PI^2 Parameters

Bob Briscoe*

05-Jul-2021

Abstract

This report gives the reasoning for the parameter settings of the reference Linux implementation of the PI^2 AQM, focusing initially on the target queue delay.

1 Introduction

This report explains the reasoning behind the parameter settings of the reference Linux implementation of the PI^2 AQM[1]. These settings are documented as a pseudocode example in Appendix A of the IETF specification of the Coupled DualQ AQM [DSBEW21]. In both cases, the PI^2 AQM is used for the Classic queue within the dual-queue structure called DualPI2, but the parameter settings for PI^2 discussed here apply irrespective of whether a PI^2 AQM stands alone or within a dual-queue structure. The discussion of the `target` parameter also applies to a PIE AQM [PPP+13].

Similar reasoning for the parameter settings was behind the technical report produced in 2015 [dSBTB15] to support standardization of the Coupled DualQ AQM. The present report comes to the same conclusion, but spells out all the details that were glossed over at that time. It focuses on the PI2 parameter settings in the following lines of Figure 2 in [DSBEW21]:

```
7:  % PI2 AQM parameters
8:  RTT_max = 100 ms            % Worst case RTT expected
9:  RTT_typ = 34 ms                       % Typical RTT

11: % PI2 constants derived from above PI2 parameters
12: target = 0.22 * 2 * RTT_typ           % qDelay target
13: Tupdate = min(RTT_typ, RTT_max/3)  % sampling interval
14: alpha = 0.1 * Tupdate / RTT_max^2 % integral gain [Hz]
15: beta = 0.3 / RTT_max          % proportional gain [Hz]
```

The choices of the maximum round trip time, `RTT_max` and of `Tupdate` on lines 8 & 13 are justified in [DSBEW21]. So this report focuses on line 12 above, which in turn depends on line 9:[2,3]

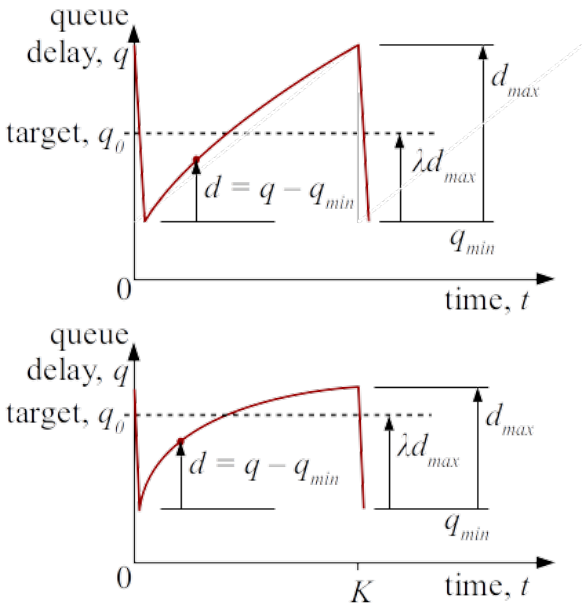

Figure 1: Definition of terms

```
9:    RTT_typ = 34 ms
12:   target = 0.22 * 2 * RTT_typ
```

Therefore the task for this report is to choose a compromise default for `target` that minimizes queue delay for Classic traffic without causing under-utilization over path RTTs that are commonly experienced by most Internet users.

2 Terminology

The schematic plots of one cycle of queue delay against time for two different congestion controllers (Reno and Cubic) in Figure 1 define some terminology for this report. We will use $q(t)$ as the time-varying queue delay in units of time, and $d(t)$ as the additional queue delay above the minimum, that is $d(t) = q(t) - q_{\min}$. q_0 is the target queue delay for a particular AQM and $d_{\max}$ is the amplitude of the cycles, in units of time. In steady state the congestion controller and AQM together keep the average queue delay at q_0, so the fraction, λ of the amplitude that sits below the average depends solely on the geometry of the sawtooth curve.

*research@bobbriscoe.net,

[1] https://github.com/L4STeam/sch_dualpi2_upstream

[2] It may be noticed that lines 9 & 12 are different from those in revision 15 of [DSBEW21], but it is planned to correct that in future versions of the draft.

[3] Justification for lines 14 & 15 is deferred to a future version of this report.

The instantaneous RTT, $R(t)$, varies because it consists of the constant base delay of the path, R_b, and variable queue delay $q(t)$, that is $R(t) = R_b + q(t)$. Alternatively, $R(t) = R_{\min} + d(t)$.

A Classic congestion controller's min window is related to its max window by the multiplicative factor, b of the congestion controller: $W_{\min} = bW_{\max}$. This leads the RTT to cycle between $R_{\min}$ and $R_{\max}$, around the average R_0. Similarly queue delay cycles around q_0 between $q_{\min}$ and $q_{\max}$.

The IETF's specification of Cubic [RXH+18] uses β for the multiplicative decrease factor, but we use b to avoid confusion with the proportional gain factor β of a PI AQM. Similarly, we use a (rather than α) for the additive increase factor of Reno, or of Cubic in its Reno-friendly mode. Where necessary, we use the subscripts 'r' or 'c' to distinguish parameters used by Reno or Cubic.

3 Scaling of Queue Variation

q_0 is intended to be the operating point that the queue cycles around under stable conditions, so we consider only long-running flows and fixed capacity links. Within this stable environment, we consider a single long-running flow as the worst-case for queue variability (and a fairly common case in access link bottlenecks).

The schematics in Figure 2 show how different Linux congestion controls vary the queue delay of a single flow around the target and how the variation scales with base RTT and link capacity. The scales of the plots are all the same, but actual numerical values of queue delay and time are irrelevant for this visualization.

The following two subsections consider how queue variation due to a single flow scales with base RTT and with link capacity. Then subsection 3.3 discusses the geometry and prevalence of different sawtooth shapes.

3.1 Scaling of Queue Variation with RTT

Assumption 1: Our goal is to find a value of `target` that prevents the queue from completely draining at the bottom of each sawtooth cycle. Therefore, the analysis in this paper assumes that is the case because, when it is not, the analysis does not need to apply. So we can assume that the delivered packet rate, r, remains constant, given we also assume link capacity remains constant for simplicity (despite being unrealistic). Then, given that the window, $W = r * R$, the RTT, R, varies in direct proportion to the window.

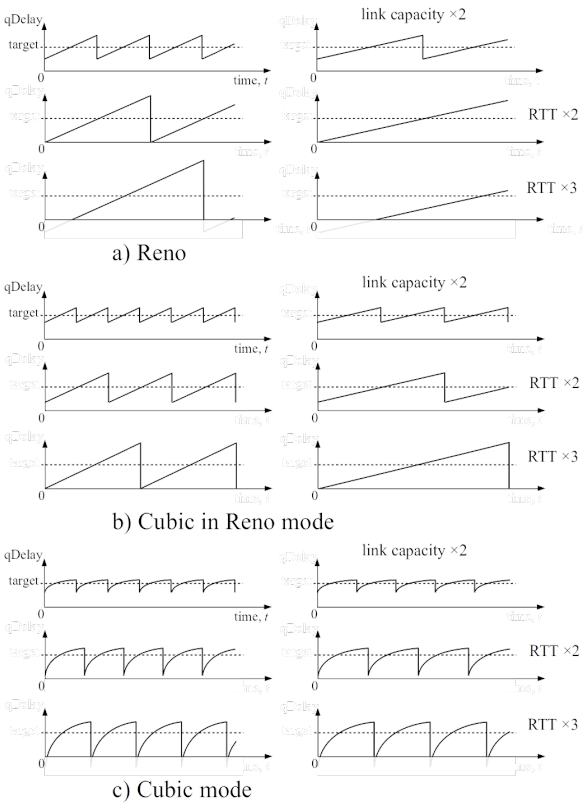

Figure 2: Scaling of queue delay variability with average RTT (increasing downwards) and link capacity (increasing to the right)

Assumption 2: As a first-order approximation, we assume that queue delay tracks the window instantly, even though it actually takes a round trip to catch up. And we don't consider smoothing of window reductions, e.g. Proportion Rate Reduction. These approximations overestimate the amplitude of queue variation a little, especially when the window reduces sharply then increases sharply (as it does when Cubic responds to congestion). However, these approximations are close enough for our purposes.

The algebra below shows that sawtooth amplitude is related to average RTT by a constant factor,

$$\begin{aligned} R_0 &= R_{\max} - (1-\lambda)(1-b)R_{\max} \\ &= R_{\max}(\lambda + b - \lambda b) \\ d_{\max} &= (1-b)R_{\max} \\ &= \frac{(1-b)}{(\lambda + b - \lambda b)} R_0. \end{aligned}$$

This is why, starting at the top left and working down the schematics for each congestion control in Figure 2, it is shown that the amplitude of the

queue variation grows linearly with average RTT.[4] This linear scaling of queue variability with RTT only relies on multiplicative decrease, so it is just as true for either mode of Cubic as it is for Reno.

3.2 Scaling of Queue Variation with Link Capacity

More link capacity allows either more flows or more throughput per flow. But in the edge links giving access to the Internet, which tend to be the bottleneck links, the number of simultaneous flows is still low, and single lone flows remain common.

If link capacity doubles, the delivered packet rate (and the average window) of a single flow doubles too. Nonetheless, the AQM keeps the average RTT, R_0, at the same operating point. For a particular congestion control, the max and min RTT are related to the average RTT by constant factors, as confirmed algebraically below, so they also remain unchanged.

$$\begin{aligned} R_{\min} &= bR_{\max} \\ R_{\max} &= R_0 + (1-\lambda)(R_{\max} - R_{\min}) \\ &= R_0 + (1-\lambda)(1-b)R_{\max} \\ R_{\max} &= \frac{R_0}{(1-(1-\lambda)(1-b))}. \end{aligned}$$

Therefore, $q_{\max}$ and $q_{\min}$ also remain unchanged as link capacity scales (shown in the right-hand column of Figure 2). Incidentally, this also means that queue variation is no different between the upstream and downstream, even if the capacity is asymmetric.

Incidentally, the scaling of the cycle duration (along the horizontal time axis in Figure 2) is not directly relevant to queue variation, but Appendix D briefly explains why it is relevant to utilization.

3.3 Sawtooth Geometry

The fraction, λ, of the sawtooth amplitude that lies below the average is important when determining the target queue delay. For a Reno-style sawtooth, Appendix A gives a good approximation as:

$$\lambda = \frac{(2+b_r)}{3(1+b_r)}.$$

And, for $b \geq 1/2$ a sufficient approximation is $\lambda \approx 1/2$ (see Appendix A for the full approximation conditions). For a Cubic sawtooth, Appendix B proves that

$$\lambda = 3/4,$$

whatever the value of b_c.

The "Great TCP Congestion Control Census" [MSJ+19] conducted by Mishra *et al* in Jul–Oct 2019 found that Cubic was the most used by nearly 31% of the Alexa top 20k web sites, but BBR was approaching 18%, and already had a larger share of the Alexa top 250, as well as contributing 40% by downstream traffic share.[5] Of the 51% of the Alexa top 20k sites that were not using either Cubic or BBR, 19% were split between eight other known controllers, the greatest shares being for YeAH and CTCP or Illinois at under 6% each. The remaining 32% were unidentifiable, including sites that were unresponsive or did not serve anything large enough to be testable. As part of that remaining 32%, nearly 17% of the total were using an unknown congestion controller and further investigation found nearly 6% of the total were using an undocumented Akamai controller.

BBRv2 supports L4S when it detects ECN marking, so it is unlikely to use the Classic queue. This leaves 67% of sites that use some form of Classic congestion control, of which 46% use Cubic and the remainder is split across a dozen or so other algorithms, many of which, like Cubic, attempt to be friendly to Reno at low BDP.

Based on recent predictions, more than two-thirds of Internet traffic now emanates from Content Distribution Networks (CDNs) or cloud services distributed to locations close to, and often within, the metro area or regional network of the end-user's ISP [LR20].

Figure 3 illustrates how the Cubic congestion control would invariably run in its Reno mode for CDN traffic over a PI^2 AQM. The figure visualizes the average CDN RTT[6] under load against average fixed downstream bandwidth per household for the top 43 countries ranked by number of Internet users (see Appendix C for the detailed data and sources). The base RTT from Appendix C is inflated by 15 ms to model the target queue delay of the PI^2 AQM. It can be seen that nearly all the points are below the upper limit of Cubic's Reno-friendly mode for a single flow (the two unlabelled points furthest from the curve are Uzbekistan and Iraq, with Iran, Venezuela, Columbia and Thailand just above it). However, there is a question mark over the CDN RTT in China (see Appendix C).

[4] At least, it does while the sawteeth do not drain the queue completely, by Assumption 1. Where this assumption breaks down—in the plots labelled 'RTT ×3' for a) Reno and c) Cubic mode—for visualization purposes light grey traces extrapolate where the plots would be if the queue could be negative.

[5] The census did not investigate congestion controls used by QUIC.

[6] The study did not measure fixed and mobile separately. RIPE Atlas probes are generally connected to fixed access links although some are connected via Ethernet to mobile broadband.

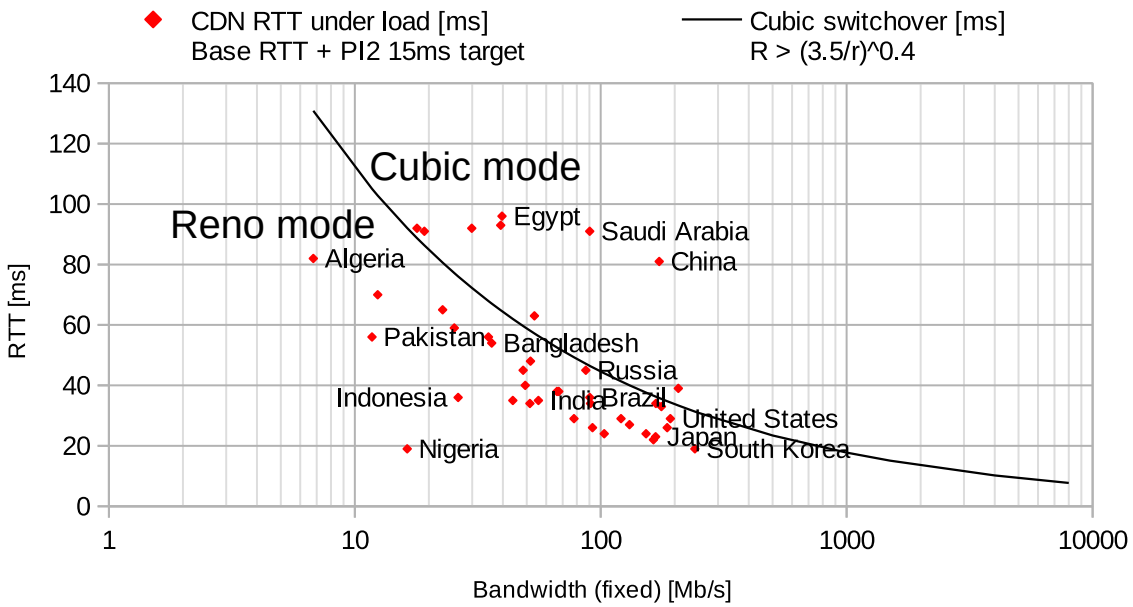

Figure 3: Scatter-plot per country of average user to CDN RTT under load against average fixed downstream access bandwidth. RTT is taken as if it is under load over the PI^2 AQM under study, so RTT = base RTT plus 15 ms. Only the 43 countries with the most Internet users are plotted, representing 90% of Internet users. The top ten are labelled as well as those at the extremes. The curve overlaid on the plot is where the Cubic congestion control in Linux switches over from Reno mode to pure Cubic mode

As link rates continue to scale, the points are expected to shift inexorably to the right. However, Cubic is likely to remain predominantly in Reno mode for some considerable time to come because, as CDN deployment continues, the points are also expected to shift downwards as base RTTs reduce below 20 ms then below 10 ms as they have done in the more mature deployments in Europe, N America and the Pacific rim. Also remember that we have chosen to examine the worst case of a single downstream flow; whenever there are more simultaneous flows, the points would shift back to the left, into the Reno region. And, where capacity is asymmetric, upstream flows would also sit further to the left.

The switchover curve between Reno and Cubic assumes the Linux implementation of Cubic with a packet size of 1500 B; aggressiveness constant $C = 0.6$; multiplicative decrease factor $b_c = 0.7$; and additive increase factor $a_c = 1$ (as opposed to $a_c = 3(1 - b_c)/(1 + b_c) = 0.53$, which is recommended in RFC 8312 for friendliness to Reno and used in BSD-derived implementations, which includes those in Apple devices). The packet rate converges to Equation 1 in Cubic's Reno mode or to Equation 2 in cubic mode, as given below.

$$r_{\text{creno}} = \frac{1.68}{p^{1/2}R} \tag{1}$$

$$r_{\text{cubic}} = \frac{1.17}{p^{3/4}R^{1/4}} \tag{2}$$

At the same loss probability, p, the packet rate, r_{cubic} equals r_{creno} when the switchover RTT is

$$R = \left(\frac{3.5}{r}\right)^{2/5} \tag{3}$$

In summary, the prevalent sawtooth geometry of Classic traffic is likely to be dominated for some time by the Reno mode of Cubic, with $\lambda \approx 1/2$ and $b_c = 0.7$.

4 Typical Base RTT

A globally typical RTT for CDN traffic is calculated in Appendix C. The average RTT for each country is weighted by the Internet user population in that country, as collated on Wikipedia [Wik20] from multiple primary sources. The countries are ranked in order of user population until 90% of the total Internet users in the world is covered. The

CDN RTT per country is based on measurements by Beganović using RIPE Atlas probes deployed by self-selected volunteers in what is claimed to be the largest Internet measurement infrastructure in the world [Beg19].

The resulting weighted average RTT to CDNs is 34 ms. However, these is a question-mark over some of the latency figures, given the measurements were all taken to 7 CDNs with global coverage, which might not be representative of the market in certain countries, particularly China (see Appendix C for details).

As a sanity check, 34 ms compares reasonably well with the global averages given on Ookla's Speedtest Global Index page:

- 20 ms fixed and 37 ms mobile (Apr 2021 data);
- 24 ms fixed and 42 ms mobile (Apr 2020 data).

Ookla's data is collected from self-selecting users who use speedtest's algorithm to find the closest CDN-based servers [Ook21]. The page gives a single global figure without details of the method used.

5 Default `target`

When selecting a global default for `target`, the aim is to ensure that the AQM keeps queue delay reasonably low while not compromising utilization for a large majority of users. Ideally a latency figure for say the 75th or 90th percentile of users would be used to derive `target`, but that data is not available globally.

Therefore, a 'safety factor' is applied to the average RTT between users and CDNs, which has to allow for the statistical distribution of RTTs to CDNs, particularly for users in rural areas [KKFR15], who will be further from the nearest CDN and who are also likely to have least bandwidth and therefore be least willing to see it eaten by under-utilization. The safety factor also has to allow for flows between clients and servers other than in CDNs. As an interim educated guess, we apply the safety factor, $f = 2$.

Next we draw together all the strands of the analysis of sawtooth scaling and geometry in section 3, in order to derive a default `target`. We want fR_{typ} to sit at least at the minimum of the saw-teeth, R_{min}, which is related to the target queue as follows

$$\begin{aligned} q_0 &= \lambda(R_{\text{max}} - R_{\text{min}}) \\ &= \frac{\lambda(1-b)}{b} R_{\text{min}}. \end{aligned}$$

Therefore:

$$\boxed{\texttt{target} \approx \frac{\lambda(1-b)f}{b} R_{\text{typ}}} \qquad (4)$$

We call $\frac{\lambda(1-b)}{b}$ the geometry factor. The geometry factors of a selection of congestion controls (CCs) are tabulated below (Cubic in Reno mode is abbreviated to CReno). The geometry parameters are taken from the current Linux implementation, but they only depend on the multiplicative decrease factor, so they are also as recommended in the RFCs and as used in BSD-derived implementations.

CC	λ	b	$\frac{\lambda(1-b)}{b}$
Reno	$1/2$	$1/2$	$1/2 = 0.50$
CReno	$1/2$	0.7	$3/14 = 0.21$
Cubic	$3/4$	0.7	$9/28 = 0.32$

Taking account of the mix of congestion controls discussed in subsection 3.3, but without modelling all the minor players, we use a weighted average of about 90% CReno; 10% Cubic, which gives a geometry factor of about 0.22. Thus, for PI^2 we suggest setting the default to:

$$\begin{aligned} \texttt{target} &= 0.22 * 2 * 34\,\text{ms} \\ &= 15\,\text{ms}. \end{aligned}$$

Over time, as CDN deployment continues, R_{typ} will continue to reduce, so the default `target` could be reduced in future. That in turn will reduce RTT further, with the knock-on effect of keeping more Cubic flows in Reno mode, thus reinforcing the applicability of the lower `target` for AQMs.

Other implementations intended for particular link technologies might use a different default today. For instance, the Low Latency DOCSIS specification [DOC19] uses `target` = 10 ms, which perhaps makes sense because cable technology is less likely to extend to rural areas, so the distribution around the average RTT is likely to be considerably tighter. By a similar argument, the default `target` for mobile networks might need to be greater than 15 ms, depending on how well 5G meets its aspirations to reduce base RTT.

Of course, operators are free not to use the default `target` for out-of-the-ordinary environments. For instance, they could configure a higher `target` for satellite links and remote rural locations; or a lower `target` for highly concentrated urban deployments.

A Average Queue Over a Reno Sawtooth

The following analysis determines the fraction λ of the amplitude of a Reno sawtooth that sits below its average. It is generalized for any additive increase of a segments per round. Terminology and assumptions are defined in the body of the paper (section 2 & section 3).

Reno's congestion window increases by a segments per round,

$$W_r(j) = W_{\min} + ja,$$

where j is an index of the rounds since the last reduction. By the same reasoning as in section 3, while the link is not underutilized, Reno's RTT is directly proportional to its congestion window:

$$R_r(j) = R_{\min} + \frac{ja}{r},$$

where a/r is the delay added to the queue by one round of additive increase. For brevity we will use $R_a = a/r$ to denote this addition to the RTT per round. By our assumption that the queue is never allowed to drain completely, we can remove the minimum queue delay from the equation and focus solely on the additional delay above the minimum,

$$d(j) = jR_a.$$

Then the fraction of the amplitude that sits below the average,

$$\begin{aligned}\lambda &= \frac{\mathbb{E}\{d(t)\}}{d_{\max}} \\ &= \frac{\mathbb{E}\{d(t)\}}{R_{\min}(1-b)/b}.\end{aligned}$$

Within a cycle, the queue above the minimum averaged over time, $\mathbb{E}\{d_r(t)\}$, is the extra queue above the minimum in each round weighted by the duration of each round then divided by the sum of the weights. The duration of each round is the RTT itself. Thus,

$$\lambda = \frac{\sum_{j=0}^{J}(R_{\min} + jR_a)jR_a \Big/ \sum_{j=0}^{J}(R_{\min} + jR_a)}{R_{\min}(1-b)/b}$$

Using the standard result for a sum of squares, then simplifying:

$$\begin{aligned}&= \frac{J^2 R_{\min}R_a/2 + (J^3/3 + J^2/2 + J/6)R_a^2}{(JR_{\min} + J^2R_a/2)R_{\min}(1-b)/b} \\ &= \frac{3JR_{\min}R_a + (2J^2 + 3J + 1)R_a^2}{(6R_{\min}^2 + 3JR_{\min}R_a)(1-b)/b}\end{aligned}$$

The maximum value of j under stable conditions can be found by equating the additive increase over a cycle to the multiplicative decrease,

$$JR_a = R_{\min}(1-b)/b.$$

Substituting for J, then collecting terms and simplifying further,

$$\begin{aligned}\lambda &= \frac{3R_{\min}^2 + (2R_{\min}^2(1-b)/b + 3R_{\min}R_a + R_a^2 b/(1-b)}{(6R_{\min}^2 + 3R_{\min}^2(1-b)/b)} \\ &= \frac{(2+b)/b + 3R_a/R_{\min} + b/(1-b)R_a^2/R_{\min}^2}{3(1+b)/b} \\ &= \frac{(2+b)}{3(1+b)} + \frac{b}{(1+b)}\frac{R_a}{R_{\min}} + \frac{b^2}{3(1-b^2)}\left(\frac{R_a}{R_{\min}}\right)^2 \\ &\approx \frac{(2+b)}{3(1+b)} \qquad \text{if} \quad R_a \ll R_{\min}\sqrt{(1-b^2)}\Big/b\,.\end{aligned}$$

Then, for standard Reno with $b = 1/2$,

$$\lambda \approx 5/9 \approx 0.556.$$

And for Cubic-Reno with $b = 0.7$,

$$\lambda \approx 9/17 \approx 0.529.$$

$\lambda \approx 1/2$ is a good enough approximation for many purposes, including for the present paper.

B Average Queue Over a Cubic Sawtooth

The following analysis determines the fraction λ of the amplitude of a Cubic sawtooth that sits below its average. Terminology and assumptions are defined in the body of the paper (section 2 & section 3).

The formula for the congestion window of a Cubic sawtooth is defined in IETF RFC 8312 [RXH+18] as,

$$W_c(t) = W_{\text{cmax}} + C(t-K)^3,$$

where C is a constant (recommended as 0.4 in RFC 8312 but currently 0.6 in Linux) and:

$$K = \left(\frac{W_{\text{cmax}}(1-b)}{C}\right)^{\frac{1}{3}},$$

where b is the multiplicative decrease factor already defined in section 2 (recommended as 0.7).

By the same reasoning as in section 3, while the link is not underutilized, Cubic's RTT is directly proportional to its congestion window:

$$\begin{aligned}R_c(t) &= R_{\text{cmax}} + \frac{C(t-K)^3}{r}, \\ K &= \left(\frac{rR_{\text{cmax}}(1-b)}{C}\right)^{\frac{1}{3}}.\end{aligned}$$

The average RTT over a cycle, $\mathbb{E}\{R_c(t)\}$, is then

$$\begin{aligned} R_{c0} &= \frac{1}{K}\int_0^K R_{\text{cmax}} + \frac{C(t-K)^3}{r}dt, \\ &= \frac{1}{K}\left[R_{\text{cmax}}t + \frac{C(t-K)^4}{4r}\right]_0^K \\ &= R_{\text{cmax}} - \frac{CK^3}{4r} \end{aligned}$$

Substituting for K:

$$= R_{\text{cmax}}\frac{(3+b)}{4}. \tag{5}$$

Then, for a single Cubic sawtooth, the fraction of the amplitude that sits below the average is

$$\begin{aligned} \lambda_c &= \frac{(R_{c0} - R_{\text{cmin}})}{(R_{\text{cmax}} - R_{\text{cmin}})} \\ &= \frac{R_{\text{cmax}}\left(\frac{(3+b)}{4} - b\right)}{R_{\text{cmax}}(1-b)} \\ &= \frac{3}{4}. \end{aligned}$$

Thus, λ_c is constant for any $b \in [0, 1)$.

C Typical User to CDN RTT

Beganović [Beg19] provides the average RTT measured using ICMP ping from probes in each country to sites known to be served by CDNs. The data was collected from RIPE Atlas probes deployed by volunteers around the world, and was last updated on 17 Apr 2019.

The data is tabulated below and visualized in Figure 4. At the bottom of the table, an average is derived, weighted by the population of Internet users in each country (taking the countries with the highest Internet user populations until 90% of the world's total Internet users are covered). The per-country data on numbers of Internet users was taken from Wikipedia [Wik20], which in turn used population figures for each country, usually from the US Census Bureau, and various estimates of the percentage of Internet users in each country, mostly provided by the ITU.

The measurements were taken to the following seven global CDNs:

- Akamai
- AWS Cloudfront
- Microsoft Azure
- Cloudflare
- Google Cloud CDN
- Fastly
- Cachefly

The data point for China seems uncharacteristic for countries of similar size and market maturity. It is possible that it is suspect, perhaps because measurements to large Chinese CDN providers such as the following were not included in the RIPE Atlas study: ‘

- Alibaba Cloud
- Baidu Cloud
- BaishanCloud
- ChinaCache
- Tencent Cloud

Given users in China make up nearly a quarter of the global total, the weighted average would be sensitive to any large error in the CDN latency for users in China. For instance, if the latency figure just for China was reduced from 66ms to 20ms (bringing it in line with India), the global weighted average would drop from 34ms to 24ms.

D Scaling of Cycle Duration

Scaling of the cycle duration is not directly relevant to the setting of PI^2 parameters, but it does affect utilization in an indirect but important way. A Classic congestion control responds to a single loss or ECN mark, so losses and ECN marks have to be completely absent during a cycle for a flow to maintain full utilization. The longer the duration of each cycle, the more likely that some extraneous event will occur, e.g. the arrival of a brief flow or loss due to a transmission error. This noise sensitivity of Classic flows becomes the dominant determinant of utilization the more flow rate scales (see footnote 6 of Jacobson & Karels [JK88]).

This scaling of cycle duration is important to understand, as follows:

- Additive increase of a constant amount of data per round trip causes the duration of a single flow's sawtooth cycle to double for every doubling of link rate. This can be seen for a) Reno and b) Cubic in Reno mode in the right-hand column of Figure 2.
- In contrast, the cycle duration of a purely Cubic congestion control scales with the cube-root of bandwidth-delay product (BDP). So, as link capacity or RTT doubles, the duration of the cycles of a single flow grow by $2^{1/3} \approx 1.26$.

Note, though, that the *amplitude* of Cubic's queue-delay variation still scales like Reno, i.e. linearly with RTT and invariant with link capacity, because it is determined by the multiplicative decrease.

‘

Country	Population	% of popul'n	Internet users [Wik20]	Fixed bandwidth (Mb/s) [Ook21]	CDN latency (ms) [Beg19]
China	1,427,647,786	69.27%	988,990,000	172.95	66
India	1,366,417,754	55.31%	755,820,000	55.76	20
United States	324,459,463	96.26%	312,320,000	191.97	14
Indonesia	266,911,900	79.56%	212,354,070	26.31	21
Brazil	213,300,278	75.02%	160,010,801	90.3	21
Nigeria	205,886,311	66.15%	136,203,231	16.33	4
Russia	143,989,754	82.39%	118,630,000	87.01	30
Japan	127,484,450	91.27%	116,350,000	167.18	8
Bangladesh	164,945,471	70.41%	116,140,000	36.02	39
Pakistan	213,756,286	47.10%	100,679,752	11.74	41
Mexico	128,972,439	69.01%	89,000,000	48.35	30
Iran	83,020,323	94.06%	78,086,663	19.17	76
Germany	82,114,224	94.74%	77,794,405	120.93	14
Philippines	104,918,090	69.58%	73,003,313	49.31	25
Vietnam	97,338,579	70.04%	68,172,134	66.38	23
United Kingdom	66,181,585	98.22%	65,001,016	92.63	11
Turkey	80,745,020	76.88%	62,075,879	34.95	41
France	64,979,548	89.32%	58,038,536	192.25	14
Egypt	101,545,209	53.91%	54,740,141	39.66	81
Italy	60,416,000	83.65%	50,540,000	90.93	19
South Korea	50,982,212	96.94%	49,421,084	241.58	4
Spain	46,750,321	90.70%	42,400,756	186.4	11
Thailand	69,037,513	52.89%	36,513,941	206.81	24
Poland	38,382,576	90.40%	34,697,848	130.98	12
Canada	36,624,199	92.70%	33,950,632	167.61	19
Argentina	44,271,041	75.81%	33,561,876	51.51	19
South Africa	56,717,156	56.17%	31,858,027	43.91	20
Colombia	49,065,615	62.26%	30,548,252	53.73	48
Ukraine	44,222,947	66.64%	29,470,000	67.52	23
Saudi Arabia	32,938,213	82.12%	27,048,861	90.24	76
Malaysia	31,624,264	80.14%	25,343,685	103.34	9
Morocco	35,739,580	61.76%	22,072,765	25.37	44
Taiwan	23,626,456	92.78%	21,920,626	163.85	7
Australia	24,450,561	86.54%	21,159,515	77.88	14
Venezuela	31,977,065	64.31%	20,564,451	17.9	77
Algeria	41,318,142	47.69%	19,704,622	6.78	67
Ethiopia	104,957,438	18.62%	19,543,075	12.39	55
Iraq	38,274,618	49.36%	18,892,351	29.88	77
Uzbekistan	31,910,641	52.31%	16,692,456	39.2	78
Myanmar	53,370,609	30.68%	16,374,103	22.75	50
Netherlands	17,035,938	93.20%	15,877,494	152.94	9
Peru	32,165,485	48.73%	15,674,241	51.81	33
Chile	18,054,726	82.33%	14,864,456	176.48	18
		% world		Averages weighted by Internet users	
Above countries		90.15%	4,292,105,058	103.32	34
World		100.00%	4,761,334,541		

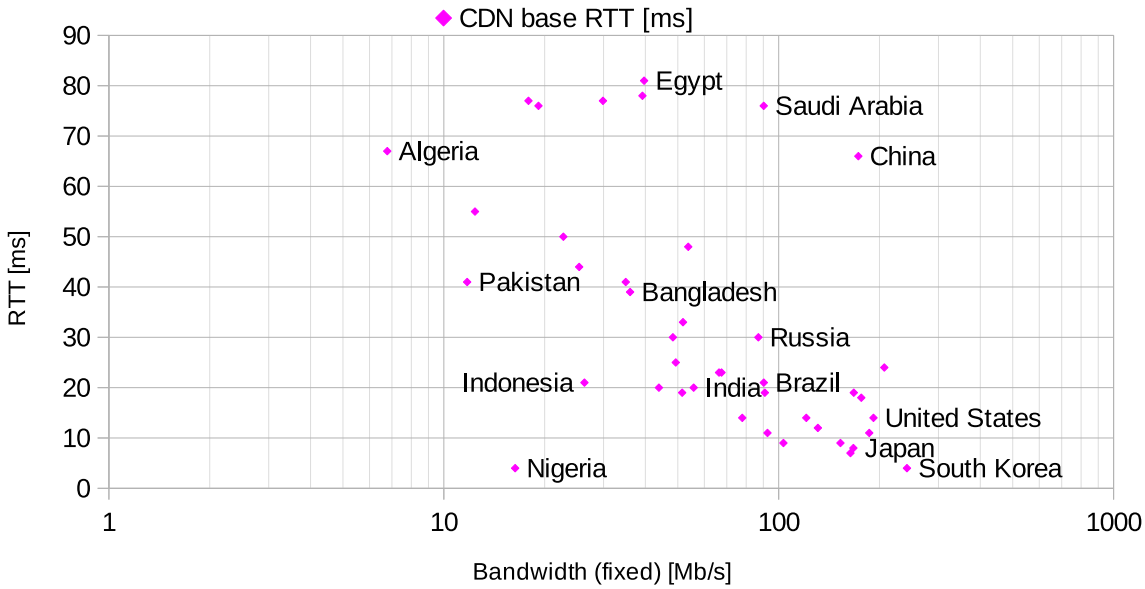

Figure 4: Scatter-plot per country of average base RTT from users to CDNs and average fixed access bandwidth. Only the 43 countries with the most Internet users are plotted, representing 90% of Internet users. The top 10 are labelled as well as those at the extremes

References

[Beg19] Emir Beganović. Analysing Global CDN Performance. Blog, RIPE Labs, August 2019. Online: https://labs.ripe.net/author/emirb/analysing-global-cdn-performance/.

[DOC19] Data-Over-Cable Service Interface Specifications DOCSIS® 3.1; MAC and Upper Layer Protocols Interface Specification. Specification CM-SP-MULPIv3.1-I17-190121, CableLabs, January 2019.

[DSBEW21] Koen De Schepper, Bob Briscoe (Ed.), and Greg White. DualQ Coupled AQMs for Low Latency, Low Loss and Scalable Throughput (L4S). Internet Draft draft-ietf-tsvwg-aqm-dualq-coupled-15, Internet Engineering Task Force, May 2021. (Work in Progress).

[dSBTB15] Koen de Schepper, Olga Bondarenko, Inton Tsang, and Bob Briscoe. 'Data Center to the Home': Ultra-Low Latency for All. Technical report, RITE Project, June 2015. http://riteproject.eu/publications/.

[Jac88] Van Jacobson. Congestion Avoidance and Control. *Proc. ACM SIGCOMM'88 Symposium, Computer Communication Review*, 18(4):314–329, August 1988.

[JK88] Van Jacobson and Michael J. Karels. Congestion Avoidance and Control. Technical report, Laurence Berkeley Labs, November 1988. (a slightly modified version of the original published at SIGCOMM in Aug'88 [Jac88]).

[KKFR15] Chamil Kulatunga, Nicolas Kuhn, Gorry Fairhurst, and David Ros. Tackling Bufferbloat in capacity-limited networks. In *2015 European Conference on Networks and Communications (EuCNC)*, pages 381–385, 2015.

[LR20] Humberto La Roche. CDN Caching and Video Streaming Performance. Blog, August 2020.

[MSJ+19] Ayush Mishra, Xiangpeng Sun, Atishya Jain, Sameer Pande, Raj Joshi, and Ben Leong. The Great Internet TCP Congestion Control Census. *Proc. ACM on Measurement and Analysis of Computing Systems*, 3(3), December 2019.

[Ook21] Ookla. Speedtest Global Index. http://www.speedtest.net/global-index, April 2021.

[PPP+13] Rong Pan, Preethi Natarajan Chiara Piglione, Mythili Prabhu, Vijay Subramanian, Fred Baker, and Bill Ver Steeg. PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem. In *High Performance Switching and Routing (HPSR'13)*. IEEE, 2013.

[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.

[Wik20] List of countries by number of Internet users. Online: https://en.wikipedia.org/wiki/List_of_countries_by_number_of_Internet_users, 2019–2020.

Document history

Version	Date	Author	Details of change
00A	01-Jun-2021	Bob Briscoe	First draft
01	02-Jun-2021	Bob Briscoe	Changed Figure 3 to RTT under load following review by Vidhi Goel. Numerous minor corrections.
01A	04 Jun 2021	Bob Briscoe	Minor corrections following review.
02	05-Jul-2021	Bob Briscoe	Following review by Koen De Schepper, altered terminology, clarified different RTTs, added more rationale and added avg Reno qDelay appendix.

