Packet queue control through a rose-colored lens

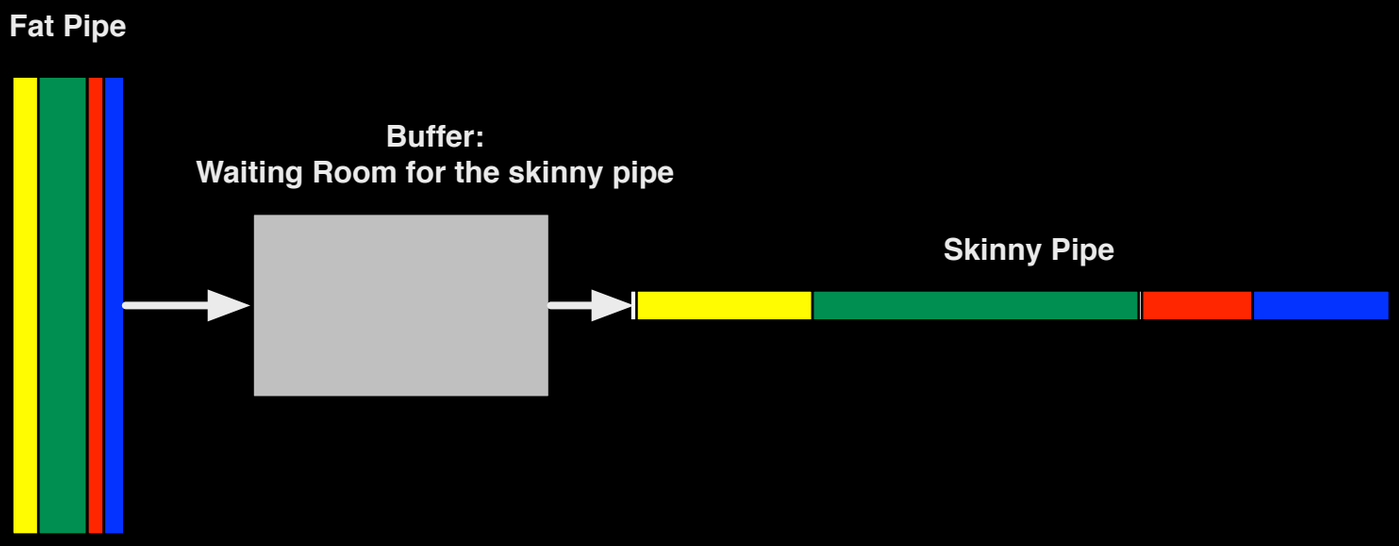
Kathleen Nichols
April 26, 2011

Acknowlegement to Van Jacobson for brilliant insights, baffling misdirections, heated disagreements, and the shared joy of discovery

- Packet networks must have buffers to absorb short-term arrival rate fluctuations
- A pesky side effect is the tendency for buffers at busy fast-to-slow transitions to fill up causing excessive delay and removing the ability to absorb bursts creating long *persistent queues* of packets
- This became a hot topic in Internet circles starting at the end of the 80s
- The idea of randomly dropping packets and other "drop preference" approaches to Adaptive Queue Management (AQM) were proposed
- Random Early Detection algorithm paper published by Floyd and Jacobson in 1993 generated a lot of interest
- Issues with the algorithm and misconceptions about dropped packets have kept AQM from being deployed

The Internet has packet buffers to handle the normal burstiness of statistically multiplexed networks - good

Persistent or standing queues of packets develop at "fast to slow" transitions - not so good



A 1KB packet takes 0.08 ms @ 100 Mbps, 1.6ms @ 5Mbps, 16ms at 500 Kbps

- A long-lived transfer should try to keep the "pipe" full: means there should be a round trip time (RTT) worth of packets in flight. This "pipe" is just the latency of traversing links along the way

- Buffers for small to moderate links should be order 1-2 RTTs (100ms) to handle transients, otherwise severe dropping during connection start ups causes timeouts and link starvation

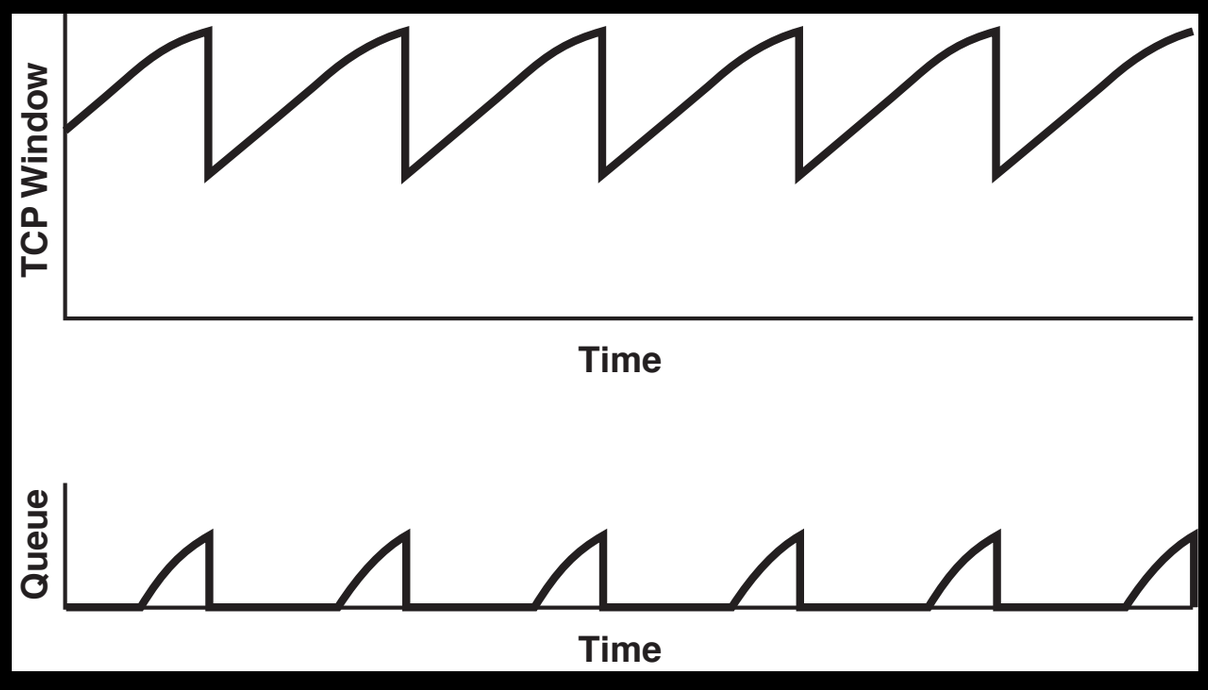
- A persistently full buffer means good link utilization *but* an order of RTT delay *added* to each packet (more if buffers are oversized)

- If instead a small packet queue persists in each buffer, good link utilization and a fraction of the delay

- RED and other AQM attempt to ensure this "sweet spot": always something ready to send on the link but not a lot of packets hanging around

To control traffic, understand it

- Most packets transported using TCP/IP
  - Lots more variations in 2011 than in 1990s
  - One invariant - takes an RTT+ for response (to a drop)
  - Multiple drops to same flow within an RTT cause pathologies (e.g. timeouts)
- Want to orchestrate dropping pattern to keep the *network link* pipe full, not the network buffers

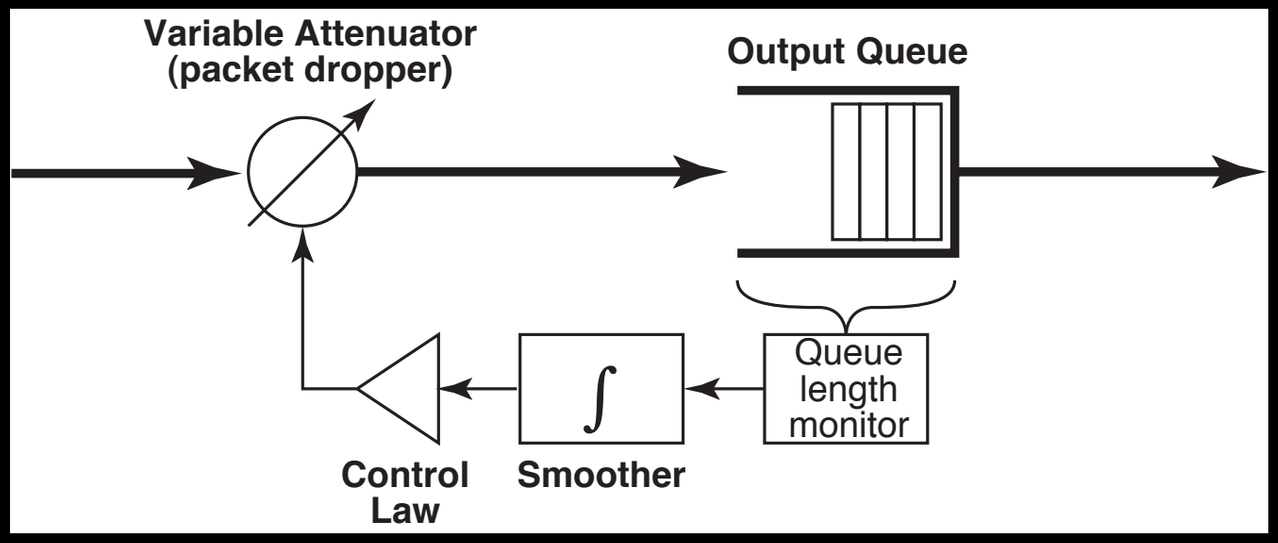


TCP opens its congestion window until it gets a drop. As the number of packets in the congestion window grows, eventually creates a queue at bottleneck link

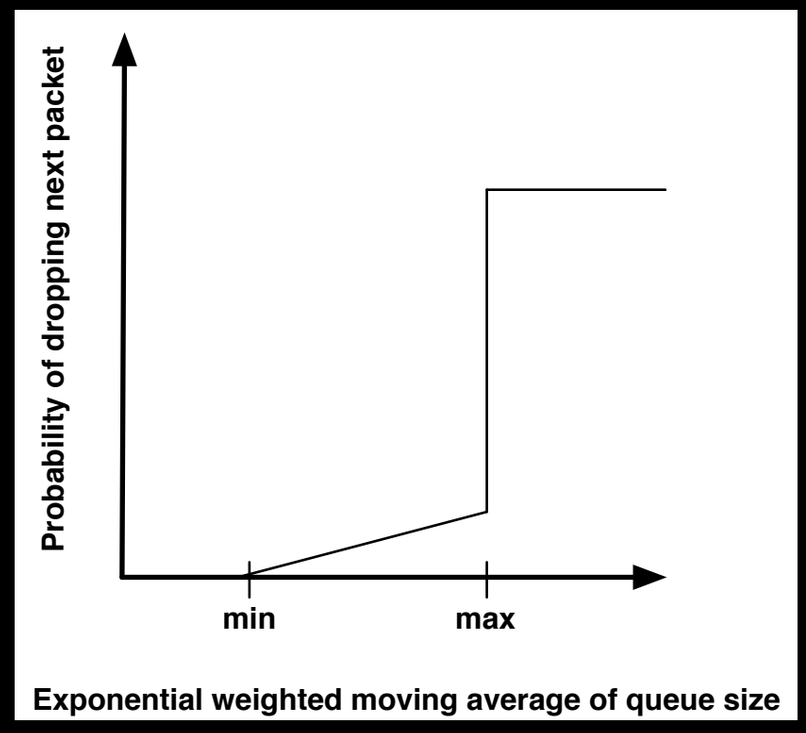
Major Features of RED

- Uses a running estimate of the average queue size over recent past to indicate congestion
- Uses this value as an input to a control law that determines the probability that the next arriving packet is dropped
- This probability is used to calculate a count of packets to arrive before a drop is performed. Drop probability increases with average queue
- If the buffer is full, packets are "tail dropped"

RED Queue Contoller



Control law has three operating ranges: above max, drops 100%



- Output queue size is observed and its samples are smoothed with an EWMA
- EWMA is used to select a probability for the packet dropper
- Packet dropper uses this to randomly select a count of packet arrivals before dropping

Floyd and Jacobson's RED algorithm became the *de facto* AQM implemented

- It was simple
- It was elegant
- Less well known at that time...it was seriously flawed

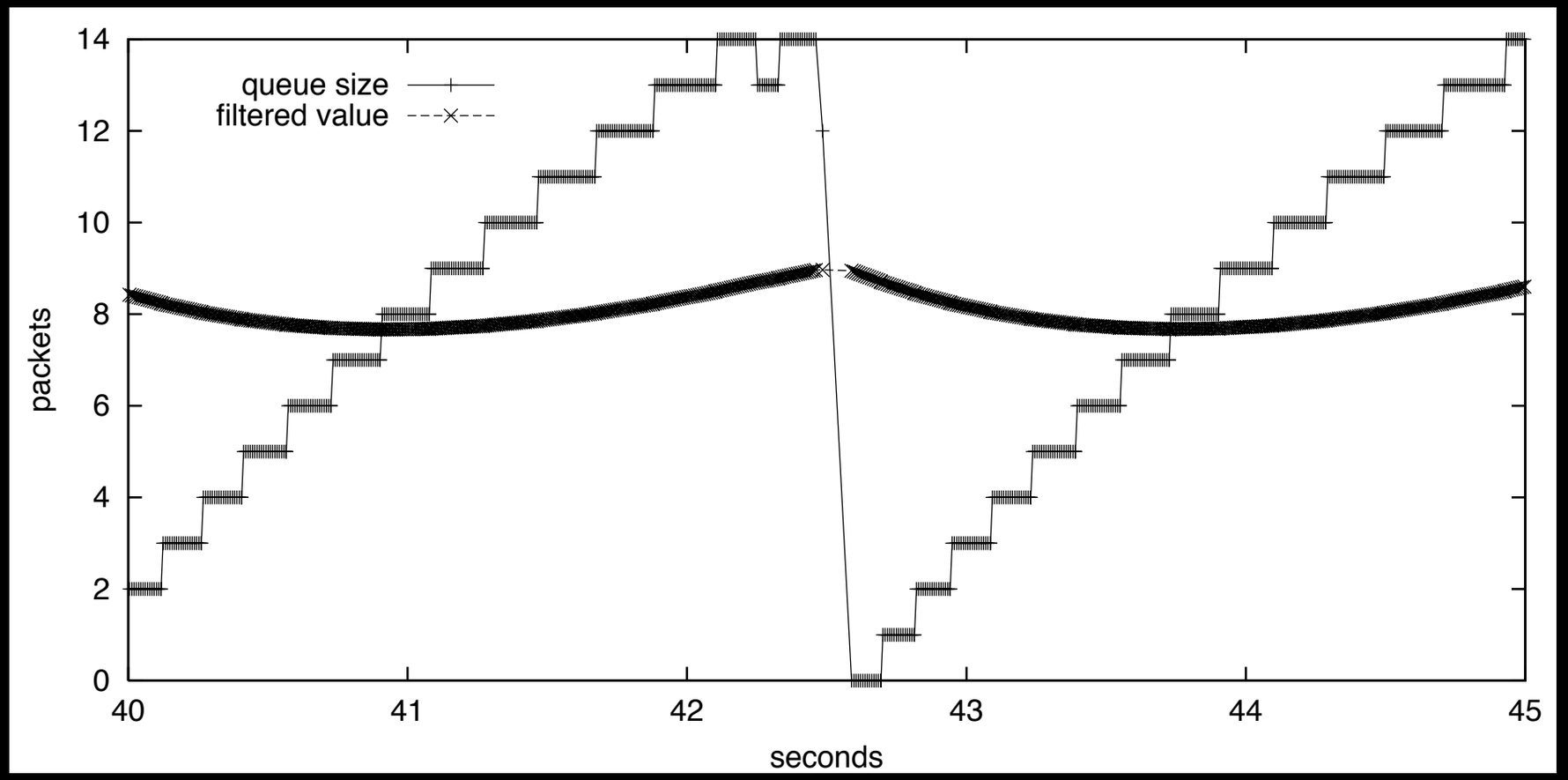
In 1998

- RFC 2309 (aka "The RED Manifesto") was published, recommending that network operators use AQM, specifically RED
- At BAL, my manager says "hey, we need to know how to adjust RED for different bandwidths"
- Van says "it's trivial"
- So I figure it sounds like fun...

Issues

- EWMA turns out to be problematic
  - can lead to drops on an empty(ing) queue
  - how long to average/filter? Don't follow transient bursts, do follow growing queue
- Queue checked on packet arrivals only
  - Queue size in packets, not time or bytes
- Control law was flawed
  - forced drops (i.e., drop every arrival), not probabilistic drops are the norm
  - heavily and unscientifically parameterized: min_{th}, max_{th}, w_q...more like religion

93 RED's "average" for an FTP ramping up



1.5 Mbps bottleneck, 10 Mbps arrival link, "ack every packet"

Used Floyd's 98 recommendations for parameters: min_{th} =5 pckts, max_{th} =20 pckts, p=0.1, w_q = 0.002. RED drops too infrequent

"RED light"- the search for something better (98-01 or so)

- Filter to find the *persistent (or standing) queue*
- Don't drop arrivals to a rapidly emptying queue
- Use network science to find control law

Turned out it's not so easy:

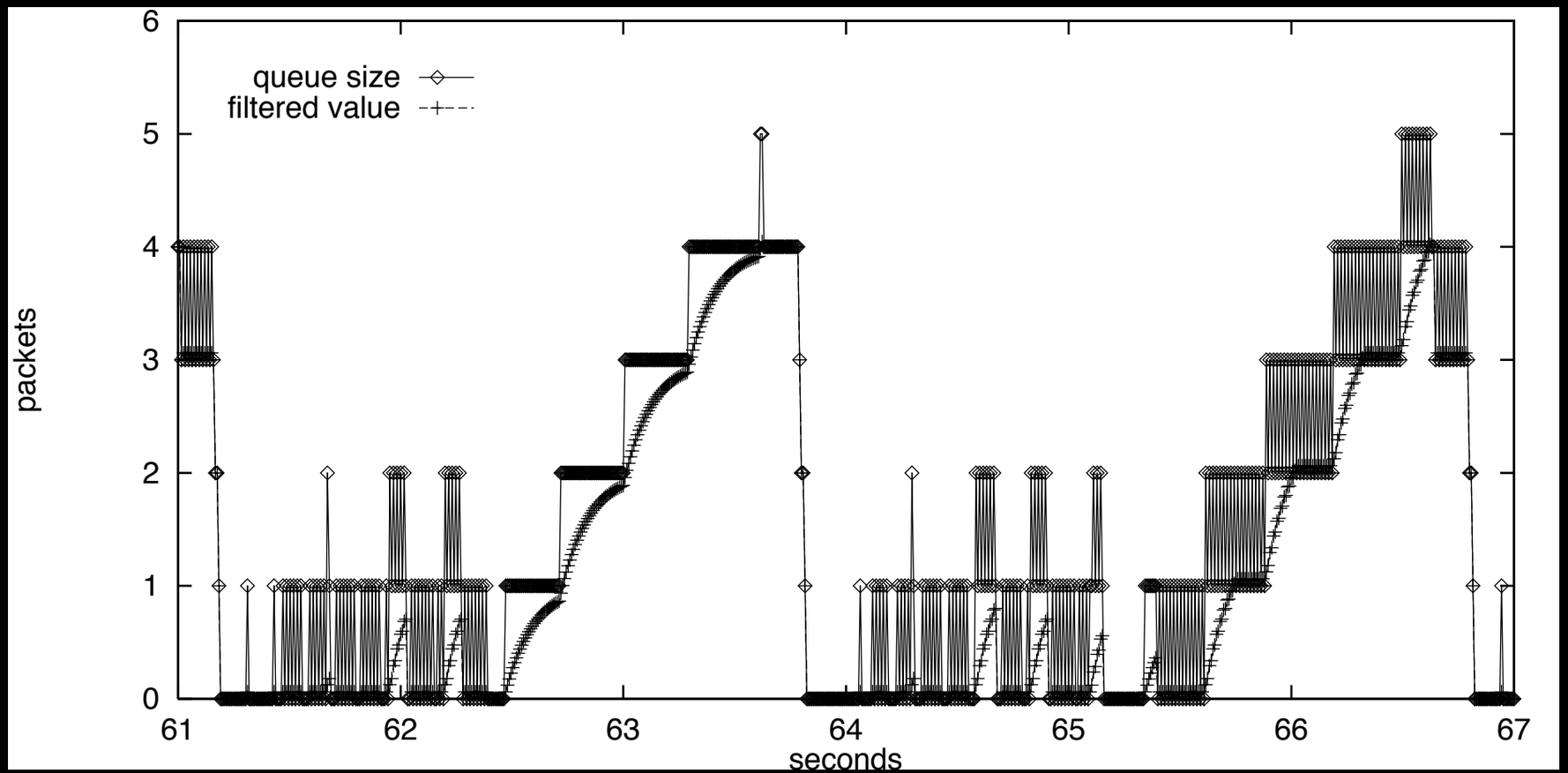
> "Everything works with all long-lived FTPs; nothing works with lots of short transfers (mice)" (for a loose definition of "works")
>
> Just what <u>is</u> the "persistent queue"? How should we detect it?

RL Filtering for Persistent Queue

- Modified filter: follow an emptying queue down, use EWMA* otherwise (i.e. when filter value > q size)
- Use a smaller filter gain (i.e., average over shorter time)
- Used time-based sampling of the queue ~1ms general recommendation - for smaller bandwidths, ~time to send a maximum sized packet.
- Sampling in general should be randomized
- To follow the queue, but not too closely, a filter gain equal to the inverse (rounded down to next power of two) of the number of samples in an RTT (round trip time)

*We later realized sticking with EWMA was probably a mistake

RL at work on the single FTP

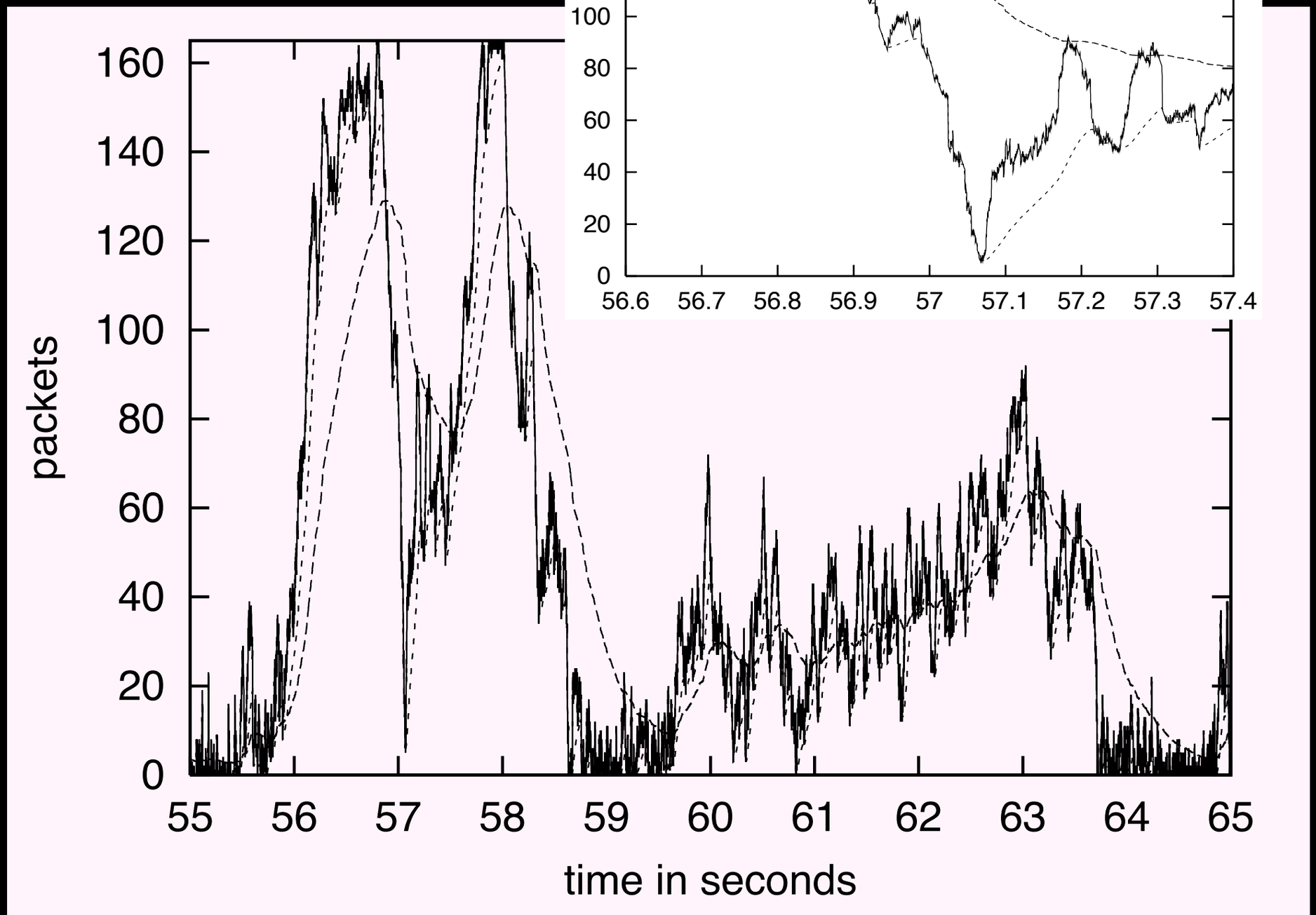


sampled at 1 ms; filter gain of 1/16

Recall that we want to keep a pipesize of packets in flight, but we don't want to let the queue build up too much or we are just introducing delay. Used a minimum threshold of 30 ms. (One max packet takes 8 ms)

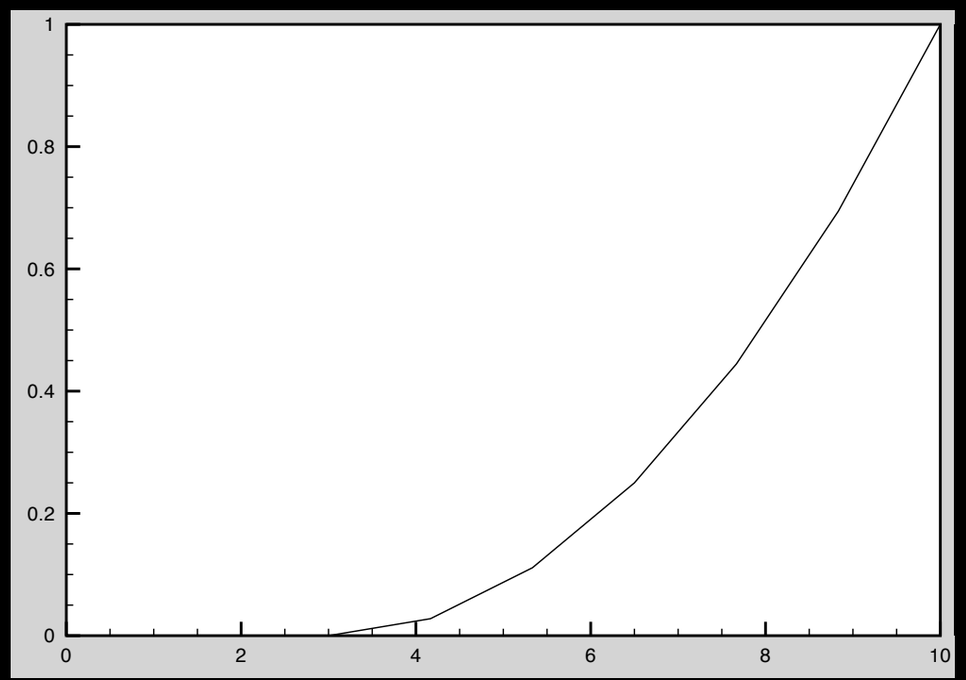
Single FTP	median queue	queue 95th percentile	% tail drops
93	6 pckts	10 pckts	32
light	1 pckts	5 pckts	1

93 RED and RED Light filters on mixed traffic



Sample from simulation: 10Mbps link, mix of traffic. Note that ewma stays high while queue is decreasing

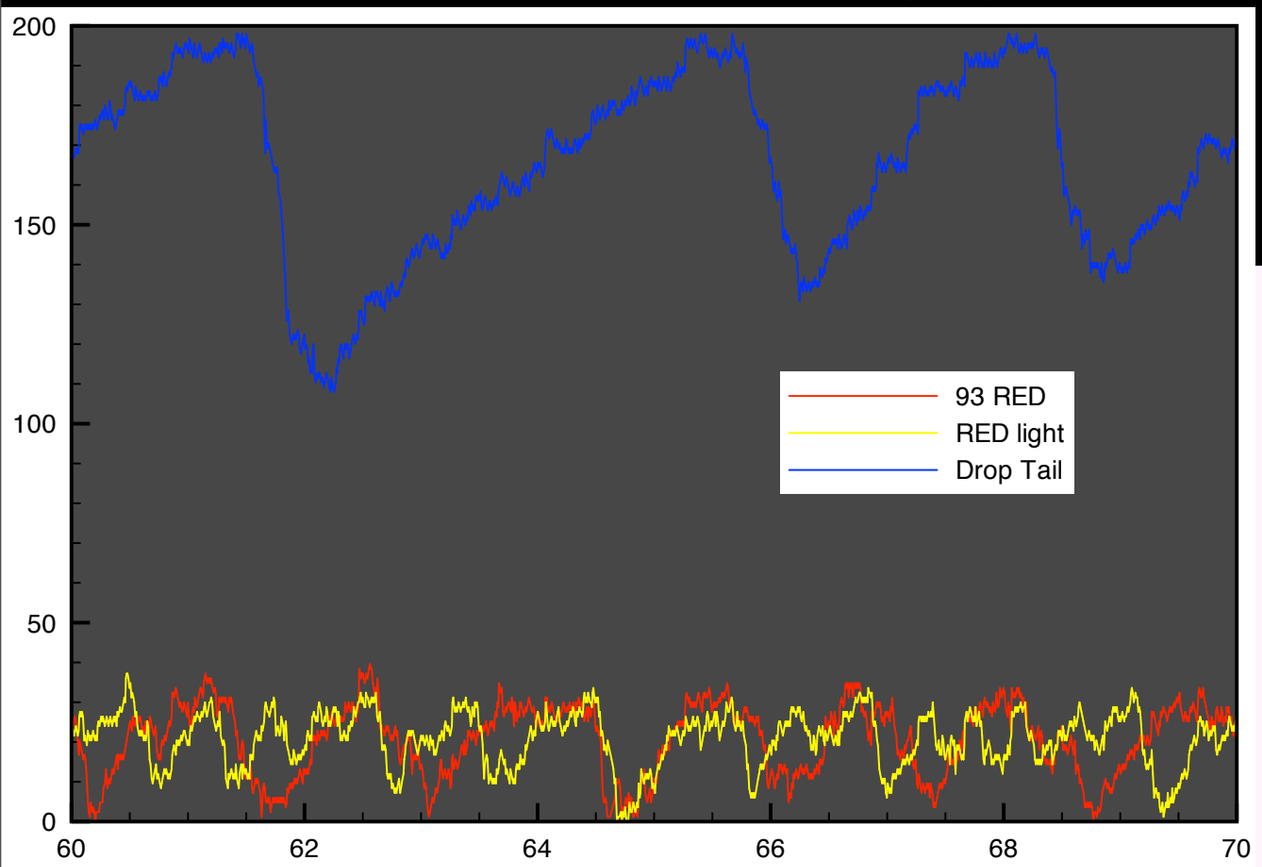
- Van started looking for a control law that would take advantage of the leverage of a TCP drop and drop more if it was apparent (from response) that the multiplexing level was high or flows were not responding
- I started looking for a good way to determine how much persistent queue to tolerate and how to see what's going on
- The search continues, but a decade ago we said:
  - Keep PQ under 10-30% of the buffer size (if a bandwidth-delay product) - smaller for larger bandwidths
  - Small bandwidths (under 4 Mbps) need to be treated differently; provide at least five packets of buffer, allow at least two packets of PQ
  - Exponential control law



Not perfect, but better

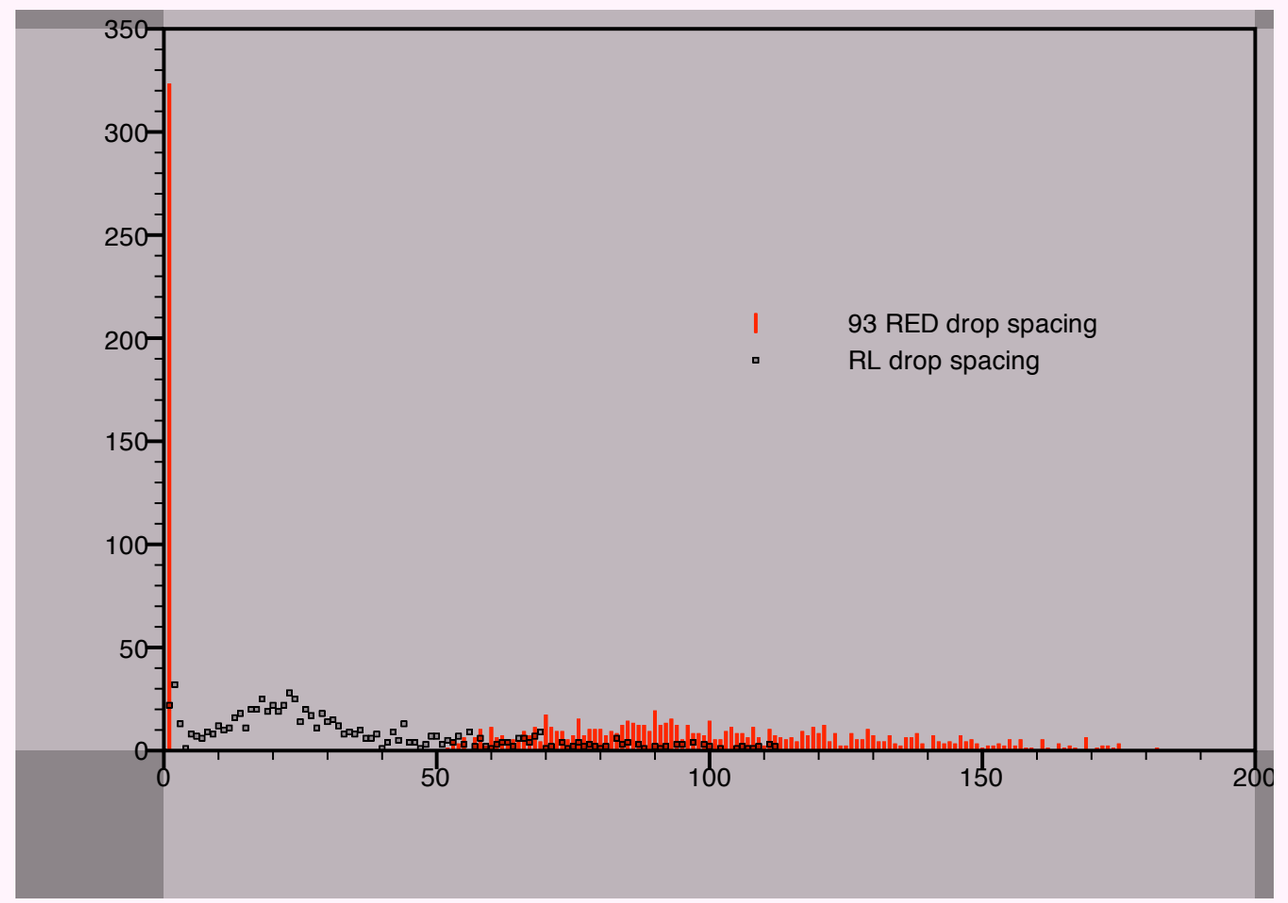
The easy case: long-lived TCPs (elephants)

Instantaneous Queue size (ms)



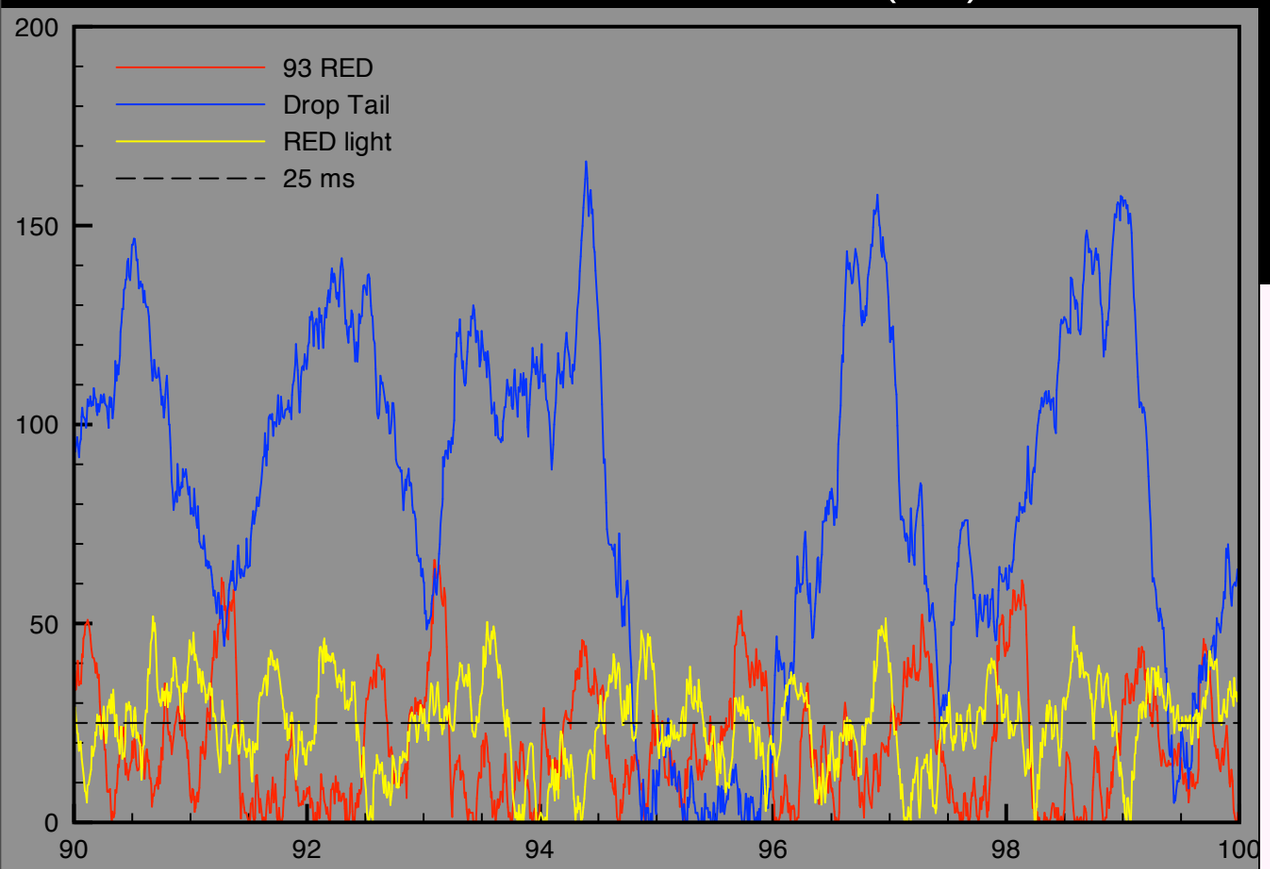
queue shows ten seconds of a 180 sec simulation

Number of packets between drops

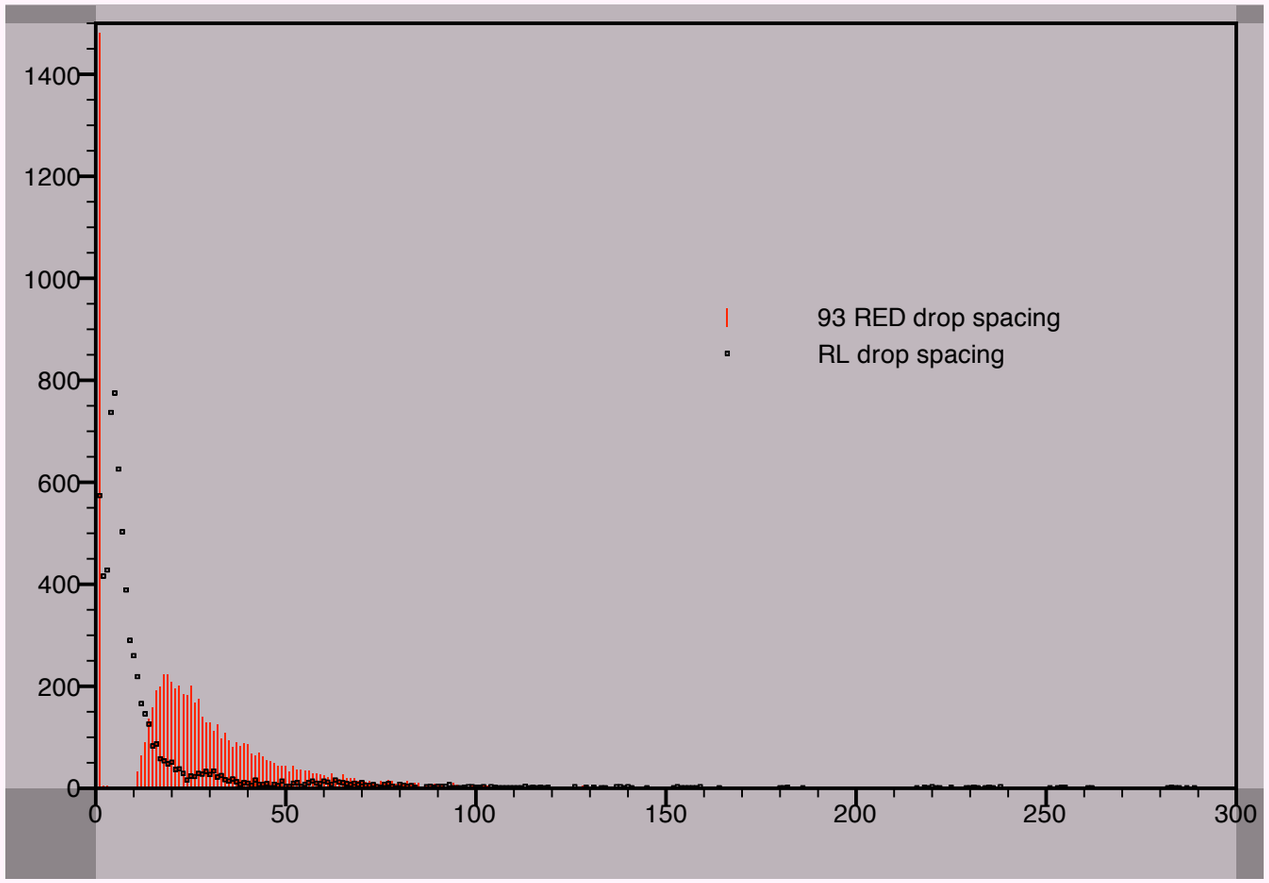


But 93 RED is mostly doing forced drops!

Not perfect, but better



The hard case: elephants meet web mice

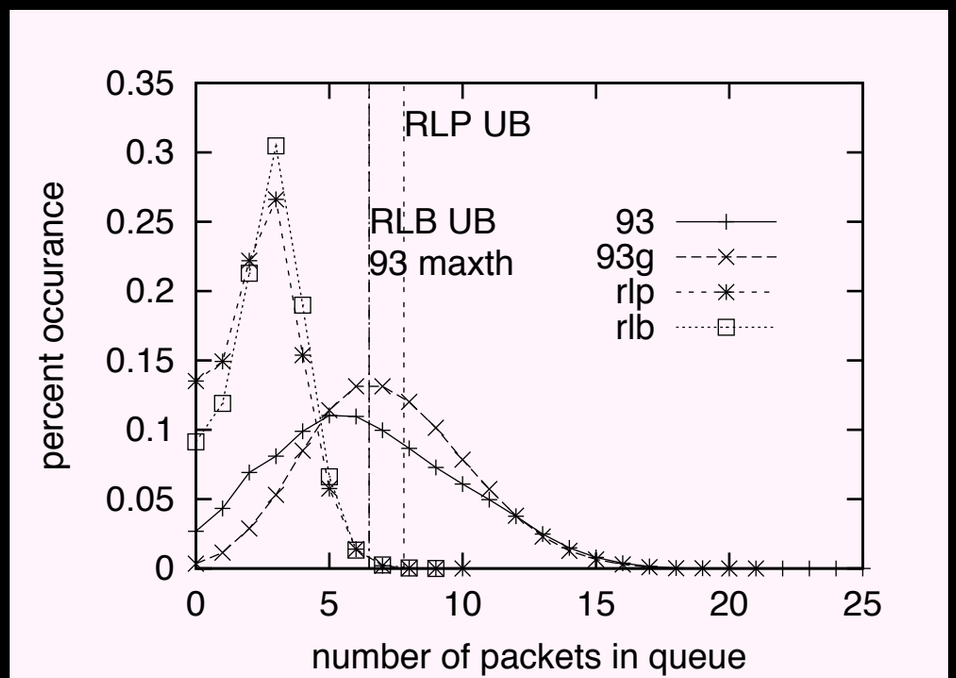


But the RED light controls better:

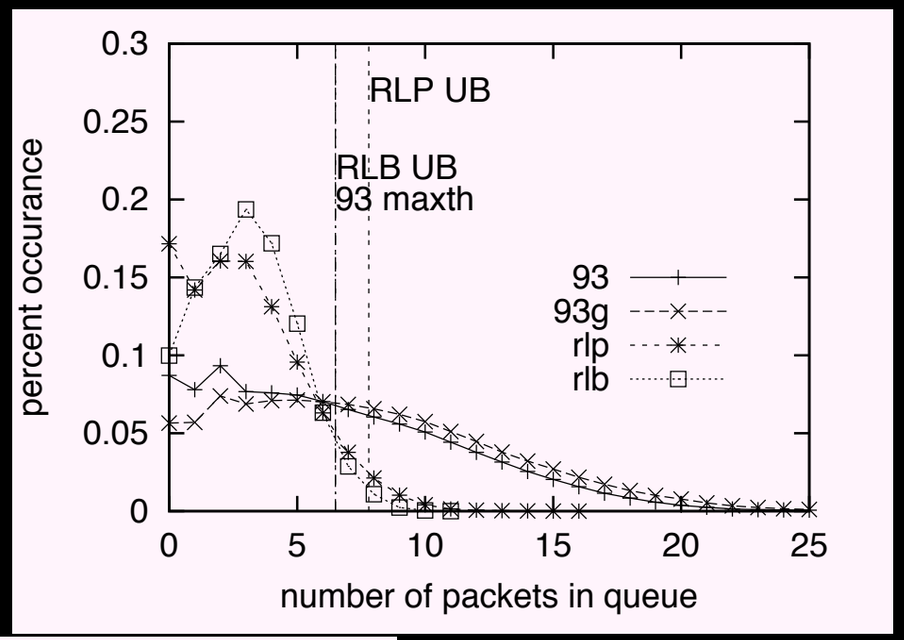
forced drops lead to timeouts

Density of Queue Sizes:

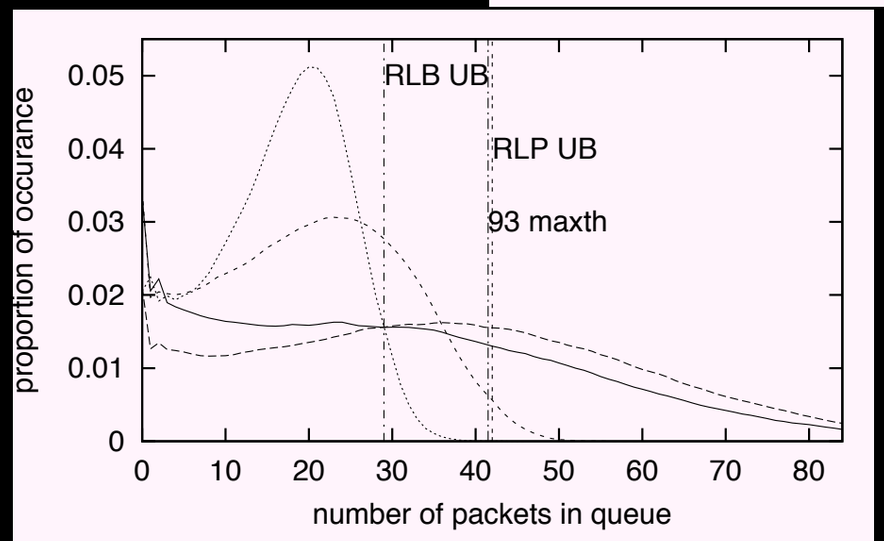
1.5 Mbps link, all large FTPs



mix medium FTPs and mice



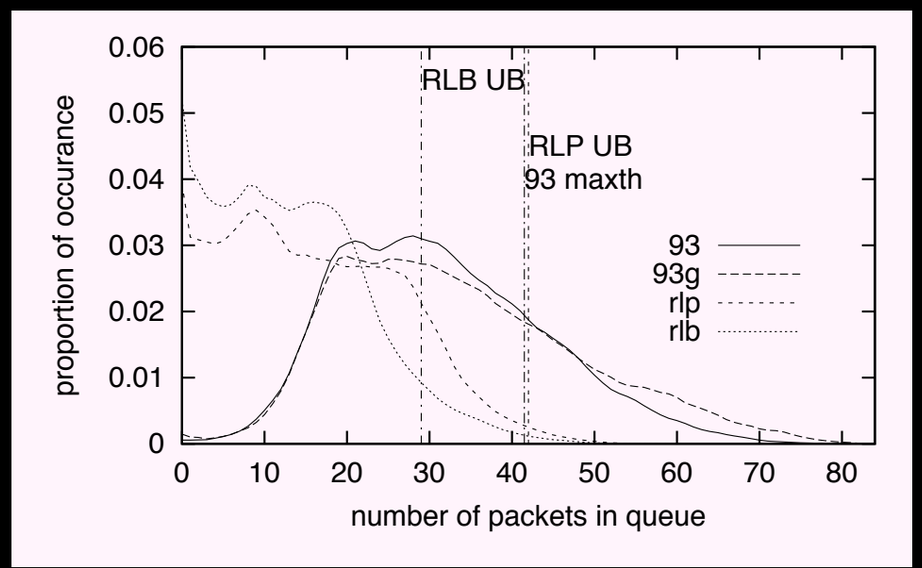
1.5Mbps link



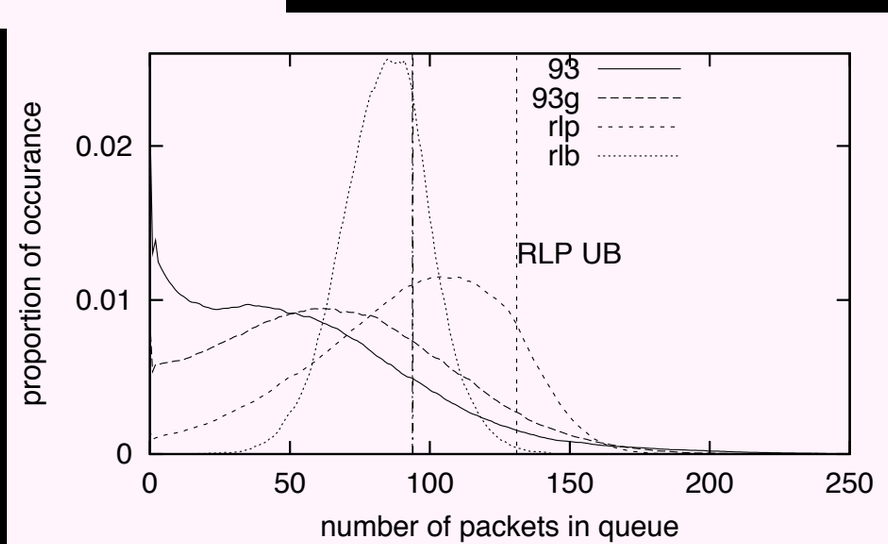
10Mbps link

93 RED params adjusted for bw

10Mbps link, all large FTPs



45 Mbps link



All utilizations were good but RL always better or equal to 93

- Early on, Van talks about it at NANOG, emails parameters with Sean Doran. I write a Bay Architecture Lab Tech Report
- Submitted for publication a couple of times
  - "the authors clearly don't understand RED"
  - one reviewer was offended by the toilet, Van's illustrative control system
- Lixia Zhang puts a draft on a web page for her students
- We do some other related work:
  - large rate links (Cisco TR)
  - RED policer, RED penalty box (Cisco patents)
- Some other papers published in early 2000s found problems with setting RED parameters, but didn't look "beneath the covers" to see what was happening
- Life goes on...

Revisiting RED

- The world has changed a bit (traffic patterns)
- Some other AQMs: BLUE, FRED
- In about 2005/6, Van starts thinking of the queue minimum as robust measure of persistent queue
- Jim Gettys starts getting passionate about "bufferbloat"
- Van asks me if I have any/all of the old RED light documents.
- I check out Jim's website and have a long discussion with Van about revisiting RED...

  (While making these slides, it occurs to me my "visible indicator" of how long I've been at this is now a teenager)

References

- V. Jacobson, “A Rant on Queues”, Talk at MIT Lincoln Labs, Lexington, MA, July, 2006, http://pollere.net/talks.html
- V. Jacobson, "Notes on Using RED for Queue Management and Congestion Avoidance", talk at NANOG 13, ftp://ftp.ee.lbl.gov/talks/vj-nanog-red.pdf, see also http:// www.nanog.org/mtg-9806/agen0698.html, Dearborn, MI, June, 1998.
- Sean Doran, RED Experience and Differentiated Queueing, talk at NANOG 13, http:// adm.ebone.net/smd/red-1.html, Dearborn, MI, June, 1998
- R. Braden et al., "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC2309, April 1998.
- V. Jacobson, reported in “Minutes of the Performance Working Group”, Proceedings of the Cocoa Beach Internet Engineering Task Force, Reston, VA, Corporation for National Research Initiatives, April, 1989.
- A. Mankin, “Random Drop Congestion Control”, Proceedings of SIGCOMM ’90, September 24-27, 1990.
- S. Floyd and V. Jacobson, Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Transactions on Networking, August 1993.

"RED Light"

- Draft paper on our work "RED in a different light"
- Paul Baran who invented much of the foundation of packet switching while at RAND in the early 1960's. (www.rand.org/publications/RM/baran.list.html) wrote in RM-3638-PR (1964), describing a Communications Control Console (or Priority Control Console):

"since gross changes in loading are slowly occurring phenomena; rather, he will normally leave the controls set to fixed positions, except when a crisis or overload approaches as indicated by the red warning light. He then decides which users with growing demands should deprive others with less important duties, and to what degree."

Is that cool or what?