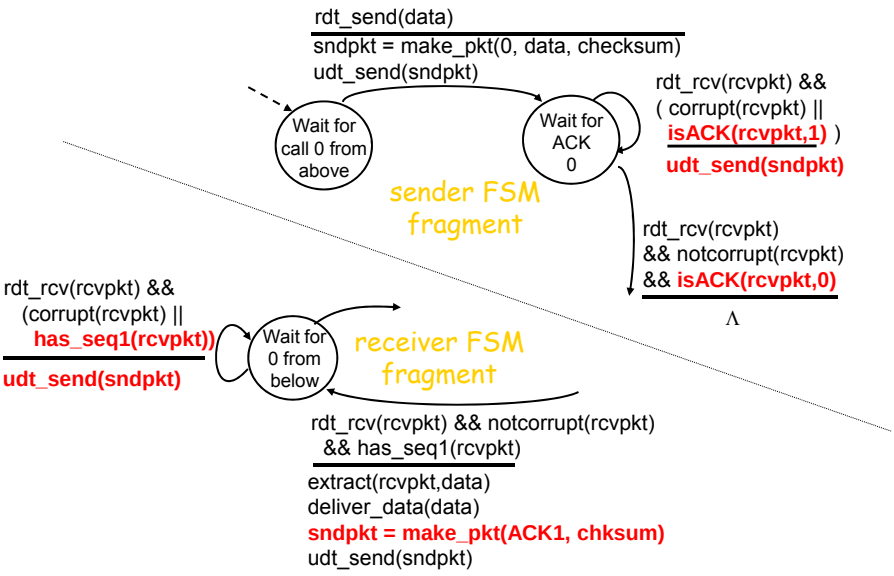
- Gelen paket çiftmi kontrol edilir
  - Bulunulan durum gelen paket için seq. no, 0 veya 1 olaraak bekler

14/101

rdt2.2: NAK kullanılmayan protokol

- ACK kullanarak rdt2.1 ile aynı fonkisyonu görür
- NAK yerine, alıcı en son doğru alınan paket için ACK paket gönderir
  - Alıcı paketin seq numarasını bilmelidir
- Alıcıdaki duplicate ACK : paketin *retransmit edilmesini sağlar*

rdt2.2: gönderici ve alıcı kısımları



rdt3.0: channels with errors *and* loss

New assumption: underlying channel can also lose packets (data or ACKs)

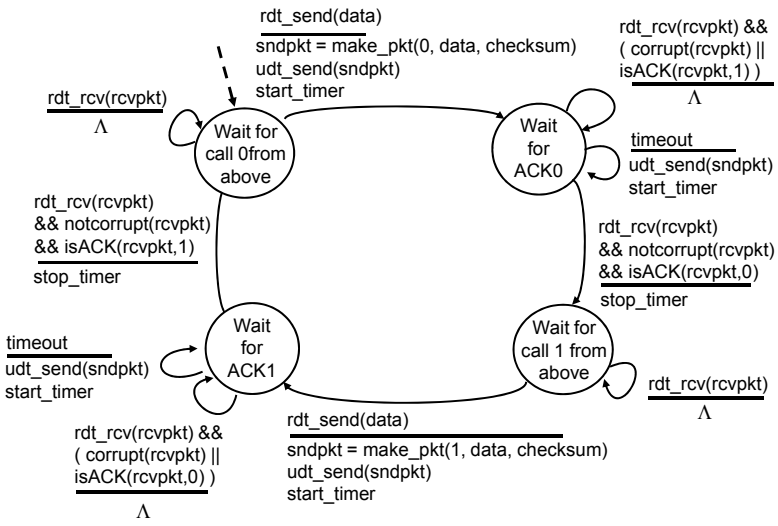
- checksum, seq. #, ACKs, retransmissions will be of help, but not enough

Approach: sender waits "reasonable" amount of time for ACK

- retransmits if no ACK received in this time
- if pkt (or ACK) just delayed (not lost):
  - retransmission will be duplicate, but use of seq. #'s already handles this
  - receiver must specify seq # of pkt being ACKed
- requires countdown timer

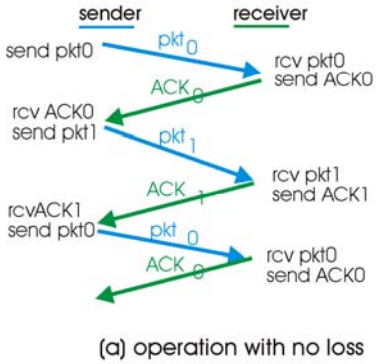
17/101

rdt3.0 sender

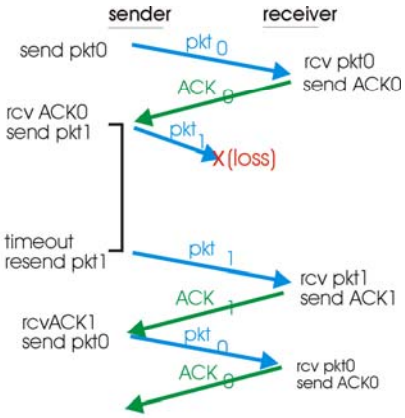


18/101

rdt3.0 in action



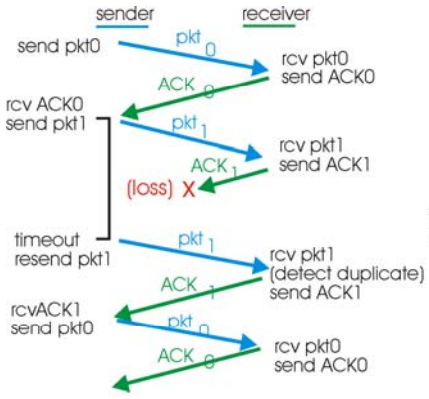
(a) operation with no loss



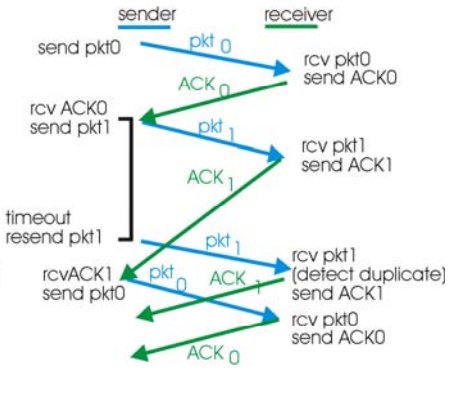
(b) lost packet

19/101

rdt3.0 in action



(c) lost ACK



(d) premature timeout

20/101

Ders konuları

- Reliable Data Transferin Prensipleri
  - Reliable data transferin performansı
  - Pipeline kullanan protokoller
  - Go-Back-N
  - Selective repeat

Performance of rdt3.0

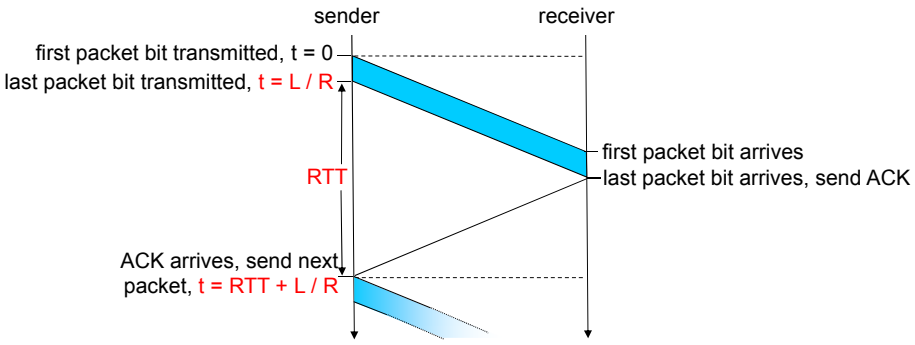
- rdt3.0 works, but performance stinks
- example: 1 Gbps link, 15 ms e-e prop. delay, 1KB packet:

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

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

- U_{sender}: utilization – fraction of time sender busy sending
- 1KB pkt every 30 msec -> 33kB/sec thruput over 1 Gbps link
- network protocol limits use of physical resources!

rdt3.0: stop-and-wait operation



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

23/101

Ders konuları

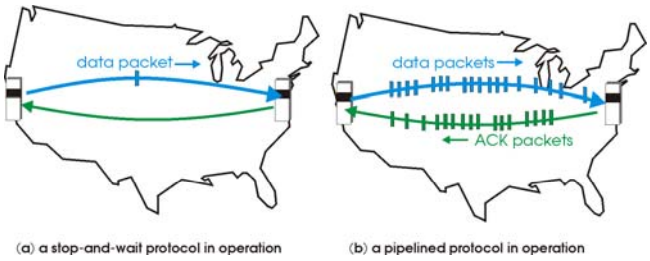
- Reliable Data Transferin Prensipleri
  - Reliable data transferin performansı
  - Pipeline kullanan protokoller
  - Go-Back-N
  - Selective repeat

24/101

Pipelined protocols

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

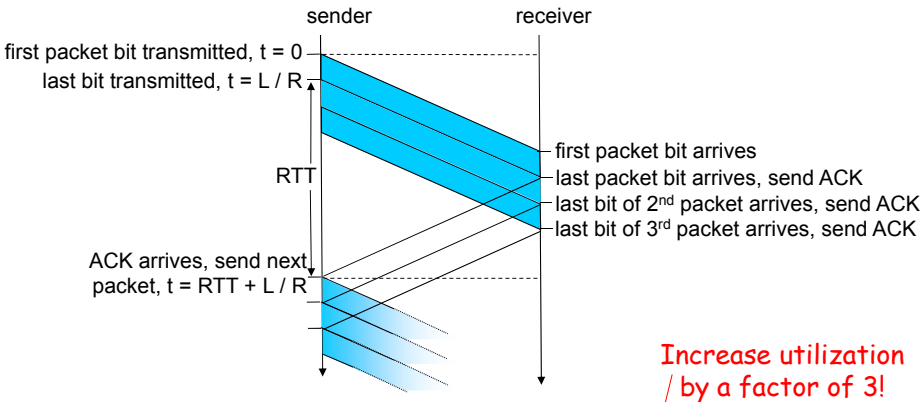
- range of sequence numbers must be increased
- buffering at sender and/or receiver



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

25/101

Pipelining: increased utilization



Increase utilization by a factor of 3!

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

26/101

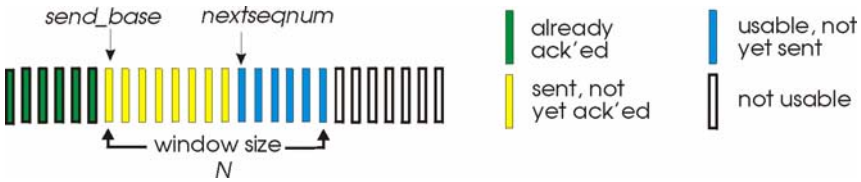
Ders konuları

- Reliable Data Transferin Prensipleri
  - Reliable data transferin performansı
  - Pipeline kullanan protokoller
  - Go-Back-N
  - Selective repeat

Go-Back-N

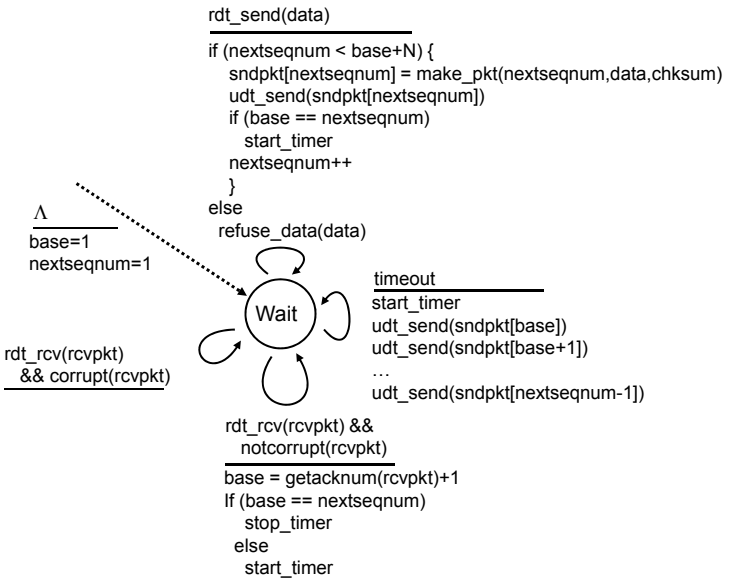
Sender:

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

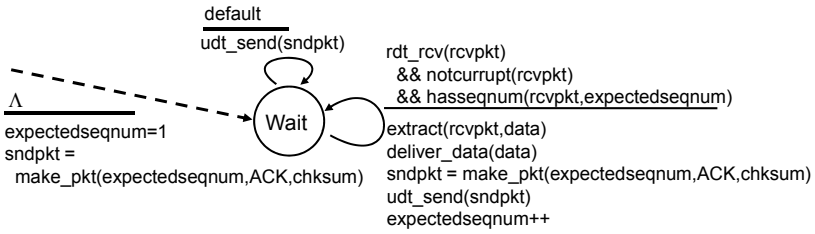


- ACK(n): ACKs all pkts up to, including seq # n - "cumulative ACK"
  - may deceive duplicate ACKs (see receiver)
- timer for each in-flight pkt
- *timeout(n):* retransmit pkt n and all higher seq # pkts in window

GBN: sender extended FSM



GBN: receiver extended FSM

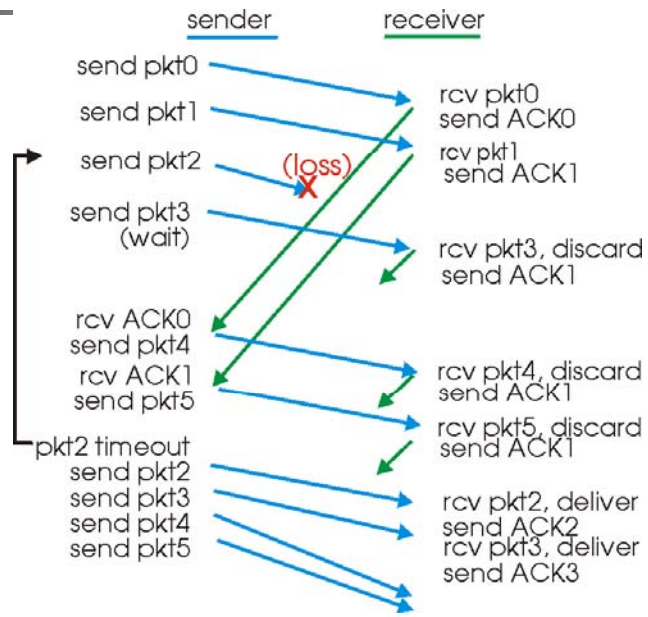


ACK-only: always send ACK for correctly-received pkt with highest *in-order* seq #

- may generate duplicate ACKs
- need only remember **`expectedseqnum`**

- out-of-order pkt:
  - discard (don't buffer) -> no receiver buffering!
  - Re-ACK pkt with highest in-order seq #

GBN in action



31/101

Ders konuları

- Reliable Data Transferin Prensipleri
  - Reliable data transferin performansı
  - Pipeline kullanan protokoller
  - Go-Back-N
  - Selective repeat

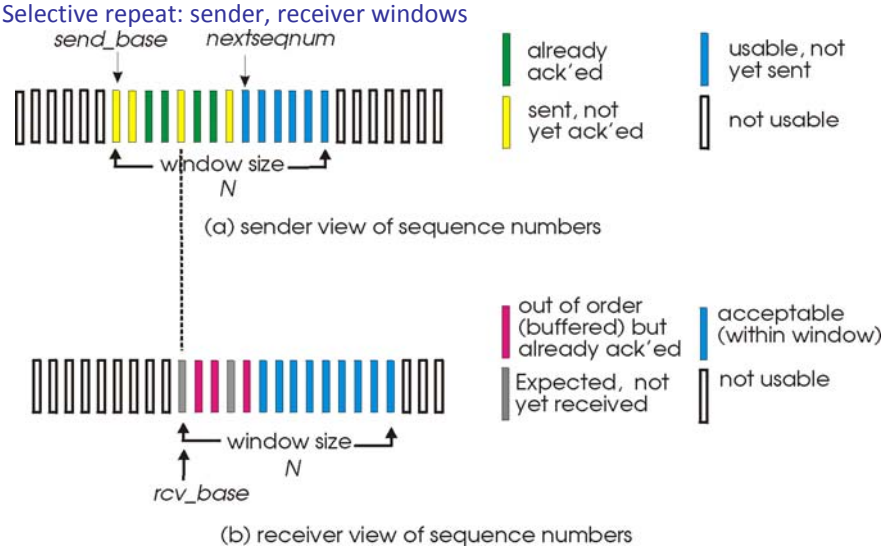
32/101

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

33/101

Selective repeat: sender, receiver windows



34/101

Selective repeat

sender

data from above :

- if next available seq # in window, send pkt

timeout(n):

- resend pkt n, restart timer

ACK(n) in [sendbase,sendbase+N]:

- mark pkt n as received
- if n smallest unACKed pkt, advance window base to next unACKed seq #

receiver

pkt n in [rcvbase, rcvbase+N-1]

- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

pkt n in [rcvbase-N,rcvbase-1]

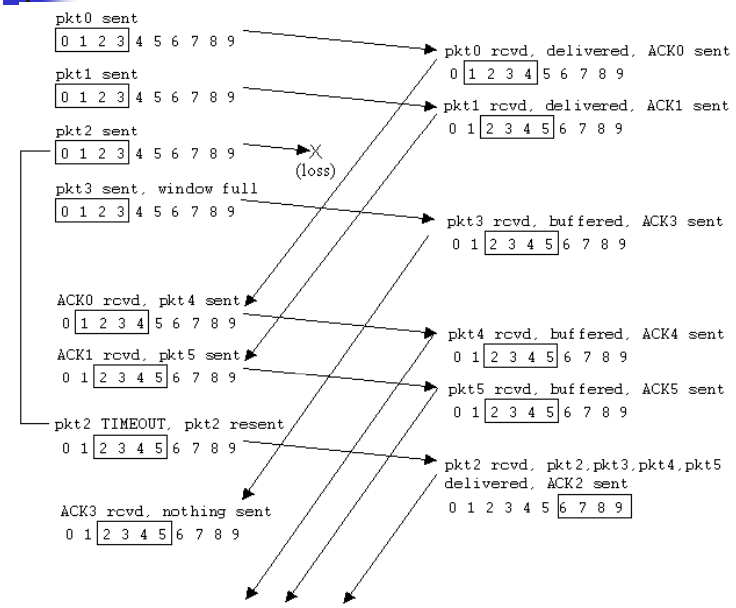
- ACK(n)

otherwise:

- ignore

35/101

Selective repeat in action



36/101

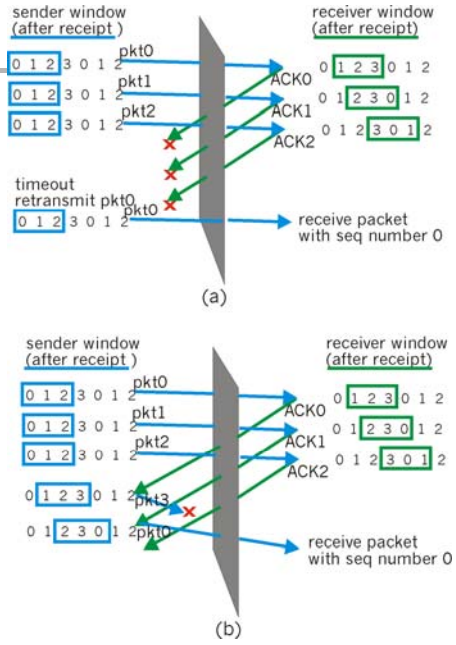
Selective repeat: dilemma

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?



37/101