Transport Layer
Part 7

Mark Allman
mallman@case.edu

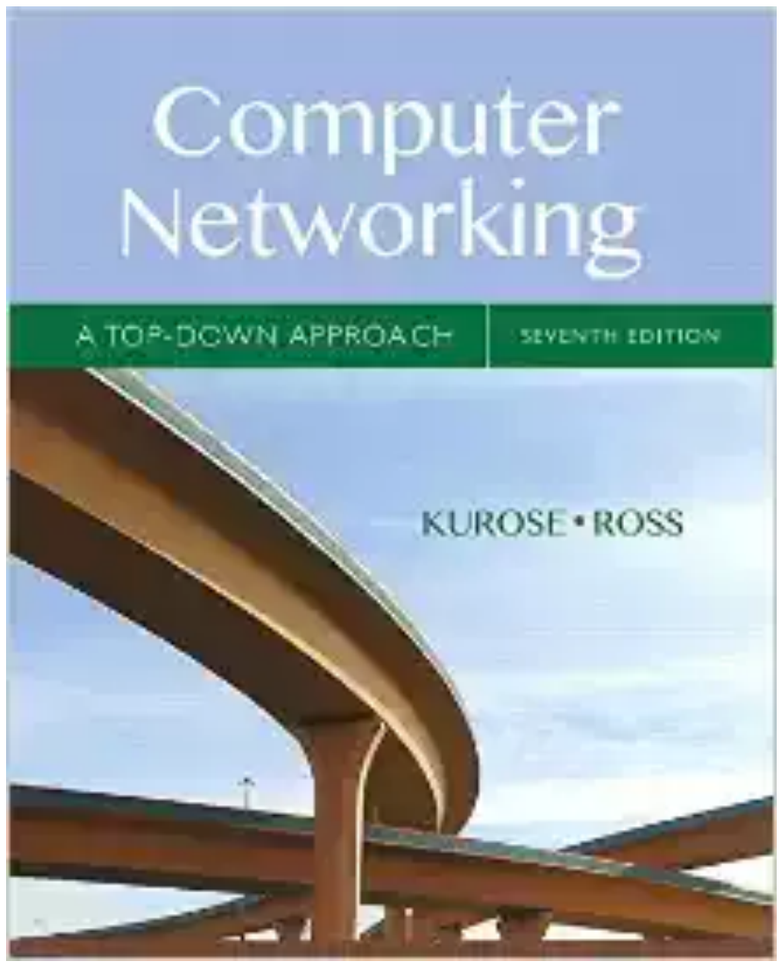
EECS 325/425
Fall 2018

"Wednesday just don't go …
Thursday goes too slow …"

These slides are more-or-less directly from the slide set developed by Jim Kurose and Keith Ross for their book "Computer Networking: A Top Down Approach, 5th edition".

The slides have been lightly adapted for Mark Allman's EECS 325/425 Computer Networks class at Case Western Reserve University.

Reading Along ...



- 3.5: Connection-oriented transport: TCP
  - reliable data transfer

Fast Retransmit

Fast Retransmit

- time-out period
  often relatively long:
  - long delay before
    resending lost packet

Fast Retransmit

- time-out period
  often relatively long:
  - long delay before
    resending lost packet
- detect lost segments
  via duplicate ACKs.
  - sender often sends
    many segments back-to-
    back
  - if segment is lost, there
    will likely be many
    duplicate ACKs.

Fast Retransmit

- time-out period often relatively long:
  - long delay before resending lost packet
- detect lost segments via duplicate ACKs.
  - sender often sends many segments back-to-back
  - if segment is lost, there will likely be many duplicate ACKs.
- if sender receives 3 ACKs for the same data, it supposes that segment after ACKed data was lost:
  - fast retransmit: resend segment before timer expires

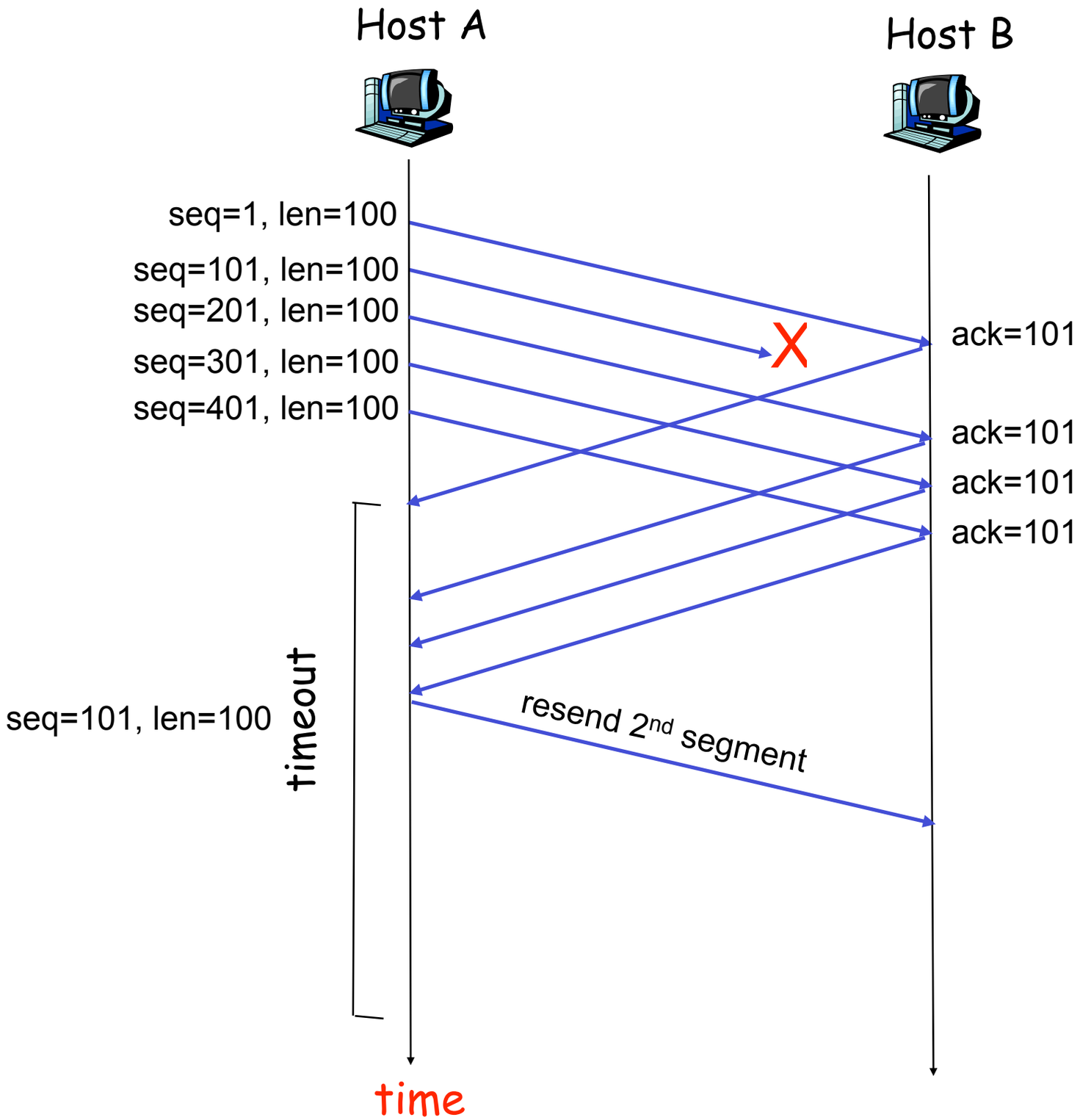


Figure 3.37 Resending a segment after triple duplicate ACK

Fast retransmit algorithm:

```
event: ACK received, with ACK field value of y
            if (y > SendBase) {
                SendBase = y
                if (there are currently not-yet-acknowledged segments)
                    start timer
              }
            else {
                increment count of dup ACKs received for y
                if (count of dup ACKs received for y = 3) {
                    resend segment with sequence number y
                 }
```

a duplicate ACK for already ACKed segment

fast retransmit

SR or GBN?

SR or GBN?

- Is this selective repeat?

SR or GBN?

- Is this selective repeat?

- To get something akin to selective repeat we use the Selective Acknowledgment (SACK) option
  - Indicates blocks of (begseq,endseq) sequence numbers that have arrived

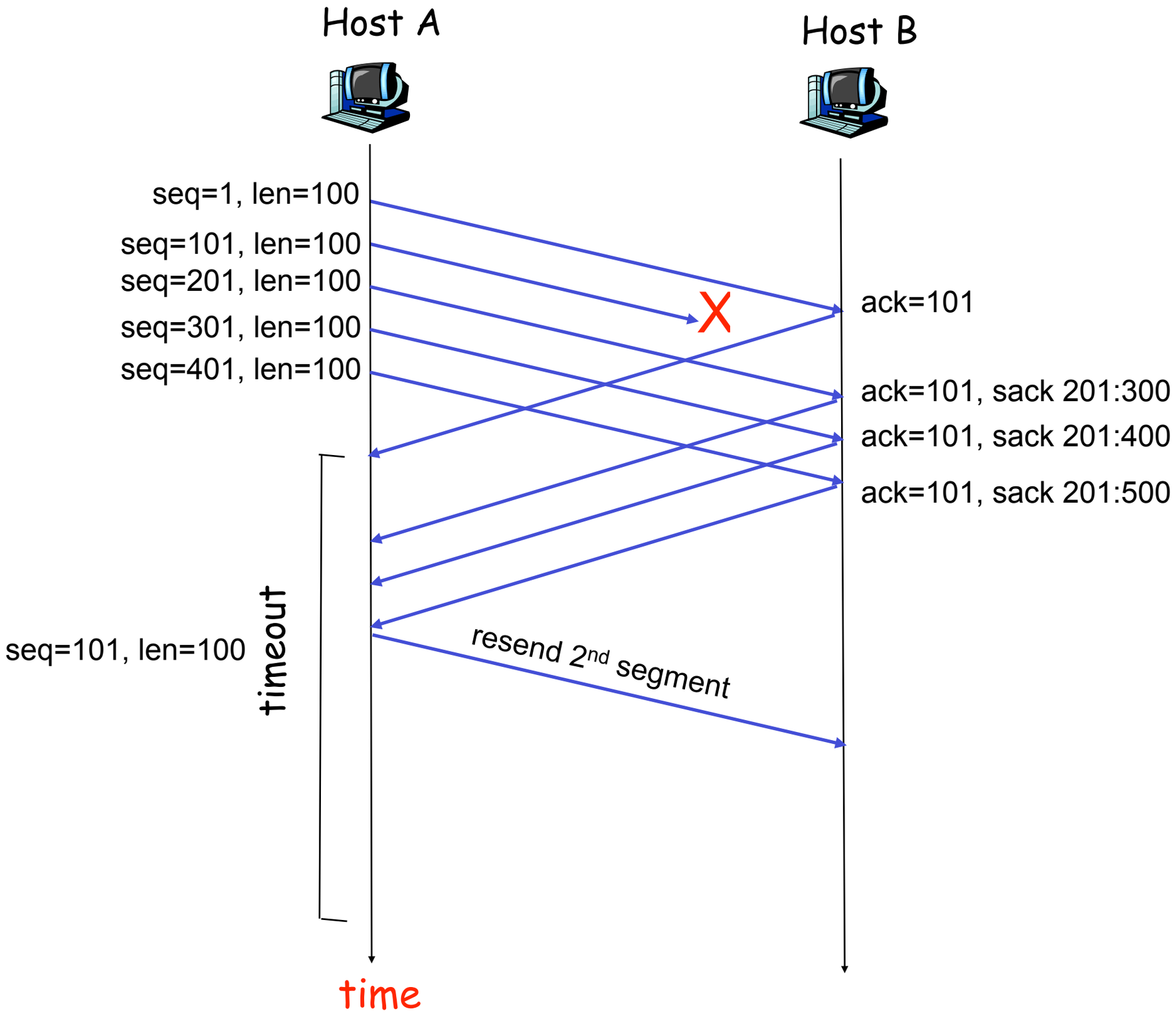


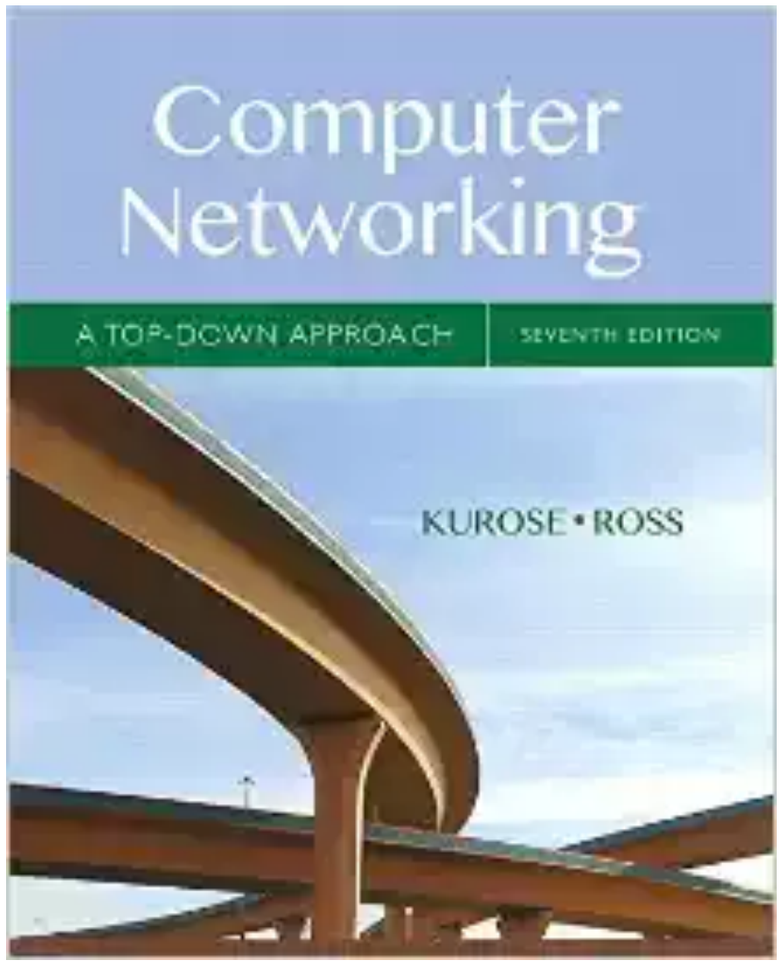
Figure 3.37 Resending a segment after triple duplicate ACK

SACK

- Fundamental change
  - cumulative ACK allows us to focus on only a single packet at a time
  - SACK allows us to concentrate on multiple packets at a time
    - i.e., the reality of a pipelined system

- SACK algorithms get complicated
  - see RFC 6675 for an example

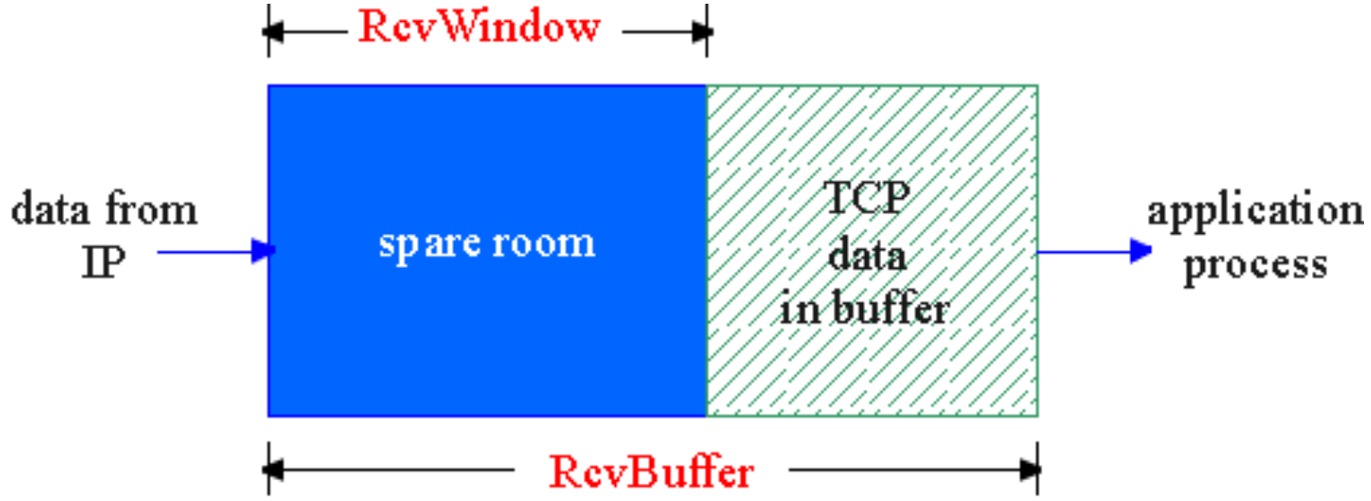
Reading Along ...



- 3.5: Connection-oriented transport: TCP
  - flow control

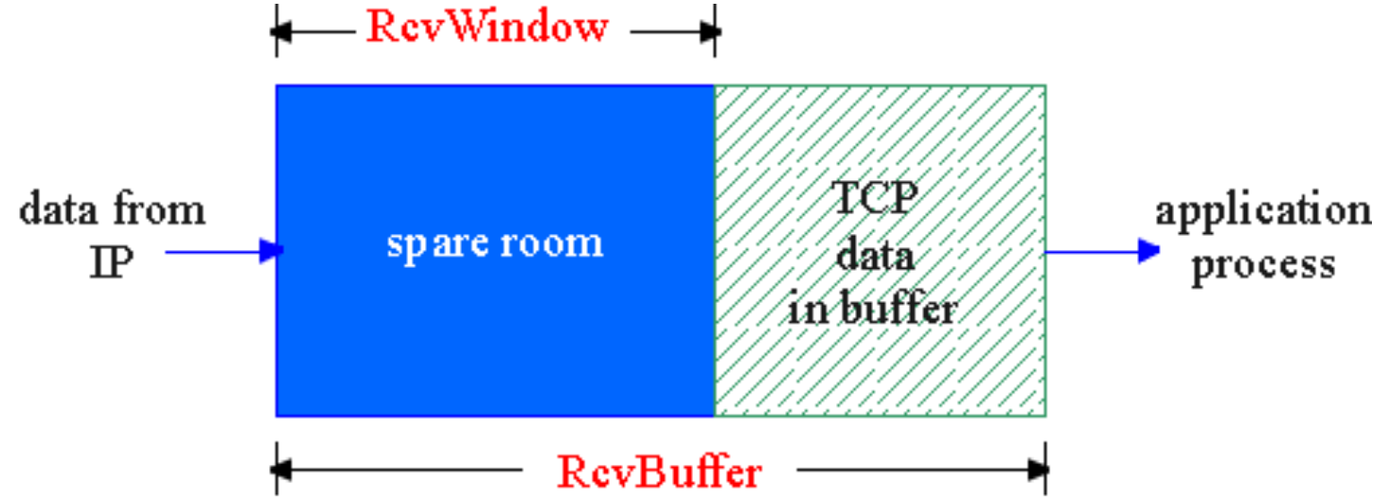
TCP Flow Control

- receive side of TCP connection has a receive buffer:



TCP Flow Control

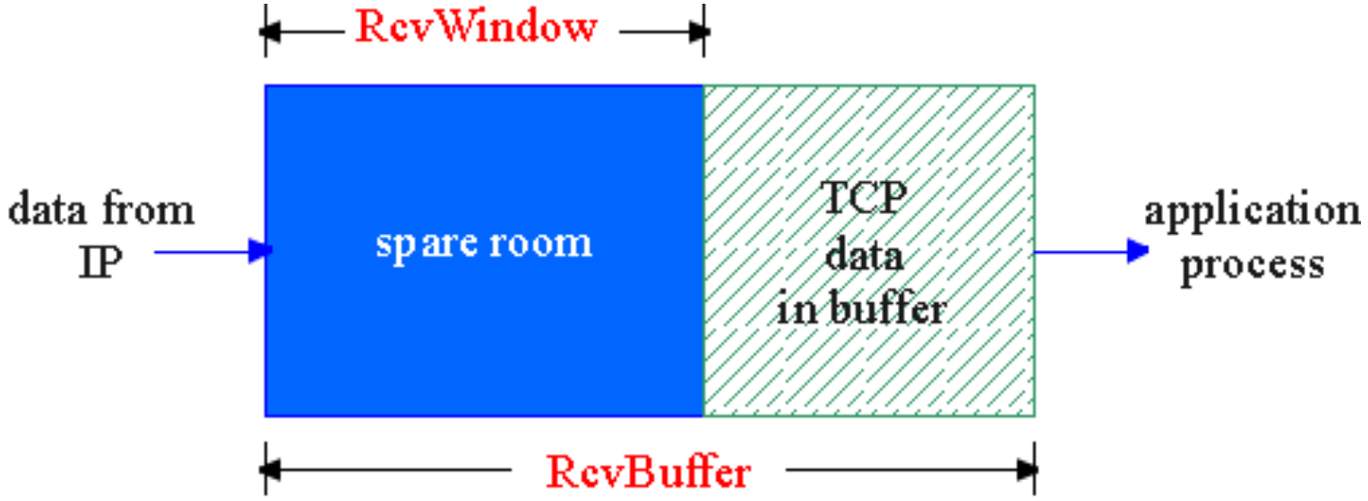
- receive side of TCP connection has a receive buffer:



- app process may be slow at reading from buffer

TCP Flow Control

- receive side of TCP connection has a receive buffer:



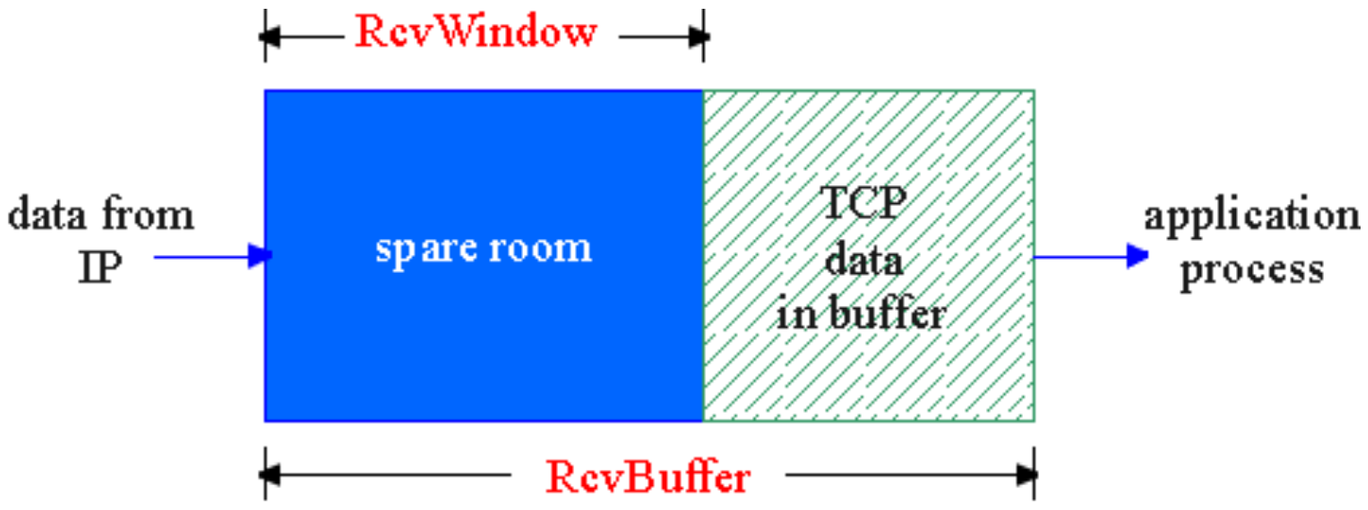
- app process may be slow at reading from buffer

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

TCP Flow control: how it works



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

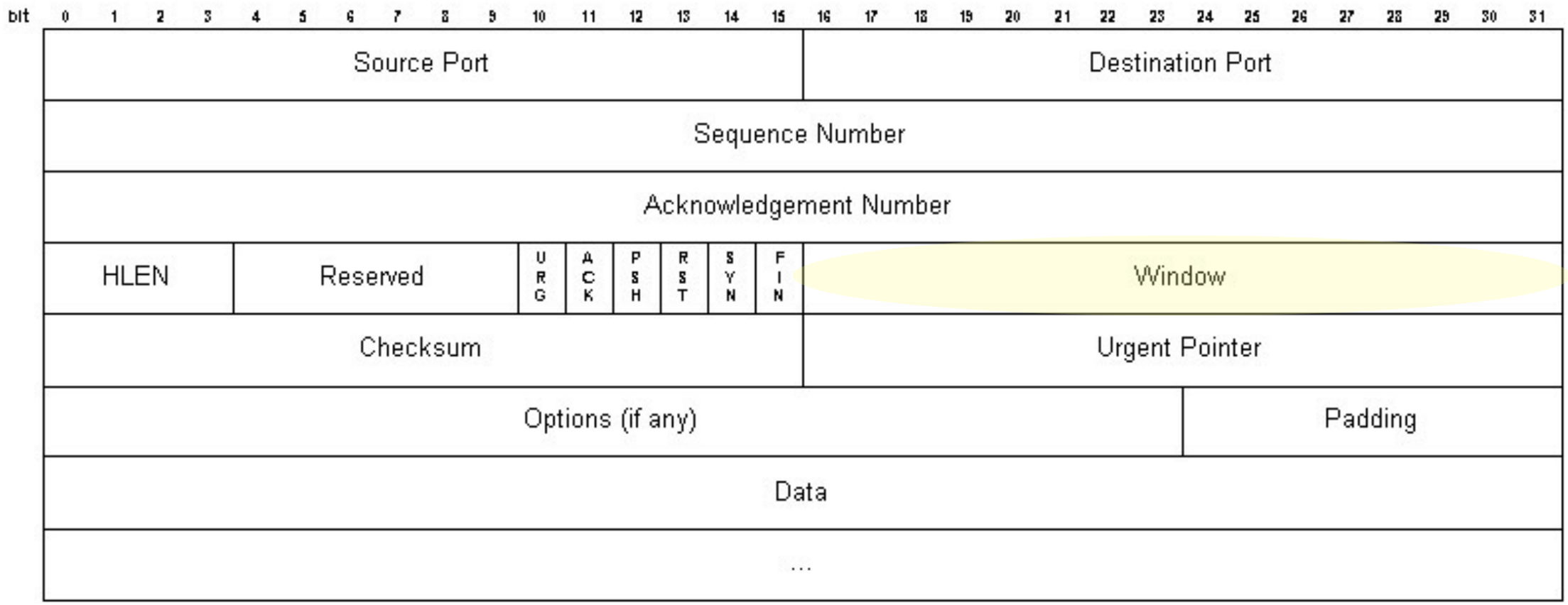
- spare room in buffer

= **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

Advertised Window



Advertised Window

Advertised Window

- 16 bits = 64KB

Advertised Window

- 16 bits = 64KB
- What are the implications of an advertised window limit of 64KB?

Advertised Window

- 16 bits = 64KB
- What are the implications of an advertised window limit of 64KB?
  - this is the max we can send at one time

Advertised Window

- 16 bits = 64KB
- What are the implications of an advertised window limit of 64KB?
  - this is the max we can send at one time
  - i.e., per RTT

Advertised Window

- 16 bits = 64KB
- What are the implications of an advertised window limit of 64KB?
  - this is the max we can send at one time
  - i.e., per RTT
- Is 64KB enough?

Advertised Window

- 16 bits = 64KB
- What are the implications of an advertised window limit of 64KB?
  - this is the max we can send at one time
  - i.e., per RTT
- Is 64KB enough?
  - No!
  - This is a place where TCP's original design has been eclipsed by reality

Advertised Window

- ReqAdvWindow = BW x RTT
- Or, BW = Window / RTT
- We need to be able to fill the so-called "bandwidth-delay product" of a network path to fully utilize the capacity of the path

Advertised Window

Advertised Window

Bandwidth	RTT Delay (sec)	Req. Window	TCP Max Utilization

Advertised Window

Bandwidth	RTT Delay (sec)	Req. Window	TCP Max Utilization
10Mbps	0.1	131 KB	50%

Advertised Window

Bandwidth	RTT Delay (sec)	Req. Window	TCP Max Utilization
10Mbps	0.1	131 KB	50%
100Mbps	0.001	13 KB	100%

Advertised Window

Bandwidth	RTT Delay (sec)	Req. Window	TCP Max Utilization
10Mbps	0.1	131 KB	50%
100Mbps	0.001	13 KB	100%
1Gbps	0.0001	13 KB	100%

Advertised Window

Bandwidth	RTT Delay (sec)	Req. Window	TCP Max Utilization
10Mbps	0.1	131 KB	50%
100Mbps	0.001	13 KB	100%
1Gbps	0.0001	13 KB	100%
1Gbps	0.002	268 KB	24%

Advertised Window

Bandwidth	RTT Delay (sec)	Req. Window	TCP Max Utilization
10Mbps	0.1	131 KB	50%
100Mbps	0.001	13 KB	100%
1Gbps	0.0001	13 KB	100%
1Gbps	0.002	268 KB	24%
622Mbps	0.5	41 MB	0.2%

Advertised Window

Bandwidth	RTT Delay (sec)	Req. Window	TCP Max Utilization
10Mbps	0.1	131 KB	50%
100Mbps	0.001	13 KB	100%
1Gbps	0.0001	13 KB	100%
1Gbps	0.002	268 KB	24%
622Mbps	0.5	41 MB	0.2%

Many common scenarios—and some uncommon ones!—require windows TCP cannot encode.

Window Scaling

Window Scaling

- Window Scaling introduced (RFC 1323)

Window Scaling

- Window Scaling introduced (RFC 1323)

- TCP option included in the SYN that offers a "window scale" …. i.e.,

Window Scaling

- Window Scaling introduced (RFC 1323)

- TCP option included in the SYN that offers a "window scale" …. i.e.,
  - the number of bits a sender will shift the true window right before sending

Window Scaling

- Window Scaling introduced (RFC 1323)

- TCP option included in the SYN that offers a "window scale" …. i.e.,
  - the number of bits a sender will shift the true window right before sending
  - the number of bits a receiver must shift the received window to the left before using

Window Scaling

- Window Scaling introduced (RFC 1323)

- TCP option included in the SYN that offers a "window scale" …. i.e.,
  - the number of bits a sender will shift the true window right before sending
  - the number of bits a receiver must shift the received window to the left before using
- Must be agreed to in the SYNACK

Window Scaling

Window Scaling

- Example with scale factor of 2

Window Scaling

- Example with scale factor of 2
- Sender:
  - true_window = 140 KB = 143,360 bytes = 10 00110000 00000000

Window Scaling

- Example with scale factor of 2
- Sender:
  - true_window = 140 KB = 143,360 bytes = 10 00110000 00000000
  - advwin = true_window >> 2 = 10001100 00000000 = 35840

Window Scaling

- Example with scale factor of 2
- Sender:
  - true_window = 140 KB = 143,360 bytes =
    10 00110000 00000000
  - advwin = true_window >> 2
    = 10001100 00000000 = 35840
- Receiver:
  - received_win = 10001100 00000000 = 35840

Window Scaling

- Example with scale factor of 2
- Sender:
  - true_window = 140 KB = 143,360 bytes =
    10 00110000 00000000
  - advwin = true_window >> 2
    = 10001100 00000000 = 35840
- Receiver:
  - received_win = 10001100 00000000 = 35840
  - true_window = received_win << 2
    = 10 00110000 00000000 = 140 KB

Window Scaling

Window Scaling

- Can shift by up to 14 bits
- I.e., windows can now grow to 2^30 or 1GB

Window Scaling

- Can shift by up to 14 bits
- I.e., windows can now grow to 2^30 or 1GB
- Accommodates most (all?) current network paths
  - some are currently pushing to support use of 15 bits of scaling and hence windows with 2^31 or 2GB