RED:
Floyd and Jacobson [Floyd93a]

CSci551: Computer Networks
SP2006 Thursday Section

John Heidemann

Key ideas

- goal to reduce congestion
  - also wants to keep queue short
- probabilistic detection
  - RED: Random Early Detection
    - random: to encourage fairness
    - early: signal the senders to slow down *before* there's congestion
    - detection: hope to detect and prevent congestion
- remember, in TCP congestion avoidance, it's always sending a little bit more

E-mail Question: Global Synchronization

- define "global synchronization"
  - when independent processes become synchronized
  - problem if you want to assume random behavior
- example: airport gate arrival
  - at LAX, people go through security randomly, arrive at gate randomly
    - if they have to wait at the gate to talk to an agent, they all see *average* delay
    - variance is low
  - at IAD, people go through security randomly, but they have to take a shuttle bus to the gate; this *synchronizes* them
    - so if they have to talk to an agent at the gate, one always sees zero delay, and one always sees maximum delay
    - over all people, delay is the same, but *variance is high*
  - (people hate variance!)

Why we need active queue management (RFC-2309)

- Lock-out problem
  - drop-tail allows a few flows to monopolize the queue space, locking out other flows
  - want to allow rapid convergence on fairness
- Full queues problem:
  - drop tail maintains full or nearly-full queues during congestion
  - want short queues to allow bursts (not persistent queues)

Prior Work

- 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 lock-out problem but not the full-queues problem
- what is needed to reduce queues? RED: random droppping for fairness, and early dropping to prevent congestion (earlier than drop head)

Solving the full queues problem

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

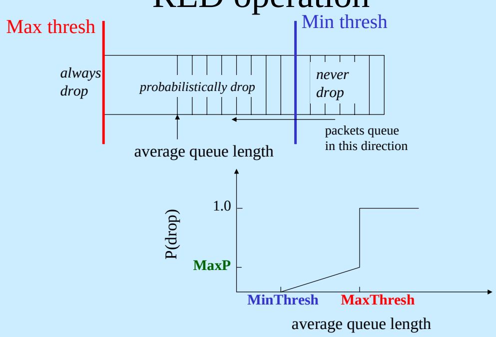
Differences with DEC-bit

- DECbit sends signal when queue is long
- RED has min-thresh/max-thresh
- sender reacts to RED as if a drop
  - MD on one signal
- in DEC bit… look at many bits, only do MD when most of the bits are congestion
- RED emphasizes randomness (not DECbit)

RED Goals

- Detect incipient 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
  - burst traffic is short term, not long-term congestion
  - burst traffic happens commonly in the internet (ex.: at TCP connection start or resumption, also in applications, like clicking on web links)
  - designed with TCP in mind
- Some protection against ill-behaved users

RED operation



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 − w_q)avg + w_q q
        else
            m ← f(time − q_time)
            avg ← (1 − w_q)^m avg
    if min_th ≤ avg < max_th
        increment count
        calculate probability p_a:
            p_b ← max_p(avg − min_th)/(max_th − min_th)
            p_a ← p_b/(1 − count · p_b)
        with probability p_a:
            mark the arriving packet
            count ← 0
    else if max_th ≤ avg
        mark the arriving packet
        count ← 0
    else count ← −1
when queue becomes empty
    q_time ← time
```


Queue estimation

- Standard EWMA:
  - avg' := (1-w_q) avg + w_q qlen
- Upper bound on w_q depends on min_{th}
  - want to set w_q to allow a certain burst size to pass without reacting
- Lower bound on w_q to detect congestion relatively quickly

⇒ w_q around 0.002

Thresholds

- min_{th} determined by the utilization requirement
  - Needs to be high for fairly bursty traffic
- max_{th} set to twice min_{th}
  - Rule of thumb
  - Difference must be larger than queue size increase in one RTT
    - Bandwidth dependence

Packet marking

- Marking probability based on queue length
  - $Pb = max_p(avg - min_{th}) / (max_{th} - min_{th})$
- Just marking based on Pb can lead to clustered marking -> global synchronization
- Better to bias Pb by history of unmarked packets
  - Pb = Pb/(1 - count*Pb)

Marking vs. Dropping

- RED technically talks about *marking* packets
  - but ECN is late to the Internet
  - ⇒ uses dropping if marking not available

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

Reviewing RED

- what is the key idea behind RED?
  - xxx

Setting RED Parameters

- not completely obvious
- guidelines
  - w_q (queue EWMA constant): >0.001, to react to bursts “fast enough”
  - minthresh: “high enough to maximize power” (allowing some bursts)
  - maxthresh:
    - should be about the bw-delay product (to handle full burst of data)
    - should be at least 2x minthresh
- maybe *no* very good parameters , at least for web traffic
  - see “Tuning RED for Web Traffic”, Christiansen et al, SIGCOMM 2000

Other observations

- xxx