COMP2001J Computer Networks

Lecture 12 – Transport Layer
(Flow/Congestion Control)

Dr. Shen WANG (王燊)

shen.wang@ucd.ie

北京工业大学北京－都柏林国际学院
BEIJING–DUBLIN INTERNATIONAL COLLEGE AT BJUT

Warm-up Questions

- Compare the implementation of flow control in data link layer and transport layer.

- What is congestion control? How TCP implement it?

Outline

- Reliable Transmission
  - Sliding window
  - Acknowledgement
  - Retransmission

- TCP Flow Control

- TCP Congestion Control

Reliable Transmission

- Once the TCP connection is established, it implies the current network condition is good enough to support reliable transmission.

- TCP ensures the end-to-end transmission is reliable, which implies the received byte stream does not have:

1. Out-of-order segments

2. Segment loss

3. Duplicated segments

Recap – Data Link Layer

- Review Lecture 4 – Data Link Layer-b

- How data link layer implements reliable communication?
  - Acknowledgement
  - Retransmission

- How data link layer implements flow control?
  - Stop-and-Wait
  - Sliding-Window
    - Go-back-N
    - Selective Repeat
  - ARQ extension

Acknowledgement

- Sequence Number
  - Sequence Number is used to uniquely identify each byte in TCP byte stream.
  - It sequences a TCP byte stream by giving a continuously increasing number for each byte.
  - Thus, it can also be used to ensure if a byte stream is in order.

Acknowledgement

- Acknowledgement Number
  - Once a byte/segment is reliably received, the receiver needs to send a **acknowledgement** TCP segment to the sender indicates that this byte/segment is correctly received.
  - The acknowledgement number is incremented by one at the sequence number of the received byte.
  - Thus, the acknowledgement can not only confirms the byte is received, but also confirms the byte is received in a correct order.

Retransmission

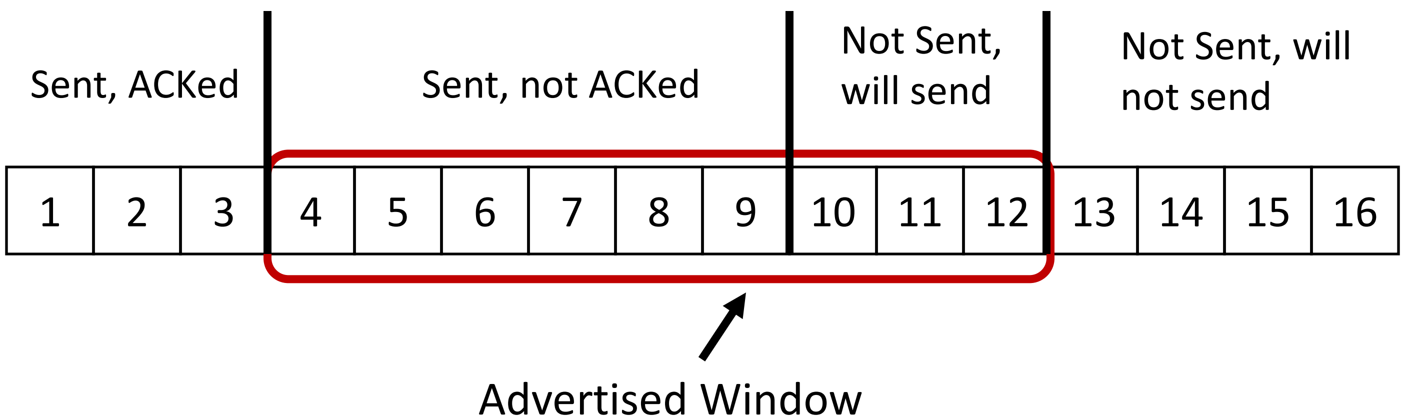
- The sender sets a **timer** for each TCP segment it sends out.

- If a timeout occurs, the sender still not yet receive its corresponding acknowledgement, then the sender considers the segment is lost and then **retransmit** this segment.

Sliding Window

- Why flow control?
  - To adjust the rate of sending and receiving
  - Using sliding window
- LastByteAcked:3
- LastByteSent:9
- LastByteAbleToSend:12

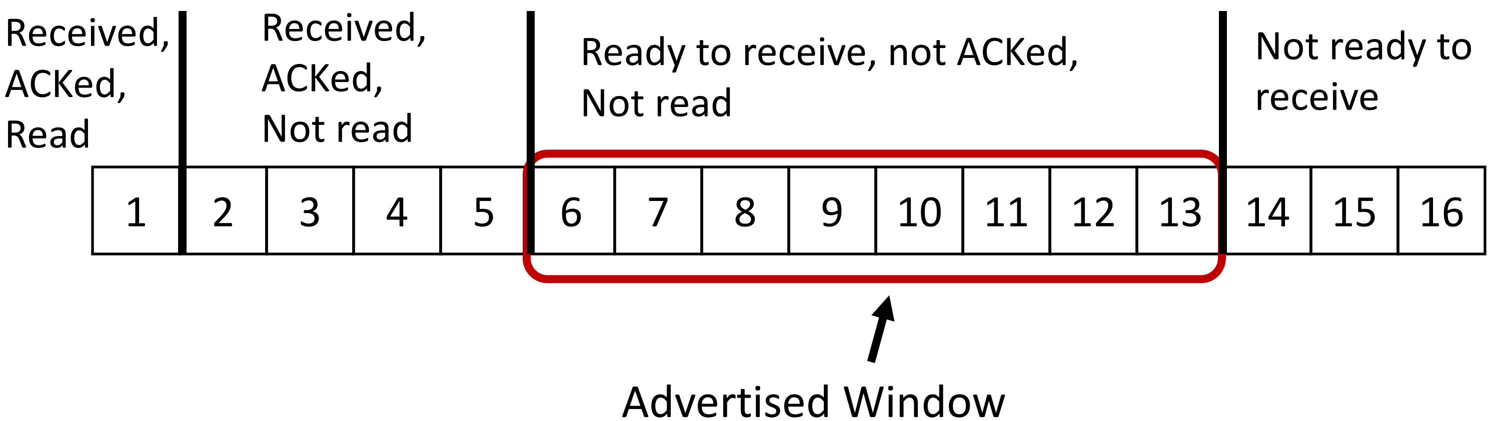
The data structure the SENDER maintains



Sliding Window

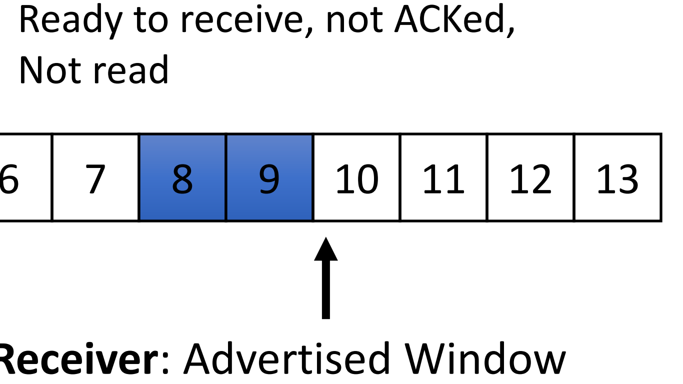
- LastByteRead: 1
- NextByteExpected:6
- MaxRcvBuffer: 13 – 2 + 1 = 12

The data structure the RECEIVER maintains



Example

- Both sender and receiver agreed: 1, 2, 3
- Receiver ACKed, but sender not received: 4, 5
  - Maybe on the way, or maybe lost
- Sender sent: 6, 7, 8, 9, only 8, 9 arrives, but Receiver cannot ACK them



Example

- Suppose:
  - ACK for 4 arrives to the sender
  - but ACK for 5 is lost
  - data 6 and 7 are lost, too.

- Then:
  - If timeout for 5, 6, 7, then retransmit
  - Receiver finds that 5 already received before, then drops
  - 6 arrives to receiver, then receiver sends ACK to sender
  - Unfortunately, data 7 is lost again

Example

- If timeout for the same segment occurs twice, TCP doubles the timer duration for expiration.

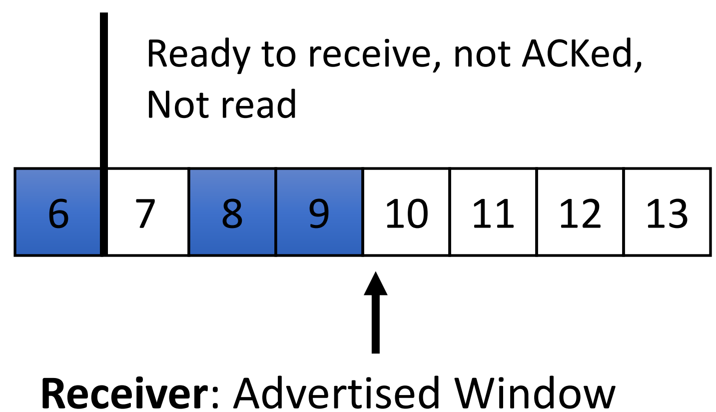
- If timeout occurs again for the third time, it indicates that the network condition is bad, then the sender should cancel this transmission.

Fast Retransmission

- When the sequence number of the segment received by the receiver is **greater** than the receiver's expectation
- Then, the receiver immediately sends 3 repeat ACKs to the sender
- Thus, when 3 redundant ACKs received by the sender, it can retransmit before the timeout (**fast** retransmission)

Example

- If the receiver finds that, 6, 8, 9 already received, only 7 does not arrive, then it sends 3 ACKs for 6, expecting the next one: 7
- When the sender receives the 3 ACKs for 6, it identifies the segment 7 is lost, then retransmit before timeout occurs.



Selective Acknowledgement (SACK)

- In the optional field of TCP header, a SACK could be added for receiver to send the “screenshot” of the receiver’s buffer to the sender.

- For example, in this case, the receiver can also send:

- ACK6, SACK8, SACK9

- Then, the sender can also recognize that 7 is lost and should be retransmitted.

Retransmission

- How should this timeout be determined?
  - **RTO** (**Retransmission TimeOut**)
  - Should at least be greater than the **RTT** (round-trip time)

- If sets too long, it degrades network performance.

- If sets too small, it triggers unnecessary retransmission, which leads to duplicated bytes.

- Due to the fact that the underlying IP network infrastructure is unpredictable, RTT varies a lot at Transport Layer

RTT varies a lot at Transport Layer

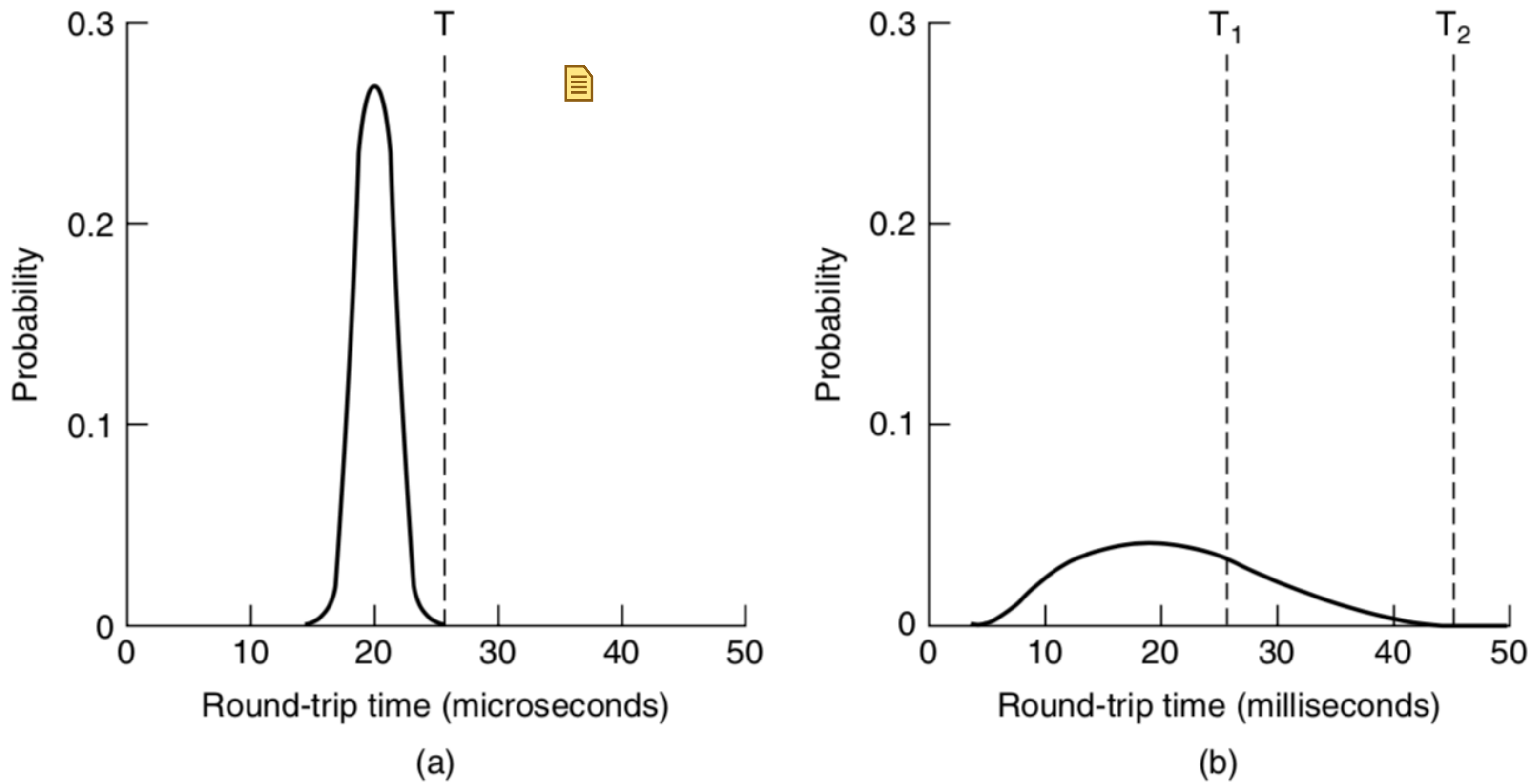


Figure 6-42. (a) Probability density of acknowledgement arrival times in the data link layer. (b) Probability density of acknowledgement arrival times for TCP.

Adaptive retransmission algorithm

- For each connection, TCP maintains a variable, ***SRTT*** (Smoothed Round-Trip Time), that is the best **current estimate** of the round-trip time to the destination.

- TCP measures how long the acknowledgement took, say, ***R***.

$$SRTT = \alpha \times SRTT + (1 - \alpha) \times R$$

  - $0 \leq \alpha < 1$, typically, $\alpha = {}^{7}/_{8}$

Adaptive retransmission algorithm

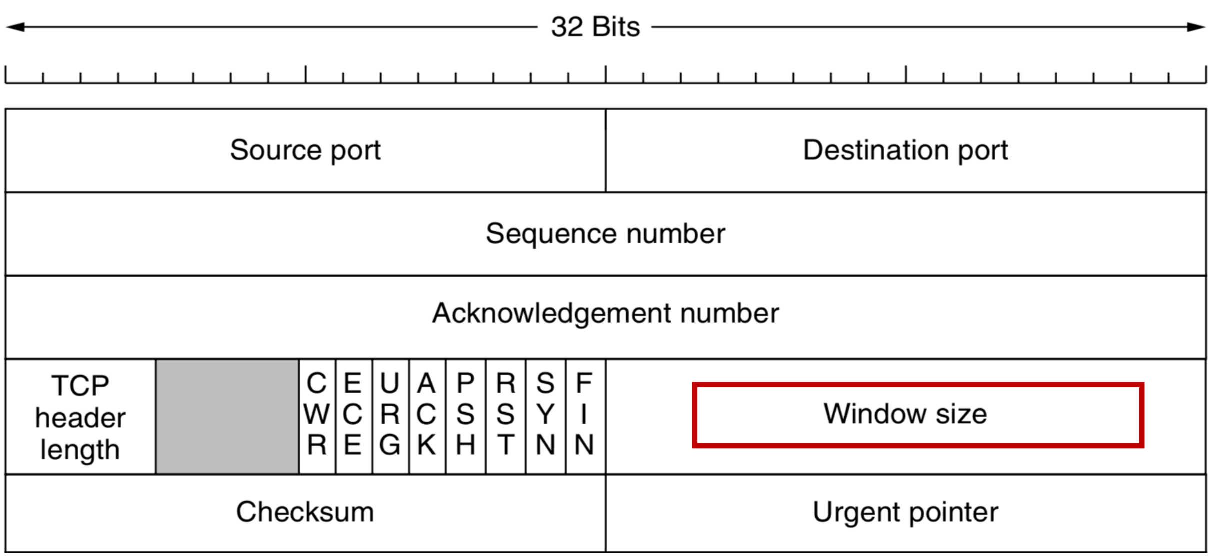
- ***RTTVAR*** (Round- Trip Time VARiation)
  - RTTVAR is not exactly the same as the standard deviation (it is really the mean deviation), but it is close enough in practice.

- $RTTVAR = \beta \times RTTVAR + (1 - \beta)|SRTT - R|$
  - $0 \leq \beta < 1$, typically, $\beta = {}^{3}/_{4}$

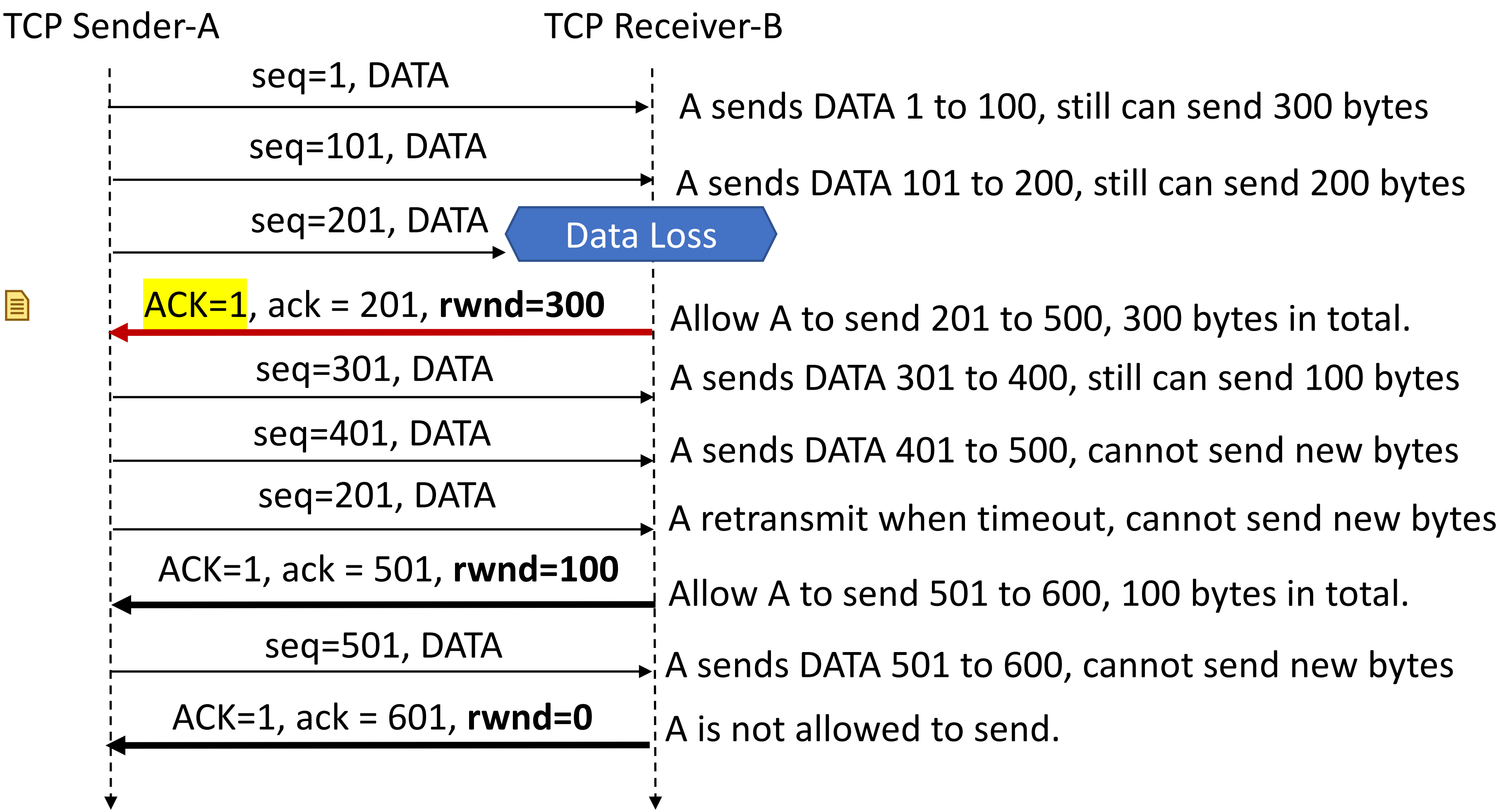
$$\boldsymbol{RTO = SRTT + 4 \times RTTVAR}$$

TCP Flow Control

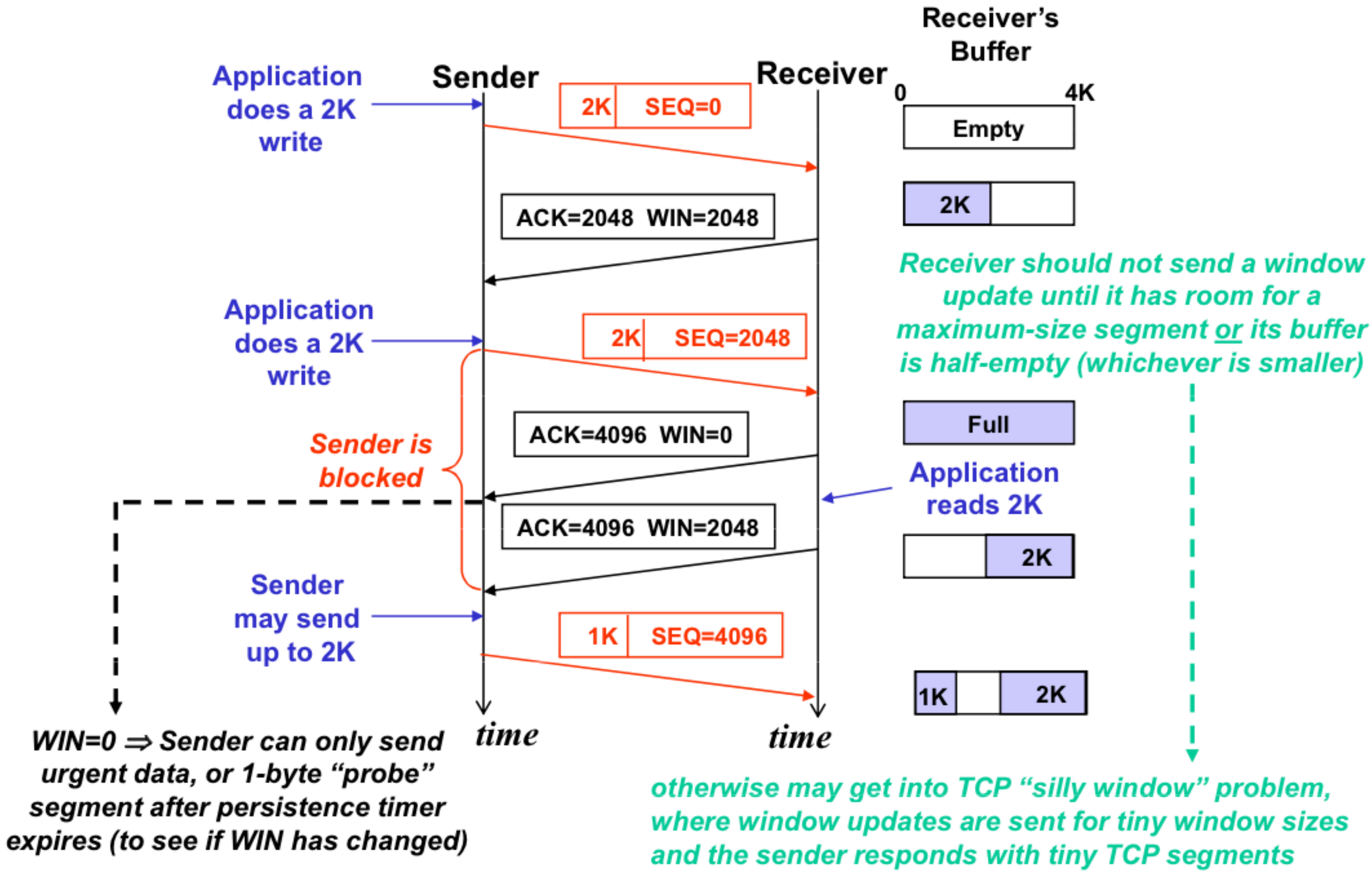
- TCP uses **variable-sized** sliding window
- Receiver window, $rwnd$: the maximum number of bytes that the receiver can receive continuously.



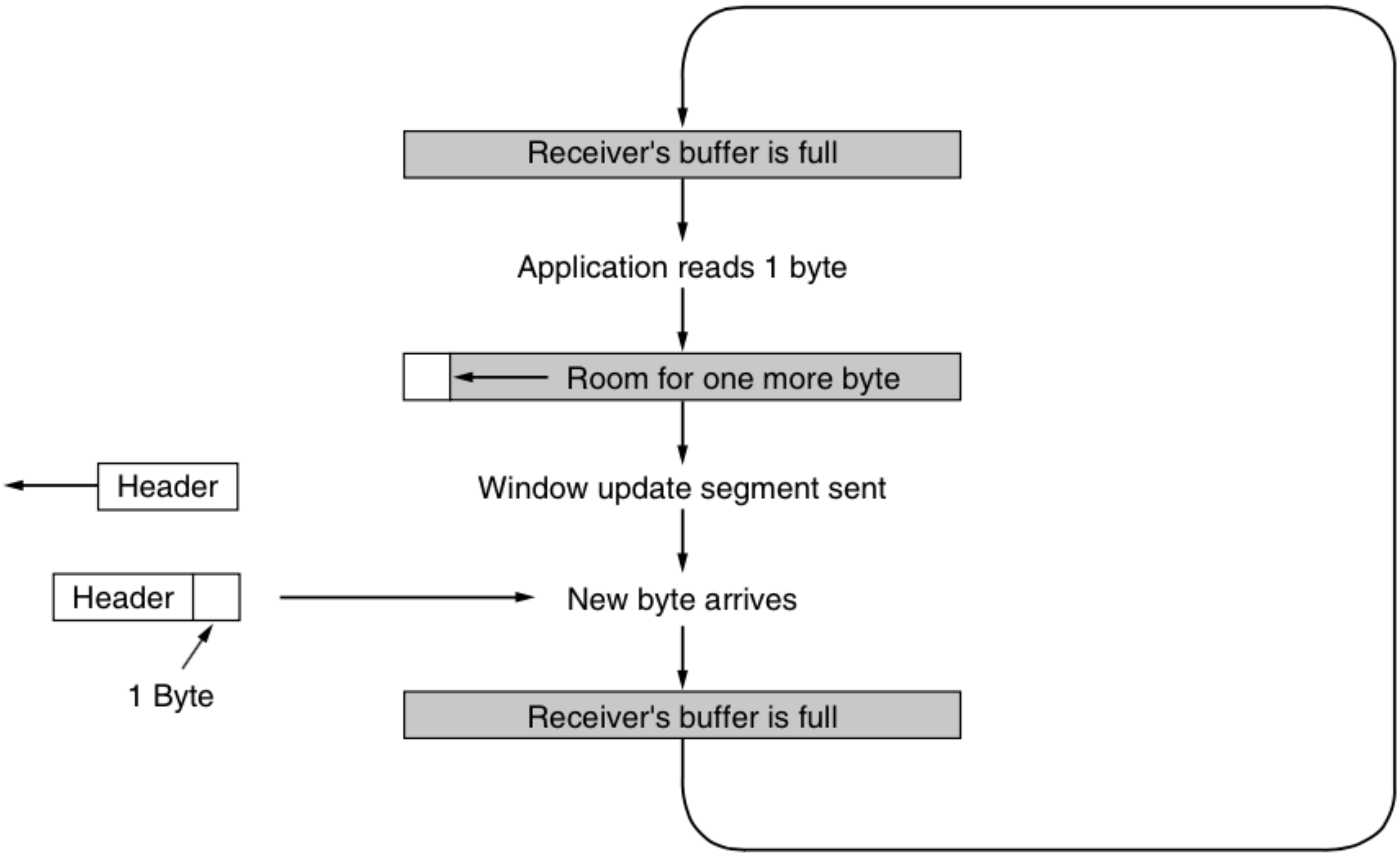
TCP Flow Control



Silly Window Syndrome



Silly Window Syndrome



TCP Congestion Control

- Flow Control
  - $rwnd$ tries to avoid to fully occupy receiver's buffer

- Congestion Control
  - Congestion Window $cwnd$: number of MSSs allowed to <u>send</u> at any given time.
  - $cwnd$ tries to avoid to fully occupy **the whole network**

- In practice, the maximum number of MSSs that a TCP sender can send is determined by the smaller one between $cwnd$ and $rwnd$
  - LastByteSent – LastByteAcked <= min{cwnd, rwnd}

- TCP sender recognizes the network is congested, when packet loss occurs
  - Timeout
  - 3 repeat ACKs

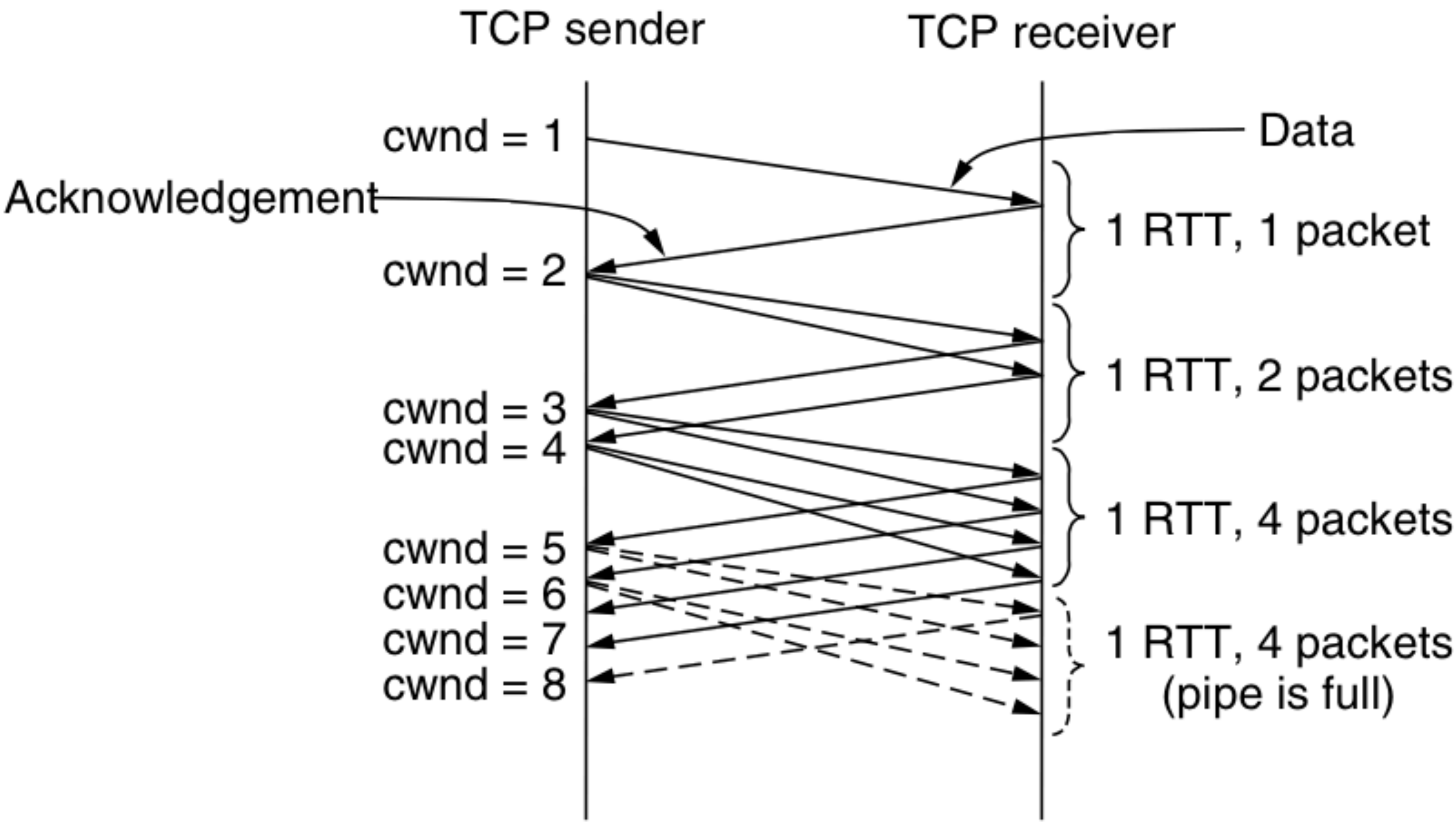
TCP Congestion

- $ssthresh$: *slow start* threshold for $cwnd$, measured in MSS. Once this threshold is met, the *slow start* finishes!

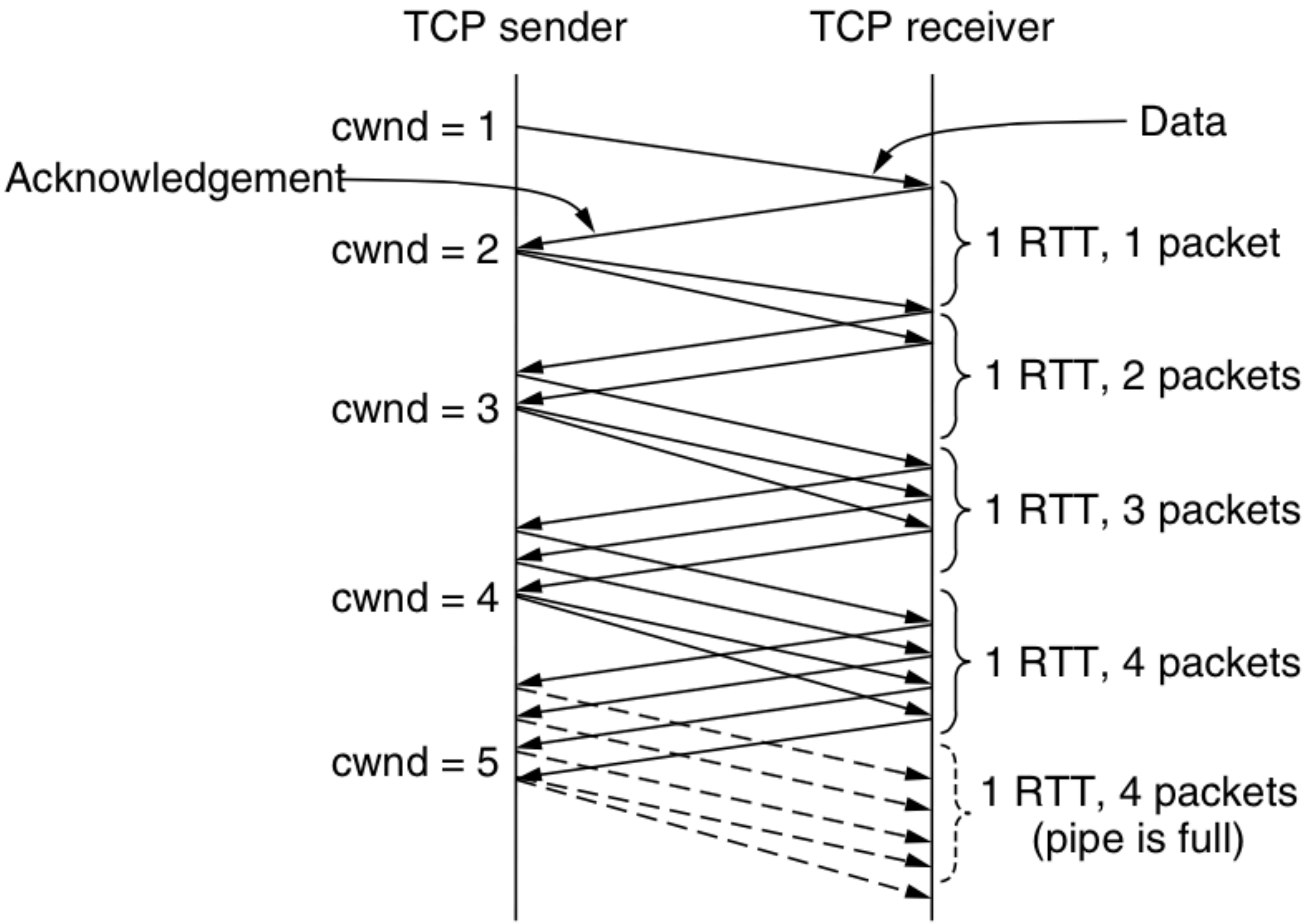
- $cwnd < ssthresh$:
  - Sender is in *Slow Start*, $cwnd$ grows **exponentially** from 1 MSS after each RTT

- $cwnd > ssthresh$:
  - Sender is in *Congestion Avoidance*, $cwnd$ grows **linearly** by 1 MSS after each RTT, using "Additive Increase"

Slow Start


TCP sender
TCP receiver
cwnd = 1
Data
Acknowledgement
1 RTT, 1 packet
cwnd = 2
1 RTT, 2 packets
cwnd = 3
cwnd = 4
1 RTT, 4 packets
cwnd = 5
cwnd = 6
cwnd = 7
cwnd = 8
1 RTT, 4 packets
(pipe is full)

Congestion Avoidance



TCP Congestion Control

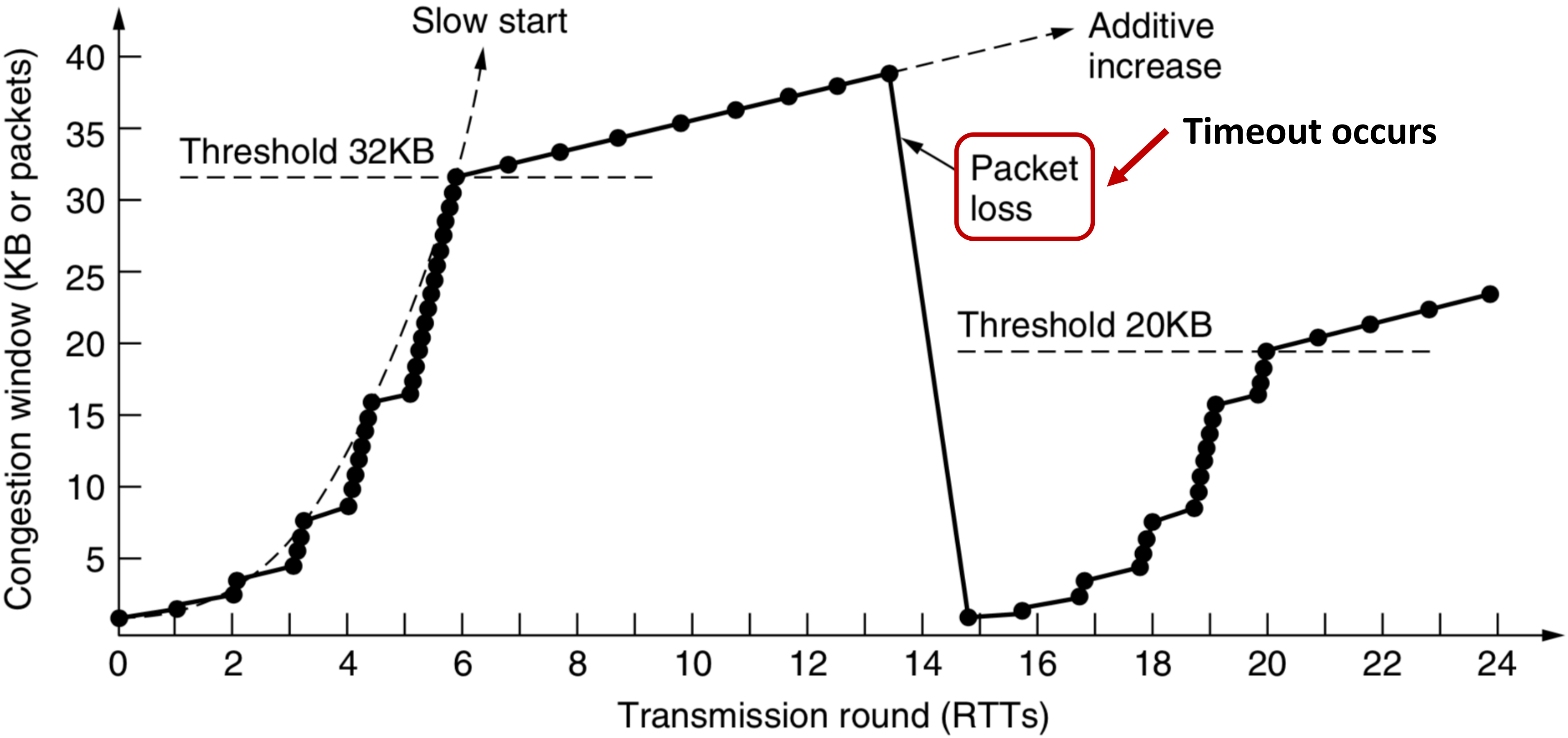
- Start with *Slow Start* until:
  - $ssthresh$ is met……. go to *Congenstion Avoidance*
  - Packet loss occurs…. go to *Packet Loss*

- *Congestion Avoidance*:
  - Additive increase for $cwnd$
  - When packet loss occurs …. go to *Packet Loss*

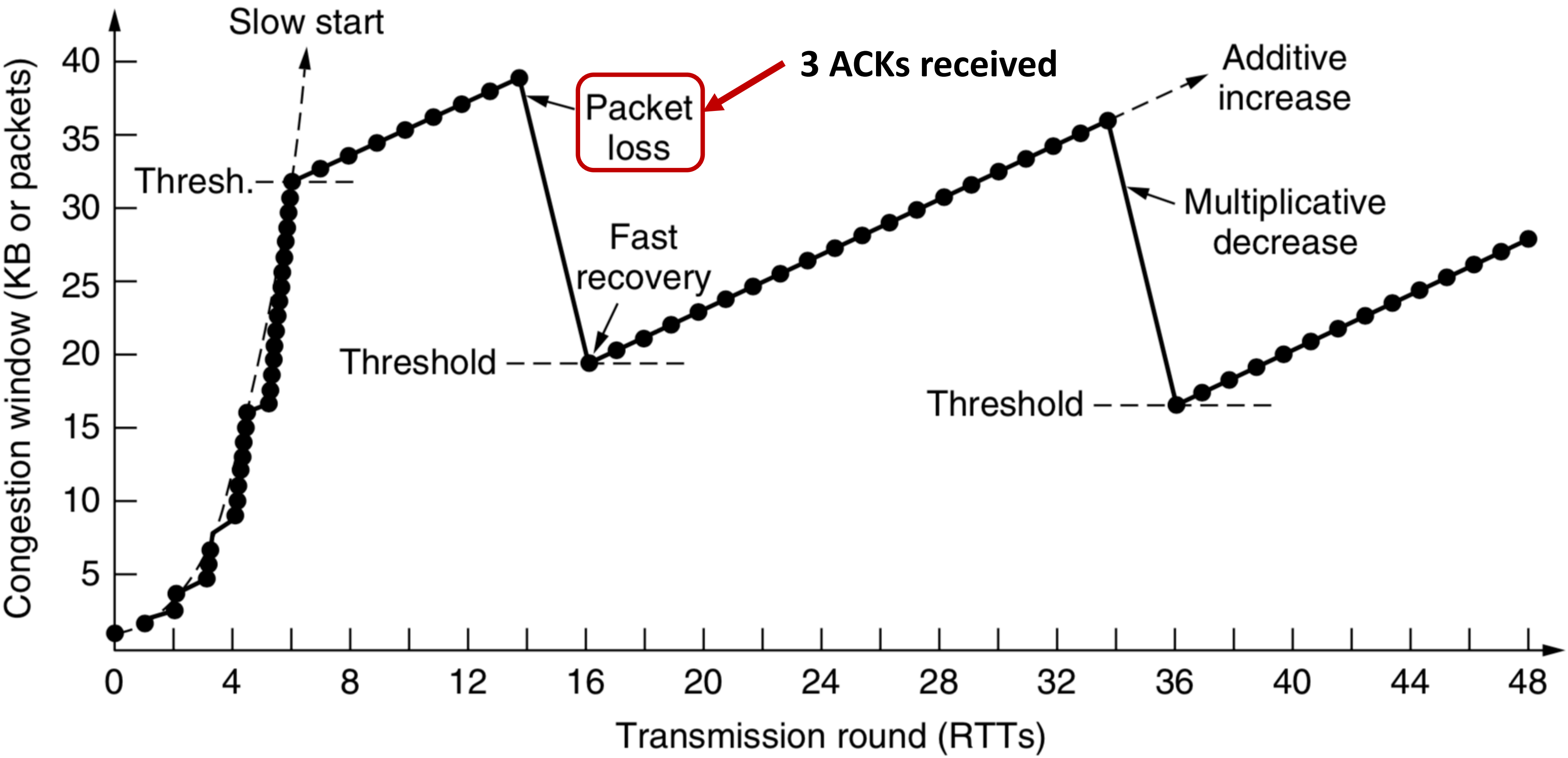
- Packet Loss:
  - Reset: $ssthresh = \frac{1}{2}cwnd$
  - If recognized by "timeout", then use **TCP Tahoe**:
    - $cwnd = 1$(MSS) , this is called "Multiplicative Decrease"
    - go to *Slow Start*
  - If recognized by "3 ACKs", then use **TCP Reno**:
    - $cwnd = ssthresh$, this is also called "Multiplicative Decrease"
    - go to *Congestion Avoidance*

TCP - Tahoe

Tahoe is out-of-date now, as it suddenly resets $cwnd$ to 1 from normally a very large value, which often leads to a network lag.



TCP - Reno



Slow start
3 ACKs received
Packet loss
Additive increase
Thresh.
Fast recovery
Multiplicative decrease
Threshold
Threshold
Congestion window (KB or packets)
40
35
30
25
20
15
10
5
0
4
8
12
16
20
24
28
32
36
40
44
48
Transmission round (RTTs)