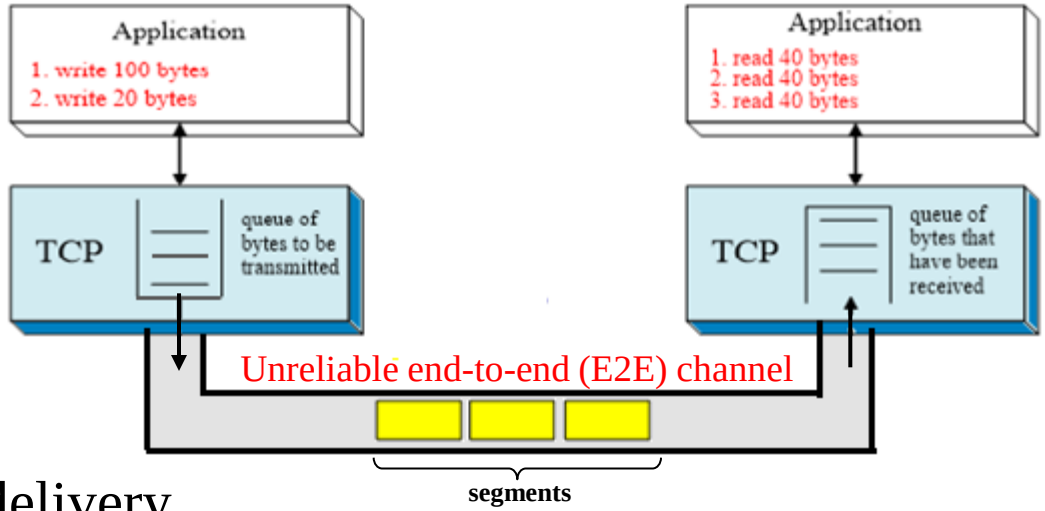
Chapter 15, 20

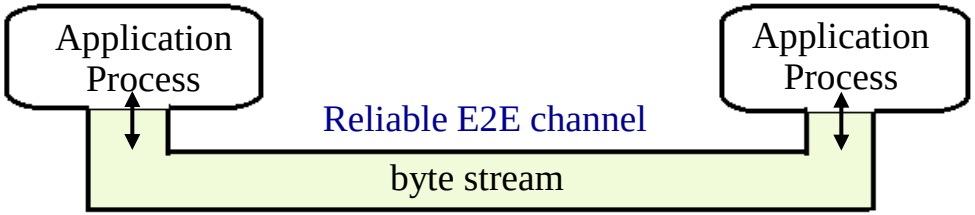
Transmission Control Protocol (TCP)

Reliable Byte Stream Service

- To the application layer, TCP handles data as a sequence of bytes
- To the lower layers, TCP handles data in blocks (segments).
  - Sender TCP: byte stream is broken up into segments

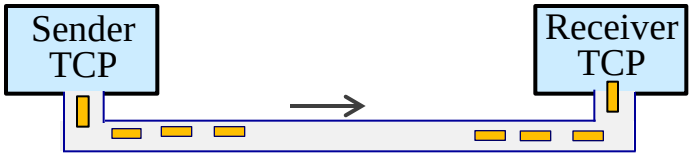


- Reliable delivery
  - Receiver TCP: sends ACKs for successfully received segments.
  - Sender TCP: If an ACK is not received in time, the segment is retransmitted

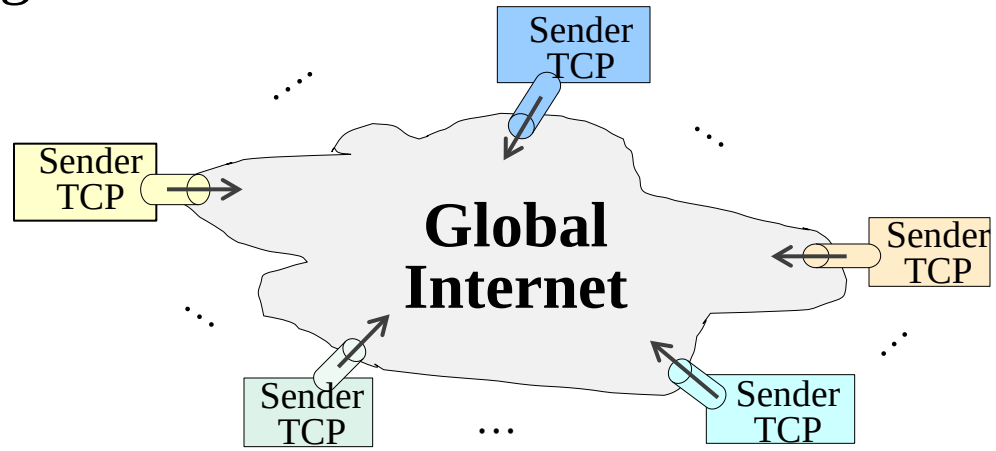


Connection-oriented reliable end-to-end transmission

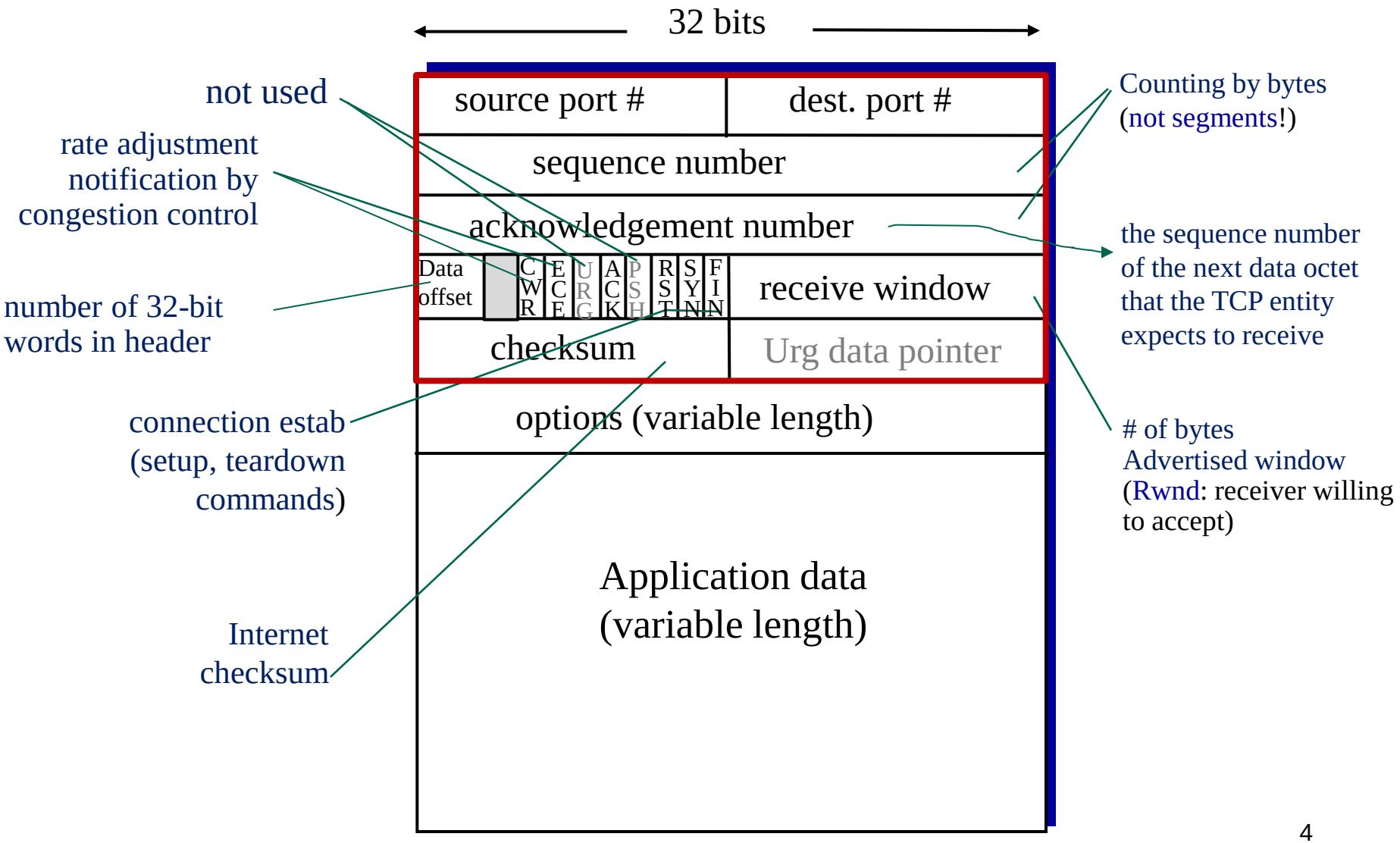
- Connection management
- Error control
- Flow control



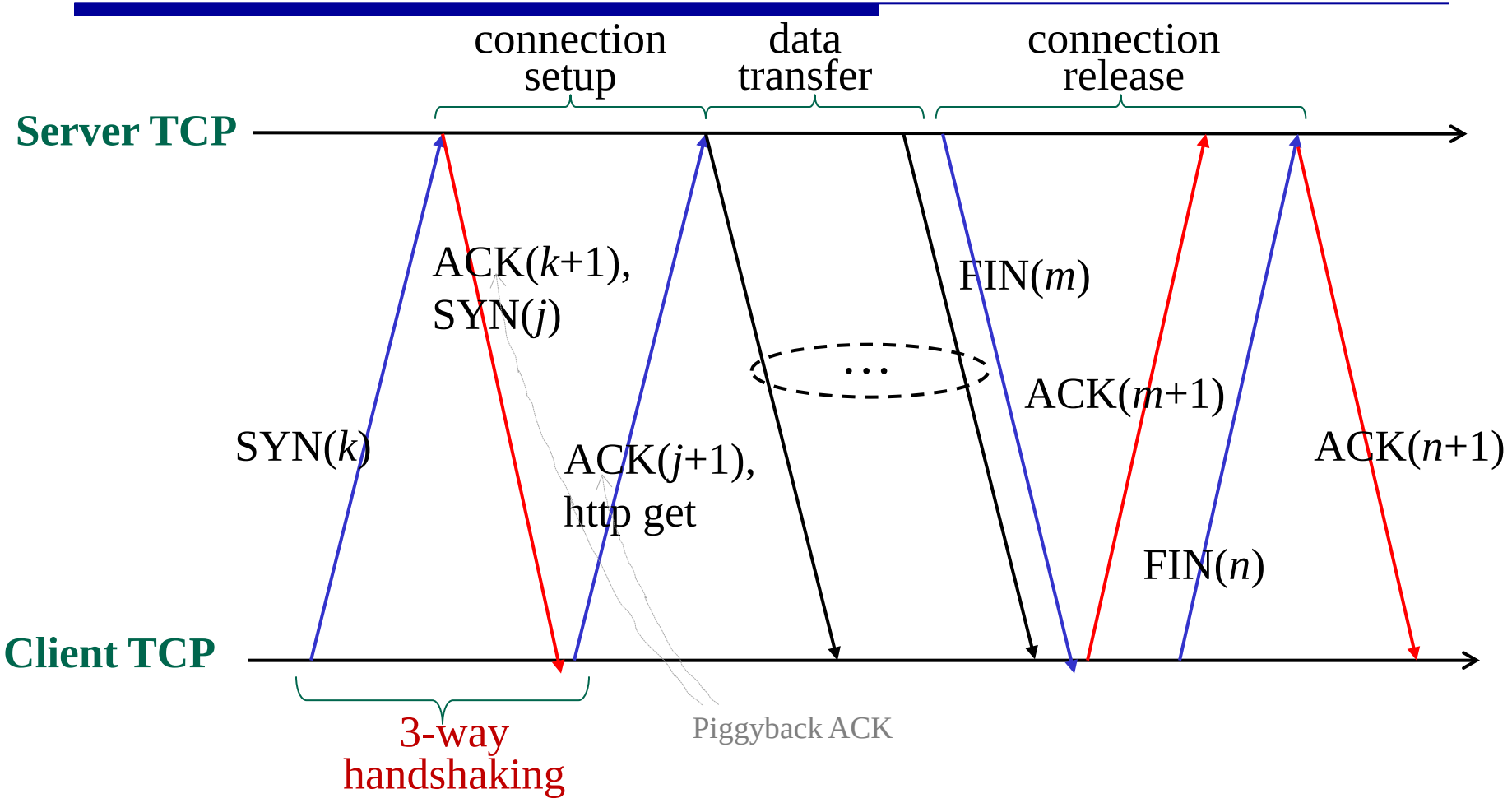
- Congestion control



TCP Header


32 bits
source port #
dest. port #
sequence number
acknowledgement number
Data offset
C W R
E C E
U R G
A C K
P S H
R S T
S Y N
F I N
receive window
checksum
Urg data pointer
options (variable length)
Application data (variable length)
not used
rate adjustment notification by congestion control
number of 32-bit words in header
connection estab (setup, teardown commands)
Internet checksum
Counting by bytes (not segments!)
the sequence number of the next data octet that the TCP entity expects to receive
# of bytes Advertised window (Rwnd: receiver willing to accept)

TCP Connection



To allow ECN-capable for explicit congestion control, in connection setup phase,
- TCP client: SYN, ECE and CWR flag setting
- TCP server: ACK, SYN, ECE flag setting

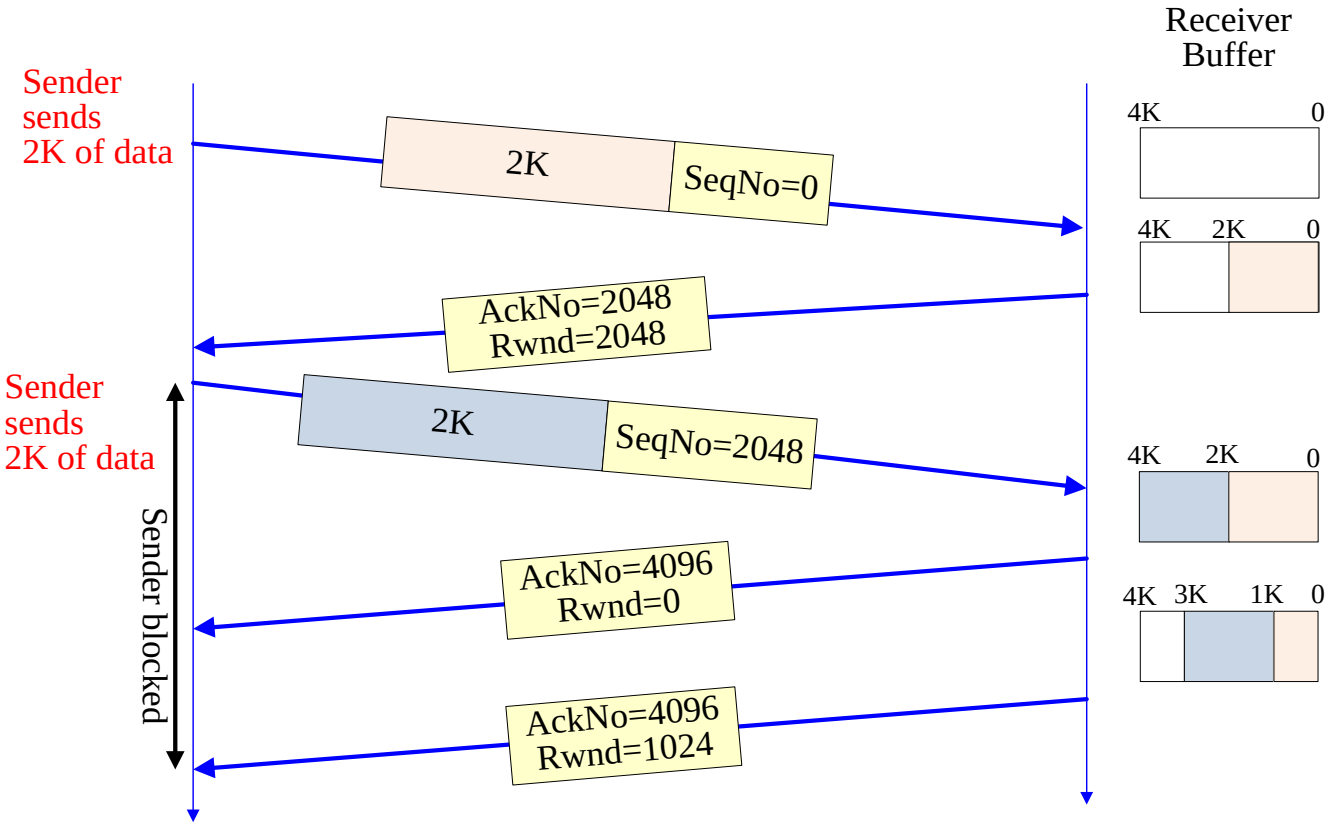
→ **IP header: ECN=01,10**

TCP Error/Flow/Congestion Control

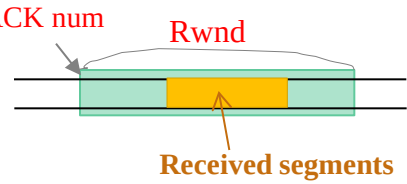
- The goal of each of the control mechanisms is different.
  → but, the implementation is combined
- Window
  - Advertised Window (Rwnd):
    - flow control between the TCP sender and TCP receiver
    - A amount of data granted by the receiver, starting from acknowledgement number
  - Congestion Window (Cwnd):
    - A window size for network congestion control
  - Allowed window (Sender's Window)
    - The amount of data that TCP is currently allowed to send without receiving further ACK's
    - Allowed window = min (Rwnd, Cwnd)

TCP Flow Control: Rwnd

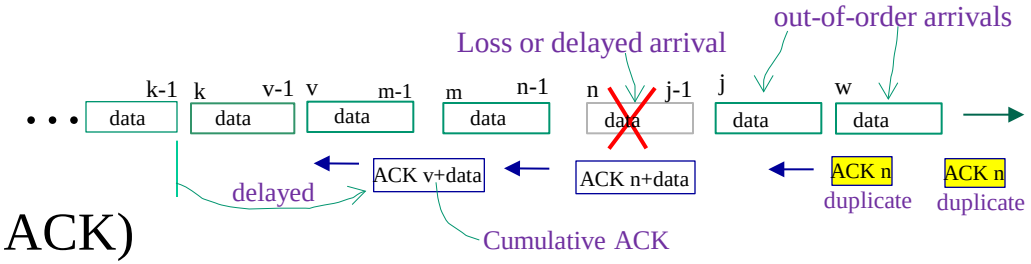
The receiver returns two parameters (Ack #, Rwnd) to the sender, meaning that "I am ready to receive new data with seqNo= AckNo, AckNo+1, …., AckNo+Rwnd-1"



TCP Error Control (1/2)



- Segment Acceptance Policy of Receiver
  - In-window: accept all segments within the receive window
  - In-order: Accept only segments that arrive in-order
  - Implementation-dependent

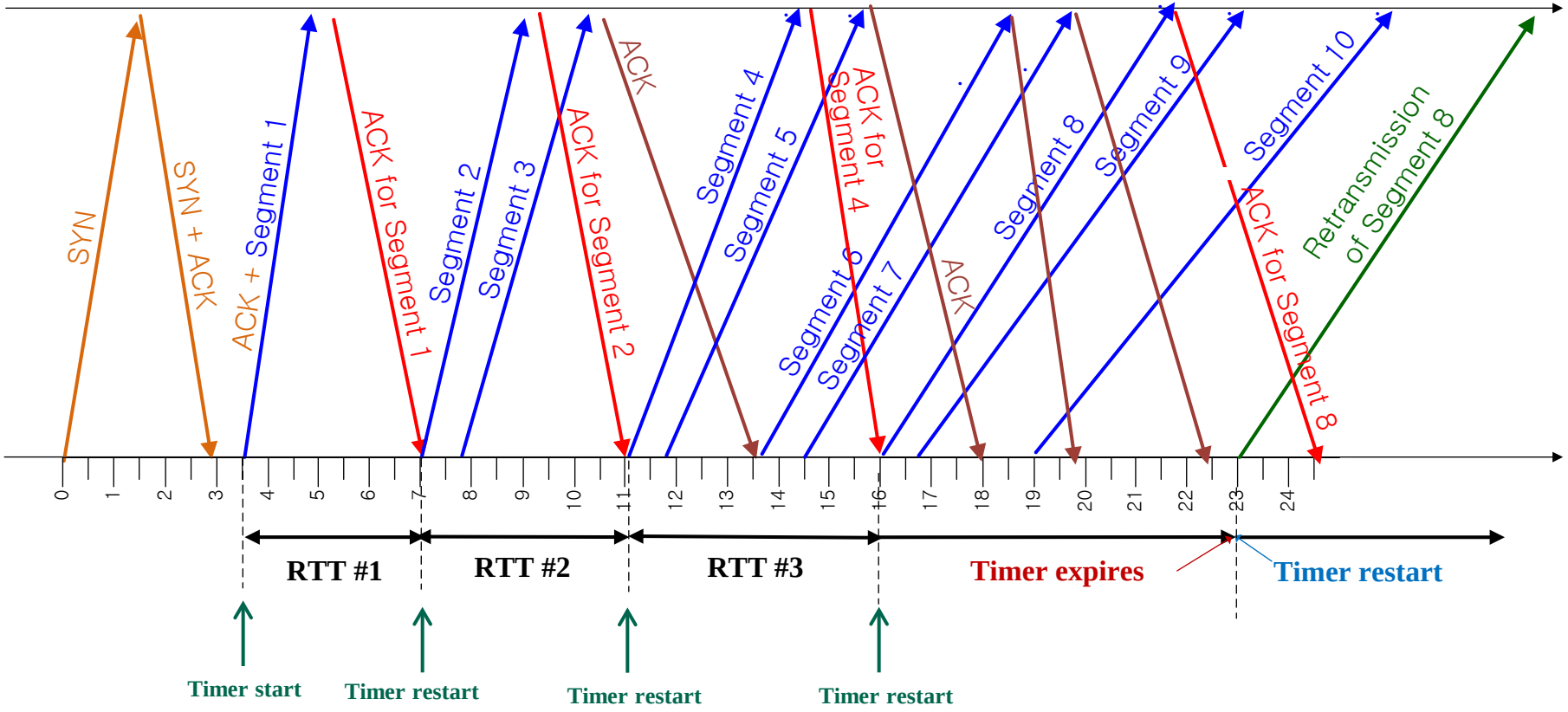


- Acknowledgement number
  - Delayed ACK (for piggyback ACK)
    - The receiver waits for the outbound segment for piggyback ack until timer expires. At time-out (200ms), send an empty segment with AckNo
    - But, at out-of-order segment arrival, immediately sends a duplicate ACK.
  - Cumulative ACK: the receiver has received up to (AckNo-1) in-order.
- Retransmit Policy of the Sender
  - First-only: retransmit only the segment at the queue front (Selective-repeat).
  - Batch: retransmit all segments in the queue (go-back-N)
  - Implementation-dependent

TCP Error Control (2/2)

- A TCP sender retransmits a segment when
  1. a timeout event occurs.
  2. three duplicate ACKs has been received (Fast Retransmission)
- Retransmission Timer
  - TCP sender maintains one retransmission timer for each connection
  - when the timer reaches the retransmission timeout (RTO) value, the sender retransmits the first segment that has not been acknowledged
- Retransmission timer is started
  1. when a packet with payload is transmitted and
     - timer is also not running or
     - when an ACK arrives that acknowledges new data (no duplicate ACK) which lets the timer start
  2. when a segment is retransmitted
- Retransmission timer is stopped when
  - all segments are acknowledged

Round Trip Time (1/2)



Retransmission Timer

- How to set the TCP timeout value (RTO) ?
  - Based on round trip time (RTT), but should be longer than RTT
    - consider that RTT varies.
  - too short: premature timeout, unnecessary retransmissions
  - too long: slow reaction to segment loss
- How to estimate RTT?
  - SampleRTT:
    - measured time from a segment transmission until its ACK receipt
    - ignore retransmissions
  - "Smoother" change of estimated RTT is desirable
    - Average of several recent measurements, not just current sampleRTT
    - Greater weight for more recent measurement

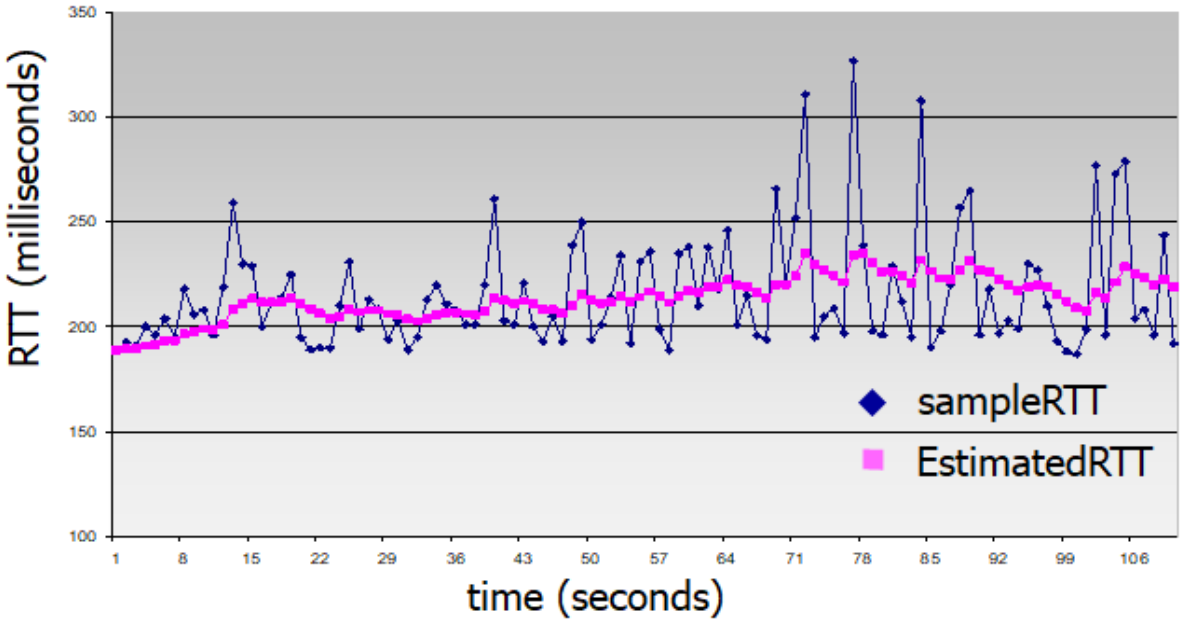
Moving average

Round Trip Time (2/2)

$$\text{EstimatedRTT} = (1- \alpha) \times \text{EstimatedRTT} + \alpha \times \text{SampleRTT}$$

$$\alpha s_n + (1-\alpha)\alpha s_{n-1} + (1-\alpha)^2 \alpha s_{n-2} + \cdots + (1-\alpha)^n \alpha s_0$$

- exponential weighted moving average
- influence of past sample decreases exponentially fast
- typical value: $\alpha = 0.125$



Retransmission Timeout (RTO)

- RTO: EstimatedRTT + "safety margin"
- Safety margin
  - SampleRTT deviation from EstimatedRTT:
  - $DevRTT = (1-\beta) \times DevRTT + \beta \times |SampleRTT - EstimatedRTT|$
  - typically, $\beta = 0.25$

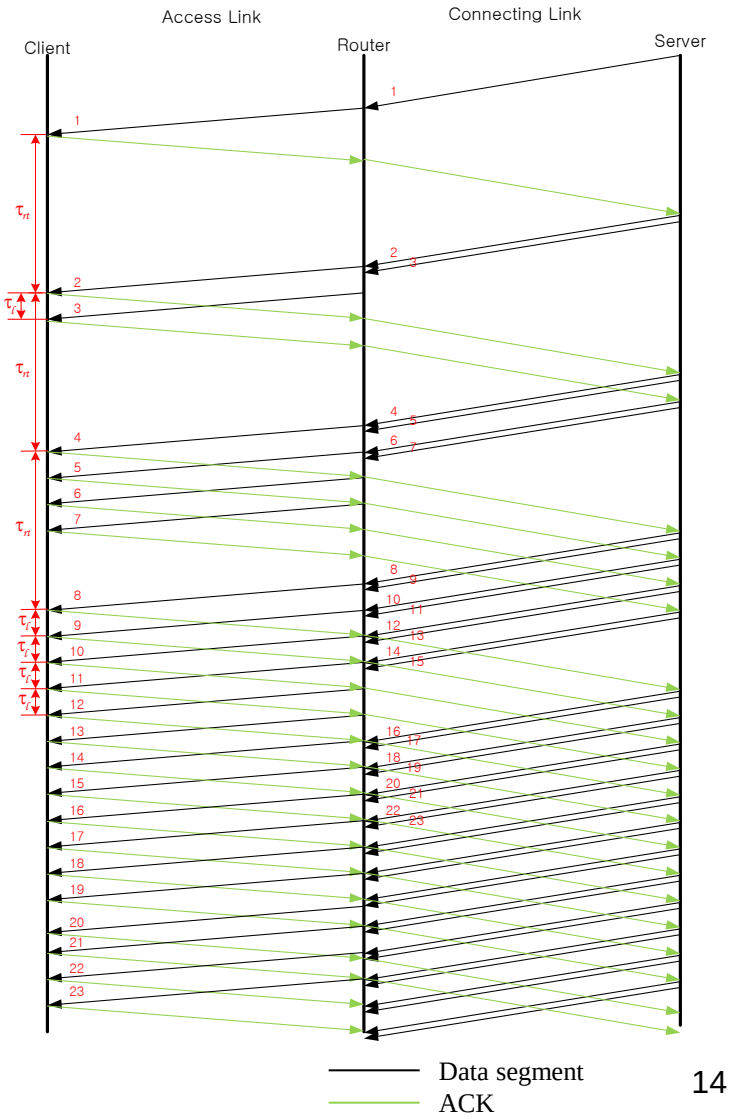
safety margin

$$RTO = EstimatedRTT + 4 \times DevRTT$$

TCP Congestion Control (1/3)

- Slow Start
  - Congestion window (Cwnd) = 1
  - Cwnd is increased by 1 for each ACK until Cwnd reaches to a slow start threshold (ssthresh)

    :Multiplicative increase
  - At initialization
- Additive Increase
  - For Cwnd>ssthresh, increase Cwnd by 1 for each round-trip time



For simplicity, let's assume
- Rwnd and Cwnd represent the number of segments (not bytes)
- Rwnd >> Cwnd

TCP Congestion Control (2/3)

- When a timeout occurs (Congestion Avoidance)
  1. Set a slow start threshold (ssthresh) to half the current congestion window : ssthresh ← Cwnd/2
  2. Set Cwnd =1
  3. Performs the Slow Start process (Multiplicative Increase) until Cwnd = ssthresh
  4. For Cwnd>ssthresh, Additive Increase

TCP Congestion Control (3/3)

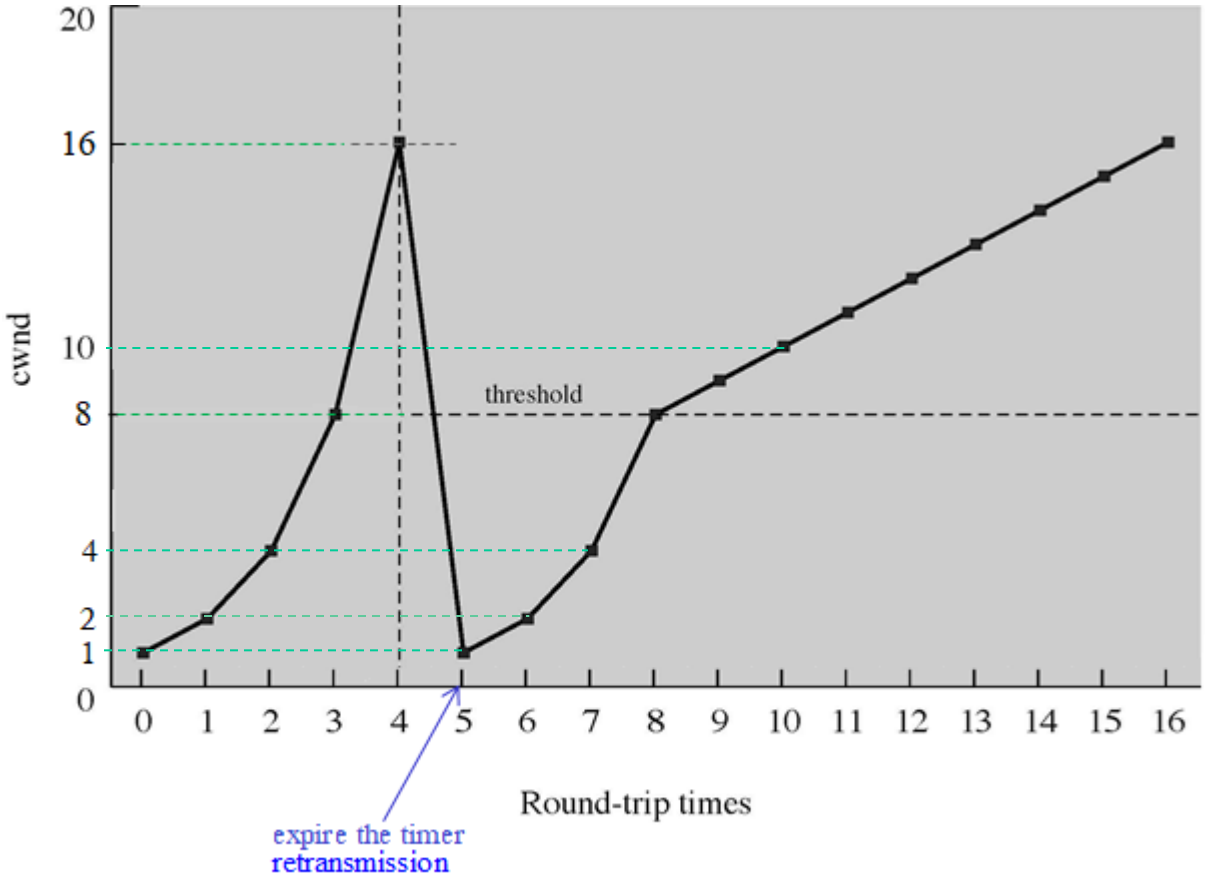
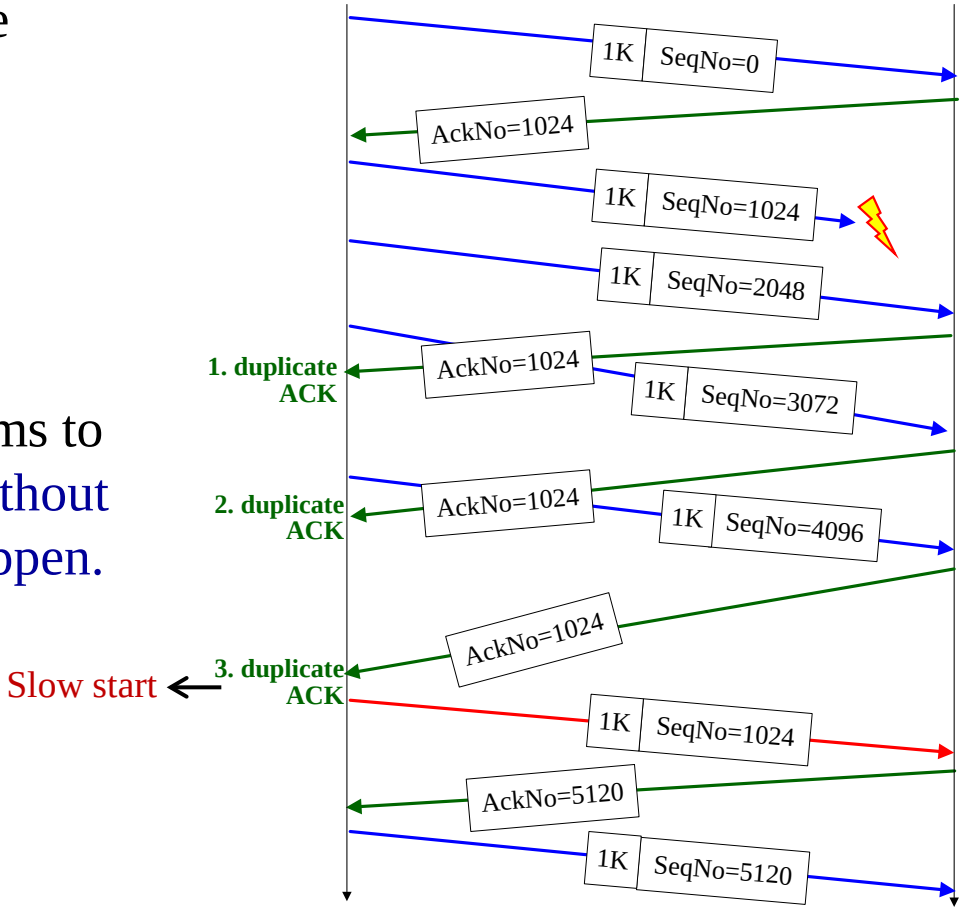


Figure 20.13 Illustration of Slow Start and Congestion Avoidance

Fast Retransmission

- If three duplicate ACKs are received, the TCP sender believes that a segment has been lost.
- Then, TCP performs a retransmission of what seems to be the missing segment, without waiting for a timeout to happen.



Fast Recovery

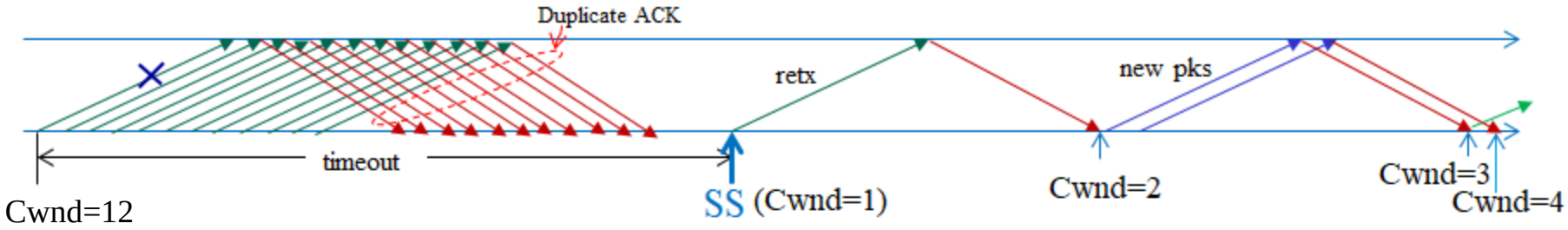
- Modification to fast retransmission
- If the sender receives *n* duplicate ACKs (typically *n*=3),
  - retransmit the loss segment immediately
  - ssthresh = Cwnd/2

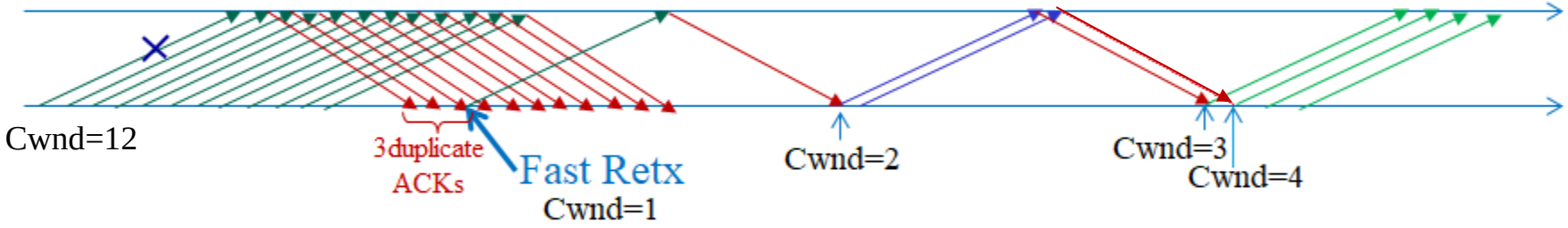
  } Fast retransmission

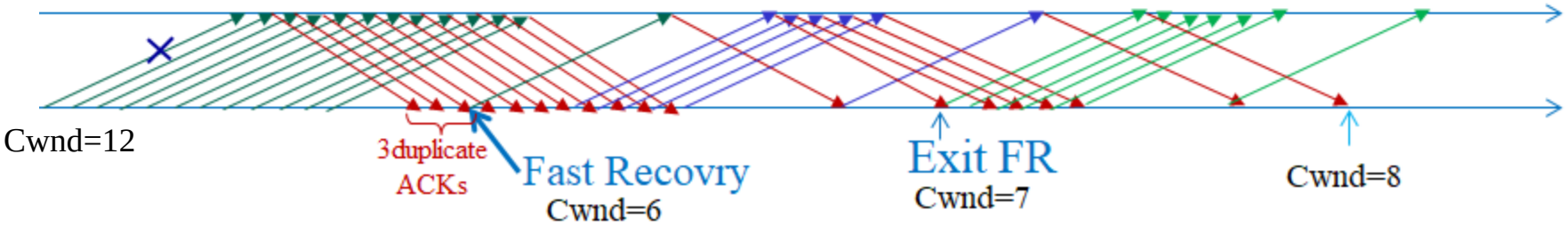
  - Cwnd = ssthresh
  - *ndup*
    - If (*n*+1) duplicate ACK's are received (i.e., when receiving one additional duplicate ACK after retransmission)

      ⇒ *ndup* is set to (*n*+1) and increments every duplicate ACK received.
    - If *ndup* reaches half the old_cwnd

      ⇒ the sender transmits new packets for each additional duplicate ACK.
  - Upon receipt of an ACK for new data, the sender exits Fast Recovery by setting *ndup* to 0 and incrementing Cwnd.

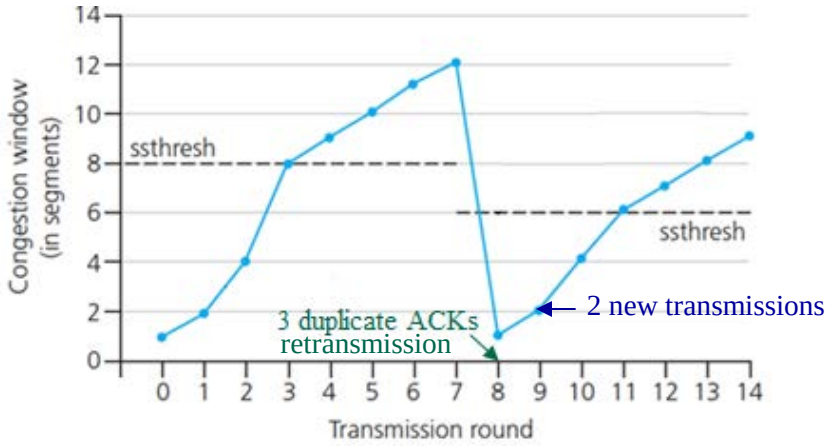

Duplicate ACK
retx
new pks
timeout
Cwnd=12
SS (Cwnd=1)
Cwnd=2
Cwnd=3
Cwnd=4


Cwnd=12
3duplicate
ACKs
Fast Retx
Cwnd=1
Cwnd=2
Cwnd=3
Cwnd=4

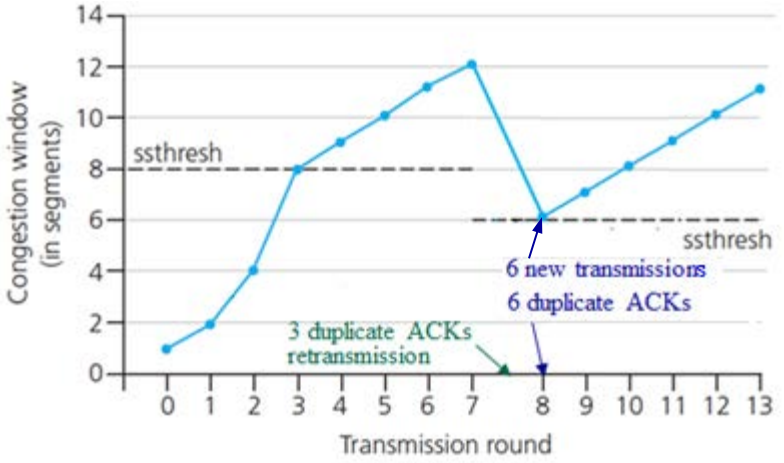

Cwnd=12
3duplicate
ACKs
Fast Recovry
Cwnd=6
Exit FR
Cwnd=7
Cwnd=8

Tahoe TCP, Reno TCP

- Tahoe TCP
  - Slow Start
  - Congestion Avoidance
  - Fast retransmit

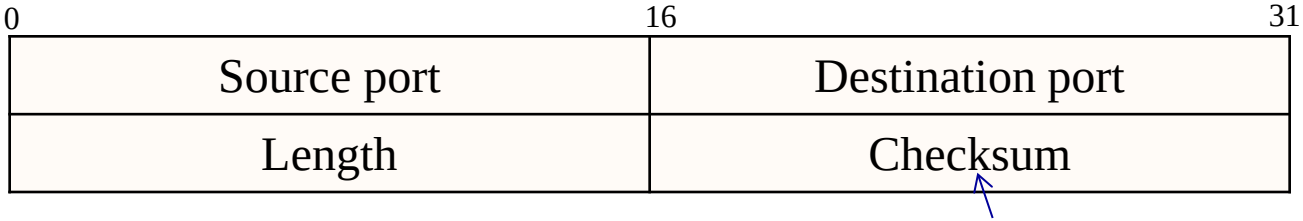


- Reno TCP
  - Slow Start
  - Congestion Avoidance
  - Fast Recovery



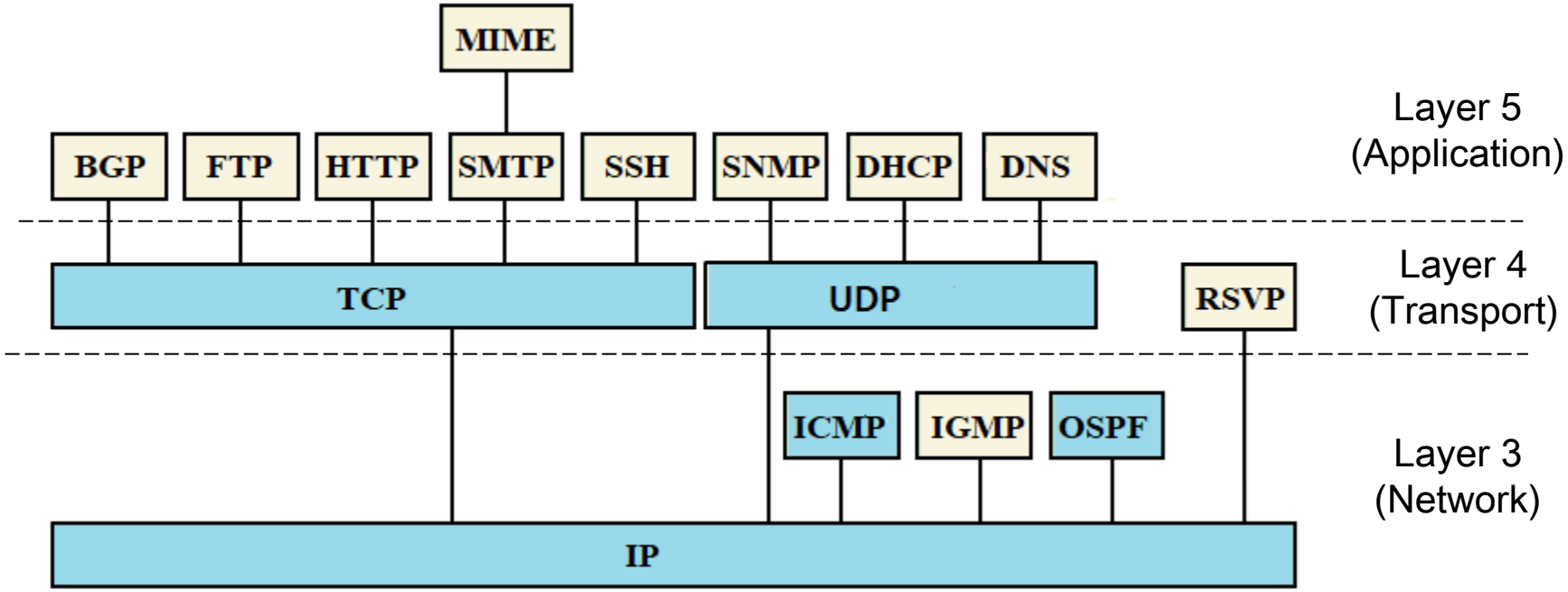
UDP (User Datagram Protocol)

- Transport layer protocol
- Connectionless service for applications
- Unreliable service
  - Deliver and duplicate protection are not guaranteed
  - It merely adds a port addressing capability to IP
- UDP header (8 bytes)



Optional field
- If not used, set to zero
- If error is detected, the segment is discarded and no further action is taken

Some Application Protocols in TCP/IP Protocol Suit



BGP = Border Gateway Protocol
FTP = File Transfer Protocol
HTTP = Hypertext Transfer Protocol
ICMP = Internet Control Message Protocol
IGMP = Internet Group Management Protocol
IP = Internet Protocol
MIME = Multipurpose Internet Mail Extension
OSPF = Open Shortest Path First
RSVP = Resource ReSerVation Protocol
SMTP = Simple Mail Transfer Protocol
SNMP = Simple Network Management Protocol
SSH = Secure Shell
TCP = Transmission Control Protocol
UDP = User Datagram Protocol

학기 마무리 잘 하고

여름방학

즐겁게 보내세요!