


Copyright © William C. Cheng




## Key Ideas

- Want better fairness without having per-flow state
- Avoid synchronization
- drop packets randomly
- Allow bursty traffic (internet traffic is very bursty)
  - by keep queues small, can handle bursts in your buffer
- Do early congestion feedback
  - drop packets early (before queue is full)
  - get people to slow down quickly (don't wait)

Computer Communications - CSC1 551

Copyright © William C. Cheng




## Why Active Queue Management? (RFC2309)

- Lock-out problem
  - drop-tail allows a few flows to monopolize the queue space, locking out other flows (due to synchronization)
- Full queues problem:
  - drop tail maintains full or nearly-full queues during congestion; but queue limits should reflect the size of bursts we want to absorb, not steady-state queuing

Computer Communications - CSC1 551

Copyright © William C. Cheng




## Solving the Full Queues Problem

- Drop packets before queue becomes full (early drop)
  - Intuition: notify senders of incipient congestion
  - example: early random drop (ERD):
    - if  $qlen > \text{drop level}$ , drop each new packet with fixed probability  $p$
    - does not control misbehaving users

Computer Communications - CSC1 551

Copyright © William C. Cheng



# CS551

## Random Early Detection (RED)


[Floyd93a]

Bill Cheng

<http://merlot.usc.edu/cs551-f12>

Computer Communications - CSC1 551

Copyright © William C. Cheng




## Random Early Detection (RED)

- Motivation:
  - high bandwidth-delay flows have large queues to accommodate transient congestion
  - TCP detects congestion from loss - after queues have built up and increase delay
- Aim:
  - keep throughput high and delay low
  - accommodate bursts (leave some room)

Computer Communications - CSC1 551

Copyright © William C. Cheng



## Other Options

- Random drop:
  - packet arriving when queue is full causes some random packet to be dropped
- Drop front:
  - on full queue, drop packet at head of queue
- Random drop and drop front solve the lockout problem but not the full-queues problem

Computer Communications - CSC1 551

Copyright © William C. Cheng

12

### Thresholds

- min<sub>h</sub> determined by the utilization requirement
- needs to be high for fairly bursty traffic
- max<sub>h</sub> set to twice min<sub>h</sub>
- rule of thumb
- difference must be larger than queue size increase in one RTT
- bandwidth dependence

Computer Communications - CSC 551

Copyright © William C. Cheng

10

### Red Algorithm

```

Initialization:
avg ← 0
count ← 1
for each packet arrival
  calculate the new average queue size avg:
  if the queue is nonempty
    avg ← (1 - wq)avg + wqq
  else
    m ← f(time - qtime)
    avg ← (1 - wq)avg
  if minh ≤ avg < maxh
    increment count
  calculate probability ps:
  ps ← max(avg - minh, 0) / (maxh - minh)
  ps ← ps / (1 - count / m)
  mark the arriving packet
  with probability ps
  count ← 0
  else if maxh ≤ avg
    mark the arriving packet
    count ← 0
  else count ← 0
  when queue becomes empty
    else count ← -1
  else count ← 0
  
```

Computer Communications - CSC 551

Copyright © William C. Cheng

8

### RED Goals

- Detect incipient (soon to happen) congestion, allow bursts
- keep power (throughput/delay) high
- keep average queue size low
- assume hosts respond to lost packets
- Avoid window synchronization
- randomly mark packets
- Avoid bias against bursty traffic
- Some protection against ill-behaved users

Computer Communications - CSC 551

Copyright © William C. Cheng

11

### Queue Estimation

- Standard EWMA:  $avg = (1 - w_q) avg + w_q q$
- Upper bound on  $w_q$  depends on min<sub>h</sub>
- want to set  $w_q$  to allow a certain burst size L
- Equation: 
$$L + 1 + \frac{L}{1 - w_q} > \frac{L}{1 - w_q}$$
- Ex: min<sub>h</sub>=5 and L=50,  $w_q > 0.0042$
- Lower bound on  $w_q$  to detect congestion relatively quickly
- Ex:  $w_q = 0.002$

Computer Communications - CSC 551

Copyright © William C. Cheng

9

### RED Operation

Computer Communications - CSC 551

Copyright © William C. Cheng


7

### Differences With DEC-bit

- Random marking/dropping of packets
- Exponentially weighted queue lengths
- Senders react to single packet
- Rationale: exponential weighting better for high bandwidth connections
- no bias when weighting interval different from roundtrip time, since packets are marked randomly
- random marking avoids bias against bursty traffic

Computer Communications - CSC 551

Copyright © William C. Cheng




### Red Penalty Box

- ▶ With RED, monitor history for packet drops, identify flows that use disproportionate bandwidth
- ▶ Isolate and punish those flows

Computer Communications - CSC1 551

Copyright © William C. Cheng




### RED Variants

- ▶ FRED: Fair Random Early Drop (Sigcomm, 1997)
  - = maintain per flow state only for active flows (ones having packets in the buffer)
- ▶ CHOKe (choose and keep/kill) (Infocom 2000)
  - = compare new packet with random pkt in queue
  - = if from same flow, drop both
  - = if not, use RED to decide fate of new packet

Computer Communications - CSC1 551

Copyright © William C. Cheng




### Extending RED for Flow Isolation

- ▶ Problem: what to do with non-cooperative flows?
- ▶ Fair queuing achieves isolation using perflow state
  - = expensive at backbone routers
- ▶ Pricing can have a similar effect
  - = but needs much infrastructure to be developed
- ▶ How can we isolate unresponsive flows without per-flow state?

Computer Communications - CSC1 551

Copyright © William C. Cheng



### Packet Marking

- ▶ Marking probability based on queue length
  - =  $P_b = \max_p(\text{avg} - \text{min}_h) / (\text{max}_h - \text{min}_h)$
- ▶ Just marking based on  $P_b$  can lead to clustered marking
  - = global synchronization
- ▶ Better to bias  $P_b$  by history of unmarked packets
  - =  $P_b = P_b / (1 - \text{count} \times P_b)$
  - where count is the number of unmarked packets that have arrived since the last marked packet

Computer Communications - CSC1 551