CPE348: Introduction to Computer Networks

Lecture #18: Chapter 6

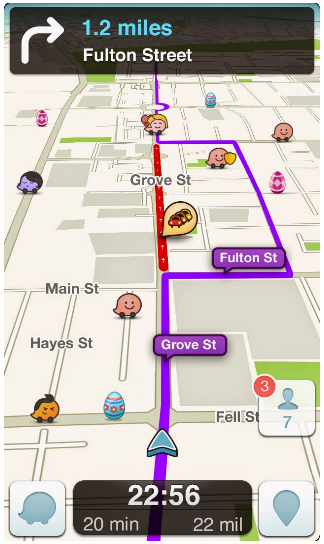
Jianqing Liu
Assistant Professor of Electrical and Computer Engineering, University of Alabama in Huntsville

jianqing.liu@uah.edu
http://jianqingliu.net

TCP Congestion Control Overview

- Early on, we talked about:
  - TCP congestion control protocols;

- In this lecture, we will talk about:
  - TCP congestion avoidance protocols
    - DEC bit
    - Slow Start
    - Fast Retransmit and Fast Recovery



Congestion Avoidance Mechanism

- TCP congestion control is the strategy once congestion happens, as opposed to avoiding congestion in the first place.
- An appealing alternative is congestion avoidance
  - Has not yet been widely adopted,
  - Tries to predict when congestion is about to happen
  - Reduces the rate at which source sends data just before packets start getting lost.

Congestion Avoidance Mechanism

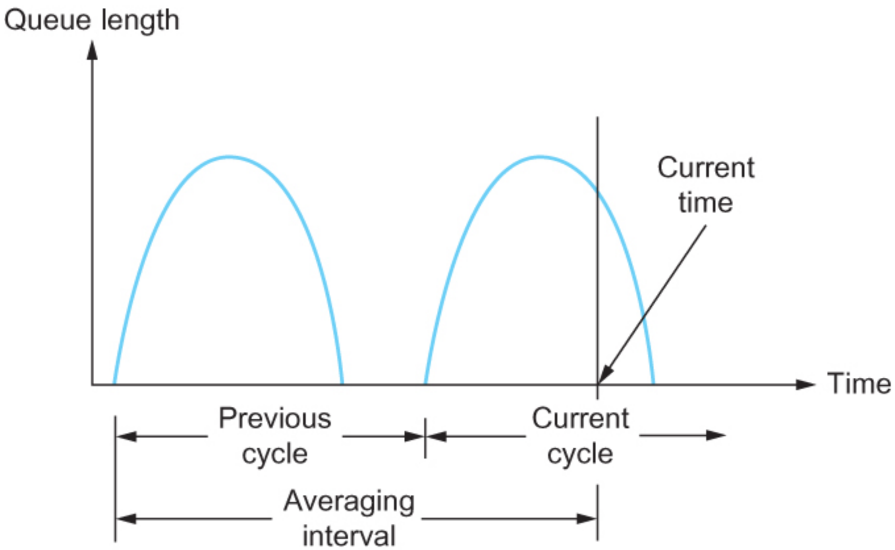
- DEC Bit
  - Initially developed for Digital Network Architecture (DNA), a connectionless network with a connection-oriented transport protocol;
  - This mechanism could, also be applied to TCP/IP.
  - The idea is to more evenly split the responsibility for congestion control between the routers and the end nodes.

Congestion Avoidance Mechanism

- DEC Bit
  - Each router monitors the load it is experiencing and explicitly notifies the end nodes when congestion is about to occur.
  - This notification is implemented by setting a binary congestion bit in the packets that flow through the router; hence the name DECbit.
  - The destination host then copies this congestion bit into the ACK it sends back to the source.
  - Finally, the source adjusts its sending rate so as to avoid congestion

Congestion Avoidance Mechanism

- DEC Bit – router participation
  - The router calculates the average queue length over the specified interval.



Computing average queue length at a router

Congestion Avoidance Mechanism

- DEC Bit – source participation
  - The source records how many of its packets resulted in some router setting the congestion bit.
  - In particular, the source maintains a congestion window, just as in TCP
  - The source watches to see what fraction of the last window’s worth of packets resulted in the congestion bit being set.

Congestion Avoidance Mechanism

- DEC Bit – source participation
  - If less than 50% of the packets had the bit set, then the source increases its congestion window by one packet.
  - If 50% or more of the last window's worth of packets had the congestion bit set, then the source decreases its congestion window to 0.875 times the previous value.

Congestion Avoidance Mechanism

- DEC Bit – source participation
  - 50% was chosen as the threshold based on analysis that showed it to correspond to the peak of the power curve.
  - The “increase by 1, decrease by 0.875” rule was selected because AIMD makes the mechanism stable.

Congestion Avoidance Mechanism

- Random Early Detection (RED)
  - A second congestion avoidance mechanism is called random early detection (RED)
    - When a router detects that congestion is imminent, it notifies the source to adjust its congestion window.

Congestion Avoidance Mechanism

- The difference between RED and DECbit
  - RED does not explicitly send a congestion notification message to the source
  - RED is implemented such that it *implicitly notifies* the source of congestion by dropping one of its packets.
  - The source is then notified by the subsequent timeout or duplicate ACK.

Congestion Avoidance Mechanism

- Cont'
  - The router drops the packet earlier than it has to
  - This action notifies the source that it should decrease its congestion window sooner than without RED.
  - The router drops a few packets before it has exhausted its buffer space completely
    - Causes the source to slow down its transmission rate
    - The router hopes that this will not cause to drop more packets later on.

Aggressive method!

Congestion Avoidance Mechanism

- Cont' – how to drop packets
  - Consider a simple FIFO queue on a router;
  - With RED, a router could decide to drop each arriving packet with some drop probability whenever the queue length exceeds some drop level.
  - This idea is called early random drop.

Congestion Avoidance Mechanism

- Cont' – Early Random Drop
  - RED computes an average queue length
    - AvgLen is iteratively computed as

      AvgLen = (1 − Weight) × AvgLen + Weight × SampleLen
    - where 0 < Weight < 1 and SampleLen is the length of the queue when a sample measurement is made.
    - Weight is a constant that determines how sensitive AvgLen is to variations in SampleLen
    - In most cases, the queue length is measured every time a new packet arrives at the router.

Congestion Avoidance Mechanism

- Cont' – Early Random Drop
  - Next, RED has two queue length thresholds : MinThreshold and MaxThreshold.
  - When a packet arrives at the router, RED compares the current AvgLen with them, according to the following rules:
    - if AvgLen ≤ MinThreshold

      → queue the packet
    - if MinThreshold < AvgLen < MaxThreshold

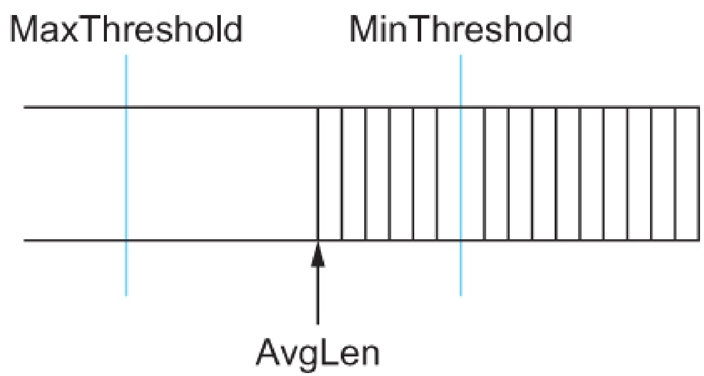
      → calculate probability P

      → drop the arriving packet with probability P
    - if AvgLen ≥ MaxThreshold

      → drop the arriving packet

Congestion Avoidance Mechanism

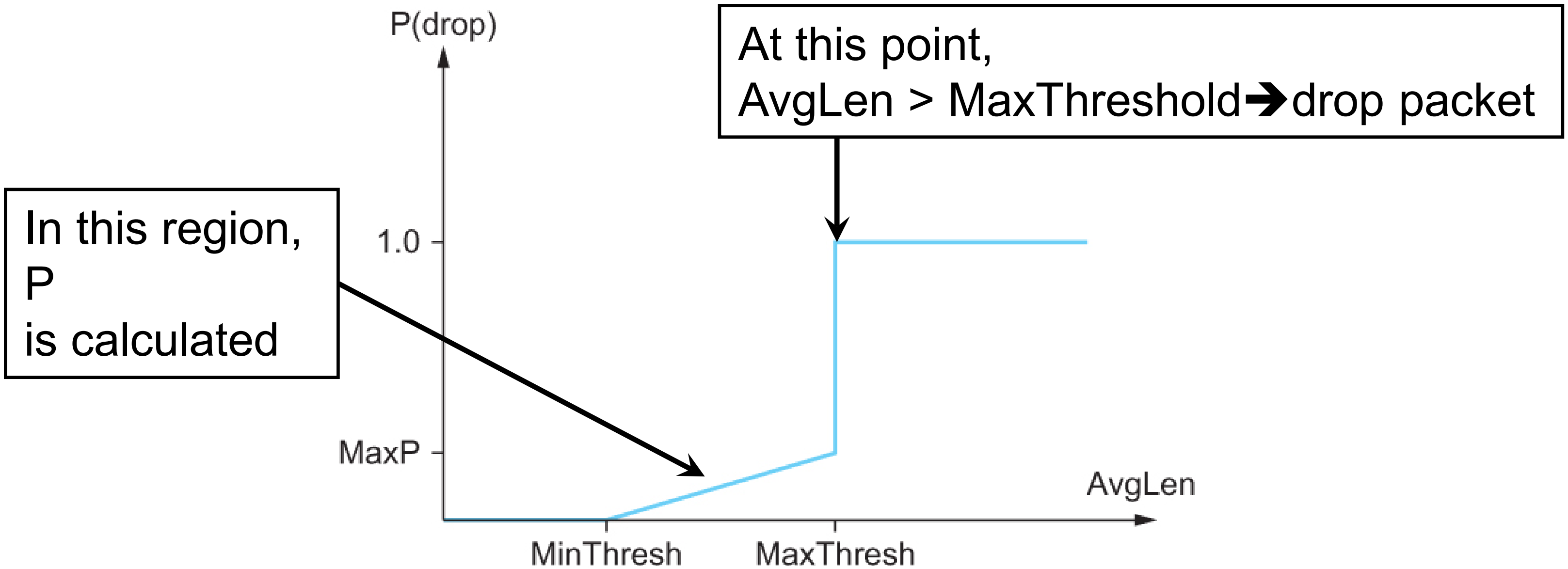
- Cont' – Early Random Drop



RED thresholds on a FIFO queue

Congestion Avoidance Mechanism

- RED – how dropping probability is calculated:



Drop probability function for RED

Congestion Avoidance Mechanism

- Source-based Congestion Avoidance
  - Sources watch for some indication that congestion will happen soon;
  - For example, source may notice an increase in the RTT for each successive packet being sent.

Catch a sign!

Congestion Avoidance Mechanism

- Source-based Congestion Avoidance
  - One algorithm using RTT measurements:
    - The congestion window normally increases as in TCP
    - After every two round-trip delays, the algorithm
      - Averages the minimum and maximum RTT seen so far
      - Checks the current RTT against the average value
      - If the current RTT is greater than the average, source decreases the congestion window by 1/8.

Congestion Avoidance Mechanism

- Source-based Congestion Avoidance
  - Another algorithm decides CongestionWindow based on changes to the RTT and the window size.
  - The congestion window is adjusted once every two RTTs based on the product
    - (CurrentWindow − OldWindow)×(CurrentRTT − OldRTT)
    - For a positive value, decrease the window size by 1/8;
    - For a non-positive value, increase the window by one maximum packet size.

Congestion Avoidance Mechanism

- Source-based Congestion Avoidance
  - A third algorithm the CongestionWindow based on the flatness of the sending rate
  - The CongestionWindow is adjusted every RTT based on the measured throughput
    - Every RTT increase the congestion window by 1
    - Compare the throughput achieved with the larger window to the throughput achieved with previous congestion window size.
    - If the difference is less than ½ of the throughput when only one packet was in transit (measured at beginning of connection with congestion window at 1) then decrease window by 1.