AQM: Questioning a Fixed Delay Target

Bob Briscoe, BT*

Koen De Schepper, Bell Labs

RITE – Reducing Internet Transport Latency

IETF-93, Praha, July 2015

* now independent

B. Briscoe and K. De Schepper, “Insights from Curvy RED”
BT Technical report TR-TUB8-2015-003 (Jul 2015)
<http://riteproject.eu/publications/>

R/TE
Reducing Internet Transport Latency

The authors' contributions are part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The views expressed here are solely those of the authors.

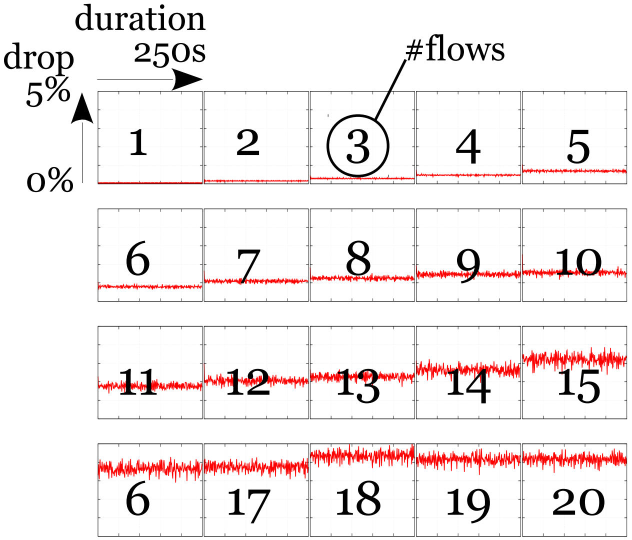
SEVENTH FRAMEWORK PROGRAMME

Questioning a fixed delay target

CoDel and PIE aim for a fixed target delay

- The AQM community has been focusing on delay
  - Don't forget loss
- In testing of PIE & fq_CoDel under high load
  - they cause significantly higher loss to keep delay down
  - loss, not queuing, becomes the dominant cause of delay

e.g. to maintain 5ms queuing delay
fq_CoDel uses 4.2% loss
to fit 20 Reno flows into 40MB/s



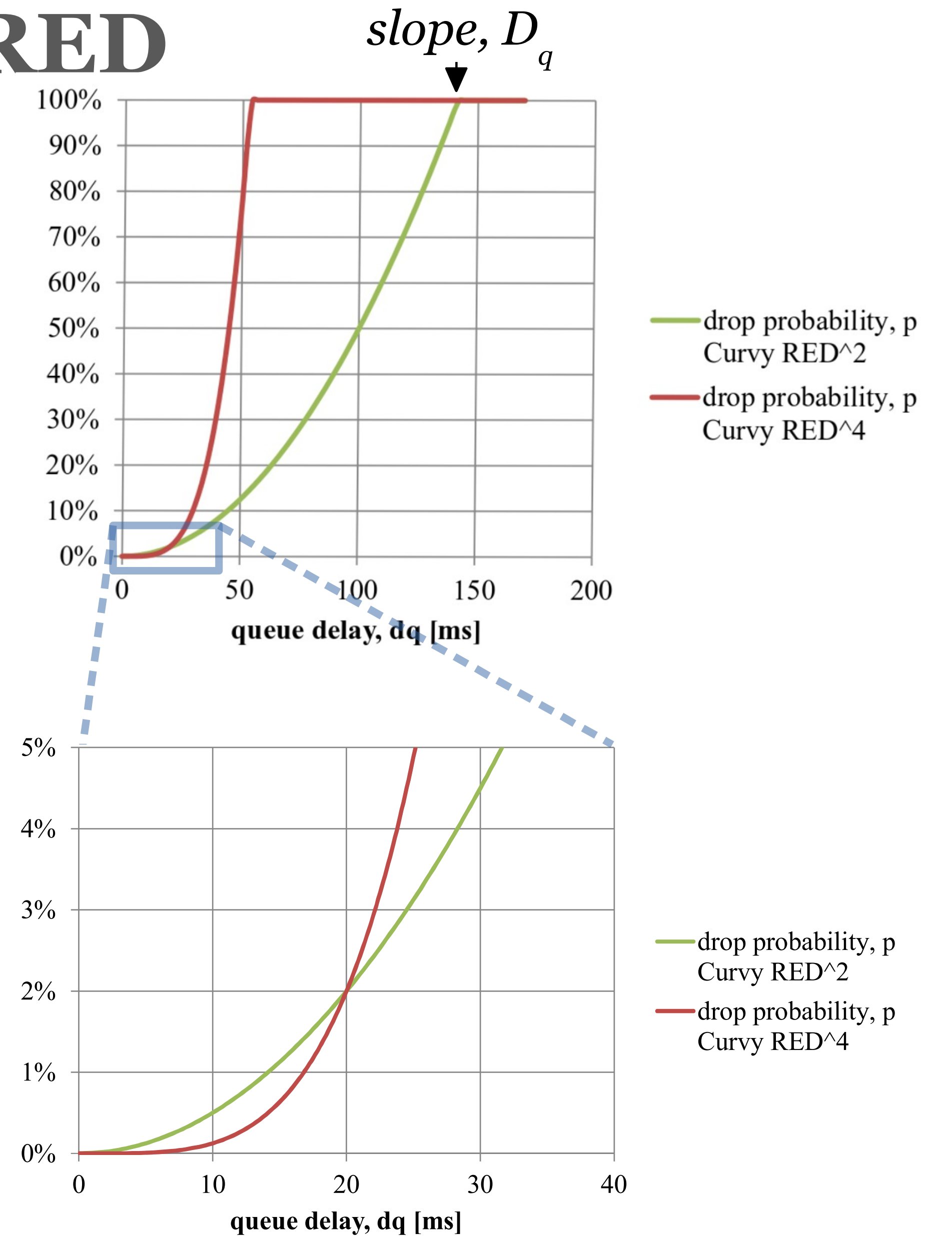
Insights from Curvy RED

What is Curvy RED?

- **Like RED except**
  - Increasing back-pressure
  - Initially hugs horiz axis
  - Continuous curve
  - Through origin

$$p = \left(\frac{d_q}{D_q} \right)^u$$

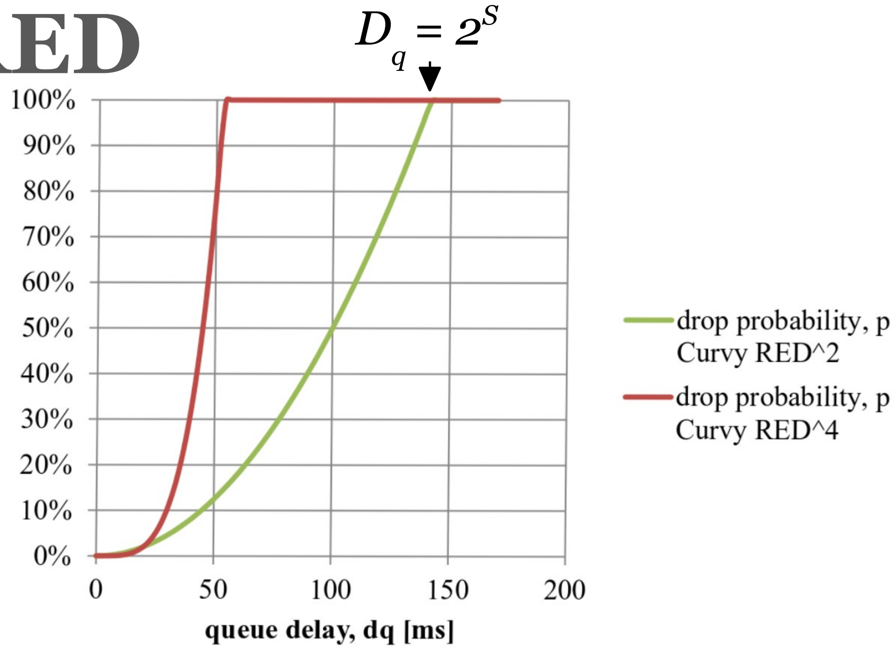
  - Simple to implement



Insights from Curvy RED

Curvy RED implementation

- $p = \left(\frac{d_q}{D_q} \right)^u$



- Whole curve including discontinuity, simply:

```
if ( (dq << S) > max(rand(),rand() )      % for u=2
   drop(pkt)
```

- or in general:

```
if ( (dq << S) > maxrand(U) )
    drop(pkt)
```

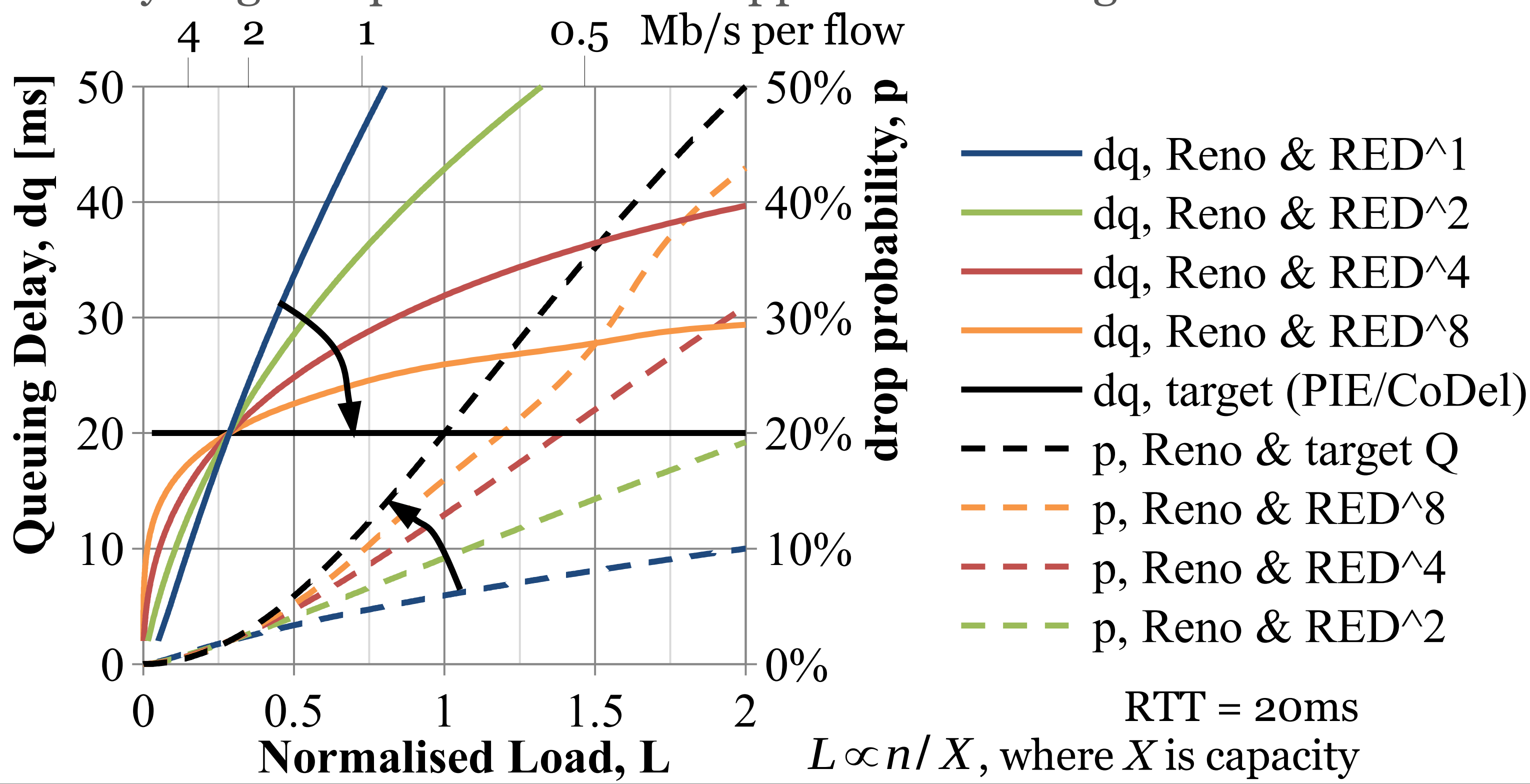
```
maxrand(u) {
     maxr=0
     while (u-- > 0)
         maxr = max(maxr, rand())
     return(maxr)
}
```

- `maxrand()` can be run out of band into buffered output

Insights from Curvy RED

Questioning a fixed delay target

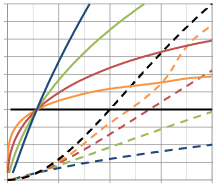
- n flows; as n⬀ AQM push-back⬀ to make each flow smaller
  - an AQM can make a TCP smaller either with higher drop or larger RTT delay
  - PIE & CoDel fix delay (**inherently infinite cUrviness**) → excessive drop
  - softer delay target requires less loss – apps survive at higher load



Insight

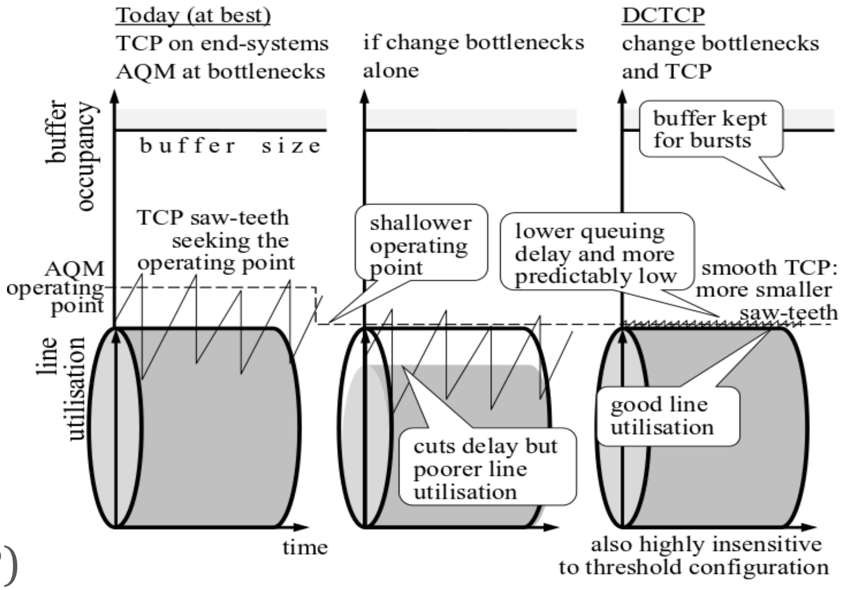
TCP creates dilemmas that no AQM can escape

If proponents of particular AQMs claim otherwise, look at delay, utilisation AND loss



1. if you squeeze delay, TCP increases loss
    - loss can become the dominant cause of delay

2. delay-utilisation tradeoff
   (we already know this one)
    - caused by TCP's large saw-teeth
    - more smaller sawteeth → excessive drop



✧ **TCP is the remaining problem**

- ECN allows you to resolve both dilemmas
- combined with scalable TCPs (e.g. DCTCP)

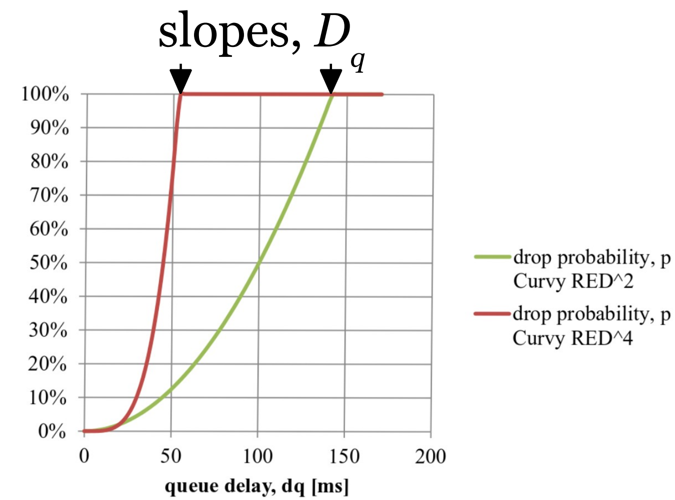
AQM: Questioning a Fixed Delay Target

RITE – Reducing Internet Transport Latency

Q&A

spare slides

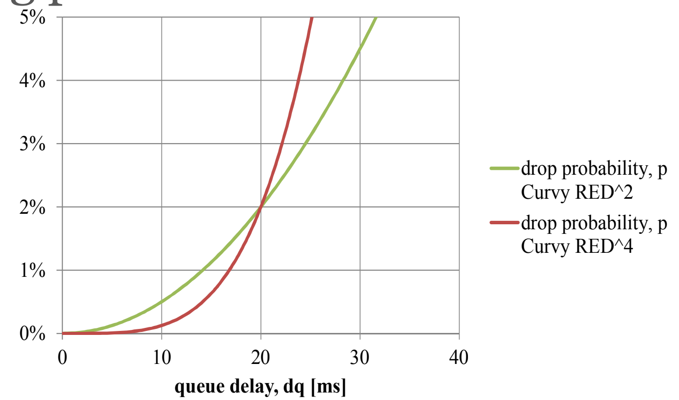
Invariance with Scale



Scaling AQM Configuration
- the Usefulness of a Design Point

- in the UI/API to Curvy RED
  - Slope D_q is not an intuitive config parameter
  - Better: use a (d_q^*, p^*) pair

    – a design point

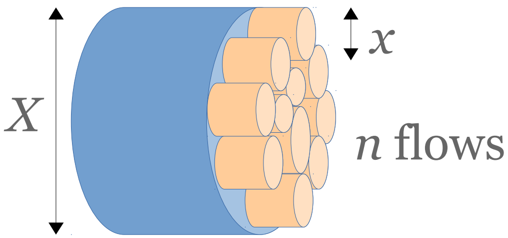


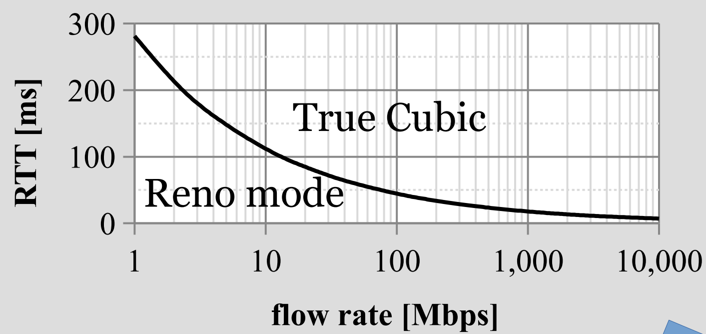
  - Represents upper end of preferred operating region
  - from which the router can easily derive D_q given cUrviness, u

$$D_q = \frac{d_q^*}{\left(p^*\right)^{1/u}}$$

Insights from Curvy RED

Analysis approach





Cubic switch-over to Reno emulation

- assume equal flows

  (1) $n \approx X/x$

- TCP formula: flow rate dependence on queuing & drop

  $$x = f(d_R, p), \quad d_R = D_R + d_q$$

  (2) $x = f(d_q, p)$

- AQM formula: relation between queuing & drop

  (3) $p = f(dq)$

- Plug (3) into (2) to get x as a function solely of p or of d_q
- Plug (2) into (1) to get n as a function solely of p or of d_q

n	# simultaneous TCP flows
X	link capacity
x	mean TCP flow rate
D_R	harmonic mean base RTT delay
d_q	queuing delay
d_R	RTT delay
p	drop probability
D_q	curvy RED slope
u	cUrviness
s	packet size
K	constant Reno: 1.22 Cubic in Reno mode: 1.68

TCP Reno: $x = \dfrac{K\,s}{d_R\sqrt{p}}$

Curvy RED: $p = \left(\dfrac{d_q}{D_q}\right)^u$