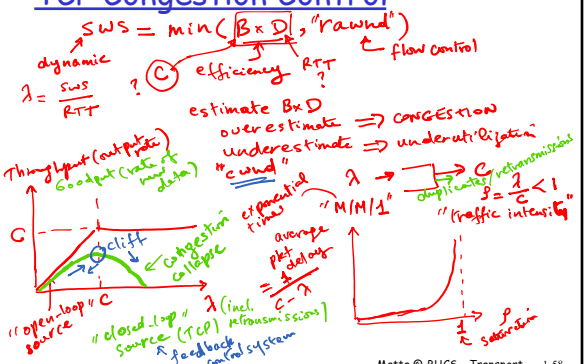
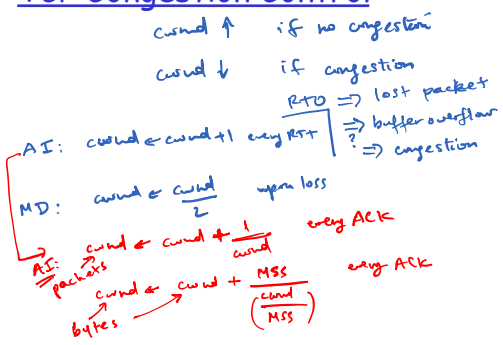
TCP Congestion Control



58

TCP Congestion Control



59

TCP Congestion Control

Additive Increase/Multiplicative Decrease

- Objective: adjust to changes in the available capacity
- New state variable per connection: CongestionWindow
  - limits how much data source has in transit

MaxWin = MIN(CongestionWindow, AdvertisedWindow)

- Idea:
  - increase CongestionWindow when congestion goes down
  - decrease CongestionWindow when congestion goes up
- *Question:* how does the source determine whether or not the network is congested?
- *Answer:* a timeout occurs
  - timeout signals that a segment was lost
  - segments are seldom lost due to transmission error
  - lost segment implies congestion

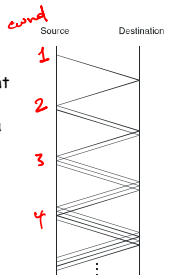
60

AIMD

- Algorithm:
  - increment CongestionWindow by one segment per RTT (***linear increase***)
  - divide CongestionWindow by two whenever a timeout occurs (***multiplicative decrease***)
- In practice: increment a little for each ACK

  Increment = (MSS x MSS)/CongestionWindow

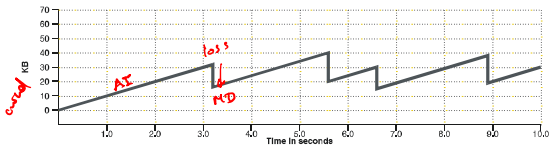
  CongestionWindow += Increment



61

Sawtooth behavior

- Example trace



62

Slow Start

63

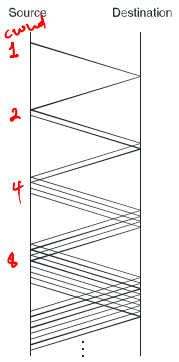
Slow Start

- Objective: determine the available capacity in the first place
  - when first starting connection
  - when connection recovers after a timeout
- Idea:
  - begin with CongestionWindow = 1 segment
  - double CongestionWindow each RTT (increment by 1 segment for each ACK)

64

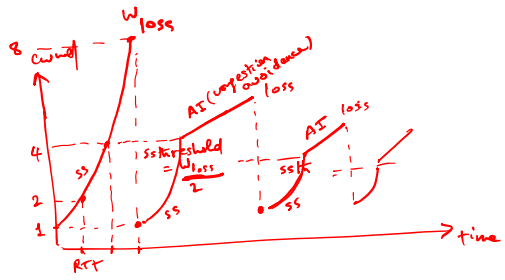
Slow Start (cont'd)

- Exponential growth, but slower than all in one blast



To reach a window W
$\log_2 W$ RTTs

65



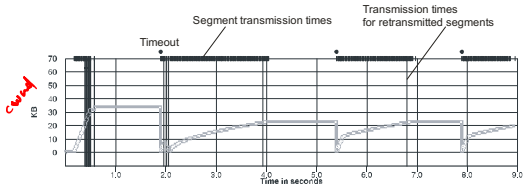
66

TCP Congestion Algorithm

On a timeout, half the current window size is recorded in ssthresh

```
if (cwnd < ssthresh) // if we're still doing
                     // slow-start, open window exponentially
    cwnd += 1
else // otherwise do Congestion Avoidance
     // linear increase
    cwnd += 1/cwnd
```

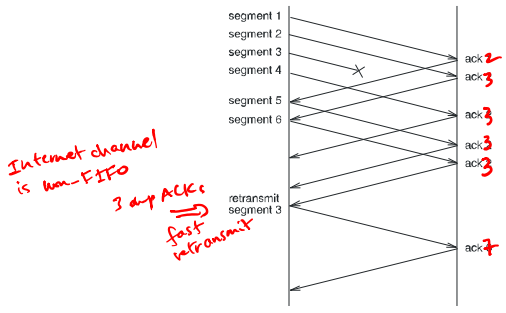

67



- Example trace
- cwnd stays flat if no ACKs are received
- Problem: lose up to half a CongestionWindow 's worth of data during slow start

68

Fast Retransmit and Fast Recovery



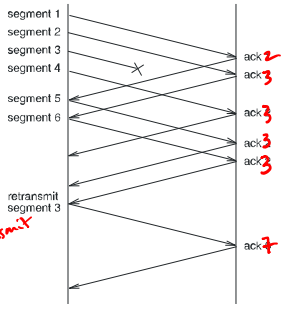
69

Fast Retransmit and Fast Recovery

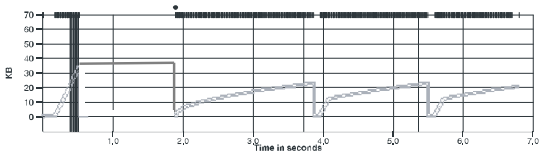
- Problem: coarse-grain TCP timeouts lead to idle periods
- Fast retransmit: use duplicate ACKs to trigger retransmission

Internet channel is non-FIFO

3 dup ACKs ⇒ fast retransmit



70



- Long periods during which cwnd stays flat are eliminated

71