Congestion Avoidance

DECbit - RED

Congestion Avoidance Mechanism

- **Congestion Control**
  - Once congestion happens, TCP will control the congestion.
  - TCP repeatedly increases the load it imposes on the network in an effort to find the point at which congestion occurs, and then it backs off from this point.
- **Congestion Avoidance**
  - to predict when congestion is about to happen and then to reduce the rate at which hosts send data just before packets start being discarded.
  - But it is not yet widely used.
- **Three Popular Methods**
  - DECbit
  - Random Early Detection (RED)
  - Source Based Congestion Control

DECbit

Congestion Avoidance Mechanism: DECbit (1/5)

- The first mechanism was developed for use on the Digital Network Architecture (DNA), a connectionless network with a connection-oriented transport protocol.
- This mechanism could, therefore, also be applied to TCP and IP.
- The idea here is to more **evenly split the responsibility** for congestion control between the **routers and the end nodes**.
- 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.

Congestion Avoidance Mechanism: DECbit (2/5)

- The destination host then **copies this congestion bit** into the ACK and it sends back to the source.
- Finally, the source adjusts its sending rate so as to avoid congestion.
- A single congestion bit is added to the packet header.
  - A router sets this bit in a packet if its **average queue length** is greater than or equal to 1 at the time the packet arrives.
- This average queue length is measured over a time interval that spans the **last busy + idle cycle, plus the current busy cycle**.
- Using a queue length of 1 as the trigger for setting the congestion bit is a trade-off between significant queuing (and hence higher throughput) and increased idle time (and hence lower delay).

Congestion Avoidance Mechanism: DECbit (3/5)

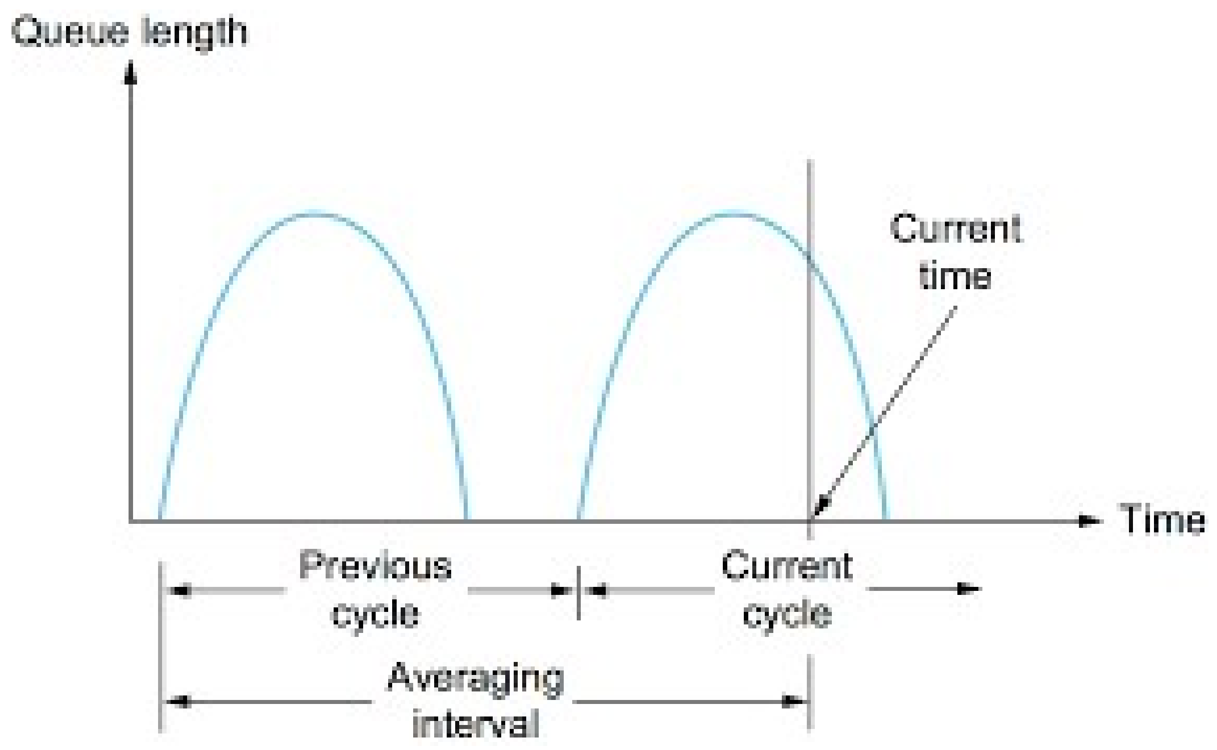
- In other words, a queue length of 1 seems to optimize the power function.
- The source records how many of its packets resulted in some router setting the congestion bit.
- In particular, the source maintains a congestion window, just as in TCP, and watches to see what fraction of the last window's worth of packets resulted in the bit being set.
- 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: DECbit (4/5)

- The value 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 additive increase/multiplicative decrease makes the mechanism stable.

Congestion Avoidance Mechanism: DECbit (5/5)

Computing average queue length at a router



RED

Congestion Avoidance Mechanism: Random Early Detection (RED) (1/6)

- It is similar to the DECbit scheme in that each router is programmed to monitor its own queue length, and when it detects that congestion is imminent, to notify the source to adjust its congestion window.
- RED, invented by Sally Floyd and Van Jacobson in the early 1990s, differs from the DECbit scheme in two major ways:
  - The first is that rather than explicitly sending a congestion notification message to the source, RED is most commonly implemented such that it **implicitly notifies the source of congestion by dropping one of its packets**.
- The source is, therefore, effectively notified by the subsequent timeout or duplicate ACK.
- RED is designed to be used in conjunction with TCP, which currently detects congestion by means of timeouts (or some other means of detecting packet loss such as duplicate ACKs).

Congestion Avoidance Mechanism: Random Early Detection (RED) (2/6)

- As the “**early**” part of the RED acronym suggests, the gateway drops the packet earlier than it would have to, so as to notify the source that it should decrease its congestion window sooner than it would normally have.
- In other words, the router drops a few packets before it has exhausted its buffer space completely, so as to cause the source to slow down, with the hope that this will mean it does not have to drop lots of packets later on.

Congestion Avoidance Mechanism: Random Early Detection (RED) (3/6)

- The second difference between RED and DECbit is, how RED decides when to drop a packet and what packet it decides to drop.
- To understand the basic idea, consider a simple FIFO queue. Rather than wait for the queue to become completely full and then be forced to drop each arriving packet, we 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. The RED algorithm defines the details of how to monitor the queue length and when to drop a packet.
- Average Queue Length Calculation: – Average queue length is calculated using weighted running average similar to the one used in the original TCP timeout computation.

Congestion Avoidance Mechanism: Random Early Detection (RED) (4/6)

  - **AvgLen** = (1 − Weight) × AvgLen + Weight × SampleLen
  - where 0 < Weight < 1 and SampleLen is the length of the queue when a sample measurement is made.
- In most software implementations, the queue length is measured every time a new packet arrives at the gateway.
- In hardware, it might be calculated at some fixed sampling interval.
- RED has two queue length thresholds that trigger certain activity: **MinThreshold and MaxThreshold**.
- When a packet arrives at the gateway, RED **compares the current AvgLen with these two thresholds**, according to the following rules:

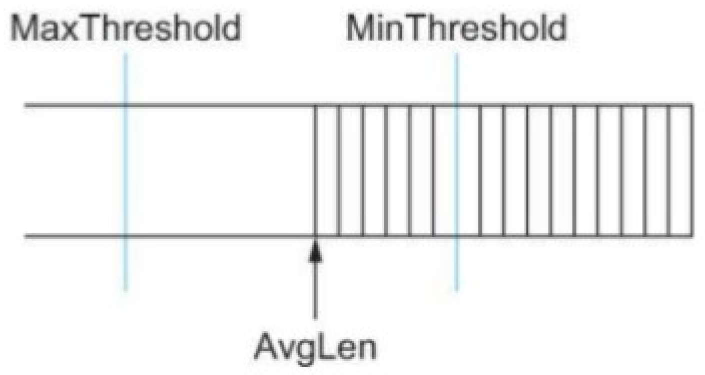
Congestion Avoidance Mechanism: Random Early Detection (RED) (5/6)

```
if AvgLen ≤ MinThreshold
    → queue the packet
if MinThreshold < AvgLen < MaxThreshold
    → calculate probability P
    → drop the arriving packet with probability P
if MaxThreshold ≤ AvgLen
    →drop the arriving packet
```

– P is a function of both AvgLen and how long it has been since the last packet was dropped.
– Specifically, it is computed as follows:
  - **TempP** = MaxP × (AvgLen – MinThreshold)/(MaxThreshold – MinThreshold)
  - **P** = TempP/(1 – count × TempP)

Congestion Avoidance Mechanism: Random Early Detection (RED) (6/6)

- Random Early Detection (RED)



RED thresholds on a FIFO queue

Source-based Congestion Avoidance

(Out of Syllabus)

Congestion Avoidance Mechanism: Source-based Congestion Avoidance (1/2)

- The general idea of these techniques is to watch for some sign from the network that some router's queue is building up and that congestion will happen soon if nothing is done about it.
- For example, the source might notice that as packet queues build up in the network's routers, there is a **measurable increase in the RTT** for each successive packet it sends.
- One particular algorithm exploits this observation as follows:
  - The congestion window normally increases as in TCP, but every two round-trip delays the algorithm checks to see if the **current RTT is greater than the average of the minimum and maximum RTTs** seen so far.
  - If it is, then the algorithm **decreases the congestion window by one-eighth**.

Congestion Avoidance Mechanism: Source-based Congestion Avoidance (2/2)

- A second algorithm does something similar. The decision as to whether or not to change the current window size is based on changes to both the RTT and the window size.
- The window is adjusted once every two round-trip delays based on the product
  - (CurrentWindow − OldWindow)×(CurrentRTT − OldRTT)
  - If the result is positive, the source decreases the window size by one-eighth;
  - if the result is negative or 0, the source increases the window by one maximum packet size.