Kathie Nichols’
CoDel

present by Van Jacobson to the
IETF-84 Transport Area Open Meeting
30 July 2012
Vancouver, Canada
• Queue forms at a bottleneck

• There’s probably just one bottleneck (each flow sees exactly one)

⇒ Choices: can move the queue (by making a new bottleneck) or control it.
Good Queue / Bad Queue
Good Queue / Bad Queue
Good Queue / Bad Queue

- Good queue goes away in an RTT, bad queue hangs around.

⇒ queue length min() over a sliding window measures bad queue ...

⇒ ... as long as window is at least an RTT wide.
Good Queue / Bad Queue

- Good queue goes away in an RTT, bad queue hangs around.

⇒ tracking min() in a sliding window gives bad queue ... 

⇒ ... as long as window is at least an RTT wide.
Good Queue / Bad Queue

- Good queue goes away in an RTT, bad queue hangs around.

→ tracking min() in a sliding window gives bad queue ...

→ ...as long as window is at least an RTT wide.
How big is the queue?

- Can measure size in bytes
  - interesting if worried about overflow
  - requires output bandwidth to compute delay

- Can measure packet’s sojourn time
  - direct measure of delay
  - easy (no enque/deque coupling so works with any packet pipeline).
Sojourn Time

• Works with time-varying output bandwidth (e.g., wireless and shared links)

• Better behaved than queue length – no high frequency phase noise

• Includes everything that affects packet so works for multi-queue links
Two views of a Queue

Top graph is sojourn time, bottom is queue size.

(one ftp + web traffic; 10Mbps bottleneck; 80ms RTT; TCP Reno)
Two views of a Queue

Top graph is sojourn time, bottom is queue size.

(one ftp + web traffic; 10Mbps bottleneck; 80ms RTT; TCP Reno)
Two views of a Queue

Top graph is sojourn time, bottom is queue size.

(one ftp + web traffic; 10Mbps bottleneck; 80ms RTT; TCP Reno)
Multi-Queue behavior
Controlling Queue

a) Measure what you’ve got
b) Decide what you want
c) If (a) isn’t (b), move it toward (b)
Controlling Queue

a) Measure what you’ve got
b) Decide what you want
c) If (a) isn’t (b), move it toward (b)

- Estimator
- Setpoint
- Control loop
How much ‘bad’ queue do we want?

• Can’t let the link go idle (need one or two MTU of backlog)

• More than this will give higher utilization at low degree of multiplexing (1-3 bulk xfers) at the cost of higher delay

• Can the trade-off be quantified?
• Setpoint target of 5% of nominal RTT (5ms for 100ms RTT) yields substantial utilization improvement for small added delay.
• Setpoint target of 5% of nominal RTT (5ms for 100ms RTT) yields substantial utilization improvement for small added delay.

• Result holds independent of bandwidth and congestion control algorithm (tested with Reno, Cubic & Westwood).
• Setpoint target of 5% of nominal RTT (5ms for 100ms RTT) yields substantial utilization improvement for small added delay.

• Result holds independent of bandwidth and congestion control algorithm (tested with Reno, Cubic & Westwood).

⇒ CoDel has no free parameters: running-min window width determined by worst-case expected RTT and target is a fixed fraction of same RTT.
Algorithm & Control Law

(see l-D, CACM paper and Linux kernels >= 3.5)
Eric Dumazet has combined CoDel with a simple SFQ (256-1024 buckets with RR service discipline). Cost in state & cycles is small and improvement is big.

- provides isolation - protects low rate CBR and web for a better user experience. Makes IW10 concerns a non-issue.

- gets rid of bottleneck bi-directional traffic problems (‘ack-compression’ burstiness)

- improves flow mixing for better network performance (reduce HoL blocking)

➡ Since we’re adding code, add fqcoodel, not codel.
Where are we?

- thanks to Jim Gettys and the ACM, have dead-tree publication to protect ideas
- un-encumbered code (BSD/GPL dual-license) available for ns2, ns3 & linux
- in both simulation and real deployment, CoDel does no harm – it either does nothing or reduces delay without affecting xput.
What needs to be done

• Still looking at parts of the algorithm but changes likely to be 2nd order.

• Would like to see CoDel on both ends of every home/small-office access link but:
  - We need to know more about how traffic behaves on particular bottlenecks (wi-fi, 3G cellular, cable modem)
  - There are system issues with deployment
Deployment Issues
Deployment Issues

Home Gateway

RTR/AP → Cable Modem → Headend

Linux kernel

Protocol stack → Device Driver → Device
Deployment Issues

Home Gateway:
- RTR/AP
- Cable Modem
- Headend

Linux kernel:
- Protocol stack
- Device Driver
- Device

Cellphone:
- Phone CPU
- 3G Modem
- RAN
- SGSN
Our thanks to:

- Jim Gettys
- CoDel early experimenters, particularly Dave Taht
- Eric Dumazet
- ACM Queue
- Eben Moglen