Insights from Curvy Random Early Detection (RED)

Bob Briscoe[*]

19 May 2015

Abstract

An active queue management (AQM) algorithm drops packets at an early stage in the growth of a queue, to prevent a capacity-seeking sender (e.g. TCP) from keeping the queue full. An AQM can mark instead of dropping packets if they indicate support for explicit congestion notification (ECN [RFB01]). Two modern AQMs (PIE [PNB+15] and CoDel [NJ12]) are designed to drop packets in order to keep queuing delay to a target value as load varies.

This memo uses Curvy RED and an idealised model of TCP traffic to explain why dropping packets to keep delay constant is a bad idea. However, it is a good idea to use ECN marking to keep delay constant. A corollary is that the dropping and marking behaviours of an AQM should be different. Recently, the requirement of the ECN standard [RFB01] that ECN must be treated the same as drop has been questioned. The insight that AQM behaviour for drop and for ECN should have different goals proves that this doubt is justified.

1 Curvy RED

Curvy RED is an active queue management (AQM) algorithm that is both a simplification and a a generalisation of Random Early Detection (RED [FJ93]). Two examples are shown in Figure 1 and Figure 2 shows a close-up of their normal operating regions.

The general formula for the drop probability, p, of a Curvy RED AQM is:

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

where d_q is the averaged[1] queuing delay and the two parameters are:

u the exponent (cUrviness) of the AQM;

D_q the slope of the AQM, or the queue delay where the curve hits 100%.

[*]ietf@bobbriscoe.net, BT Research & Technology, B54/77, Adastral Park, Martlesham Heath, Ipswich, IP5 3RE, UK

[1]The queuing delay used by Curvy RED is averaged, but this memo is only concerned with steady-state conditions, so averaging is not explained or discussed.

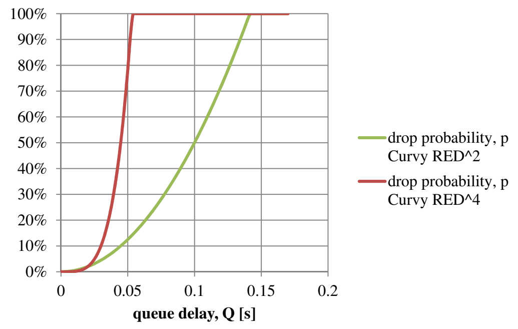

Figure 1: Two Example Curvy RED algorithms

The slopes D_q of the curves in Figure 1 are arranged so that they both pass through (20 ms, 2%), which we call the design point. All the curves in Figure 2 and Figure 3 are arranged to pass through this same operating point, which makes them comparable over the operating region of about 0–40 ms either side of this point, shown in Figure 2.

In the following, the dependence of both queuing delay and drop on load will be derived, assuming the load consists solely of equal TCP Reno flows.

The load on a link is proportional to the number of simultaneous flows being transmitted, n. If the capacity of the link (which may vary) is X and the bit-rate of each flow is x, then:

$$n = \frac{X}{x}. \qquad (2)$$

The rate of a TCP flow depends on round trip delay, d_R and drop probability, p. An accurate formula has been derived, but the simplest model [MSMO97] will suffice for the purpose of insight:

$$x = \frac{l}{d_R}\sqrt{\frac{3}{2p}}, \qquad (3)$$

where l is the average packet size.

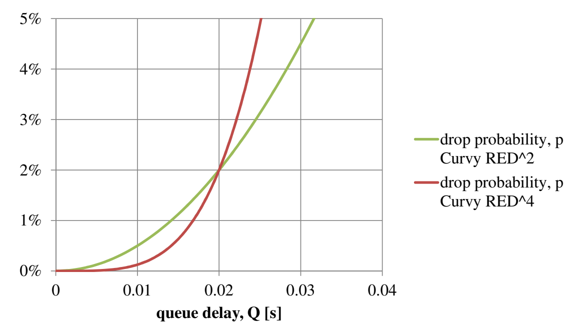

Figure 2: Normal Operating Region of The Same Two Example Curvy RED AQM algorithms

The round trip time, d_R consists of the base RTT D_R plus queuing delay d_q, such that:

$$d_R = D_R + d_q \tag{4}$$

Substituting Equation 3 Equation 4 in Equation 2:

$$\begin{aligned} n &= \frac{X(D_R + d_q)}{l}\sqrt{\frac{2p}{3}} \\ &= K\frac{D_R + d_q}{D_s}\sqrt{p}, \end{aligned} \tag{5}$$

where constant $K = \sqrt{2/3}$ and $D_s = l/X$, which is the serialisation delay for an average sized packet, which is constant while capacity is constant.

By substituting from Equation 1 into Equation 5, the number of flows can be given as a function of either queueing delay d_q or loss probability, p:

$$n = K\frac{(D_R + d_q)}{D_s}\left(\frac{d_q}{D_q}\right)^{u/2} \tag{6}$$

$$n = K\frac{(D_R + D_q.p^{1/u})p^{1/2}}{D_s} \tag{7}$$

Equation 6 & Equation 6 are plotted against normalised load in Figure 3 for an example set of Curvy RED algorithms with curviness parameters $u = 1, 2, 4, 8, \infty$, with base RTT $D_R = 20\,\text{ms}$. It can be seen that as the curviness parameter is increased, the AQM pushes harder against growth of queuing delay as load increases. However, given TCP is utilising the same capacity, it has to cause more loss (right axis) if it cannot cause more queuing delay (left axis).

In the extreme, infinite curviness represents the intent of AQMs such as CoDel and PIE that aim to clamp queueing delay to a target value (the black horizontal straight line). As a consequence, the TCP flows force loss to rise more quickly with load (the black dashed line).

Normalised load is plotted on the horizontal axis to make the plots independent of capacity or average packet size. Normalised load has units of ms and is defined as $L = nD_s/K$, with D_s in units of ms. As an example, if $X = 40\,\text{Mb/s}, l = 700\,B$, then for one Reno flow, $L = 700 * 8/40M/\sqrt{2/3} = 0.17\,\text{ms}$. Then for these example values, $L = 1$ on the horizontal axis represents $1/0.17 = 5.8$ flows.

2 Conclusions & Further Work

The curviness parameter of Curvy RED can be considered to represent the operator's policy for the tradeoff between delay and loss whenever load exceeds the intended design point. Conversely, PIE and RED embody a hard-coded policy, which dictates that holding down delay is paramount, at the expense of more loss.

Given losses from short interactive flows (e.g. Web) cause considerable delay to session completion, sacrificing loss for delay is unlikely to be the optimal policy to minimise delay. Also, it may lead real-time applications such as conversational video and VoIP to degrade or fail sooner as load increases. Allowing some additional flex in queueing delay with consequently less increase in loss is likely to give more favourable performance for a mix of Internet applications.

In addition, when load is below the design point, both PIE and CoDel tend to allow the queue to grow towards the target delay. Whereas Curvy RED gives lower queuing delay when load is light.

Another insight that can be drawn from this analysis, is that the dilemma in **??** disappears with ECN. For ECN, the AQM can mark packets without introducing any impairment. There is therefore no downside to clamping down queuing delay for ECN packets. This further supports the idea that ECN should not be treated equivalently to drop.

We intend to conduct experiments to give advice on a compromise level of curviness that best protects a range of delay-sensitive and loss-sensitive applications during high load. Although Curvy RED seems to be a useful AQM, we are not necessarily recommending it here. We are merely using the concept of curviness to draw insights.

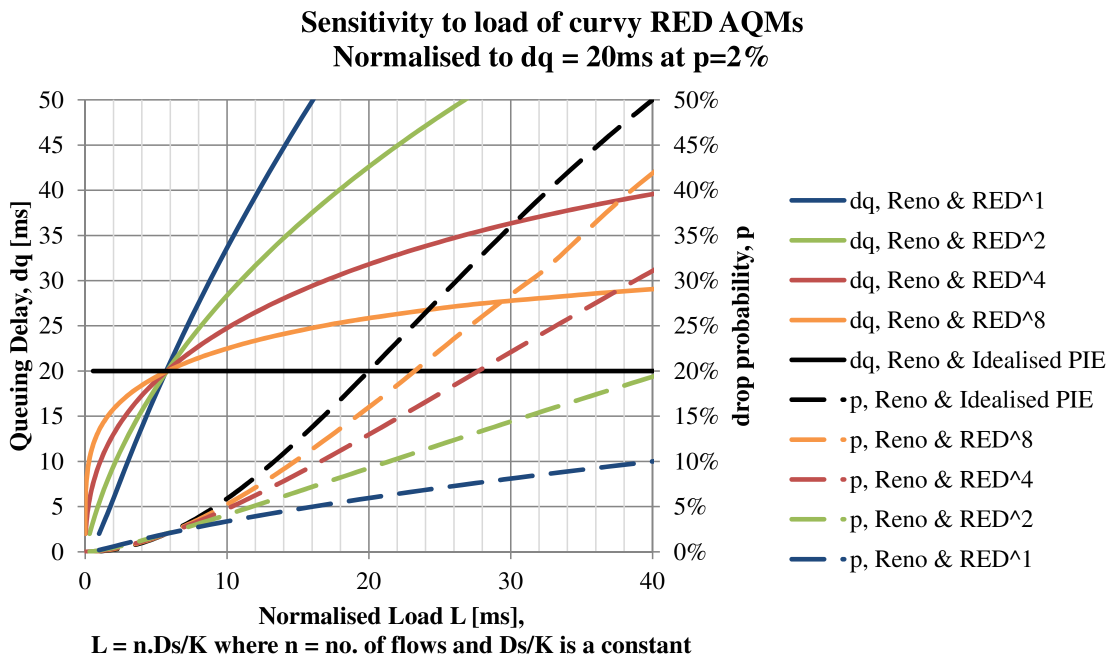

Figure 3: Dilemma between Two Impairments: Delay and Loss against Normalised Load; for a Set of Curvy RED Algorithms with Increasing Curviness

References

[FJ93] Sally Floyd and Van Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[MSMO97] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithms. *SIGCOMM Comput. Commun. Rev.*, 27(3):67–82, 1997.

[NJ12] Kathleen Nichols and Van Jacobson. Controlling queue delay. *ACM Queue*, 10(5), May 2012.

[PNB+15] Rong Pan, Preethi Natarajan, Fred Baker, Bill Ver Steeg, Mythili Prabhu, Chiara Piglione, Vijay Subramanian, and Greg White. PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem. Internet Draft draft-ietf-aqm-pie-01, Internet Engineering Task Force, March 2015. (Work in progress).

[RFB01] K. K. Ramakrishnan, Sally Floyd, and David Black. The Addition of Explicit Congestion Notification (ECN) to IP. Request for Comments 3168, Internet Engineering Task Force, September 2001.

Document history

Version	Date	Author	Details of change
00A	19 May 2015	Bob Briscoe	First Draft