Friendliness between AIMD Algorithms

Bob Briscoe*

01 Oct 2021

Abstract

This paper aims to provide a robust grounding for the additive increase factor used in the 'TCP-Friendly' mode of the CUBIC congestion control algorithm.

1 Introduction

The first IETF RFC to define the CUBIC congestion control algorithm [RXH+18] was based on the original paper introducing CUBIC [HRX08]. For 'TCP-friendly' mode, both draw on an equation in an ACIRI technical report [FHP00] when they specify the additive increase factor. The derivation of the equation in that technical report assumes a deterministic dropping (or ECN-marking) algorithm at the bottleneck, which limits its applicability. Also the technical report attempts to validate the theoretical formula empirically by simulation using a RED gateway at the bottleneck, but it leads to flow rates that are significantly different (by a factor of more than 2×) when they should be the same.

Below, an equation for the additive increase factor is derived without the assumption of deterministic dropping. Instead it is assumed that drops are synchronized between flows, which is typically the case for tail-drop queues. The resulting equation turns out to be the same as that in the technical report [FHP00]. However, the derivation here is a straightforward geometric one. It relies on fewer assumptions and no approximations; it considers variation of the RTT explicitly and it does not use loss probability at all.

The present paper is not intended to be ambitious or insightful, just pedestrian and rigorous. [BB01] provides and analyses a wider set of TCP-friendly algorithms, but does not dwell on the simple linear cases analysed here.

*research@bobbriscoe.net,

2 Terminology

Nowadays, TCP-friendly mode is more accurately known as Reno-friendly mode, given its flow rate is intended to match that of the Reno congestion control, and given that it is irrelevant which wire protocol is used, whether TCP, QUIC, SCTP, etc. The term C-Reno will be used for Cubic in Reno-friendly mode.

This paper uses the variables defined below:

a : Additive increase factor;
b : Multiplicative decrease factor;
j : Round index;
J : Rounds per sawtooth cycle;
$R(j)$: Round trip time (RTT);
$W(j)$: Congestion window;
$\check{W}$: Minimum W;
$r(j)$: Packet rate;
X_r : Reno variant of any variable X;
X_c : C-Reno variant of any variable X.

3 AIMD-Friendliness

Consider two types of Additive Increase Multiplicative Decrease (AIMD) flow with parameters (a_r, b_r) and (a_c, b_c) competing at a bottleneck, under the following assumptions:

- The buffer is large enough not to drain completely, even if all flows reduce simultaneously.
- All other factors of all the flows, particularly packet size and base RTT, are equal. When flows sharing the same bottleneck queue all have the same base RTT, they all have the same RTT, $R(j)$, at every stage, j, of their sawtooth cycles.
- The bandwidth-delay product (BDP) is low enough for all flows to remain in their AIMD mode throughout the cycle.
- All the flows have run long enough to converge on a steady state.

3.1 Synchronized (Tail-Drop) Case

For this case, it is additionally assumed that:

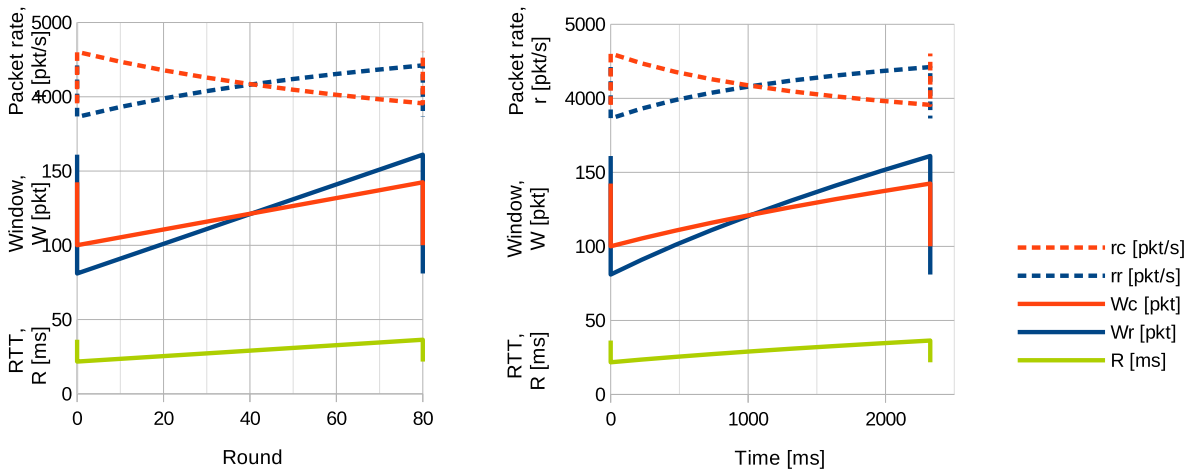

Figure 1: One synchronized sawtooth cycle of C-Reno and Reno plotted wrt. round trips (left) and wrt. time (right)

- All the flows are synchronized so that, whenever one flow experiences loss the others do too. No assumption is made about how much loss occurs at each congestion event, except that all flows experience it and they only respond to the presence of loss, not its extent.

Steady state: For each flow, the additive increase of a cycle balances with its multiplicative decrease from the max, $\check{W}/b$, to the min, $\check{W}$.

$$\begin{aligned} a_r J &= \check{W}_r/b_r - \check{W}_r \\ &= \check{W}_r(1-b_r)/b_r \qquad (1) \\ a_c J &= \check{W}_c(1-b_c)/b_c \qquad (2) \end{aligned}$$

Flow rate equality: Given the parameters a_r, b_r, b_c the aim is to derive a_c such that each flow's average rate is the same. This is equivalent to each flow transferring the same number of packets over a cycle.

As a cycle progresses, the RTT grows. So to derive the number of packets transferred over a cycle, the packet rate has to be weighted by the RTT in each cycle before being summed:

$$\begin{aligned} \sum_{j=0}^{J-1} r_c(j)R(j) &= \sum_{j=0}^{J-1} r_r(j)R(j) \\ \sum_{j=0}^{J-1} W_c(j) &= \sum_{j=0}^{J-1} W_r(j) \\ \sum_{j=0}^{J-1} \check{W}_c + a_c j &= \sum_{j=0}^{J-1} \check{W}_r + a_r j \\ J\check{W}_c + \frac{J^2 a_c}{2} &= J\check{W}_r + \frac{J^2 a_r}{2} \end{aligned}$$

Dividing through by J and substituting from Equation 1 & Equation 2:

$$\begin{aligned} \check{W}_c\left(1+\frac{(1-b_c)}{2b_c}\right) &= \check{W}_r\left(1+\frac{(1-b_r)}{2b_r}\right) \\ \frac{\check{W}_c}{\check{W}_r} &= \frac{(1+b_r)b_c}{(1+b_c)b_r} \qquad (3) \end{aligned}$$

Returning to the steady state equations, we can divide Equation 2 by Equation 1, then substitute from Equation 3:

$$\begin{aligned} \frac{a_c}{a_r} &= \frac{\check{W}_c}{\check{W}_r}\frac{(1-b_c)b_r}{(1-b_r)b_c} \\ &= \frac{(1-b_c)}{(1+b_c)}\frac{(1+b_r)}{(1-b_r)} \qquad (4) \end{aligned}$$

Plugging in Reno's AIMD factors, $a_r = 1, b_r = 1/2$:

$$\boxed{a_c = \frac{3(1-b_c)}{(1+b_c)}} \qquad (5)$$

And plugging in the multiplicative decrease factor of C-Reno recommended in [RXH+18], $b_c = 0.7$:

$$\begin{aligned} a_c &= 9/17 \\ &\approx 0.53. \end{aligned}$$

Geometric interpretation: Figure 1 shows one flow each of C-Reno and Reno competing over one synchronized sawtooth cycle. Superficially, the whole derivation of a_c above can be derived from simple triangle geometry, by drawing congestion window sawteeth that increase linearly wrt. round trips (mid-left) then setting the mid-points of the two ramps to the same height. Then Equation 3 gives the ratio between the heights of the bases of

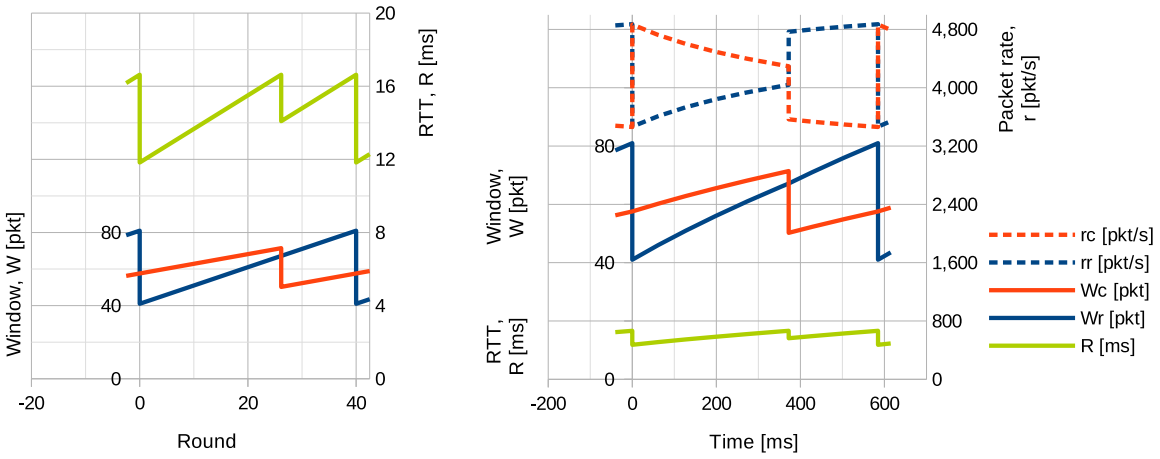

Figure 2: One desynchronized sawtooth cycle of C-Reno and Reno plotted wrt. round trips (left) and wrt. time (right)

the triangles, and Equation 4 gives the ratio of the heights of the triangles themselves.

However, it is not enough to merely assert that the average heights of these sawteeth are equal, as [FHP00] does. It is necessary to start from the goal of equal flow rates averaged over time, as the above analysis does. Given RTT grows throughout the cycle, the plots of flow-rate against time (top right in Figure 1) stretch out more to the right, forming concave curves. It is not at all obvious how to equate the averages of these two curves until they are weighted by round trip duration, which transforms them into the linear plots of window wrt. rounds (mid-left).

It is also interesting to note from Figure 1 that C-Reno's packet rate *decreases* as its window increases over the sawtooth. This is because the competing Reno flow causes the RTT to grow faster than would be the case with only C-Reno flows. If the buffer is not deep enough to hold all the synchronized sawteeth, it will be empty during the early part of the sawteeth. Then C-Reno will miss its highest packet rate and Reno will miss its lowest, so C-Reno's average packet rate will be lower than Reno's.

This might help explain the simulation results with a tail-drop queue in Floyd's report [FHP00], where C-Reno's packet rate was lower than the theory predicted.

3.2 Desynchronized (AQM) Case

For this case, instead of the assumption of synchronization, it is assumed that:

- As the queue grows, Active Queue Management (AQM) at the bottleneck selects single packets to drop or mark, so that the congestion responses of each flow tend not to coincide.[1]

The AQM case is harder to analyse than the synchronized case with tail-drop. Superficially, one could use the transformation from equal average flow rates (wrt. time) into equal window (wrt. rounds). However, there is no guarantee that the number of rounds per cycle, J is the same in each case.

If we assumed it was, we would end up with Equation 5 for C-Reno's additive increase factor a_c. Then, as shown in Figure 2, the phasing between the sawteeth would evolve so that the queue reached roughly the same depth before each reduction, i.e. the tips of the RTT sawteeth will all align at roughly the same level—the operating point of the AQM.

However, although Figure 2 shows the sawtooth reductions alternating Reno – C-Reno – Reno, this need not be the case. At 400 ms in the top-right plot, the ratio between C-Reno's and Reno's packet rates is about 51:49. So it is nearly as likely that the AQM will hit a Reno packet as a C-Reno packet, causing Reno to reduce twice in a row. If the AQM did hit C-Reno around 400 ms, at around 600 ms the ratio would be about 42:58, making Reno more likely to be hit. Overall, if a_c is set for the synchronized case, an AQM is likely to hit Reno somewhat more often than C-Reno.

This runs counter to the simulation results using

[1] In desynchronized cases, the RTT varies much less than in synchronized cases—on average the amplitude is respectively $1/\sqrt{n}$ vs. n times that of a single flow, where n is the number of flows [AKM04]. Therefore, the first assumption (that the buffer is large enough not to drain completely) is much more likely to hold in desynchronized cases.

the RED AQM in Floyd's report [FHP00] where the throughput of C-Reno was 70% of that of competing Reno flows when C-Reno used $b_c = 7/8$ and a_c was set according to Equation 5.

References

[AKM04] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. Sizing Router Buffers. *Proc. ACM SIGCOMM'04, Computer Communication Review*, 34(4), September 2004.

[BB01] Deepak Bansal and Hari Balakrishnan. Binomial Congestion Control Algorithms. In *Proc. IEEE Conference on Computer Communications (Infocom'01)*, pages 631–640. IEEE, April 2001.

[FHP00] Sally Floyd, Mark Handley, and Jitendra Padhye. A Comparison of Equation-Based and AIMD Congestion Control. Technical report, ACIRI, May 2000.

[HRX08] Sangtae Ha, Injong Rhee, and Lisong Xu. CUBIC: A New TCP-friendly High-speed TCP Variant. *SIGOPS Operating Systems Review*, 42(5):64–74, July 2008.

[RXH+18] I. Rhee, L. Xu, S. Ha, A. Zimmerman, L. Eggert, and R. Scheffenegger. CUBIC for Fast Long-Distance Networks. Request for Comments RFC8312, RFC Editor, August 2018.

Document history

Version	Date	Author	Details of change
00A	29 Sep 2021	Bob Briscoe	First draft.
00B	01 Oct 2021	Bob Briscoe	Added geometric interpretation and deterministic case.