# 6.896 Lecture 6: Congestion Control Practice

Hari Balakrishnan

September 30, 1998

## 1 Outline

- **Readings: FJ93, DKS89.**
- Why do we need gateway support to combat congestion?
- RED.
- Fair queueing.
- Discussion of project topics (if time permits).

## 2 Gateway support for congestion management

- Can't assume cooperating sources any more.
- Even if we did, biases exist that prevent fair allocation (e.g., rtt biases).
- Danger of congestion collapse: (beyond cliff of throughput-delay curve).
- What forms might congestion collapse take?
  1. *Classical collapse:* Redundant retransmissions.
  2. *Fragmentation-based collapse:* Loss of fragments causes entire packet to be retransmitted.
  3. *Control traffic-triggered collapse:* E.g., increased amounts of multicast join/prunes, RTCP traffic, packet headers, routing updates, etc.
  4. *Collapse from stale packets:* Network carries packets that are no longer wanted by user, because of long delays. Or because of ill-informed *push* data to users.
  5. *Undelivered packets:* This is the hardest form to tackle, and is caused by unresponsive traffic. Some applications also incrase their bandwidth use upon detecting loss, sometimes by increased FEC (FEC in itself isn't bad, it's the increased bandwidth use that is).
- Three ways of tackling this:
  1. Flow isolation and per-flow scheduling.
  2. Router mechanisms that encourage end-to-end adaptation to congestion.
  3. Pricing.
- We will discuss RED and its applicability to #2 and fair queueing as a mechanism for #1.

# 3 RED

- Goals: control average queue size and at the same time, and accomodate bursty traffic.
- Key idea: use randomization to avoid synchronizations and biases that could occur in deterministic approaches.
- Basic idea: If average queue size, $q_a$ is between $min_{th}$ and $max_{th}$ implies *mark* or *drop* packet with some probability, $p_a$. If $q_a > max_{th}$, then drop packet. If $q_a < min_{th}$, then do nothing.
- If marking is used and TCP and other sources understand this, we can avoid some losses. This is called *Explicit Congestion Notification (ECN)*.
- Differences from DECbit:
    1. Method for computing $q_a$. Regeneration vs. EWMA. Latter explicitly controls time constant for average calculation. Biases don't arise because of randomization in packet marking.
    2. Use of randomization. Avoids synchronizations. (Note that this is more likely to happen with TCP's 0.5 decrease factor).
    3. Which connections to notify of congestion? Claim is DECbit could penalize bursty traffic. (This isn't clear, esp. since DECbit was later modified to allow selective feedback).
    4. RED much more in tune with TCP; DECbit tuned for DNA's transport protocol.
- RED can also handle misbehaving or uncooperative sources, by penalizing them ("penalty box").
- Two key details: how to compute $q_a$ and how to compute $p_a$.

## 3.1 Computing $q_a$

- $q_a = (1 - w_q)q_a + w_q q$. $w_q$ is time constant of low pass filter. Too large implies rapid response to *transient* congestion. Too low implies sluggish behavior.
- Get upper bound based on how much burstiness you want to allow.
- Get lower bound based on how much memory you want to maintain.
- In practice, Floyd sets $w_q = 0.002$.
- $max_{th} \geq 2min_{th}$.

## 3.2 Computing $p_a$

- Initially, compute $p_b = max_p(q_a - min_{th})/(max_{th} - min_{th})$. This is standard linear interpolation.
- Method #1: Geometric r.v. Supose each packet is marked with probability $p_a = p_b$. Consider the r.v $X =$ number of packets between successive marked packets. Then $P(X = n) = (1 - p_b)^{n-1}p_b$. This is a geometric distribution.
- Unfortunately, this doesn't mark packets at fairly regular intervals. $E[X] = 1/p_b$, but because packets get marked in bunches, more synchronization than is good could occur.

- Method #2: Uniform r.v. Let $X$ be uniform in $\{1, 2, \ldots, 1/p_b\}$. This has the nice property that we want, and is achieved if we set $p_a = p_b/(1 - count p_b)$, where $count$ is the number of unmarked packets since last marked packet. $E[X] = 1/(2p_b) + 1/2$.

- $max_p$ should never have to be large. (Anyway, after $q_a > max_{th}$, all packets are dropped).

### 3.3 Encouraging end-to-end responsiveness

- Use RED drop (mark) history to classify ill-behaved flows.

- In the long run, connection's share of marked packets proportional to its share of bandwidth. Note however that RED doesn't provide flow isolation and isn't fair in the fair queueing ense of max-min fairness.

- Put them in a *"penalty box"* and don't service them. I.e., provide a *disincentive* to be unresponsive to congestion.

- Using this, it seems possible to identify long-running ill-behaved flows. But what about short or bursty bad flows?

## 4 Announcements

- Today at 4pm in NE43-518: Tom Anderson's talk on Detour.

- Next tuesday at 4pm in 37-212: Arthur Berger's talk on dimensioning bandwidth for elastic apps.

- Office hrs this afternoon 2-4.

## 5 Fair queueing

- One of the earliest practical schemes to achieve per-flow isolation in the network.

- Gateway algorithms perform (1) queue management and (2) scheduling. These are orthogonal to each other. Q: what do the terms mean?

- Queue management manages buffers. Scheduling: bandwidth and delay. Which one does RED do? What about FQ?

- Define work-conserving and non-work-conserving scheduling disciplines.

- FQ based on earlier work by Nagle on round-robin scheduling. Q: What's the main problem with this approach?

- Similar to Zhang's virtual clock algorithm.

- Goals: provide per-flow isolation, allocate bandwidth and buffers fairly, control bandwidth and delay independently.

- *max-min* fairness definition. Ask question to recap.

### 5.1 The Algorithm

- Allocating buffer space fairly is trivial. How?
- Emulate bit-by-bit round robin (RR), which is obviously fair.
- Think of packet bits being sent in rounds. Suppose there are $N_a$ active flows. In each round, one bit per flow is processed and sent out.
- Let $R(t)$ be the number of rounds in the RR service discipline until time $t$. Then, $dR/dt = \mu/N_a$. Why?
- Suppose packet first gets serviced at time $t$. In how many more rounds does it get sent out? ($P$, the size of the packet).
- Draw picture of $N$ existing flows. Suppose packet of size $P_i^\alpha$, the $i^{th}$ packet of connection $\alpha$, arrives at time $t_i^\alpha$. Then, if $S_i^\alpha$ is the starting round and $F_i^\alpha$ the finishing one, clearly $F_i^\alpha = S_i^\alpha + P_i^\alpha$ (note that we're in the "rounds" domain, not time).
- In addition, we have $S_i^\alpha = max(F_{i-1}^\alpha, R(t_i^\alpha))$.
- $R(t)$ is non-decreasing, so ordering of finish values $F_i^\alpha$ is the same as finishing times in bit-by-bit RR.
- Scheduling algorithm is now straightforward: send packets out by earliest finishing round values.
- What happens when a new flow arrives or a flow completes (times out)?
- Can show that this emulation is accurate to within one maximum-sized packet.
- Delay scheduling: use a parameter $\delta$ and modify rule according to *bids*, $B_i^\alpha = max(F_{i-1}^\alpha, R(t_i^\alpha) - \delta$. Sending order determined by bids.
- Paper has analytical and simulation results that show the good properties of fair queueing.
- New congestion control schemes in conjunction with FQ: Keshav's packet pair.
- Problems: per-flow state, could be problematic in high-speed routers (but then again, recent work shows this might not be a big problem; also see work on SFQ and C-SFQ), weights to different flows (addressed by WFQ).
- Lots of subsequent work on scheduling: Parekh & Gallager (GPS), Golestani (stop-and-go), Knightly, Zhang, and the Tenet group (RCSP etc.), Guerin et al. at IBM, and many, many others.

## 6 Discussion of project topics

- Be prepared with more than one choice. Don't want 10 people doing the same thing!
- 3-member groups OK for some projects. Come and talk to me this afternoon during office hours if you have any questions or concerns.