Queuing

Presenter: Adam Berlinsky-Schine

September 28, 2004

Readings

- Random Early Detection Gateways for Congestion Avoidance (1993)
- Sizing Router Buffers (2004)

Random Early Detection: Overview

- Congestion Avoidance goal: maintain network in region of low delay and high throughput
- RED marks (drops) packets randomly when the average queue size is greater than threshold
- By calculating average queue size over time and having mark probability increase with level of congestion, problems such as global synchronization and bias against bursty packets are overcome

Goals of RED

- Overall goal: achieve congestion avoidance by controlling average queue size
- Backwards compatible with existing TCP clients
- Avoid global synchronization
- Avoid bias against bursty traffic
- Maintain upper bound on average queue size
  - Even in the absence of cooperation from transport layer protocols

Previous Queuing Schemes

- Drop Tail
- Random Drop
- IP Source Quench
- Early Random Drop
- DECbit

Drop Tail, Random Drop, IP Source Quench

- Drop Tail: drop incoming packets when buffer is full
- Random Drop: when buffer is full, drop a random packet from the queue
- IP Source Quench: When queue length exceeds threshold, send ICMP message to source
- Problems:
  - Bias against bursty traffic
  - Global synchronization
  - No explicit bound on queue length

Early Random Drop

- If queue exceeds a maximum threshold, drop packets
  - Drop with fixed probability (on the order of p = 0.02)
  - Responds to instantaneous queue size
- Problems:
  - Not able to control misbehaving users
  - Cannot explicitly bound average queue length
  - Can still have bias against bursts

RED vs. Early Random Drop

- RED:
  - *Average* queue size is measured rather than instantaneous
  - Packet marking is a function of average queue size rather than fixed probability
  - Not limited to dropping (can set a bit)
  - Bounds average queue size

DECbit

- When *average* queue length is greater than 1, mark congestion bit in header of all packets
- Source decreases by 0.875 times if more than half of packets in the last window had congestion bit set
- Average queue length is calculated from the last cycle of busy + idle time (plus current busy time)

RED vs. DECbit

- DECbit requires changes in the end-hosts, but RED is designed to work with existing TCP
- Choosing connections to notify of congestion
  - DECbit notifies ALL sources when average queue size > threshold; RED only notifies some
- DECbit backs off less aggressively than TCP's AIMD (0.875 rather than 0.5)
- Method of computing average queue size
  - DECbit uses last busy + idle period, which can be short; RED explicitly controls time constant in time-based exponential-decay
- RED is designed to work even without cooperation from hosts – average queue size is still bounded

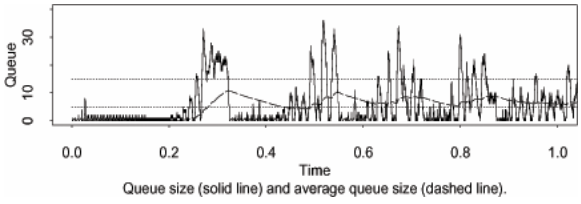
General RED Algorithm

```
for each packet arrival
   calculate the average queue size avg
   if min_th ≤ avg < max_th
      calculate probability p_a
      with probability p_a:
         mark the arriving packet
   else if max_th ≤ avg
      mark the arriving packet
```


General RED Algorithm

- Calculate average queue size
  - Determines degree of burstiness allowed
  - avg ← $(1 - w_q)$avg + w_q*q
    - Small w_q: Takes a long time to respond to congestion changes
    - Large w_q: avg rises quickly with bursts
- Calculate probability p_a
  - Determines how frequently the gateway marks packets, given level of congestion

Smoothing Effect



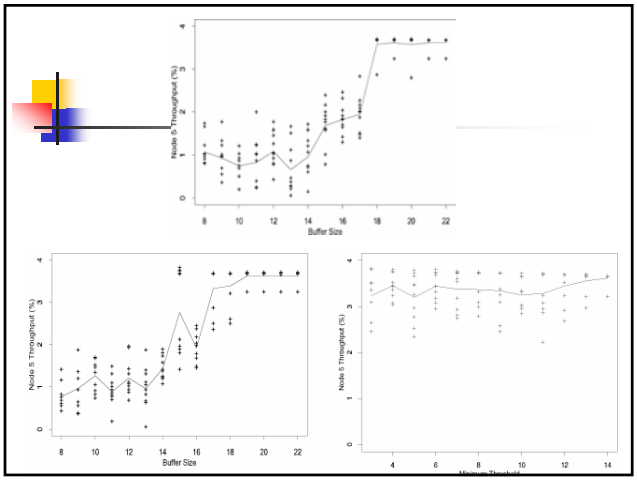
Queue size (solid line) and average queue size (dashed line).

Avoiding Global Synchronization

- Marks as few packets as possible
- Packets not marked at the same time
- Contrast with Drop-Tail: when overflow occurs, packets from many flows tend to be dropped

Avoiding Bias Against Bursty Traffic

- Only marks packets when *average* queue size over a period of time is too high
- Contrast with Drop Tail, which is more likely to overflow when a burst is received
  - Random Drop, Early Random Detection, and DECbit all have similar bias



Fairness

- Marked packets roughly proportional to a connection's bandwidth through gateway
  - Note: not true for, say, Drop-Tail
- Maintains desired average queue size even in the case of misbehaving users
  - But not fair – misbehaving users can still get more than their share of bandwidth
- Suggests mechanism for detecting misbehaving users by identifying connections with many dropped packets

Additional Benefits

- Gradual Deployment – Benefits observed if just one router implements RED
- Not restricted to FIFO – For example, could be applied to different diffserv classes

Conclusions

- RED gateways bounds calculated average queue size regardless of cooperation from transport layer
- It does so with no bias toward bursty flows, and does not tend towards global synchronization
- Applicable to a wide range of networks; various algorithm parameters allow flexibility in design decisions

Sizing Routing Buffers: Overview

- Core networks have been using a rule of thumb to determine how much buffer is needed in a router
- The rule of thumb was designed for a single flow, or several synchronized flows
- Paper argues that this is out of date, and the vast number of desynchronized flows today makes the buffer sizes way too large

Sizing Routing Buffers

- Goal: want link to be busy 100% of the time. Achieved if the buffer never becomes empty
- Tradeoff:
  - Large buffers minimize the probability that queue will empty
  - Small buffers minimize the delay caused by the routers

Rule of Thumb

- Well known rule of thumb is

  $B = RTT * C$

  (B = buffer size, RTT = Round Trip Time, C = data rate of link)
- But as C increases, this is becoming quite a burden on router design
- Authors argue that this is way too much

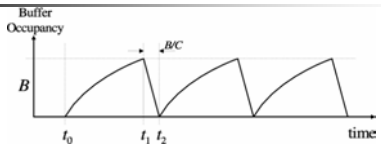
Why Not Overprovision?

- Large buffers
  - Increase delay
  - Complicate router design
  - Impede development of routers with larger capacity
  - Are expensive

Where did B = RTT*C come from?

- Accurate rule for a single flow
- When a single flow is in AIMD and a packet is dropped, the sender needs to wait for W/2 packets to flush out of pipe
- Don't want buffer to empty during this time

Where did B = RTT*C come from?



- Sender pauses for (W_{max}/2)/C seconds
- Router buffer drains over period B/C
- To avoid emptying, (W_{max}/2)/C < B/C, or W_{max}/2 < B
- Sending rate C of TCP is W/RTT, so C = W/RTT = (W_{max}/2) / 2T_p; or C * 2T_p= W_{max}/2
- Plugging into equation for B, B > C * 2T_p= C*RTT

Under- and Overbuffered

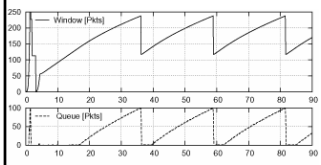


Figure 5: A TCP flow through an underbuffered router.

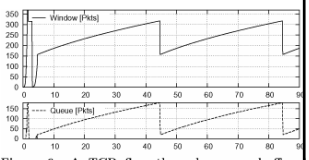


Figure 6: A TCP flow through an overbuffered router.

- Underbuffered: Loses throughput due to gaps
- Overbuffered: Increased delay, but no loss of throughput

Multiple Flows - Synchronized

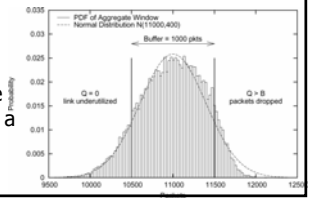
- Synchronized flows behave like amplified single flows
  - Recall that flow synchronization is common in Drop-Tail, the most common router
  - Common for fewer than 100 flows
- Still need enough buffer to account for pause from peak to trough of sawtooth

Multiple Flows - Desynchronized

- As the number of flows increases, they smooth each other out
  - Very rare for synchronized flow > 500 flows
- Still need enough buffer for peak to trough
  - But no longer a sawtooth – peaks and troughs are very small!

Multiple Flows - Desynchronized

- Model total window size as bounded random process made up of sum of sawtooths
- If start times and propagation delays are independent, aggregate window size will converge to Gaussian
- Queue occupancy
  $Q = W - 2T_p*C$
  - So Q is Gaussian too
- This is good, because now we can determine Prob(underflow) given a buffer size



Revised Rule of Thumb

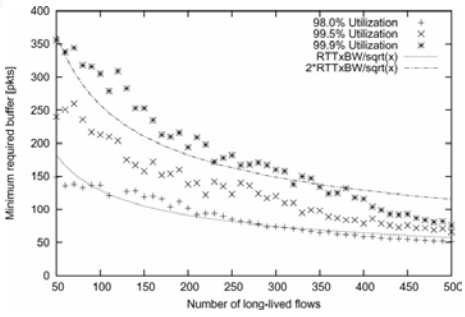
- Authors deduce new rule of thumb:
  B = RTT*C/sqrt(n)
  Where n = number of flows
  - This adds up when n > 10,000!

Router Buffer Size	Utilization
$B = 1 \cdot \frac{2\overline{T}_p \cdot C}{\sqrt{n}}$	$\text{Util} \geq 98.99\ \%$
$B = 1.5 \cdot \frac{2\overline{T}_p \cdot C}{\sqrt{n}}$	$\text{Util} \geq 99.99988\ \%$
$B = 2 \cdot \frac{2\overline{T}_p \cdot C}{\sqrt{n}}$	$\text{Util} \geq 99.99997\ \%$

Short Flows

- Defined in paper as TCP flows that do not exit slow start. Modeled as *independent* bursts of [power of two] packets
- When mixed with long-lived flows, short flows contribute little
- Short flows benefit because of decreased delays

Simulations



Measurements on a Physical Router

TCP Flows	Router Buffer $\frac{RTT \times BW}{\sqrt{n}}$	Pkts	RAM	Link Utilization (%) Model	Sim.	Exp.
100	0.5 x	64	1 Mbit	96.9%	94.7%	94.9%
100	1 x	129	2 Mbit	99.9%	99.3%	98.1%
100	2 x	258	4 Mbit	100%	99.9%	99.8%
100	3 x	387	8 Mbit	100%	99.8%	99.7%
200	0.5 x	46	1 Mbit	98.8%	97.0%	98.6%
200	1 x	91	2 Mbit	99.9%	99.2%	99.7%
200	2 x	182	4 Mbit	100%	99.8%	99.8%
200	3 x	273	4 Mbit	100%	100%	99.8%
300	0.5 x	37	512 kb	99.5%	98.6%	99.6%
300	1 x	74	1 Mbit	100%	99.3%	99.8%
300	2 x	148	2 Mbit	100%	99.9%	99.8%
300	3 x	222	4 Mbit	100%	100%	100%
400	0.5 x	32	512 kb	99.7%	99.2%	99.5%
400	1 x	64	1 Mbit	100%	99.8%	100%
400	2 x	128	2 Mbit	100%	100%	100%
400	3 x	192	4 Mbit	100%	100%	99.9%

Conclusion

- Backbone routers may have way more memory than needed
  - Theory, simulations, and physical measurements all agree
- Next step is to try it out on a real backbone router
- But good luck convincing router vendors or network operators
  - In the future, necessity may force B < RTT*C
  - Cisco very interested in this