CSCE 463/612
Networks and Distributed Processing
Spring 2024

Transport Layer VIII

Dmitri Loguinov
Texas A&M University

March 22, 2024

Chapter 3: Roadmap

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

TCP Congestion Control

- End-to-end control (no network assistance)
- Sender limits transmission:
  `LastByteSent - LastByteAcked ≤ CongWin`
- `CongWin` is a function of perceived network congestion
- The *effective* window is the minimum of `CongWin`, flow-control window carried in the ACKs, and sender's own buffer space

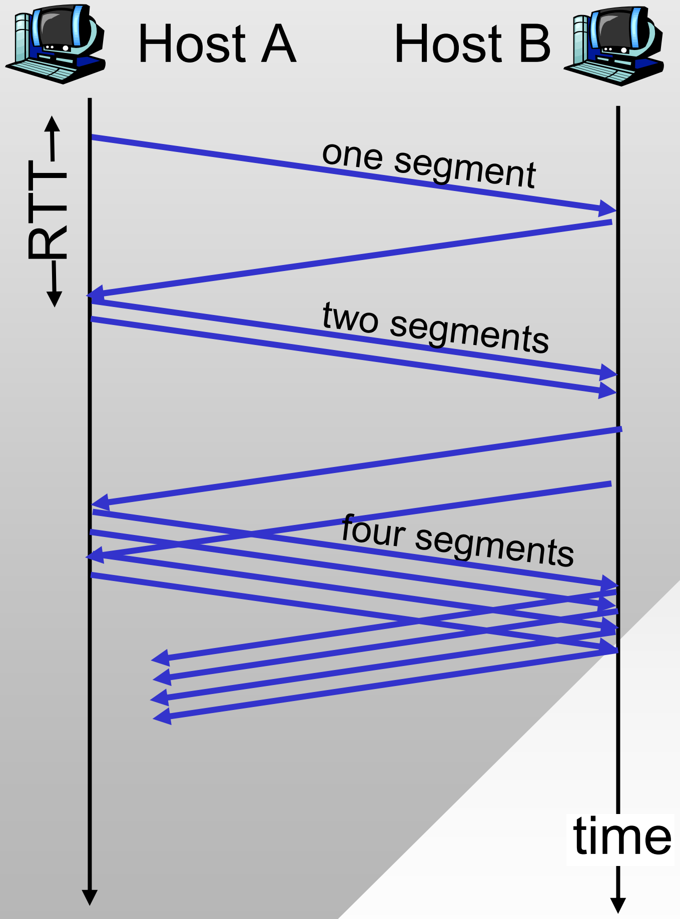
- How does sender perceive congestion?
  - Loss event = timeout or 3 duplicate acks
- TCP sender reduces rate (`CongWin`) after loss event
- Three mechanisms:
  - Slow start
  - Conservative after timeouts
  - AIMD (congestion avoidance)

TCP Slow Start

- When connection begins, `CongWin` $= 1$ MSS
  - Example: MSS = 500 bytes and RTT = 200 msec
  - Q: initial rate?
  - A: 20 Kbits/s
- Available bandwidth may be much larger than MSS/RTT
  - Desirable to quickly ramp up to a “respectable” rate
- Solution: Slow Start (SS)
  - When a connection begins, it increases rate exponentially fast until first loss or receiver window is reached
  - Term “slow” is used to distinguish this algorithm from earlier TCPs which directly jumped to some huge rate

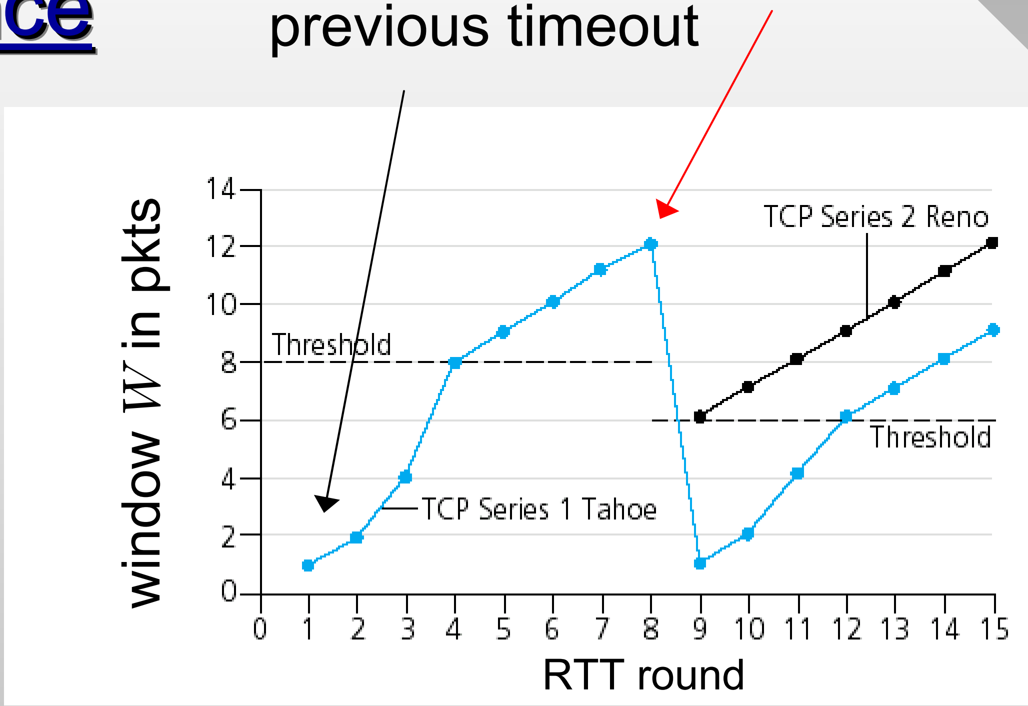
TCP Slow Start (More)

- Let W be congestion window in pkts and $B =$ `CongWin` be the same in bytes ($B = MSS * W$)
- Slow start
  - Double `CongWin` every RTT
- Done by incrementing `CongWin` for every ACK received:
  - $W = W{+}1$ per ACK (or $B = B + MSS$)
- Summary: initial rate is slow but ramps up exponentially fast



Congestion Avoidance

- TCP Tahoe loss (timeout or triple dup ACK):
  - `Threshold = CongWin/2`
  - `CongWin` is set to 1 MSS
  - Slow start until `threshold` is reached; then move to linear probing
- TCP Reno loss:
  - Timeout: same as Tahoe
  - 3 dup ACKs: `CongWin` is cut in half, then continue linear probing (called fast recovery, now part of AIMD)


loss detected via triple dup ACK

previous timeout

Fast Recovery Philosophy:

Three dup ACKs indicate that network is capable of delivering subsequent segments

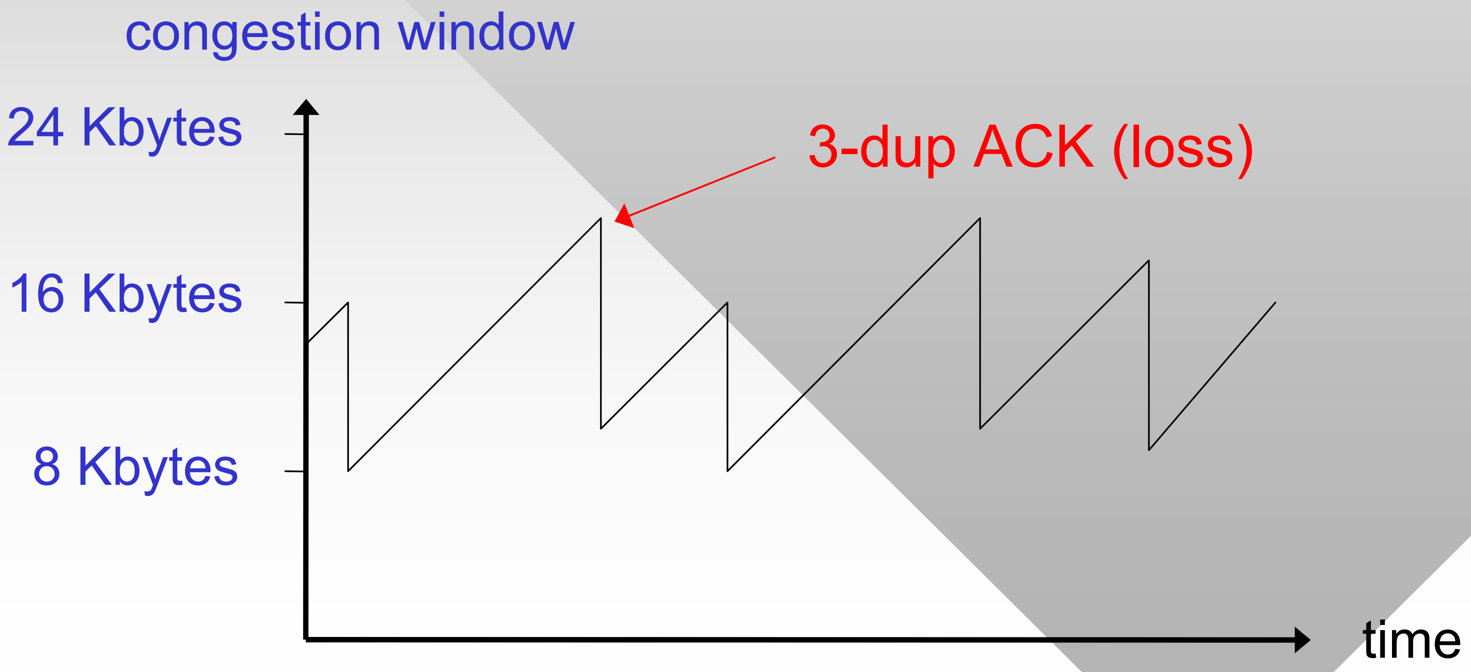
Timeout before 3-dup ACK is more alarming

TCP Reno AIMD (Additive Increase, Multiplicative Decrease)

Additive increase: increase `CongWin` by 1 MSS every RTT in the absence of loss events: *probing*

Multiplicative decrease: cut `CongWin` in half after fast retransmit (3-dup ACKs)

Peaks are different: # of flows or RTT changes



TCP Reno Equations

- To better understand TCP, we next examine its AIMD equations (congestion avoidance)
- General form (loss detected through 3-dup ACK):

$$W = \begin{cases} W + \frac{1}{W} & \text{per ACK} \\ W/2 & \text{per loss} \end{cases}$$

- Reasoning
  - For each window of size W, we get exactly W acknowledgments in one RTT (assuming no loss!)
  - This increases window size by roughly 1 packet per RTT
- Performing actions on packet arrival is lower overhead than waking up on timers

TCP Reno Equations

$$W = \begin{cases} W + \frac{1}{W} & \text{per ACK} \\ W/2 & \text{per loss} \end{cases}$$

- What is the equation in terms of $B = MSS * W$?

$$B = \begin{cases} B + \frac{MSS^2}{B} & \text{per ACK} \\ B/2 & \text{per loss} \end{cases}$$

- Equivalently, TCP increases B by MSS per RTT
- What is the rate of TCP given that its window size is B (or W)?
- Since TCP sends a full window of pkts per RTT, its ideal rate can be written as:

$$r = \frac{B}{RTT + L/R} \approx \frac{B}{RTT} = \frac{MSS * W}{RTT}$$

TCP Reno Sender Congestion Control

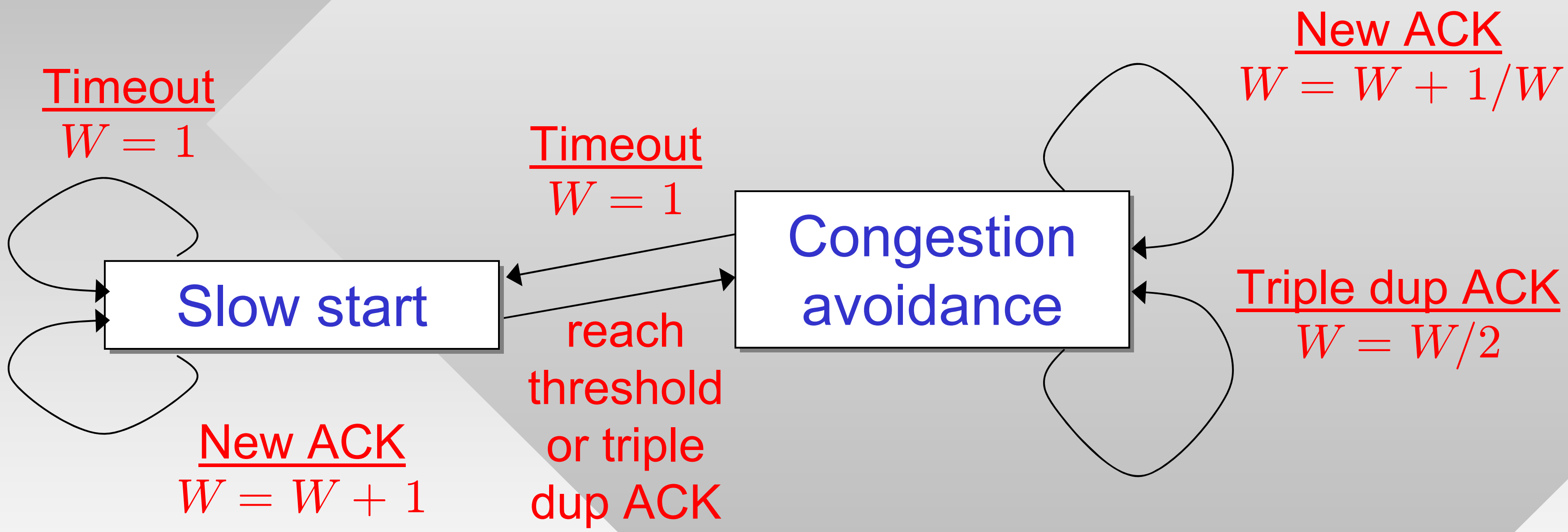
Event	State	TCP Sender Action	Commentary

ACK receipt for previously unacked data	Slow Start (SS)	CongWin += MSS,
		If (CongWin >= ssthresh) {
		Set state to "Congestion Avoidance"
		}
			Results in a doubling of CongWin every RTT

ACK receipt for previously unacked data	Congestion Avoidance (CA)	CongWin += MSS^2 / CongWin	Additive increase, resulting in increase of CongWin by 1 MSS every RTT
Loss event detected by triple duplicate ACK	SS or CA	ssthresh = max(CongWin/2, MSS) CongWin = ssthresh Set state to "Congestion Avoidance"	Fast recovery, implementing multiplicative decrease
Timeout	SS or CA	ssthresh = max(CongWin/2, MSS) CongWin = MSS Set state to "Slow Start"	Enter slow start
Duplicate ACK	SS or CA	Increment duplicate ACK count for segment being acked	CongWin and Threshold not changed

TCP Reno Congestion Control

- Summary:



Timeout
$W = 1$

Slow start

New ACK
$W = W + 1$

Timeout
$W = 1$

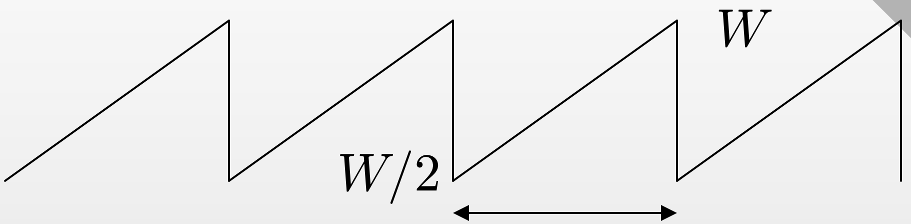
reach threshold or triple dup ACK

Congestion avoidance

New ACK
$W = W + 1/W$

Triple dup ACK
$W = W/2$

TCP Throughput

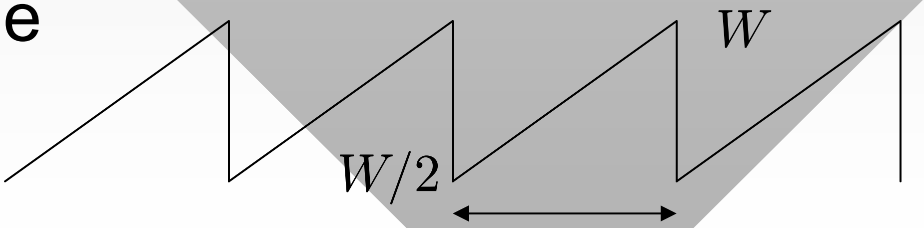


- What's the average throughout of TCP as a function of max window size W and RTT?
  - Ignore slow start and assume perfect AIMD (no timeouts)
- Let W be the window size when loss occurs
  - At that time, throughput is $W * MSS/RTT$
  - Just after loss, window drops to $W/2$, throughput is halved
- Average rate:

$$r_{av} = \frac{3}{4} \times \frac{W \times MSS}{RTT} = \frac{W_{av} \times MSS}{RTT}$$

TCP Model

- Example: 1500-byte segments, 100 ms RTT, want 10 Gbps average throughput r_{av}
  - Requires max window size $W = 111{,}111$ in-flight segments, 166 MB of buffer space ($W_{av} = 83{,}333$ packets)
  - But there are bigger issues as discussed below
- Next: derive average throughput in terms of loss rate
  - Assume packet loss probability is p
  - Roughly one packet lost for every $1/p$ sent packets
- Step 1: derive the number of packets transmitted in one oscillation cycle



TCP Model

- Examine time in terms of RTT units
    - At each step, window increases by 1 packet
- The number of packets sent between two losses:

$$sent = \frac{W}{2} + \left(\frac{W}{2} + 1\right) + \left(\frac{W}{2} + 2\right) + \ldots + W$$

- Combining $W/2$ terms, we have:

$$sent = \frac{W}{2}\left(\frac{W}{2} + 1\right) + \sum_{i=1}^{W/2} i$$

TCP Model

- Thus we arrive at:

$$sent = \frac{3}{8}W^2 + \frac{3}{4}W$$

- Step 2: now notice that this number equals $1/p$
  - Ignoring the linear term, we approximately get:

$$\frac{1}{p} \approx \frac{3}{8}W^2$$

- In other words:

$$W = \sqrt{\frac{8}{3p}}$$

TCP Model

- Step 3: writing in terms of average rate:

$$r_{av} = \frac{W_{av} \times MSS}{RTT} = \frac{\frac{3}{4}W \times MSS}{RTT} = \frac{\frac{3}{4}\sqrt{\frac{8}{3p}} \times MSS}{RTT}$$

- Simplifying:

$$r_{av} = \frac{\sqrt{3/2} \times MSS}{RTT\sqrt{p}} \approx \boxed{\frac{1.22 \times MSS}{RTT\sqrt{p}}}$$

- This is the famous formula of AIMD throughput
  - Note: homework #3 does not use congestion control and its rate is a different function of p