CS 372 Lecture #22

Reliable data transfer with TCP

- congestion control

Note: Many of the lecture slides are based on presentations that accompany *Computer Networking: A Top Down Approach,* 6th edition, by Jim Kurose & Keith Ross, Addison-Wesley, 2013.

Congestion Control

- Goals:
  - Optimize utilization
  - Handle causes of network congestion
    - Retransmission handles only the symptoms
  - Detect congestion
  - Avoid congestion (when possible)
  - React to congestion (when necessary)

Detecting congestion

- Two approaches
  - Inferred by end-to-end systems
    - delay/loss
    - used by TCP
  - Network core assistance
    - routers provide feedback to senders
    - more later about ICMP
      - Internet Control Messaging Protocol

Avoiding congestion

- Handle the <u>causes</u> … and optimize utilization
  - timeouts (see previous lecture)
  - sliding windows
    - If receiving application drains a few data bytes, receiver will advertise a small window , and sender will immediately send small segment to fill window
      - Inefficient in processing time and network bandwidth
    - TCP solution:
      - Receiver delays advertising new window
      - Sender delays sending data when window is small

Reacting to (handling) congestion

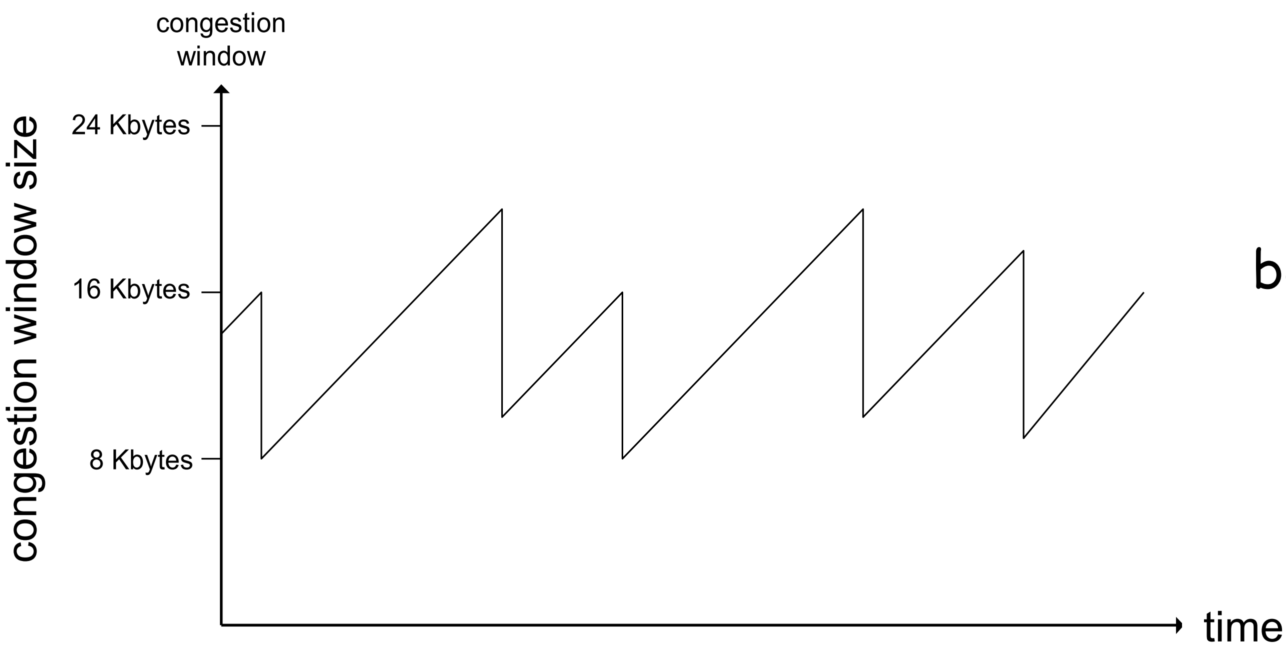
- Delicate balance
  - Too slow: under-utilization
    - waste of network resources
  - Too fast: over-utilization
    - waste of network resources

- Approach: **Adaptive transmission algorithm**
  - Define **`CongWin`** (sender's window size)
  - Slowly increase **`CongWin`** to probe for usable bandwidth
  - Decrease the sending rates when congestion is detected
  - TCP uses additive-increase, multiplicative-decrease (**AIMD**)

- Definition: MSS (Maximum Segment Size)
  - Upper limit on amount of data that can be sent in the largest link-layer frame on the sending host (usually 1460 bytes)

TCP congestion control: AIMD

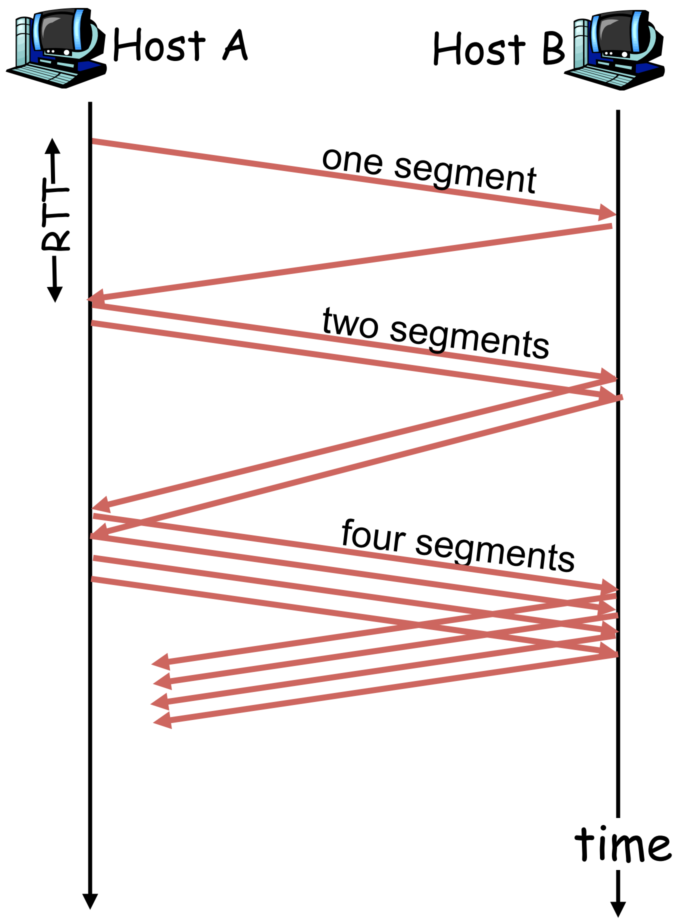
- Additive-increase, multiplicative decrease (AIMD)
  - additive increase: increase **CongWin** by 1 MSS every RTT until loss is detected
  - multiplicative decrease: cut **CongWin** in half after loss



Saw tooth behavior: probing for bandwidth

TCP congestion control: Slow Start

- When connection begins
  - **`CongWin`** = 1 MSS
  - available bandwidth may be much greater than MSS/RTT
    - avoids congestion, but need to quickly increase to a respectable rate
- increase rate exponentially until first loss event
  - double **`CongWin`** every RTT
  - must stop increasing rate if congestion is detected
- initial rate is slow but increases exponentially fast
  - must define a **`Limit`**
    - **`CongWin`** grows linearly after **`Limit`**



Refinement: inferring loss

- After 3 dup ACKs:
  - **`CongWin`** is cut in half
  - window then grows linearly
- But after timeout event:
  - **`CongWin`** is set to **1** MSS
  - window then grows exponentially to a **`Limit`**, then grows linearly

Philosophy:

•3 duplicate ACKs implies that the network is capable of delivering some segments, so OK to assume packet is lost.

•timeout indicates a "more alarming" congestion scenario

Summary: TCP Congestion Control

- When **CongWin** is below **Limit**, sender is in slow-start phase
  - **CongWin** grows exponentially.
- When **CongWin** is above **Limit**, sender is in congestion-avoidance phase
  - **CongWin** grows linearly.
- When a triple duplicate ACK occurs, **Limit** set to **CongWin/2** and **CongWin** set to **Limit**.
  - back to slow-start or congestion-avoidance
- When timeout occurs, **Limit** set to **CongWin/2** and **CongWin** is set to 1 MSS.
  - back to slow-start

- Several TCP versions (RENO, TAHOE, VEGAS, etc.) See http://en.wikipedia.org/wiki/TCP_congestion_avoidance_algorithm

Summary Lecture #22

- Definitions:
  - MSS, AIMD, congestion window
- Congestion control
  - detection
    - delay/loss
  - avoidance
    - smarter timers, sliding-windows, slow-start
  - reaction
    - adaptive transmission