Alternative Backoff: Achieving Low Latency and High Throughput with ECN and AQM

Naeem Khademi*, Grenville Armitage†, Michael Welzl*, Sebastian Zander§
Gorry Fairhurst‡ and David Ros¶

***Abstract*—A number of recently proposed Active Queue Management (AQM) mechanisms instantiate shallow buffers with burst tolerance to minimise the time that packets spend enqueued at a bottleneck. However, shallow buffering causes noticeable TCP performance degradation as a path's underlying round trip time (RTT) heads above typical intra-country levels. Using less-aggressive multiplicative backoffs in TCP can compensate for shallow bottleneck buffering. AQM mechanisms may either drop packets or mark them using Explicit Congestion Notification (ECN), depending on whether the sender marked packets as ECN-capable. While a drop may therefore stem from any type of queue, an ECN-mark indicates that an AQM mechanism has done its job, and therefore the queue is likely to be shallow. We propose ABE: "Alternative Backoff with ECN", which consists of enabling ECN and letting individual TCP senders back off less aggressively in reaction to ECN-marks from AQM-enabled bottlenecks. Using controlled testbed experiments with standard NewReno and CUBIC flows, we show significant performance gains in lightly-multiplexed scenarios, without losing the delay-reduction benefits of deploying AQM. ABE is a sender-side-only modification that can be deployed across networks incrementally (requiring no flag-day) and offers a compelling reason to deploy and enable ECN across the Internet.**

***Index Terms*—ECN, low latency, AQM, TCP, congestion control.**

I. Introduction

Recent years have seen increasing mainstream awareness of how critical low latency (delay) is to today's Internet and to end users' quality of experience. The delay experienced by any given packet is heavily influenced by routing choices (distance), link speeds (serialisation) and queuing (buffering at bottlenecks during periods of congestion). Increasing network speeds have reduced the relative contribution of serialisation, and therefore placed more focus on the size and management of bottleneck buffers [1].

A key influence on end user experience is the way bottleneck buffering interacts with capacity estimation techniques of common transport protocols. The Internet's reliance on statistical multiplexing requires buffering to absorb transient periods of congestion. Loss-based TCP algorithms will try to fill a bottleneck's available buffer space before backing off after packet loss (congestion signal). When available buffering is significantly in excess of requirements the resulting queuing delay can far outweigh any delay contributed by distance [2].

Two complementary solutions to the above issue have emerged—Active Queue Management (AQM) [3] and Explicit Congestion Notification (ECN) [4]. AQM provides earlier feedback about the onset of congestion, and thus reduces buffer filling during congestion. ECN adds new information to packets in transit allowing ECN-capable AQM bottlenecks to deliver congestion feedback earlier than is possible using packet loss, and avoiding the cost of dropping packets to signal congestion [5].

Unfortunately, modern AQM schemes interact badly with the traditional TCP response to congestion notification. Standard TCP halves its window (a *multiplicative decrease factor,* β, of 0.5) in response to packet loss or ECN. So a common rule-of-thumb is to allocate bottleneck buffering at least equivalent to a path's intrinsic 'bandwidth delay product' (BDP)[1] to avoid under-utilisation. Yet recent AQM schemes, such as Controlled Delay (CoDel) [6], [7] and Proportional-Integral controller Enhanced (PIE) [8], [9], effectively instantiate a shallow bottleneck buffer with burst tolerance. So TCP performance suffers once a path's BDP exceeds the bottleneck AQM scheme's effective buffer size, whether congestion is signalled by packet loss or ECN [10], [11].

A. Modifying TCP sender response to ECN marks

We propose and evaluate Alternative Backoff with ECN (ABE), a simple sender-side TCP enhancement that enables low latency and high throughput through AQM-managed bottlenecks even for high-BDP paths, thus providing an incentive to deploy ECN as an alternative to loss-based congestion signaling. ABE can be summarised as follows:

- Upon packet loss, a TCP sender reduces its congestion window (*cwnd*) as usual (e.g., NewReno uses $\beta = 0.5$ to reduce *cwnd* by 50%).
- Upon receipt of an ECN mark, a TCP sender reduces *cwnd* by *less* than the response for loss (uses a larger β).

ABE is based on the intuition that meaningful ECN marks are generated by AQM schemes whose congestion indications are proactively aimed at keeping buffer occupancy low. An

*Email: {naeemk,michawe}@ifi.uio.no, Networks and Distributed Systems Group, Department of Informatics, University of Oslo, Norway

†Email: garmitage@swin.edu.au, School of Software and Electrical Engineering, Swinburne University of Technology, Australia

§Email: s.zander@murdoch.edu.au, School of Engineering and Information Technology, Murdoch University, Australia

‡Email: gorry@erg.abdn.ac.uk, School of Engineering, University of Aberdeen, United Kingdom

¶Email: dros@simula.no, Simula Research Laboratory, Norway

[1] Path's intrinsic bandwidth is defined as the bandwidth at the bottleneck link.

ABE sender thus compensates by backing off less, reducing the likelihood of the shallow bottleneck buffer draining to empty. Smaller backoffs will continue to occur as the AQM continues to react, ensuring the traffic does not build a long standing queue.

A similar idea was proposed by Kwon and Fahmy in 2002 [12], who coupled a larger β with a smaller additive increase factor in order to be TCP-friendly under all circumstances. Their evaluation also assumed the bottleneck's AQM to be RED [13], which is not necessarily deployed with a shallow buffer. The concept of TCP senders reacting differently to loss and ECN marks was also noted by Ott in 2005 [14]. In more recent years it has become common to relax the requirement of strict TCP-friendliness [15]. Consequently, ABE relies solely on using a larger β when reacting to ECN marks generated by modern AQMs such as PIE and CoDel.

ABE may be deployed incrementally – an ABE-enabled TCP sender exhibits completely standard TCP behaviour if either the destination or the path do not support ECN. If an ABE sender's flow traverses a bottleneck whose AQM instantiates an unexpectedly deep queue instead, then either packet losses or successive ECN marks will ensure the sender continues to back off appropriately.

This paper is a much improved presentation of the ABE idea first outlined in a 2015 technical report [16] which is also now being considered for IETF standardization [17], [18].

B. *The past and future role of ECN*

In 2001 RFC 3168 [4] defined ECN as a simple marking method to replace packet drops as a congestion indication. Although widely implemented, ECN is still not widely used [19], in part due to early experiences of failure of ECN-enabled TCP connections to 8% of web servers tested in 2000 [20]. This improved to $< 1\%$ of web servers tested in 2004 [21], yet in 2011 there were still a significant number of paths whose middleboxes mangled the ECN field in the IP header [22].

By 2014, the majority of the top million web servers offered ECN negotiation (56% for IPv4 and 65% for IPv6) [19]. Widespread client-side support for ECN-fallback means only 0.5% of websites suffer additional connection setup latency. ECN is now supported in the Linux and FreeBSD implementations of CoDel and PIE (turned on by default in FreeBSD), and is recommended for Internet deployment [23].

In light of increased interest in AQM for managing latency, ABE provides a new incentive to also deploy and enable ECN more widely.

C. *Paper structure*

The rest of this paper is structured as follows. We motivate ABE in Section II, showing the relationship between TCP backoff, bottleneck buffer size and path BDP, and demonstrating how TCP degrades over CoDel and PIE bottlenecks at plausible domestic and international round-trip times (RTTs). ABE is introduced in Section III. Section IV evaluates ABE using controlled testbed experiments with CUBIC and NewReno TCP. Related work and future work are considered in Sections V and VI respectively. We conclude in Section VII.

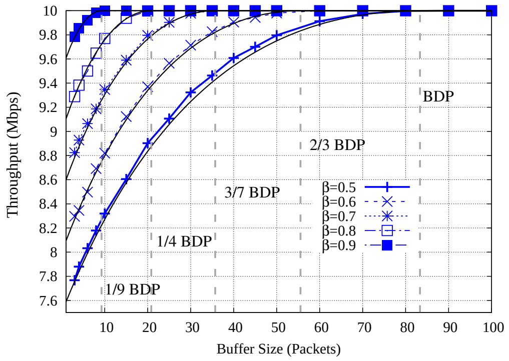


Fig. 1: Throughput of a single NewReno flow, with $C = 10\,\text{Mbps}$, $RTT = 100\,\text{ms}$ (model and simulation).

II. Background

The choice of multiplicative decrease factor $\beta = 0.5$ for standard TCP has been noted in [24] as a heuristic, conservative choice. TCP's Additive Increase, Multiplicative Decrease (AIMD) control of *cwnd* would also exhibit useful convergence properties with other values of β. To better understand the implications of AQMs enforcing small queues on long-RTT paths, we revisit how TCP performance is influenced by β, path characteristics and bottleneck queue size.

A. *TCP backoff, path characteristics and link utilisation*

Consider a single long-lived TCP flow traversing a path with round trip time of RTT_{path}, bottleneck link capacity C, and DropTail buffer of size b. Assume the TCP flow is in congestion avoidance mode, multiplicative decrease factor is $\beta \in (0, 1)$ and *cwnd* is not limited by the TCP receiver's advertised window (*rwnd*). Denote utilisation as U.

To achieve $U = 1$ the number of bytes in flight must not drop below the path's intrinsic BDP, $\delta = C \times RTT_{path}$. Given $cwnd_{max}$ as the number of unacknowledged bytes in flight just prior to backoff, it must be simultaneously true that $\beta * cwnd_{max} \geq \delta$ and $cwnd_{max} = (b + \delta)$. Solving for b gives:

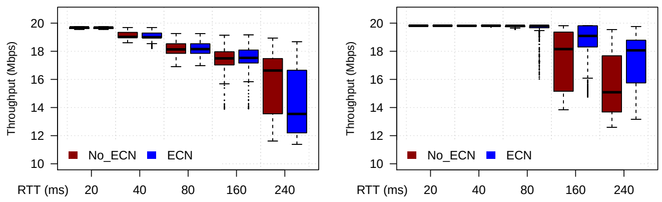
$$b \geq \delta \, \frac{1 - \beta}{\beta} \tag{1}$$

With $\beta = 0.5$, (1) yields the well-known $b \geq \delta$ rule-of-thumb for buffer sizing, with $b = \delta$ being the sweet spot and larger b merely resulting in higher queuing delays when the longer queue is filled. Rewriting equation (1) as

$$\beta \geq \frac{\delta}{b + \delta}\,, \tag{2}$$

makes clear that for given path δ, a larger β is required for (2) to hold (i.e., sustain $U = 1$) when buffer b shrinks below δ. To give this some concrete representation, Fig. 1 illustrates[2] how increasing β allows the throughput of a TCP flow running

[2] Using ns-2.35 simulation results and an analytical model described in [16].



(a) CoDel

(b) PIE

Fig. 2: Experiments confirm CUBIC TCP performance through CoDel and PIE bottlenecks degrades as RTT_{path} increases.

over a 100 ms path with 10 Mbps bottleneck to remain high even as b becomes smaller and smaller fractions of δ.

B. Using AQM with a large path RTT

Although CoDel and PIE differ in their specific details, in broad terms they both aim to directly limit the sojourn times of packets in a bottleneck queue, and hence limit the queuing delays experienced by traffic passing through the bottleneck. Unfortunately, this is also known to reduce TCP throughput when RTT is large [25], [26]. Fig. 2 uses results from a real-life experiment to illustrate how throughput drops off as RTT_{path} increases from domestic (20 ms) to international (240 ms) levels when a single CUBIC TCP flow traverses a 20 Mbps bottleneck for 90 seconds using either CoDel or PIE (at default settings) for queue management, and with or without ECN.[3] It is clear that ECN on its own provides limited assistance; simply replacing packet drops by packet marking does not mitigate the consequences of effectively small buffers.

Fig. 2 is easy to explain. For a given bottleneck rate C, the queuing delay targets of CoDel and PIE roughly equate to a particular buffer b. As RTT_{path} rises, b becomes increasingly shallow relative to the path's δ, eventually violating (1) and resulting in utilisation dropping below 100%.

At this point both intuition and (2) suggest we should increase β to offset the degree to which δ is exceeding the effective b of a CoDel or PIE bottleneck. However, the TCP sender should only apply an increased β when it is (or is likely to be) dealing with a short bottleneck queue somewhere inside the network path. We want to avoid the higher standing queues (and resulting queuing delays) that result from increasing β on connections through conventionally sized ($b \geq \delta$) bottlenecks.

III. Alternative Backoff with ECN (ABE)

ABE is based on the following observation: The negligible deployment to date of old style AQM such as RED [23] means that future ECN marks are likely generated by modern AQM mechanisms whose effective queues are quite shallow. Consequently, ABE requires TCP senders to only use an alternative, higher $\beta_{\text{ecn}} > \beta$ when reacting to ECN marks on ECN-enabled connections, and use their original β when reacting to a packet loss. ABE alters RFC 3168's requirement that TCP senders react to ECN marks as they react to packet loss [4]. A sender's additive increase is unchanged.

Section II motivated the use of ABE during a TCP connection's congestion avoidance phase on connections over high BDP paths. However, two different questions now present themselves: (a) what happens when using $\beta_{\text{ecn}} > \beta$ if the bottleneck's effective queue is not shallow *relative* to the path's δ, and (b) what impact might ABE have during slow-start?

Given that AQMs are deployed to keep latencies down, an ECN marking bottleneck is unlikely to have a large effective b. But if the path has a low δ (close to the bottleneck's effective b), using β_{ecn} is still safe – the connection will simply require a few more successive backoffs to reach where it would have been if not using ABE (or ultimately react to packet loss).

ABE can also benefit a TCP connection's transition from the slow-start phase (SS) to the congestion avoidance phase (CA). Conventionally, SS involves starting *cwnd* at an initial value (e.g. 10 MSS in some popular stacks), doubling *cwnd* every RTT while probing for path capacity, then entering CA after the first congestion indication (packet loss or mark) [4] by doing a backoff with $\beta = 0.5$.[4] This reverts *cwnd* back to the last value that successfully sent a full window of data.

However, at the point of backoff *cwnd* has overshot a path's capacity by between one packet and almost an entire BDP.[5] So while subsequently halving *cwnd* does reliably terminate the overshoot within one RTT, connections may enter CA with *cwnd* significantly below path capacity, requiring many RTTs for the flow to regain lost utilisation.

An ABE sender may mitigate this by using its higher $\beta_{\text{ecn}} > \beta$ for the backoff when transitioning from SS to CA as a result of an ECN mark. Fig. 3 depicts the issue. The solid lines each represent one NewReno flow using using $\beta_{\text{ecn}} = 0.5$, with $RTT_{path} = \{160, 240\}ms$ respectively. The $RTT_{path} = 240ms$ flow suffers under-utilisation when a small *cwnd* after SS overshoot results in a long period of slow growth in CA. The $RTT_{path} = 160ms$ flow is luckier after SS, but then a second congestion event leads to wasted capacity in CA. Dashed lines represent the same flows using $\beta_{\text{ecn}} = 0.9$. In both cases, the less aggressive backoff results in *cwnd* tending towards BDP soon after exiting SS, better path utilisation and reduced completion times for the flow.[6]

The issue is of particular interest when handling short flows, such as common web traffic. Recent trends [28] show web-related short flows are getting longer, increasing the probability that they terminate not long after entering CA with a potentially suboptimal *cwnd*. All experiments in this paper use the higher β_{ecn} for SS to CA transition.

[3] CUBIC's non-traditional use of $\beta = 0.7$ makes Fig. 2 a conservative illustration. The throughput loss is worse for NewReno flows ($\beta = 0.5$). By contrast, over the same RTT_{path} and using conventional DropTail queue management, CUBIC achieves 100% capacity (albeit with significant additional queuing delay).

[4] True for both NewReno and CUBIC (even with modern CUBIC including Hystart [27] to minimise SS overshoot).

[5] Depending on how a certain multiple of TCP's initial window is aligned with the path BDP and the queue length.

[6] On the down side, a large overshoot results in *cwnd* being reduced in multiple steps, which increases the latency over multiple RTTs after slow-start (however, the number of these RTTs is bounded by $\log_{\beta_{\text{ecn}}}(0.5)$).

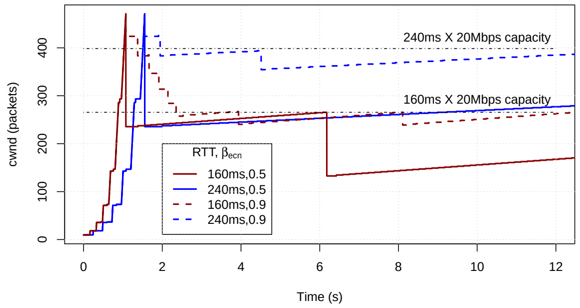


Fig. 3: The effect of overshoot at the end of slow-start for $\beta_{ecn} = \{0.5, 0.9\}$, with $C = 20\,$Mbps (test bed results).

IV. Evaluation

Our evaluation involves controlled, testbed experiments[7] whose conditions are described further in [16].

A. AQM algorithms and parameters

Our experiments considered ABE-enabled flows traversing PIE and CoDel bottlenecks. We deferred testing with FQ_CoDel [30] as it is known to trigger CoDel-like underutilisation on paths with higher RTTs [31], and we wished to explore interactions between ABE and non-ABE flows in single-queue AQM contexts.

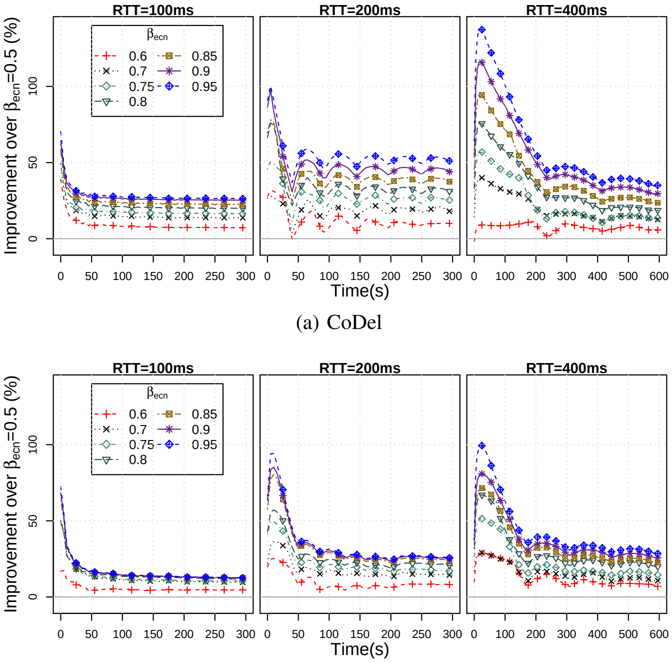
We used Linux 3.17.4 implementations of PIE[8] and CoDel with their default parameters. CoDel's *interval* and *target* were set to 100 ms and 5 ms. For PIE, *target* and *tupdate* were set to 20 ms and 30 ms, *alpha* was 0.125, *beta* was 1.25, *max_burst* (used to allow bursts of a given size to pass) was set to 100 ms and *bytemode* was turned off. Maximum queue length was set to the Linux default of 1000 packets in all scenarios.

In Linux kernel 3.17.4, PIE drops packets on ECN-enabled flows when PIE's drop/mark probability exceeds 10%. Although this ostensibly provides a defense against non-responsive ECN-enabled flows, we disabled it for all our experiments due to its detrimental impact on ECN-enabled flows that are responsive to congestion signals.

We patched our testbed's FreeBSD 10.1 and Linux (openSUSE 13.2) hosts to generate ABE-enabled NewReno and CUBIC flows respectively [33].

B. Latency vs. throughput for a NewReno or CUBIC flow

As β_{ecn} increases above β we would expect to see both improved throughput and increased queuing delays. But since ABE presumes ECN marks indicate the presence of a bottleneck using modern AQM, traffic is already being marked at low buffering thresholds, limiting the latency increase experienced by ECN-enabled traffic using $\beta_{\text{ecn}} > \beta$.



(a) CoDel

(b) PIE

Fig. 4: Improvement over $\beta_{\text{ecn}} = 0.5$ in terms of bytes received for a single NewReno flow; $C = 10\,$Mbps.

Despite some applications (such as web browsers) commonly opening multiple connections (flows), we focus our current analysis on the single-flow scenario as (1) it is the worst-case for throughput, and (2) the present trade-off is mainly determined by long-lived connections [34].[9]

Using results from live testbed experiments, Fig. 4 and Fig. 5 illustrate the latency vs throughput trade-off for a single long-lived NewReno flow traversing CoDel or PIE bottlenecks, for $RTT_{path} = \{100, 200, 400\}ms$ and $\beta_{\text{ecn}} = \{0.5, 0.6, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95\}$.

Fig. 4 shows that for either AQM we see significant gains in throughput (averaged over ten runs relative to throughput achieved with $\beta = 0.5$) as β_{ecn} increases. The improvements are particularly notable right after slow-start, and for long-lived flows at larger RTT_{path}. For example, a long-lived flow sees 24% (CoDel) and 23% (PIE) throughput improvement for $\beta_{\text{ecn}} = 0.8$ when $RTT_{path} = 400ms$.

Fig. 5 shows CDFs of estimated queuing delay[10] averaged over ten runs for the scenarios of Fig. 4. CoDel's deterministic marking policy keeps queuing delay close to its 5 ms target for all β_{ecn}. PIE also keeps delay low, with a more heavy-tailed distribution due to the randomised and burst-tolerant nature of PIE's ECN marking [38].

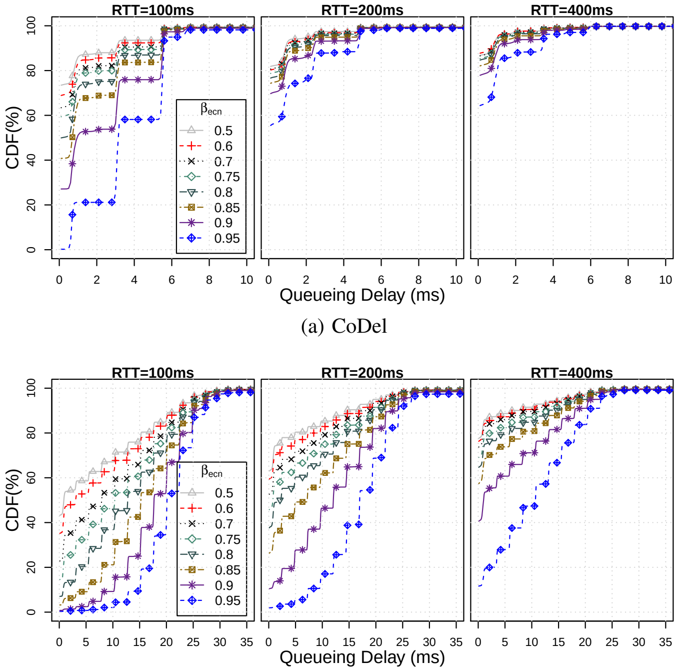
With $RTT_{path} = 100ms$ (where throughput gain is smallest and the delay increase is largest), $\beta_{\text{ecn}} = 0.95$ sees CoDel and

[7] Using TEACUP [29], an experiment automation suite for TCP that is driving a physical testbed with a dumbbell topology using FreeBSD and Linux hosts, and a Linux-based software router.

[8] Note the Linux PIE implementation between kernel 3.14 to 4.5 is based on the early [32] rather than later [9], with no major change during this period, http://lxr.free-electrons.com/diff/net/sched/sch_pie.c?v=3.14;diffval=4.5;diffvar=v.

[9] End-user experience is also likely to become more influenced by single-flow performance as browsers shift to using HTTP/2 [35] or SPDY [36].

[10] Computed from the actual RTT measured using SPP [37] minus the sum of constant path RTT configured plus 1 ms for overheads (short queue in the ACK direction plus delays at sender/receiver).



(a) CoDel

(b) PIE

Fig. 5: CDF of estimated queuing delay for a single NewReno flow; $C = 10\,\text{Mbps}$.

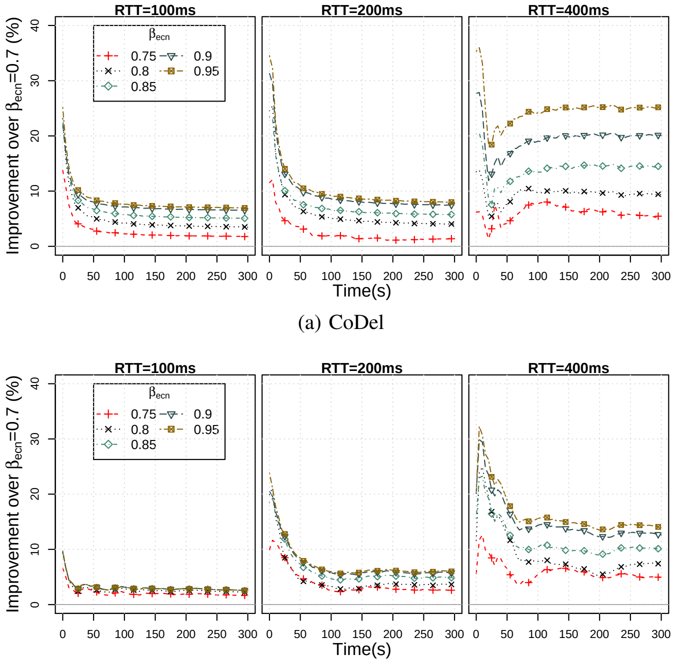
PIE add 3 ms/2 ms and 20 ms/5 ms to the 50th/90th percentile delay respectively (compared to $\beta_{\text{ecn}} = 0.5$). However, with $\beta_{\text{ecn}} = 0.85$ the increase is only 1 ms/<2 ms (CoDel) and 15 ms/3 ms (PIE). For a NewReno flow, the range $0.7 \le \beta_{\text{ecn}} \le 0.85$ appears to be a good trade-off between improved throughput and increased latency.

Fig. 6 and Fig. 7 illustrate the latency vs throughput trade-offs when the NewReno flow is replaced by a CUBIC flow.[11]

Throughput gains are evident, although smaller than Fig. 4 due to CUBIC's already aggressive additive increase and default $\beta = 0.7$. For example, when $RTT_{path} = 400ms$ a long-lived flow sees gains of 15% and 25% (CoDel) or 10% and 15% (PIE) for $\beta_{\text{ecn}} = \{0.85, 0.95\}$ respectively.

Fig. 7's queuing delay distributions are similar to NewReno's in Fig. 5. With $RTT_{path} = 100ms$ and $\beta_{\text{ecn}} = \{0.85, 0.95\}$ we see the increase in 50th/90th percentile delay of 2.5 ms/<1 ms and 5 ms/<1 ms (CoDel) and 2.5 ms/<1 ms and 5 ms/<1 ms (PIE) respectively relative to $\beta_{\text{ecn}} = 0.7$. For a CUBIC flow, the range $0.85 \le \beta_{\text{ecn}} \le 0.95$ seems a good trade-off between improved throughput and increased latency.

Fig. 8 provides an alternative illustration of how increasing β_{ecn} allows a NewReno flow to recover throughput otherwise lost when running through a CoDel bottleneck, particularly for higher RTT_{path}. Although not shown to save space, for each RTT_{path} the median RTT was basically unaffected by β_{ecn}.



(a) CoDel

(b) PIE

Fig. 6: Improvement over $\beta_{\text{ecn}} = 0.7$ in terms of bytes received for a single CUBIC flow; $C = 10\,\text{Mbps}$.

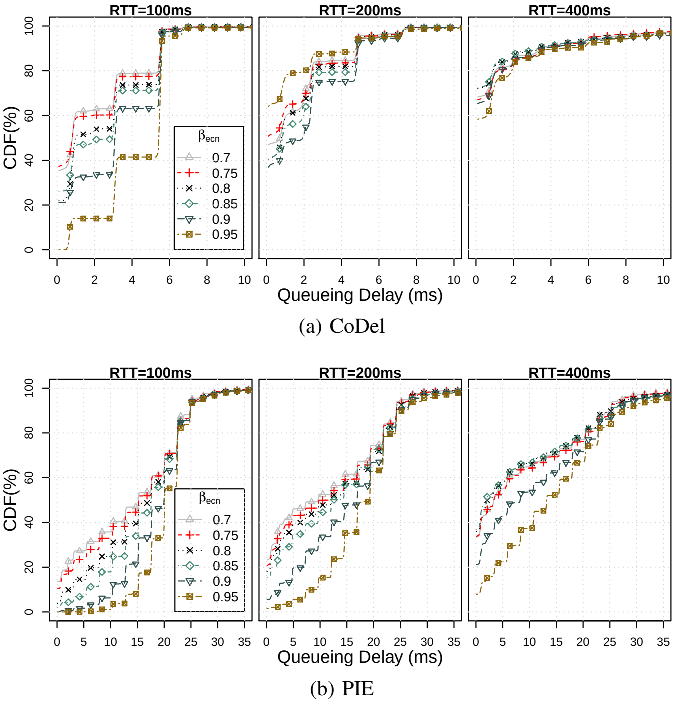
C. Convergence between bulk transfers

We also used live testbed experiments to investigate how β_{ecn} influences the time for similar flows to converge on a fair rate allocation, with convergence measured using Jain's Fairness Index [39]. Flows were started one after the other, with a delay of 30 seconds in between. When a new flow started at time $t = t_0$, for each existing flow i we measured over time the cumulative number of bytes transferred $x_i(t)$, from t_0 onwards, and computed a running fairness index $F(t)$ as

$$F(t) = \frac{\left(\sum_{i=1}^{N} x_i(t)\right)^2}{N \sum_{i=1}^{N} x_i(t)^2}$$

where N is the total number of flows active at time t. Convergence time T was defined as the time it took $F(t)$ to reach a threshold $\theta = 0.95$.
Initial tests using $N = \{2, 4, 10\}$ flows with equal RTTs did not find any consistent improvement or disadvantage in convergence times from changing β_{ecn}. Therefore we illustrate the impact of β_{ecn} on fairness using experiments with $N = 2$ flows, bottleneck speeds $C = \{1, 5, 10, 20, 40, 100\}Mbps$, $RTT_{path} = \{10, 20, 40, 60, 80, 160, 240, 320\}ms$, and different values of β_{ecn}. For each of the 48 combinations of capacity and RTT, we measured T using the default backoff factors ($\beta = 0.7$ and $\beta = 0.5$ for CUBIC and NewReno respectively), which we call T_β, and compared it to T using a higher backoff factor $\beta_{\text{ecn}} > \beta$, which we call $T_{\beta_{\text{ecn}}}$. The metric of interest is

[11] CUBIC's fast convergence mode was turned off to provide conservative results, as disabling fast convergence produced slightly worse improvements for increased β_{ecn}.



(a) CoDel

(b) PIE

Fig. 7: CDF of estimated queuing delay for a single CUBIC flow; $C = 10\,\text{Mbps}$.

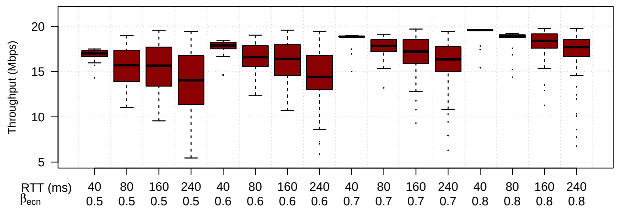
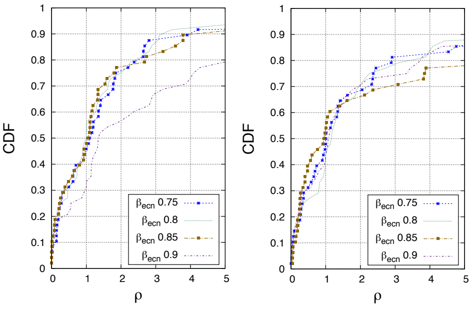


Fig. 8: Throughput improves with higher β_{ecn}, more so for large RTT_{path} (NewReno flow over CoDel, $C = 20\,\text{Mbps}$).

then the *ratio* of convergence times $\rho = T_{\beta_{\text{ecn}}}/T_{\beta}$, e.g. $\rho = 2$ means convergence to fairness with $\beta_{\text{ecn}} > \beta$ takes twice as long as it does with $\beta_{\text{ecn}} = \beta$.

Figs. 9 and 10 show the CDF of ρ, obtained from 384 and 576 experiments, respectively. We also tested with $\beta_{\text{ecn}} = \{0.55, 0.65\}$ for NewReno and $\beta_{\text{ecn}} = 0.95$ for both NewReno and CUBIC. Results were similar; we omit these lines for clarity. All experiments ran long enough to ensure convergence ($F(t) = \theta$). The distribution of ρ has a long tail due to some extreme outliers—long convergence times for the depicted β_{ecn} values and very small ones for $\beta_{\text{ecn}} = \beta$. Some were one "unlucky" flow always being affected by a congestion mark. In other cases, using $\beta_{\text{ecn}} = \beta$ saw slow-start ending at just the right value, creating an outlier for all the $\beta_{\text{ecn}} > \beta$ cases.

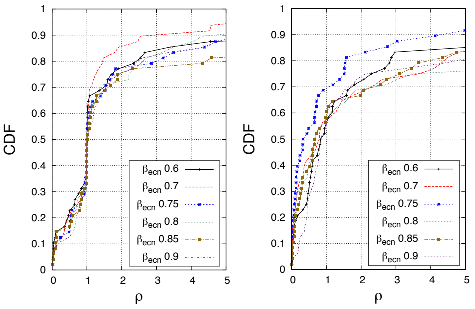
All β_{ecn} values can sometimes cause even faster convergence than the default, but for CUBIC no value of β_{ecn} could significantly improve it (generally, less than half of the cases converged faster than the default case). This is somewhat different with NewReno, where at least with PIE all the plotted values improve convergence in more than half of the cases, with a particularly good result for $\beta_{\text{ecn}} = 0.75$.



(a) CoDel

(b) PIE

Fig. 9: Distribution of the ratio ρ of convergence times $T_{\beta_{\text{ecn}}}$ (where $\beta_{\text{ecn}} > \beta$) and T_{β} (where $\beta_{\text{ecn}} = \beta$) with CUBIC.



(a) CoDel

(b) PIE

Fig. 10: Distribution of the ratio ρ of convergence times $T_{\beta_{\text{ecn}}}$ (where $\beta_{\text{ecn}} > \beta$) and T_{β} (where $\beta_{\text{ecn}} = \beta$) with NewReno.

Our results suggest that convergence time is not significantly harmed by changing β_{ecn} since in most cases the median ρ is close to, or smaller than, 1.

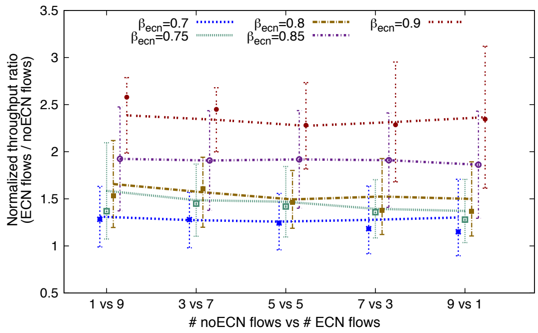


Fig. 11: Ratio of throughput of 10 ECN flows vs 10 non-ECN flows over a CoDel queue using several RTTs ($C = 10\,\text{Mbps}$). Lines intersect the arithmetic mean, dots indicate the median.

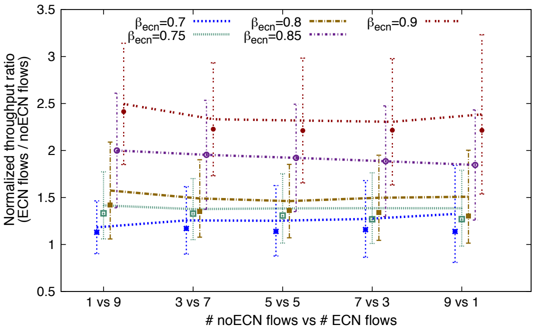


Fig. 12: Ratio of throughput of 10 ECN flows vs 10 non-ECN flows over a PIE queue using several RTTs ($C = 10\,\text{Mbps}$). Lines intersect the arithmetic mean, dots indicate the median.

D. Fairness between bulk transfers

Next, we investigated the long-term impact that flows with a large β_{ecn} value can have on traffic that does not support ECN. To study the worst case, we tested CUBIC flows with various β_{ecn} values in competition with NewReno flows that do *not* support ECN. Figs. 11 and 12 show the throughput ratios for various combinations of 10 flows across a 10 Mbps bottleneck link using either CoDel or PIE. Throughput was computed over the after-slow-start period of each flow lasting approximately 5 minutes. Ratios were normalised by weighting throughput values with the number of flows in each group. We carried out every test three times, using $RTT_{path} = \{20, 80, 160, 320\}ms$; even at $RTT_{path} = 320ms$ the trials encompassed 133 TCP congestion cycles. The error bars represent standard deviation over all runs.

The difference between CoDel and PIE is marginal, and generally a larger β_{ecn} value increased the throughput ratio, as could be expected. Our goal was to assess the potential "danger" of deploying ABE (i.e., using $\beta_{\text{ecn}} > \beta$) on the Internet. On average the increase in the throughput ratio is limited to a factor of 2.5 for the quite extreme value of β_{ecn}=0.9. For smaller β_{ecn} between 0.7 and 0.85 the average throughput ratio reduces to between 1.25 and 2. In fact, the averages for $\beta_{\text{ecn}} = \{0.75, 0.8, 0.85\}$ lie within the error bars of the ratios seen with CUBIC's default $\beta_{ecn} = 0.7$, suggesting ABE ($\beta_{\text{ecn}} > \beta$) is not much worse than no ABE ($\beta_{\text{ecn}} = \beta$) in terms of fairness for β_{ecn}up to at least 0.85.

V. Related Work

Since its inception, ECN has attracted much interest in the research community. One obvious question that papers have tried to answer was: can we get "more bang for the bits"? Given that there are two ECN bits in the IP header, they can be re-defined to distinguish between multiple levels of congestion [40], [41], or used as defined, but marked with a different rule that must be known to the end hosts [42], [43]. LT-TCP [44] proposed using ECN to distinguish between random packet losses on wireless links and congestion; NF-TCP [45] suggests using it as a means to detect congestion early and back off more aggressively than standard TCP, thereby reducing the delay caused for competing applications. The potential applications are broad—yet, none of these methods were designed to interoperate with the currently deployed mix of standards-compliant ECN-capable and -incapable routers and end hosts such that it could be gradually introduced in the Internet.

Congestion Exposure (ConEx) [46], which is based on Re-ECN [47], uses traffic shapers to hold users accountable for the congestion they cause on the whole path (as opposed to giving most capacity to the host running the most aggressive congestion control). ConEx is gradually deployable, but needs significant changes to the infrastructure (hosts must be modified to give information to the network and traffic shapers must be deployed).

Some schemes were defined for use within a configured operator domain. In Pre-Congestion Notification (PCN) [48], the ECN bits are used to inform DiffServ ingress routers of incipient congestion to better support admission control. Data Center TCP (DCTCP) [10] also updates TCP's ECN response resulting in measurable benefits when network devices update their ECN marking behaviour. DCTCP requires changing the receiver to more precisely feed back the number of ECN marks at a greater precision (i.e., more than one signal per RTT). It also requires a specific configuration of RED on routers to operate on the instantaneous queue length, which is currently only considered safe when all routers on a path use these updated marking rules. The required device support limits DCTCP to operate within one data center [49].

A recent proposal called L4S [50] investigates the possibility of gradually introducing a DCTCP-like scheme in the Internet. That proposal is very close in spirit to our work. However, just like DCTCP, it requires changing the receiver which is still ongoing IETF work [51]. Another problem with DCTCP-like schemes is the inherent unfairness against standard loss-based TCP traffic, since DCTCP tends to highly penalise these flows when sharing a common bottleneck. In order to mitigate this problem recent work proposes a dual queue AQM mechanism [52], [53] that provides a fair share of bandwidth between DCTCP and non-ECN flows. However, this mechanism cannot be deployed quickly as it requires support from all routers along a path.

ABE's response to ECN is still based on a reduction each RTT, rather than per marked segment as for DCTCP. Marking the instantaneous queue length provides faster feedback, so DCTCP-like approaches should theoretically perform better than ABE, if they can be combined with a sender behaviour that strikes the correct balance between efficiency across updated routers and compatibility across old routers. The major advantage of ABE is its simplicity. It only requires a single parameter change at the sender and it is not sensitive to the marking method (ABE works with network devices configured for DCTCP as well as other router marking policies). We argue that taking the shorter deployment path with ABE will give incentive to wider ECN deployment in the near future.

The idea of using values of $\beta \neq 0.5$ is at the basis of proposals for improving performance of long-lived TCP flows in high-speed networks. CUBIC is one example of congestion controllers tailored to high-speed links, but other similar schemes can be found in the literature. For instance, H-TCP [54] uses $\beta \in (0.5, 0.8)$, and the value of β is adapted as a function of measured minimum and maximum RTTs. In a similar vein, after loss is experienced TCP Westwood [55] sets *cwnd* to the product of the estimated available capacity and the lowest observed RTT—thus, the equivalent β is variable and dependent on network conditions. High-Speed TCP (HSTCP) [56] adapts β in the range $(0.5, 1)$, with β taking higher values for larger *cwnd* above a threshold. In these proposals, the rationale for chosing a larger β is to allow for a faster recovery of *cwnd* after loss;[12] with "standard" congestion control this can take many RTTs over paths with large BDP.

VI. Future Work

Our evaluation explored the benefits of ABE with bottlenecks managed by CoDel or PIE using the recommended CoDel and PIE AQM parameters. We also used the same β_{ecn} value in both slow start and congestion avoidance. Intuitively, one may think that using more aggressive AQM parameters for marking should advocate a higher β_{ecn} value. However, we expect there is a limit to how aggressively an AQM scheme can react, before even packets in a natural packet burst are punished by ECN marks or drops. Such questions concerning AQM tuning should be investigated in future work.

The choice of β_{ecn} would depend on the AQM parameters on the paths as well as on the transport being used. However, it is within the scope of AQM designers and network operators (and not the sender's transport) to configure network devices on the path appropriately with the parameters that keep the latency at low level, similar to the default values used by CoDel and PIE [23]. With regards to choice of transport, since the sender is aware of the congestion control mechanism it uses (e.g., CUBIC or NewReno), it can already select the β_{ecn} values recommended in this paper. An analytical justification for the choice of β_{ecn} is left for future work.

For this paper we explored the potential benefits of using a static β_{ecn} (it is a simple change in the sender). Investigating methods for dynamically adapting β_{ecn} according to network conditions is an interesting area for future work.

ECN-enabled routers need to protect themselves from overload by unresponsive traffic. To this end RFC 3168 and RFC 7567 recommend dropping packets in case of excessive traffic (e.g., at a queue length above the threshold for CE-marking). However, a drop threshold set too low can significantly harm performance. Improved overload protection methods (e.g., auto-tuning of drop thresholds) requires future research.

VII. Conclusion

This paper proposes and motivates ABE, which asks that TCP senders use a higher multiplicative decrease factor $\beta_{\text{ecn}} > \beta$ when reacting to ECN marks on ECN-enabled connections, and use their original β when reacting to a packet loss. We show that despite low implementation cost, ABE can bring important performance improvements for TCP flows traversing bottlenecks that use modern AQM techniques. ABE achieves this using standard ECN marking as specified in RFC 3168 for routers/middleboxes and TCP receivers. It also defaults to a conservative behaviour whenever ECN is not supported along the path, ensuring ABE is incrementally deployable.

As our results in §IV show, the choice of β_{ecn} is important but not overly critical, in the sense that ABE seems robust and offers performance improvements across a range of values of β_{ecn}. Setting $\beta_{\text{ecn}} \in [0.7, 0.85]$ for NewReno (cf. $\beta = 0.5$) and $\beta_{\text{ecn}} \approx 0.85$ for CUBIC (cf. $\beta = 0.7$) seems to provide reasonable trade-offs between latency, throughput and fairness across a range of scenarios.

We expect methods like ABE to encourage more use of ECN, and as use increases, we believe the time will become ripe to revisit proposals for alternative ECN marking, based on the techniques discussed in §V. When use becomes widespread, router/middlebox manufacturers will have an incentive to implement these improved ECN specifications to further optimise performance. Hosts using ABE will then also be able to update their β_{ecn}.

The use of ECN must be initiated by the TCP client. This makes ABE immediately applicable for use cases where a host updated with ABE initiates a connection and then transmits data, such as uploads to the Cloud, Web 2.0 applications, etc. In use cases where a client initiates a connection to retrieve data from a server, such as web surfing, ABE requires the server to be updated. An example of the expected performance at the server side with ECN, but without ABE, is shown in Fig. 2. Given such results and the increasing ability of routers to correctly pass ECN-enabled packets, no significant disadvantage is to be expected in this case, and the impact of ABE will increase as servers introduce support for it.

In conclusion, experimental results have been presented that show real benefit from updating TCP's backoff in response to ECN marks (β_{ecn}), and we assert that this can provide a significant performance incentive towards greater use of the ECN across the Internet. ABE is now also being considered for standardization within the IETF [17], [18].

VIII. Acknowledgements

This work was partly funded by the European Community under its 7th Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The views expressed are solely those of the authors.

References

[1] B. Briscoe, A. Brunstrom, A. Petlund, D. Hayes, D. Ros, I. J. Tsang, S. Gjessing, G. Fairhurst, C. Griwodz, and M. Welzl, "Reducing Internet Latency: A Survey of Techniques and Their Merits," *IEEE Communications Surveys Tutorials*, vol. 18, pp. 2149–2196, 3rd quarter 2016.

[2] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *ACM Queue*, vol. 9, pp. 40:40–40:54, Nov 2011.

[3] R. Adams, "Active Queue Management: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 15, pp. 1425–1476, 3rd quarter 2013.

[12] These mechanisms also adapt the window-increase parameter α to increase performance.

[4] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP." RFC 3168 (Proposed Standard), Sep 2001. Updated by RFCs 4301, 6040.

[5] G. Fairhurst and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)." RFC 8087 (Informational), Mar 2017.

[6] K. Nichols and V. Jacobson, "Controlling Queue Delay," *ACM Queue*, vol. 10, pp. 20:20–20:34, May 2012.

[7] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled Delay Active Queue Management." Internet Draft, https://tools.ietf.org/html/draft-ietf-aqm-codel, Mar 2017.

[8] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem.," in *IEEE HPSR*, (Taipei, Taiwan), Jul 2013.

[9] R. Pan, P. Natarajan, F. Baker, and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem." RFC 8033 (Experimental), Feb 2017.

[10] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," in *ACM SIGCOMM*, (New Delhi, India), Aug 2010.

[11] D. Bansal and H. Balakrishnan, "Binomial Congestion Control Algorithms," in *IEEE INFOCOM*, (Anchorage, AK, USA), Apr 2001.

[12] M. Kwon and S. Fahmy, "TCP Increase/Decrease Behavior with Explicit Congestion Notification (ECN)," in *IEEE ICC*, (New York, New York, USA), May 2002.

[13] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, pp. 397–413, Aug 1993.

[14] T. J. Ott, "Transport Protocols in the TCP Paradigm and Their Performance," *Telecommun. Syst.*, vol. 30, pp. 351–385, Dec 2005.

[15] B. Briscoe, "Flow Rate Fairness: Dismantling a Religion," *ACM SIGCOMM CCR*, vol. 37, pp. 63–74, Apr 2007.

[16] N. Khademi, M. Welzl, G. Armitage, C. Kulatunga, D. Ros, G. Fairhurst, S. Gjessing, and S. Zander, "Alternative Backoff: Achieving Low Latency and High Throughput with ECN and AQM." CAIA Technical Report 150710A, http://caia.swin.edu.au/reports/150710A/CAIA-TR-150710A.pdf, 10 July 2015.

[17] D. Black, "Explicit Congestion Notification (ECN) Experimentation." Internet Draft, https://tools.ietf.org/html/draft-ietf-tsvwg-ecn-experimentation, Mar 2017.

[18] N. Khademi, M. Welzl, G. Armitage, and G. Fairhurst, "TCP Alternative Backoff with ECN (ABE)." Internet Draft, https://tools.ietf.org/html/draft-ietf-tcpm-alternativebackoff-ecn, Feb 2017.

[19] B. Trammell, M. Kühlewind, D. Boppart, I. Learmonth, G. Fairhurst, and R. Scheffenegger, "Enabling Internet-wide Deployment of Explicit Congestion Notification," in *PAM*, (New York), Mar 2015.

[20] J. Padhye and S. Floyd, "Identifying the TCP Behavior of Web Servers," in *ACM SIGCOMM*, (Stockholm, Sweden), Aug 2000.

[21] A. Medina, M. Allman, and S. Floyd, "Measuring the Evolution of Transport Protocols in the Internet," *ACM SIGCOMM CCR*, vol. 35, pp. 37–52, Apr 2005.

[22] S. Bauer, R. Beverly, and A. Berger, "Measuring the State of ECN Readiness in Servers, Clients,and Routers," in *ACM IMC*, (Berlin), Nov 2011.

[23] F. Baker and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management." RFC 7567 (Best Current Practice), Jul 2015.

[24] V. Jacobson, "Congestion Avoidance and Control," in *Symposium Proceedings on Communications Architectures and Protocols*, SIGCOMM, (New York, NY, USA), pp. 314–329, ACM, 1988.

[25] N. Khademi, D. Ros, and M. Welzl, "The New AQM Kids on the Block: Much Ado about Nothing?," Technical Report 434, University of Oslo, Dept. of Informatics, Oct 2013.

[26] J. Schwardmann, D. Wagner, and M. Kühlewind, "Evaluation of ARED, CoDel and PIE," in *EUNICE*, (Rennes, France), Sep 2014.

[27] S. Ha and I. Rhee, "Taming the Elephants: New TCP Slow Start," *Computer Networks*, vol. 55, pp. 2092–2110, Jun 2011.

[28] "HTTP Archive." http://httparchive.org/trends.php.

[29] S. Zander and G. Armitage, "TEACUP v1.0 - A System for Automated TCP Testbed Experiments." CAIA Technical Report 150529A, http://caia.swin.edu.au/reports/150529A/CAIA-TR-150529A.pdf, 29 May 2015.

[30] T. Hoeiland-Joergensen, P. McKenney, D. Täht, J. Gettys, and E. Dumazet, "FlowQueue-Codel." Internet Draft, https://tools.ietf.org/html/draft-ietf-aqm-fq-codel, Mar 2016.

[31] C. Kulatunga, N. Kuhn, G. Fairhurst, and D. Ros, "Tackling Bufferbloat in Capacity-limited Networks," in *EuCNC*, (Paris, France), Jun 2015.

[32] R. Pan, P. Natarajan, F. Baker, G. White, B. VerSteeg, M. S. Prabhu, C. Piglione, and V. Subramanian, "PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem." Internet Draft, https://tools.ietf.org/html/draft-ietf-aqm-pie-02, Aug 2015.

[33] N. Khademi, "ABE Linux and FreeBSD patches." http://heim.ifi.uio.no/naeemk/research/ABE. [Accessed on 27 April 2017].

[34] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing Router Buffers," in *ACM SIGCOMM*, (Portland, Oregon, USA), Sep 2004.

[35] M. Belshe, R. Peon, and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)." RFC 7540 (Proposed Standard), May 2015.

[36] Y. Elkhatib, G. Tyson, and M. Welzl, "Can SPDY Really Make the Web Faster?," in *IFIP Networking*, (Trondheim, Norway), Jun 2014.

[37] S. Zander and G. Armitage, "Minimally-Intrusive Frequent Round Trip Time Measurements Using Synthetic Packet-Pairs," in *IEEE LCN 2013*, Oct 2013.

[38] N. Khademi, D. Ros, and M. Welzl, "The New AQM Kids on the Block: An Experimental Evaluation of CoDel and PIE," in *IEEE INFOCOM WKSHPS*, (Toronto, Ontario, Canada), Apr 2014.

[39] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems," Technical report TR-301, DEC Research, Sep 1984.

[40] A. Durresi, L. Barolli, R. Jain, and M. Takizawa, "Congestion Control Using Multilevel Explicit Congestion Notification," *Journal of Information Processing Society of Japan*, vol. 48, pp. 514–526, Feb 2007.

[41] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One More Bit is Enough," *IEEE/ACM Trans. Netw.*, vol. 16, pp. 1281–1294, Dec 2008.

[42] N. Vasic, S. Kuntimaddi, and D. Kostic, "One Bit is Enough: A Framework for Deploying Explicit Feedback Congestion Control Protocols," in *COMSNETS*, (Bangalore, India), Jan 2009.

[43] I. A. Qazi, L. L. H. Andrew, and T. Znati, "Congestion Control with Multipacket Feedback," *IEEE/ACM Trans. Netw.*, vol. PP, p. 1, Dec 2012.

[44] B. Ganguly, B. Holzbauer, K. Kar, and K. Battle, "Loss-Tolerant TCP (LT-TCP): Implementation and Experimental Evaluation," in *IEEE MILCOM*, (Orlando, Florida, USA), Oct 2012.

[45] M. Arumaithurai, X. Fu, and K. Ramakrishnan, "NF-TCP: Network Friendly TCP," in *IEEE LANMAN*, (Long Branch, New Jersey, USA), May 2010.

[46] B. Briscoe, R. Woundy, and A. Cooper, "Congestion Exposure (ConEx) Concepts and Use Cases." RFC 6789 (Informational), Dec 2012.

[47] B. Briscoe, A. Jacquet, C. D. Cairano-Gilfedder, A. Salvatori, A. Soppera, and M. Koyabe, "Policing Congestion Response in an Internetwork Using Re-Feedback," *ACM SIGCOMM CCR*, vol. 35, pp. 277–288, Aug 2005.

[48] M. Menth, B. Briscoe, and T. Tsou, "Precongestion Notification: New QoS Support for Differentiated Services IP Networks," *IEEE Communications Magazine*, vol. 50, no. 3, pp. 94–103, 2012.

[49] S. Bensley, D. Thaler, P. Balasubramanian, L. Eggert, and G. Judd, "Datacenter TCP (DCTCP): TCP Congestion Control for Datacenters." Internet Draft, https://tools.ietf.org/html/draft-ietf-tcpm-dctcp, Mar 2017.

[50] B. Briscoe, K. D. Schepper, and M. B. Braun, "Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Architecture." Internet Draft, https://tools.ietf.org/html/draft-briscoe-tsvwg-l4s-arch, Mar 2017.

[51] M. Kuehlewind, R. Scheffenegger, and B. Briscoe, "Problem Statement and Requirements for Increased Accuracy in Explicit Congestion Notification (ECN) Feedback." RFC 7560 (Informational), Aug 2015.

[52] O. Bondarenko, K. De Schepper, I.-J. Tsang, B. Briscoe, A. Petlund, and C. Griwodz, "Ultra-low Delay for All: Live Experience, Live Analysis," in *ACM MMSys*, (Klagenfurt, Austria), May 2016.

[53] K. De Schepper, B. Briscoe, O. Bondarenko, and I. Tsang, "DualQ Coupled AQM for Low Latency, Low Loss and Scalable Throughput." Internet Draft, https://tools.ietf.org/html/draft-briscoe-tsvwg-aqm-dualq-coupled, Oct 2016.

[54] D. Leith and R. Shorten, "H-TCP: TCP for High-speed and Long-distance Networks," in *PFLDnet*, (Argonne, Illinois, USA), Feb 2004.

[55] C. Casetti, M. Gerla, S. Mascolo, M. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," in *ACM MOBICOM*, (Rome, Italy), Jul 2001.

[56] S. Floyd, "HighSpeed TCP for Large Congestion Windows." RFC 3649 (Experimental), Dec 2003.