Fighting the bufferbloat: on the coexistence of AQM and low priority congestion control

Y.Gong[a], D.Rossi[a], C.Testa[a,1], S.Valenti[a,2], M.D.Täht

[a] *Télécom ParisTech, 23 Avenue d'Italie, 75013 Paris, France*

Abstract

Nowadays, due to excessive queuing, delays on the Internet can grow longer than the round trip time between the Moon and the Earth – for which the "bufferbloat" term was recently coined. Some point to active queue management (AQM) as the solution. Others propose end-to-end low-priority congestion control techniques (LPCC). Under both approaches, promising advances have been made in recent times: notable examples are CoDel for AQM, and LEDBAT for LPCC.

In this paper, we warn of a potentially fateful interaction when AQM and LPCC techniques are combined: namely, AQM resets the relative level of priority between best-effort and low-priority congestion control protocols. We validate the generality of our findings by an extended set of experiments with packet-level ns2 simulation, considering 5 AQM techniques and 3 LPCC protocols, and carry on a thorough sensitivity analysis varying several parameters of the networking scenario. We complete the simulation via an experimental campaign conducted on both controlled testbeds and on the Internet, confirming the reprioritization issue to hold in the real world at least under all combination of AQM policies and LPCC protocols available in the Linux kernel. To promote cross-comparison, we make our scripts and dataset available to the research community.

Keywords: Bufferbloat, AQM, Scavenger protocol, Simulation, Experiments

1. Introduction

Internet delays, especially in the uplink direction, "are now as common as they are maddening" [1]. The root cause for these delays can be identified with excess buffering also known as "bufferbloat" inside a network. Although this is nothing new [2], the situation has deteriorated in recent years due to mainly two facts: (i) TCP's loss-based design forces the bottleneck buffer to fill before the sender reduces its rate and (ii) the availability, due to Moore's law, of relatively large memories in front of low-capacity ADSL and cable modems translate into multiple seconds worth of queuing delay [3].

Some point to *local active queue management (AQM)* techniques (i.e., affect the scheduling and discard packets in the buffer differently from a traditional FIFO discipline) as the ultimate solution to reduce queuing delay. Others prefer another direction: the engineering of *end-to-end flow and congestion control (CC)* alternatives to best-effort TCP, and specifically aiming at lower than best-effort priority. Irrespectively of pro-AQM [4] or pro-CC [5] positions, in this work we instead focus on the coexistence of best-effort TCP CC and Low Priority CC (LPCC) transiting a bottleneck link governed by AQM. In the rest of this section we explain the reasons supporting the relevance and timeliness of this goal.

Active queue management is not a new research field, with numerous techniques proposed over the years such as RED [6], SFQ [7], DRR [8], CHOKe [9], SQF [10] and, very recently, CoDel [11]. Yet, despite numerous AQM proposals, they have so far encountered limited adoption. The difficulties in tuning RED [12] are well known, and the computational cost of Fair Queuing was, back in the 90s, considered to be prohibitive (see [1] for an historical perspective). The situation has however started to change, with operators worldwide implementing AQM policies in the upstream of the ADSL modem (e.g., in France, Free implements SFQ since 2005 [13], and Orange starts to deploy SQF [10]) to improve the quality of user experience.

Congestion and flow control may have different goals, such as controlling the streaming rate over TCP connections as done by YouTube or Netflix, or aggressively protecting user QoE as done by Skype over UDP, or to provide *low-priority* bulk transfers service toward the Cloud (e.g., Picasa background upload or Microsoft Background Intelligent Transfer Service BITS, etc.). Similarly, research in the CC domain has produced many proposals for low-priority (or background) transfers, such as NICE [14], TCP-LP [15] or, more recently, LEDBAT [16]. In terms of adoption, while TCP-LP and NICE have been around in the

Email addresses: yixi.gong@enst.fr (Y.Gong), dario.rossi@enst.fr (D.Rossi), claudio.testa@orange.com (C.Testa), silvio.valenti@gmail.com (S.Valenti), dave.taht@gmail.com (M.D.Täht)

[1] Current affiliation: Orange.

[2] Current affiliation: Google Inc.

Linux kernel for about a decade, they have seldom been used[3]. However, ignited by the ease of application-layer deployment, scavenging CC services are now becoming popular: examples of this trend are represented by Picasa's background upload option and the adoption of a LPCC by BitTorrent. Indeed, BitTorrent recently abandoned TCP in favor of LEDBAT, a "low extra delay background transport" protocol implemented at the application-layer over a UDP framing[4]. According to Brahm Cohen, and as confirmed by our measurement [18], LEDBAT is now "the bulk of all BitTorrent traffic, [...] most consumer ISPs have seen the majority of their upload traffic switch to a UDP-based protocol" [19]. Concerning the penetration of BitTorrent traffic, while downlink traffic is nowadays dominated by video streaming, BitTorrent remains the top-1 contributor in the uplink direction. As pointed out by a recent report [20] from the Canadian broadband management company Sandvine, BitTorrent is the top-1 application on uplink traffic, and can be credited for over one third of all upload traffic in North America, Latin America and Asia Pacific. Additionally, BitTorrent represent not less than 10% of the aggregated uplink and downlink traffic, and is still the top-1 application in terms of aggregated traffic volume in the Asia Pacific region. Finally, as median Internet traffic increases, so does the overall BitTorrent traffic.

From the above observation, we gather an increasing adoption trend of both AQM techniques and LPCC protocols, which already coexist in the current Internet. Yet, studies have so far focused on AQM or LPCC, and only seldom jointly considered these two aspects. As such, interactions between AQM and LPCC is, at this stage, poorly understood. In this paper, we show a potentially fateful interplay between AQM and LPCC: namely, AQM resets the relative level of priority between best-effort and low-priority congestion control protocols. Intuitively, this arises from the fact that one of the typical design goals of AQM is to enforce fairness among flows, to penalize the most aggressive heavy-hitter flows and to protect the newly starting and short-lived ones. This is in sharp contrast with LPCC design goal, which instead aims at utilizing the excess capacity, without however interfering with standard TCP transfers, fact that AQM inhibits by penalizing TCP more than LPCC. As this interplay resets the relative level of priority among CCs, we refer to this issue simply as "reprioritization" in the following.

The remainder of this paper is organized as follows. First, Sec. 2 overviews closest related work. Then, by means of `ns2` packet-level simulation, Sec. 3 illustrates the problem, showing that the phenomenon is fairly general and holds under any combination of AQM techniques and LPCC protocols. To simplify and promote cross-comparison, we make the set of our `ns2` scripts available to the scientific community at [21]. Additionally, a thorough sensitivity analysis in Sec. 4, considering network parameter variation such as buffer size, link capacity, flow duty cycle and RTT delay, confirms that the reprioritization holds under any tested network scenario as well. Furthermore, via testbed and Internet experiments, Sec. 5 testifies the phenomenon to hold in the real world as well, though we limit the validation to a representative subset of the larger design space explored by the simulation. The set of scripts we used in the experimental approach is again available at [21]. Finally, a discussion of the root causes and feasible solution is covered in Sec. 6, while Sec. 7 summarizes our findings.

2. Related work

It would be extremely cumbersome to retrace over 20 years of Internet research in these few pages (we refer the reader to [1], where without providing a complete picture, it does however present a historical viewpoint). We extend this viewpoint by reporting in Fig. 1 a timeline of the AQM and LPCC algorithms used in this paper. The timeline clearly shows a temporal separation of the two research topics, which in our opinion helps understand why the AQM vs. LPCC interaction assessed in this paper was not previously exposed. It is also out of the scope of this paper to provide an overview of AQM and CC research, for which we point the reader to the sources appeared in this timeline.

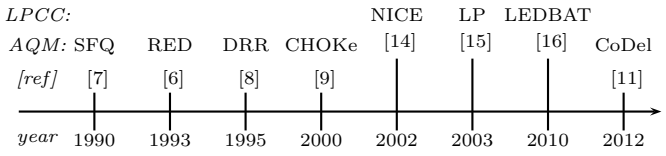

Figure 1: Timeline of AQM and LPCC algorithms.

2.1. Separated AQM vs. LPCC studies

Studies on scheduling and active queue management were very popular during the 90s (e.g., SFQ [7], RED [6], DRR [8]), still active at the beginning of the early 2000s (e.g., CHOKe [9]) and declined since then. Yet, we currently see a resurgence of the topic, in terms of novel proposals (e.g., CoDel [11], AFpFT [22]) and further research [23, 24, 25], as also testified by the very recent proposal to create a dedicated IETF AQM WG [26].

Studies on low-priority congestion control protocols started in early 2000s, with several contributions such as NICE [14], LP [15], 4CP [27, 28] and LEDBAT [16]. While it is out of the scope of this paper to provide a full overview of the above protocols, we refer the reader to [29] for a more thorough survey, and to [30] for a simulation-based comparison

[3] In recent kernels, NICE is no longer available as a kernel module, whereas LP it available but not among the TCP flavors allowed by default via `net.ipv4.tcp_allowed_congestion_control`

[4] The protocol is usually referred to either as LEDBAT (in the IETF community [16]) or as uTP (in the BitTorrent BEP [17] community). To avoid ambiguity, in this paper we employ its IETF name.

of NICE, LP and LEDBAT, showing that LEDBAT has the lowest level of priority among all LPCC protocols.

In terms of AQM vs. LPCC trend, it is worth to highlight an interesting similarity between the most recent approaches of either class (i.e., CoDel and LEDBAT), as both explicitly control queuing delay: they both employ a *target delay* parameter upon which dropping decisions or congestion window modifications are based respectively.

2.2. Joint AQM and LPCC studies

Our main focus concerns fairness of the capacity share among heterogeneous control protocols (TCP and LPCC) on a bottleneck governed by AQM. Concerning this aspect, closest work to ours is represented by [25, 31, 32].

Yet, we point out that generally researchers are trying to ensure fairness between different heterogeneous TCP versions, as done via simulation in [25], which only considers high-priority TCP variants where fairness is desirable, whereas in our settings *unfairness would be desirable*, as it would imply that low-priority property of LPCC is maintained.

While not the authors' main focus, the interplay of AQM and LEDBAT is exposed via an experimental approach in [31]: in one of the tests, authors experiment with a home gateway that implement some (non-specified) AQM policy other than DropTail. When LEDBAT and TCP are both marked in the same "background class" the "TCP upstream traffic achieves a higher throughput than the LEDBAT flows but significantly lower than" that under DropTail [31]. This fact is also recognized by the LEDBAT RFC, which states that under AQM it is possible that "LEDBAT reverts to standard TCP behavior, rather than yield to other TCP flows" [16].

Finally, in [32], we analyse the AQM vs. LPCC reprioritization issue via a fluid model, describing system dynamics of heterogeneous congestion control protocols (namely, TCP and LEDBAT) competing on a bottleneck link governed by AQM (namely, RED). Thus, [32] limitedly considers a single pair of LPCC and AQM instances – unlike in this work.

Hence, the interplay of AQM and LPCC has been anecdoctically covered, though a broad and deep study is missing so far. This is precisely our main goal: building on our previous work [33], this work presents the first systematic study of the interaction between LPCC and TCP flows under the presence of AQM. The present work extends [33] by carrying an extensive sensitivity analysis, and by employing a mixed simulative and experimental methodology to further extend the validity and generality of our results. To facilitate cross-comparison and independent validation of our results, we make our scripts and datasets available to the research community at [21].

3. Reprioritization

The aim of our `ns2` simulation campaign is to test the validity of the reprioritization phenomenon under the largest possible set of scenarios. Instead of thoroughly listing in this paper all details and low level configurations of the `ns2` environment, we directly make our scripts available[5] at [21]. We include a number of representative AQM techniques among the previously listed ones: namely, SFQ [7], RED [6], DRR [8], CHOKe [9] and CoDel [11]. Similarly, we consider a number of representative LPCC techniques such as NICE [14], TCP-LP [15] and LEDBAT [16]. As for standard best-effort TCP, we consider the IETF NewReno variant[6]. Notice that some of these modules are not directly available in `ns2` (version 2.33), therefore we patch it to support CHOKe [9], CoDel [11], LEDBAT and NICE (the last two LPCCs use our own open source implementations [36]).

A couple of points are worth stressing. Although we are aware of the fallacies of RED (as are the authors of CoDel [11]), we believe that it needs to be considered as a reference benchmark (to which indeed CoDel is compared in [11]). Additionally, note that RED is one of the few AQM policies available on common recent Linux kernels shipped with standard distributions, we believe it would not make sense to exclude it from this study even due to its known performance issues [37, 12]. Similarly, we are aware of the latecomer unfairness issues of LEDBAT [38, 39], but its widespread use makes it necessary to be considered in the mix.

Throughout this paper, we express system performance mainly in terms of the link utilization η, the share of the link exploited by the TCP aggregate $TCP\%$, and the average queue length $E[Q]$. For convenience, we can either express $E[Q]$ in number of packets (possibly normalized over the buffer size $E[Q]/Q_{max}$ to gauge the bufferbloat intensity), or as the actual packet sojourn time in the queue (a more direct measure of the user quality of experience).

In what follows, we aim at conveying the most relevant message we gather from simulation in a straightforward way. We therefore incrementally extend the breadth of our results by initially considering an illustrated example of the reprioritization phenomenon. We then extend our horizon by observing the impact of AQM policies, and finally the joint impact of AQM and LPCC.

3.1. A motivating example

As previously outlined, while the underlying ideas and knobs that AQM and LPCC can exploit are similar (e.g., controlling delay under LEDBAT and CoDel), their interplay can have negative consequence: we find *that AQM can induce a reprioritization of CC*. In other words, current

[5] Due to known difficulties in reproducing research results [34], particularly with RED [35], we invite the reader to use these scripts instead of trying to closely mirror our scenarios.

[6] We point out that, in recent times both Linux and Windows have drifted from IETF recommendation, selecting Cubic and Compound as default TCP flavors respectively. Yet, as both Cubic and Compound are designed to be more aggressive than NewReno, we henceforth expect reprioritization to hold for these flavors as well.

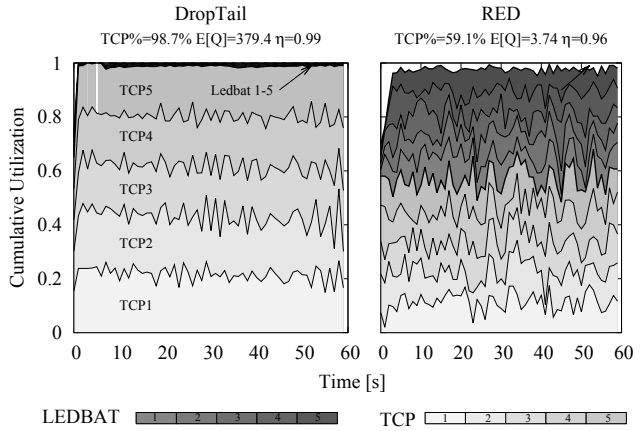

Figure 2: Illustrated example of problems arising from the interaction of AQM and CC techniques (ns2 simulation for AQM=RED, CC=TCP+LEDBAT).

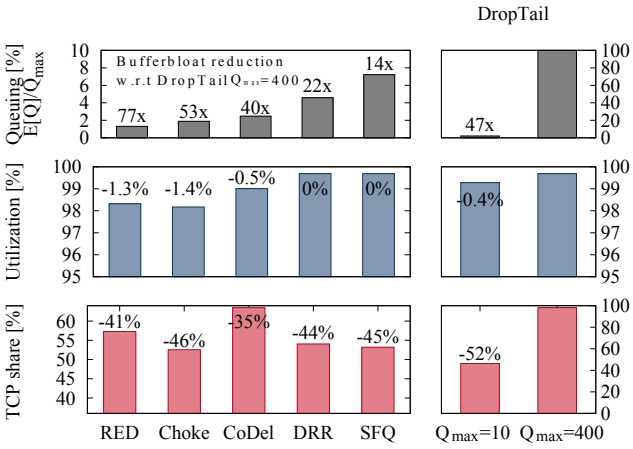

Figure 3: Impact of AQM policies on the bufferbloat intensity $E[Q]/Q_{max}$ (top) link utilization η (middle) and TCP% breakdown (bottom).

scavenging protocols can successfully realize a lower-than-best-effort priority only if the bottleneck buffer operates according a DropTail discipline.

Fig. 2 illustrates the phenomenon: we stress that, while the picture depicts simulation results gathered in a very specific case, the remainder of this paper will show the phenomenon to hold under a large range of AQM+LPCC scenarios (using both real-world experiments and `ns2` simulations). The picture shows a breakdown of the link utilization when 5 TCP NewReno (each of which is represented with a different shade of light-gray) and 5 LEDBAT (dark-gray) backlogged flows sharing the same bottleneck. Capacity is set to 10 Mbps and the buffer has room for 500 packets (600 ms worth of delay); for the sake of simplicity, delays are homogeneous across flows. The left plot reports the case of a DropTail queuing discipline, while the right one reports the case of RED. The plots are annotated with further statistics concerning the average queue size in packets $E[Q]$, the capacity share exploited by the aggregate TCP%, and the average link utilization η.

In the DropTail case, the LEDBAT protocol operates in a lower-than-best-effort mode: we see that this delay-based scavenging protocol successfully exploits the spare capacity left unused by NewReno (as shown in [38]). The TCP aggregate uses the bulk of the capacity (TCP%=99%), with a fair share among TCP flows (due to homogeneous delay). However, the queuing delay approaches half a second because nearly 400 full-size TCP packets are queued on average. Clearly, bufferbloat would be even worse for lower (e.g., ADSL-like) capacities, or larger buffer sizes (common defaults for home gateways are well in excess of 1000 packets).

In the RED case, while it is successful in limiting the queue size (less than 4 packets on average), this comes at the cost of (i) a slight 3% reduction of the link utilization, (ii) a complete reset of the relative level of priority between flows. In the case depicted in the figure, the share is now fair among all LEDBAT and NewReno flows, so that LEDBAT operates in a best-effort mode, and is thus as aggressive as TCP. *While an AQM fixes the bufferbloat, it destroyes the relative priority among CC protocols.*

While the interaction between LEDBAT and AQM is pointed out in [31] and mentioned by the draft [16], we believe both the extent and depth of the problem to be underestimated. As for the extent, Sec. 4 summarizes results of over 3,000 simulations, showing that the phenomenon just illustrated *is a fairly general problem, arising from the interaction of AQM and any LPCC protocol.* As for the depth, Sec. 5 not only confirms the phenomenon to hold in the real world via Internet experiments on multiple AQM and LPCC techniques, but also shows that the *relative level of priority seldom reverses under AQM.*

3.2. Impact of AQM policy

For simplicity, in this section, we consider an equal number $N = 5$ of best-effort TCP NewReno and low-priority flows sharing a link having capacity $C = 4$ Mbps and a buffer size $Q_{max} = 400$packets (corresponding to a maximum bufferbloat of 1.2 seconds). All flows are backlogged, start at time $t = 0$, and have a homogeneous delay $RTT = 50$ ms. Simulations last for 60s, and 10 runs are repeated for each parameter settings. A larger number of settings is reported in Sec. 4.

For the time being, we fix the LPCC to LEDBAT, and examine the impact of different AQM techniques. Fig. 3 reports the bufferbloat intensity $E[Q]/Q_{max}$ (top), link utilization η (middle) and TCP% breakdown (bottom) for varying AQM policies, and for DropTail as a comparison (right). As expected, at the price of a slight decrease in the link utilization, AQM reduces the intensity of the bufferbloat: compared to a persistently full DropTail buffer, queue size and delay is from 14 to 77 times smaller, using SFQ and RED respectively. This implies that, under AQM the queuing delay is always less than 85 ms (SFQ,

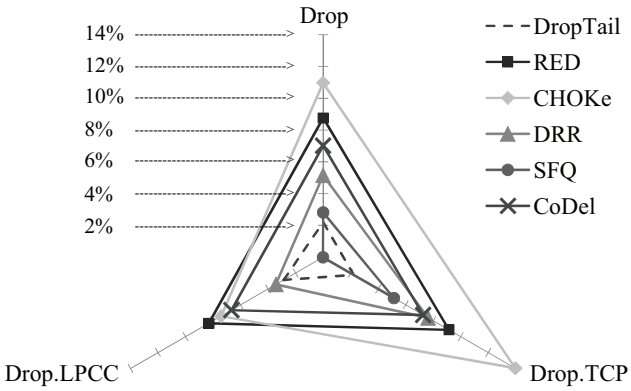

Figure 4: Impact of AQM policies on the drop rate.

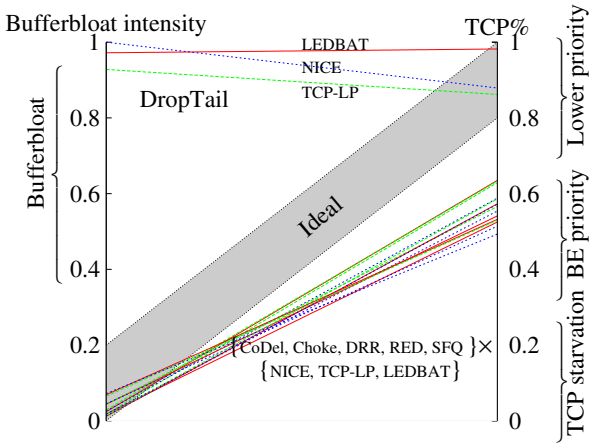

Figure 5: Joint impact of AQM and LPCC on the queuing delay and TCP breakdown.

worst case[7]), and generally much lower. However, we see that the capacity share of the TCP aggregate can drastically reduce 35%-46% (under CoDel and CHOKe respectively). Notice that this result alone extends the validity of the phenomenon observed under unknown[8] AQM policies in [31]. Importantly, we point out that simply reducing the DropTail buffer size is not a solution to the problem either: notice that (i) as shown in Fig. 3, shorter DropTail buffer behaves like AQM, as bufferbloat is reduced at the expense of reprioritization and (ii) in case of varying link bandwidth such as WiFi (where rates can vary from few to several tens of Mbps), there is no fixed buffer size that would not translate into bufferbloat. Intrinsically, when the queue length is short, either due to AQM or tiny buffers, LPCC cannot reliably infer congestion signals from delay measurement.

Finally, Fig. 4 shows a Kiviat chart of the impact of AQM policies on drop rate of different flow types, considering average drop rate (top axis), TCP drop rate (bottom right axis) and LPCC drops (bottom left axis). Compared to DropTail, all AQMs increase drop rate of all flows (due to early drop decisions) but not enough to influence the link utilization. Interestingly, DropTail penalizes TCP as much as LPCC, whereas AQMs generally penalizes TCP more than LPCC – the root cause of reprioritization. Drop rates for TCP are worst in the case of CHOKe, which is designed to leverage power of two choices to especially penalize heavy-hitter flows. In case of SFQ scheduling, notice that LPCC almost experiences no losses.

Summarizing, reprioritization of LEDBAT holds under any AQM. Additionally, the use of tiny buffers is not helpful in terms of reinstating priorities between LPCC and best-effort TCP.

[7]It has to be noted that, strictly speaking, SFQ is a scheduling discipline and not an AQM mechanism: as such, longer delay is expected since no "early" drops occur.

[8]Authors just mention that "different prioritized traffic classes" are implemented in the "modern home gateways" used in their experiments.

3.3. Impact of AQM policy and LPCC protocol

We now explore the full product of LPCC flavors and AQM techniques. Under DropTail, it is known [30] that LEDBAT achieves the lowest priority against best-effort TCP, followed by NICE and TCP-LP. We illustrate results as a *parallel coordinate* plot in Fig. 5. Having noticed that link utilization is subject to small variations, we consider the two main metrics of interest: namely the queuing delay and the TCP share. For convenience, we normalize the bufferbloat over the queue size and report its normalized intensity $E[Q]/Q_{max} \in [0,1]$ on the left y-axis, and report the normalized TCP share of the link capacity $\frac{TCP}{TCP+LPCC} \in [0,1]$ on the right y-axis. In the parallel coordinate plot, each (LPCC, AQM) pair is represented as a line, joining the delay and TCP share figures. For reference purposes, we put a light-gray strip representing the ideal case where the queue is short but priorities are unaltered.

We see that the three (LPCC,DropTail) combinations appear as horizontal lines on the top of the figure: this happens since, under DropTail, bufferbloat has maximal intensity while TCP monopolizes the bottleneck. We see that this holds for any LPCC protocol, with a slight separation of the curves reflecting the order of priority observed in [30] (from top to bottom, the least to the most aggressive LPCC with respect to TCP).

Excluding DropTail, we also see that all (LPCC,AQM) combinations are very close, as AQM implies low bufferbloat but jeopardizes CC priorities. Specifically, as under AQM all TCP and LPCC flows have roughly the same priority, the lines fall in the best-effort (BE) priority range, below the ideal strip, with best-effort TCP getting slightly more than half of the link share.

Summarizing, reprioritization holds under any considered LPCC and AQM. The choice of a specific (LPCC,AQM) combination has only very limited impact on the system performance – and, in any case, is not helpful in reinstating priorities between LPCC and best-effort TCP.

Table 1: Sensitivity analysis parameters

Variable parameters

Parameter	*Default*	*Range*	*Sec.*
Qmax (pkt)	400	100, 200, 300, 400	4.1
C (Mbps)	4	0.25,0.5,1,2,4,6,8,10	4.1
Workload (%)	100	10 - 100 (interval 10)	4.2
RTT (ms)	6	2, 4, 6, 8, 10	4.3

Fixed parameters

Parameter	*Value*
AQM	DropTail, RED, CHOKe, DRR, SFQ, CoDel
LPCC	LEDBAT
No.TCP flows	5
No.LPCC flows	5
Duration (s)	60
No.Runs	10

4. Sensitivity analysis

To further extend the breadth of our findings, we conduct over 3,000 simulations to investigate the impact of other network parameters such as buffer size Q_{max}, link capacity C, heterogeneous RTT delay, and flow duty cycle. We first discuss each of the above factors separately, and finally compactly report their impact. We anticipate that under all explored circumstances[9] the early outlined phenomenon remains valid.

The parameters used in our analysis are listed in Tab. 1 (as parameters listed in the table are by no means exhaustive, we invite the reader to use the provided scripts [21] shall they reproduce these experiments). We let 10 flows (5 TCP + 5 LEDBAT) competing at the bottleneck at the same time for 60s, results are obtained from the average of 10 runs for each simulation. We will explain the varied parameters investigated in detail in its corresponding sub-section.

4.1. Buffer size and capacity

We explore over 30 combinations of link capacity varying in the 250Kbps to 10Mbps range, and buffer size between 100 and 400 packets (detailed values can be found in Tab. 1).

First, when fixing the value of buffer size, we observe a decrease of packet drop rate as the link capacity increases, while the link utilization is mostly unaffected by capacity variations. This is expected since TCP ramps up to exploit the full capacity both in case of DropTail or AQM.

If we instead fix the capacity, then the impact of the buffer size is limited in case of AQM. This is again expected, since AQM is configured to keep the buffer short irrespectively of its size: in other words, the average queue size is $E[Q] \approx Q_{max}$ under DropTail for any Q_{max}, whereas $E[Q] < Q_{max}$ under any AQM technique irrespectively of Q_{max}.

[9] We avoid reporting some of which (such as scenarios with a larger number of flows, or dynamic flow arrivals patterns, etc.) in full details, to privilege clarity over completeness.

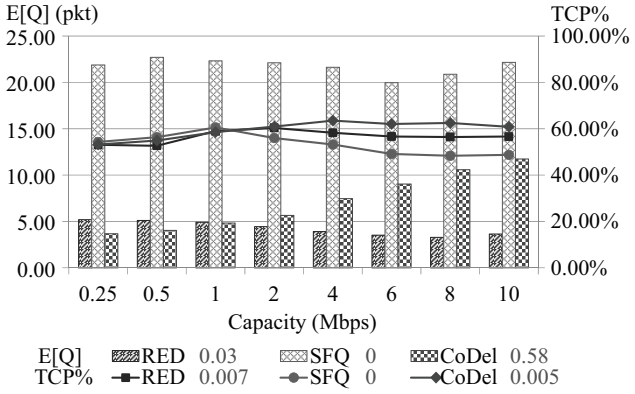

Figure 6: Sensitivity analysis: average buffer occupancy and TCP% share for varying capacity (x-axis) and buffer sizes (standard deviation across buffer size is reported in the legend).

We elaborate more on capacity variations in Fig. 6, which reports the average queue size $E[Q]$ (in packets, left y-axis) and TCP% breakdown (right y-axis) as a function of the link capacity (x-axis) for some AQM mechanisms (namely we limitedly report RED, SFQ and CoDel to avoid cluttering the picture). The picture reports the average of the metrics over all buffer sizes, and the legend is further annotated with the largest standard deviation across buffer size (i.e., for any given capacity, we evaluate the standard deviation of the metrics of interest across different buffer sizes, and report the largest across all capacities). As the standard deviation is very low, this confirms reprioritization results to hold across the explored scenarios.

As expected, capacity does not have impact on average queue size $E[Q]$ except for CoDel, which controls the packet-sojourn time: as a result, larger capacities translate into a larger number of packets to be allowed in the buffer under the same target delay configuration (hence, a larger standard deviation). The TCP% breakdown is also unaffected under any capacity and buffer size combination tested, confirming TCP% manages to maintain its priority level over LPCC only under DropTail.

4.2. Flow duty cycle

We next observe that networks are rarely used by backlogged flows: hence, we include duty cycle model simulating different workloads to make our simulation more realistic. We engineer the scenario to simulate different workloads,by controlling the individual flow duty cycle (from 10% to 100% with a 10% step). For each flow, we turn it ON and OFF alternatively during the entire simulation duration. During each ON period, the flow continuously transmits data, and during an OFF period, the flow remains idle and resets its window parameters. At the end

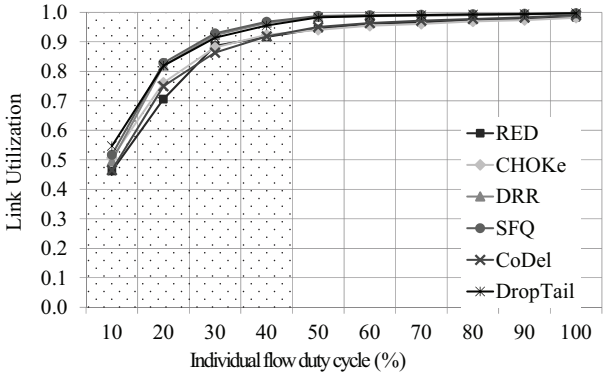

Figure 7: Sensitivity analysis: Impact of flow duty cycle on the link utilization.

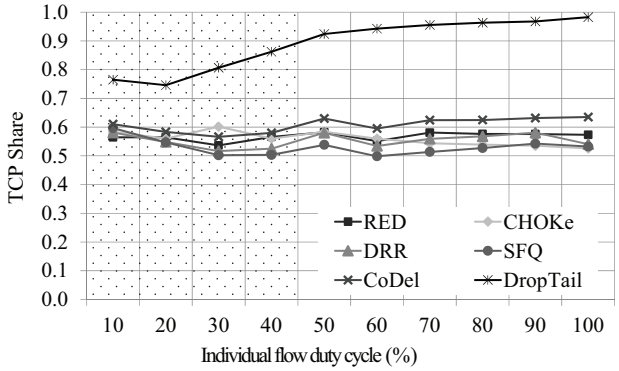

Figure 8: Sensitivity analysis: Impact of flow duty cycle on the TCP breakdown.

of each OFF period (including the initial state), we randomly generate the duration of next ON period, which follows an exponential distribution with average λ. Also, the same procedure is employed at the end of each ON period, generating the duration of next OFF period with average duration μ. Thus, the individual flow duty cycle is defined as $\lambda/(\lambda+\mu)$.

By design, the duty cycle model affects link utilization, so that we can individuate two regimes. Regimes are illustrated as two colored regions in Fig. 7, which reports the average link utilization as a function of the individual flow duty cycle. When individual duty cycle is low, the aggregated load remains, on average, well below the link capacity. Also, as flows are often idle, they will operate in isolation, only seldom interact. Still, due to TCP bursty nature, the buffer can occasionally be filled with burst of packets belonging to the same congestion window, possibly triggering AQM reactions. In the underload regime, we therefore expect each flow to get a fair share of the link. As individual flow duty cycle increases, and the aggregated load approaches the link capacity, packets of different flows now mix in the buffer, and we expect AQM decisions to penalize the more aggressive flows – which as we previously observed will induce reprioritization. Also in the overload regime, we therefore expect each flow to get a fair share of the link.

As shown in Fig. 8, the expected behavior holds. In both the underload zone (shaded left part of Fig. 7 and Fig. 8) and overload region (white right part), TCP and LPCC always fairly share the link. In case only one flow is active at a time, it will monopolize the bottleneck in both TCP and LPCC cases. When two (or more) flows are active at the same time, in case one of these flows is best-effort TCP, then its AIMD dynamics will let the queue increase and trigger AQM reaction, to the advantage of LPCC flows.

Extending results of Fig. 4, we report per-protocol drop rate under different AQMs and workloads in the scatter plot of Fig. 9. The diagonal line corresponds to an equivalent drop rate of packets of both TCP and LPCC: as it can

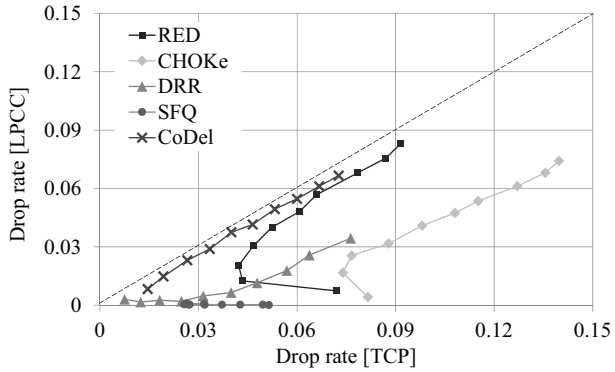

Figure 9: Sensitivity analysis: Impact of flow duty cycle on drop rate.

clearly be seen, under all considered duty cycle conditions TCP is more heavily penalized with respect to LPCC. Note that the line of CoDel is very close to the diagonal, which means its penalization of TCP is the slightest while SFQ has the heaviest penalization. Additionally, we see that fairness in the loss rate translates in higher TCP share: notice indeed that the loss behavior is reflected in Fig. 8, showing that CoDel and SFQ grant the highest and lowest TCP% breakdown respectively.

4.3. RTT Delay Heterogeneity

Finally, we acknowledge that rarely flows have homogeneous delay in the real life. Yet, as it is known that TCP exhibits RTT unfairness, it makes sense to carefully build the scenario so to avoid RTT unfairness biasing our conclusions. First, we observe that congestion control protocol in use end-to-end (i.e., TCP vs. LPCC) is not correlated with the RTT, as this is an end-host decision: it follows that TCP and LPCC aggregates have no a priori reason to have different RTT distributions. Second, we observe that as the queuing delay is low due to AQM, the propagation delay is the dominant component of the RTT, which controls the rate at which acknowledgements are received, which in turn controls how fast the sending

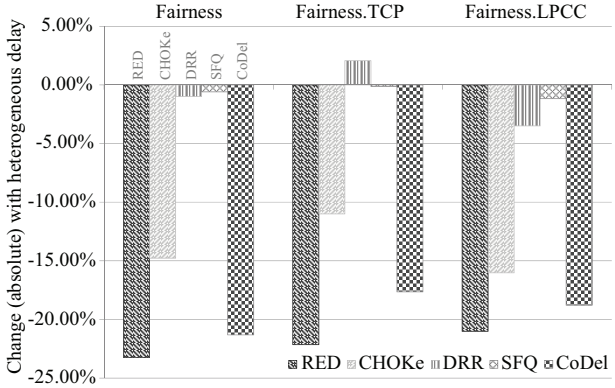

Figure 10: Sensitivity analysis: Impact of heterogeneous RTT delay on inter/intra-protocol fairness.

rate can change: it follows that the relative difference of RTT between heterogeneous flows, more than the absolute RTT value, is important to gauge response to RTT heterogeneity.

As such, indicating by abuse of language with TCP and LEDBAT the set of flows using that congestion control protocol, we engineer the scenario so that (i) both the TCP and the LPCC sets have the same $E[RTT]$, (ii) the $E[RTT]$ of both sets is the same as in the homogeneous case, (iii) within one set, there are no flows having the same RTT delay, (iv) across two sets, there are one TCP and one LEDBAT flows having exactly the same RTT delay. Specifically, we assign $RTT_k = 2k$ ms for the k-th flows (with $k \in [1, 5]$) in the TCP and LEDBAT sets (so that $E[RTT] = 6ms$ is the same as the default setting).

Unsurprisingly, we verify that RTT heterogeneity does not affect our conclusion in terms of reprioritization. While RTT unfairness affects the performance of individual flows within an aggregate, we see that each aggregate as an entirety can fairly compete with each other on the RTT (i.e., there is a balanced mix of opportunistic flows with small RTT and penalized flows with large RTT in both sets). We resort to the classical Jain fairness index $F_X = (\sum_{i=1}^{N} T_i)^2/(N \sum_{i=1}^{N} T_i^2)$, with T_i throughput of the i-th connection, is calculated within homogeneous TCP and LPCC aggregate (intra-protocol fairness), as well as over the total set of flows (inter-protocol fairness). We visualize the impact of RTT heterogeneity in Fig. 10, where we report the relative error in terms of fairness with respect to the homogeneous scenario.

Fig. 10 shows that RTT heterogeneity decreases both inter-protocol and intra-protocol fairness up to 20%. Though different AQMs have rather noticeable different impact on fairness decrease, each of them influences TCP and LPCC aggregate in a similar way.

It is more interesting to observe the intra-protocol fairness, i.e., with respect to a set of flows using homogeneous congestion control protocol, which may change significantly depending on the AQM policy. We find that intra-protocol fairnesses of TCP (F_{TCP}) and LPCC (F_{LPCC}) are quite close with respect to each AQM. To rule out the impact of congestion control protocols heterogeneity, we measure the average Jain fairness index $F = (F_{TCP} + F_{LPCC})/2$ as an indicator of average intra-protocol fairness[10]. We have that fairness under CoDel ($F = 0.81$) is only slightly better than RED ($F = 0.78$) and significantly smaller than SFQ ($F = 0.99$). Notice that CoDel fairness range is coherent with [11], which however does not address a comparison with SFQ and reports significantly smaller fairness values for RED (under a more heterogeneous scenario).

From the result, we can conclude that the extent up to which heterogeneous RTT delays can affect inter-protocol and intra-protocol fairness, is closely related to AQM drop decision-making mechanism. Due to the fact that AQM like RED and CHOKe work on the entire buffer status, it follows that with heterogeneous delay, packets will be dropped at more varied intervals, resulting in a lower fairness both *intra-protocol* and *inter-protocol*. For instance, CHOKe uses sampling to find heavy hitters, dropping both packets if the sampled packet and the newly incoming one both belong to the same flow (whereas RED would only drops the new packet): hence, this design choice increases the fairness among flows (with respect to RED) even when heavy hitters are flows opportunistically exploiting RTT heterogeneity. However, scheduling decisions of the DRR and SFQ policies affect individual flow (or stochastic flow aggregates), thus they can always ensure almost perfect fairness under these two scenarios.

Table 2: Sensitivity analysis

Coefficient of variation (CoV)		η	$E[Q]$	$TCP\%$
AQM		0.0073	**0.6916**	0.080
LPCC	DropTail	0	0.038	0.072
	AQM	0.0047	0.034	0.048
Link capactiy	DropTail	0.0003	**1.136**	0.035
	AQM	0.0042	**0.604**	0.050
Flow duty	DropTail	0.259	**1.309**	0.120
cycle	AQM	0.313	**1.067**	0.057
RTT delay	DropTail	0.0003	0.022	0.001
heterogeneity	AQM	0.0248	0.052	0.048

4.4. Summary

Tab. 2 summarizes the results of the sensitivity analysis reporting the coefficient of variation $CoV(X) = \sigma(X)/E[X]$ of the metric X of interest (i.e., link efficiency η, average queue size in packets $E[Q]$ and the TCP aggregate $TCP\%$) due to each of the studied parameters (i.e., link capacity C, buffer size Q, flow duty cycle, RTT delay, etc.). Notice that metrics of interest all have different units and value

[10] Notice that $\frac{1}{2}F_{TCP} + \frac{1}{2}F_{LPCC}$ differs from evaluating the fairness over the whole aggregate $F_{\mathcal{W}} = TCP \cup LPCC$, as in the latter case we would be measuring the inter-protocol effect as well.

ranges: CoV describes the dispersion in a way that does not depend neither on the metric unit, nor on its scale. Hence, this choice allows a relative comparison across metrics, and makes CoV especially suitable in our context.

For parameters other than AQM, we include the result calculated both under DropTail and AQM: i.e., to exclude the impact due to AQM (shown in the first row of the table) we compute the $CoV(X)$ considering each AQM policy in isolation, then report the average $E[CoV(X)]$. The result shows that most values of the CoV are small, which confirms that results are only minimally affected by any of the tested parameters, and especially for the TCP%, confirming the generality of the reprioritization phenomenon illustrated in Fig. 5.

We highlight (in bold) CoV value larger than 0.5 in Tab. 2, which are of straightforward interpretation. It can be seen indeed that AQM policy can have noticeable impact on average queue size $E[Q]$ (recall Fig. 3) due to both their inner working mechanism (e.g. RED limits the queue explicitly while SFQ schedules the flows but allow a proportional increase of packets in the queue with increasing flows) as well as the variation of their default parameters (untested in this sensitivity analysis due to known difficulty in tuning AQM [12], and so not accounted for in CoV). Obviously, buffer size influences heavily the average queue size under DropTail (CoV=1.136), whereas flow duty-cycle translates into a wider range of queue sizes under both DropTail (CoV=1.309) and AQM (CoV=1.067), as queues are possibly empty unlike in backlogged workload.

Summarizing, reprioritization holds under most networking scenarios. While the specific scenario settings may have an impact on fairness and queue size statistics, they have nevertheless only very limited impact on the reprioritization phenomenon.

5. Experimental results

The goal of our experimental campaign is to confirm the occurrence of the reprioritization phenomenon in the real world. Configuration details and scripts used for these experiments are available at [21]. Since our aim is not to propose any new AQM or LPCC, nor to replicate the full set of simulations shown early in Sec. 3, we select the only ones that are already available in recent Linux 3.2 kernel: namely, RED [6] and SFQ [7] for AQM and TCP-LP [15] for the LPCC protocol family. As previously pointed out, LEDBAT cannot be excluded from the mix since it is, by far, the most successful LPCC: as such, we employ the `libUTP` [40] application-level implementation of BitTorrent that we already analysed in [41].

Incidentally, while our choice is forced by the availability of AQM and LPCC implementation in Linux, in reason of results of the previous sections we expect the performance of other AQM+LPCC combinations to fall into the boundaries defined by {RED,SFQ}×{TCP-LP,LEDBAT}. On the one hand, this is due to the fact that RED vs. SFQ yield to small vs. large queuing delays respectively; on the other hand, it is known that TCP-LP and LEDBAT have the most and the least aggressive behaviour in the LPCC family [30].

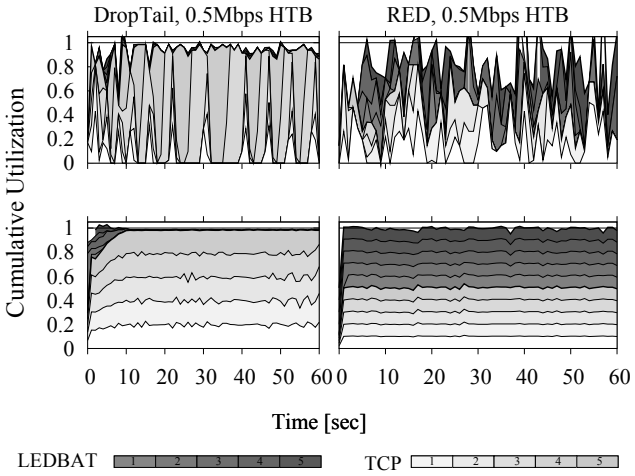

Figure 11: Testbed experiments: Utilization breakdown among TCP and LEDBAT, for different AQM techniques (DropTail, RED, SFQ), capacities (500 Kbps, 10 Mbps) and emulation technique (HTB, PHY).

For any AQM and LPCC combination, we explore some experimental settings, including varying bottleneck capacity C, configuration of AQM parameters, number of flows N in the bottleneck, in both an emulated testbed and in a real Internet deployment.

5.1. Testbed experiments

In the testbed experiments, we directly connect two PCs through a crossover Ethernet cable. Capacity limitations are either emulated through a Hierarchical Token Bucket (HTB) of the standard Linux traffic control `tc` suite (as in Fig. 2), or natively by forcing $C = 10$ Mbps PHY Ethernet through `ethtool` (which is a more reliable option than HTB). In both cases, we turn off most features that could possibly interfere with the experiments (e.g., jumbo frames, TCP segmentation offload, interrupt coalescing). To emulate a WAN setup, we inflate RTT delay by a constant amount equal to 30 ms using `netem`. We configure the `tc` queuing discipline to either DropTail (i.e., the default `pfifo_fast`), RED (with state of the art configuration) or SFQ.

We generate backlogged traffic during 60 seconds experiments, using `iperf` for all TCP flavors (both best-effort and low-priority) and simple client/server applications provided by `libUTP`[11] for the application-layer LEDBAT implementation. For best-effort TCP, we report results using NewReno, the protocol of choice at the IETF,

[11] libUTP is the uTorrent Transport Protocol (uTP) library implementation of LEDBAT, released under the MIT license at [40]. Version d4685a3 (May 2012) was used in all experiments, though we point out that libUTP has been very stable since, with only 4 commits, all of which were not related to congestion control issues.

and leave the study of Compound (default TCP flavor in Windows) and Cubic (default in Linux) for future work[12].

As for queuing delay measurement, we point out that dark buffers may lay at multiple points in the kernel stack (e.g., TCP buffers, device driver), and that buffering may occur outside the host (e.g., in the ADSL modem in Internet experiments). Hence, we opt for a simple methodology that mimics the way in which NICE measures the queuing delay: specifically, we monitor the RTT through a low frequency ICMP ping command and estimate queuing delay samples as $Q_i = \mathrm{RTT}_i - \min_{j \le i} \mathrm{RTT}_j$. Notice that, as the reverse direction is carrying only ACK data and is thus not congested, we are safe in assuming that the RTT variation is due to queuing at the sender side.

A further example of the temporal evolution of the utilization breakdown of TCP vs. LEDBAT flows is depicted in Fig. 11 for different AQM techniques (DropTail, RED, SFQ), capacities (500 Kbps, 10 Mbps) and emulation technique (HTB, PHY). The phenomenon early shown in Fig. 2 is thus confirmed for different AQMs and testbed settings: shortly, AQMs induce reprioritization of heterogeneous CC flows. We also see that, while when the capacity is abundant the breakdown is smooth, the opposite happens when capacity is scarce (which additionally leads to unfairness at short timescales).

We conduct a more systematic experimental testbed campaign that is summarized in Tab. 3, reporting the TCP breakdown and average queuing delay for an emulated HTB capacity of $C = 500$ Kbps. The total number of flows varies in $N \in \{2, 10\}$, with flows equally split in the best-effort TCP and LPCC families. Experiments report the average over 3 independent runs. As we may now expect, DropTail leads to bufferbloat of multiple seconds[13], which is instead solved by both RED and SFQ. The picture may change completely under RED, depending on the LPCC flavor and the number of flows: indeed, while for $N = 10$ the reprioritization happens for both LEDBAT (TCP%=49.3) and TCP-LP (57%), in the case that $N = 2$ flows compete for the same access, the LEDBAT+NewReno combination, through RED, possibly *results in TCP NewReno starvation* (1.7%) while the opposite happens under TCP-LP+NewReno (97.5%).

With respect to reprioritization, which is a general issue holding for a large number of LPCC vs. AQM combinations (as we show in this paper) and AQM settings (as we show in [32]), we believe the starvation phenomenon to be of more limited interest (as it happens for a single AQM, and furthermore non-systematic, thus tied to specific configurations). Yet, the phenomenon is still worth elucidating further, as these differences reflect the LPCC dynamics of LEDBAT and TCP-LP. The latter is AIMD-controlled, and its low priority stems from a slower recovery *after* losses that the AIMD dynamics force. The former is delay-based and PID-controlled: by limiting the queue size, it will seldom be penalized under RED. Whenever the queue grows due to best-effort TCP AIMD, LEDBAT reduces its own window, and so the chances that a packet will get dropped are very low. Whenever TCP experienced a timeout and abruptly shrank its window, LEDBAT instead grows its window again, but in reason of its low target delay, it will limit the amount of queuing and again prevent its packets from being dropped.

Table 3: Testbed: LEDBAT vs. TCP-LP

		5+5 flows		1+1 flows	
		TCP%	E[Q] ms	TCP%	E[Q] ms
LEDBAT	FIFO	94.9	6437.3	99.5	5304.3
	RED	49.3	11.3	1.7	9.5
	SFQ	76.1	106.4	57.7	15.1
TCP-LP	FIFO	50.8	7870.6	65.8	7471.9
	RED	57.0	21.0	97.5	2.7
	SFQ	49.8	144.2	50.0	25.2

Under SFQ, TCP starvation phenomena are avoided. All flows get hashed in different buckets at enqueue time, and since hash buckets are queried in a round robin fashion, this guarantees that each flow is able to send data in turn – including the best-effort TCP that was heavily penalized under RED, though reprioritization still occurs. However, the queuing delay under SFQ (which has a default buffer size of 127 packets) grows up to about 100-150 ms, thus approaching the limit of what is considered to be harmful for interactive communication[42].

5.2. Internet experiments

We then perform additional experiments on the wild Internet. Since we have already shown the reprioritization to hold for different combinations of LPCC and AQM, our aim here is to disproof that these are artifacts that only arise in testbeds due to, e.g., the extremely controlled settings or, conversely, due to unexpected interactions inside the emulation layer.

In order to replicate a setup as close as possible to the typical user scenario, the sender is connected with 802.11g WiFi to an ADSL box. The receiver is connected to another ADSL box of another ISP through the Ethernet interface. The minimum RTT delay between the two hosts is approximately 50 ms, and the capacity between the hosts only slightly exceeds 500 Kbps (so that it can be compared with the testbed).

We carry on experiments only for the LEDBAT LPCC,

[12] While Windows, and thus Compound, constitutes the largest portion of hosts, it would be difficult to replicate the same methodology. Preliminary tests with Cubic suggest the phenomenon to hold; at the same time, we incur in problems with Cubic vs. TCP-LP since the latter appears to be *more aggressive* than Cubic under DropTail– so we prefer to examine the issue at a later time.

[13] In our settings, this is due to our emulated capacity limitations, coupled to the default Network Interface Card (NIC) queue length, which for Ethernet NICs is set to 1000 packets in Linux (`txqueuelen!`).

using two configurations[14] of RED with results summarized in Tab. 4. It can be seen that TCP starvation persists under RED when the number of flows is small: we stress that we observe TCP starvation only under the RED+LEDBAT combination, since as previously explained RED tries to penalize flows proportionally to the queue they create, while LEDBAT is designed to precisely avoid queuing.

This unexpected phenomenon is worth pointing out, as it may add other reasons not to deploy RED other than those listed in [37]. At the same time, it is also worth highlighting that the phenomenon appears to be non-systematic and possibly arises from specific configuration and network environment (that are possibly hard to reproduce [34, 35]). Additionally, starvation is not observed under scheduling disciplines as SFQ, which may play a more important role than RED in the near future (see discussion Sec. 6). It follows that the practical relevance of this starvation phenomenon is expected to be significantly smaller with respect to the repriorization phenomenon.

Finally, we experiment with RED*, a configuration inspired by the fluid model in [32], which reinstates the relative CC priorities, but at the cost of an already sizeable bufferbloat (in the order of several hundreds milliseconds, which makes it thus an impractical solution). The main difference between RED* vs. RED configurations is that RED* sets a larger min_{th} (16500B) compared to that in RED (1500B). As the queuing delay equivalent to min_{th} exceeds the default LEDBAT queueing delay target value (respectively, 264ms vs 100ms), this partly allows relative prioritization of end-to-end TCP vs. LEDBAT protocols. Intuitively, in this case our configuration starts dropping packets after the LEDBAT queuing delay target, so that TCP has the chance to grow its congestion window at the expense of LEDBAT (that by its LPCC nature will backoff in presence of best-effort TCP) before one of its packets get dropped by AQM; however, TCP recovery is in this case fast enough, and the additional buffer space beyond the LEDBAT target is large enough, to let TCP prevails over LEDBAT.

Table 4: LEDBAT: Testbed vs. Internet Experiments

		5+5 flows		1+1 flows	
		TCP%	E[Q] ms	TCP%	E[Q] ms
Testbed	FIFO	94.9	6437.8	99.5	5304.3
	RED	49.3	11.3	1.7	9.5
	RED*	71.9	763.9	98.2	333.8
Internet	FIFO	97.0	6551.7	98.9	916.0
	RED	2.6	45.0	5.1	29.1
	RED*	63.8	745.2	86.7	305.4

6. Discussion

This work points out possible negative issues arising from the interaction of AQM and CC techniques. Specifically, under AQM it is likely that LPCC techniques will become as aggressive as best-effort TCP (and can mislead at least one flavor of TCP to starvation under RED). We now discuss the implication of these findings.

As it seems that AQM and LPCC will have to coexist, there is a need to find possible ways out of the negative interplay we have shown in this paper. A general solution is hard to find, as testified by the current standpoint after over 20 years of research. Yet, a patch to the problem may be within reach, but may be hidden by radical positions in favor of either AQM or LPCC. While some see low priority protocols as useful in the transient period until AQM will be deployed [31], others are not convinced by AQM and propose "the end to end approach as the solution to bufferbloat and just forget about changing router behavior." [5]. Arguing that a practical solution requires a compromise between both extremes, we agree with more moderate viewpoints that "AQM is just one piece to the solution of bufferbloat" [11].

Consider indeed that an ideal solution (recall Fig. 5) should guarantee low access delay irrespectively of the mix of CC protocols and maintain the relative level of priority among flows in the mix as well. Since even a single AIMD flow may bufferbloat the bottleneck, the solution *needs scheduling and AQM*. At the same time, to avoid the CC reprioritization phenomenon, we argue that classification capabilities will be needed in AQM to account for flows' *explicitly advertised* level of priority. Notice that while in the more general case classification has failed to be adopted (IP TOS field, DiffServ, etc.), and the ability to claim *higher* priority could be easily gamed, in a hybrid AQM vs. LPCC world it makes sense for flows to claim a *lower* priority.

Other avenues in AQM and LPCC remain to be explored. First, AQMs evaluated in this paper either implement drop (RED/CoDel/CHOKe) or scheduling (SFQ) strategies. At the same time, hybrid AQM techniques that *jointly* exploit fair queuing with early drop are appearing – as the `fq_codel` technique that will make its way in future Linux kernels, starting from 3.5 [43]. Though further testing will be needed on these new AQMs, we argue that the reprioritization problem will remain: intrinsically, AQM and scheduling aim at fairness, whereas LPCC aims at the contrasting objective of unfairness with respect to TCP.

Another sensible question is whether it would be possible to differentiate priorities at a finer grain within the LPCC class: for instance, it is known that different targets in LEDBAT yield to starvation in case of backlogged flows

[14] Both RED and RED* parameters are set based on recommended settings proposed in [6]. RED* is a configuration inspired by [12, 32] and crafted ad hoc by a standard trial and error procedure. We stress once more that configuration details and scripts used for these experiments are available at [21].

under a DropTail discipline [30]; at the same time, is not clear what the behavior would be under AQM.

7. Conclusion

In this paper, we study the interaction between Active Queue Management (AQM) and Low-priority Congestion Control (LPCC). We consider a fairly large number of AQM techniques (i.e., RED, CHOKe, SFQ, DRR and CoDel) and LPCC protocols (i.e., LEDBAT, NICE, and TCP-LP), studying system performance (mainly expressed in terms of TCP% breakdown, average queue size $E[Q]$, and link utilization η) with both simulative and experimental methodologies.

Summarizing our main findings, we observe that *AQM resets the relative level of priority between best-effort TCP and LPCC*. That is to say, the TCP share of the bottleneck capacity drops dramatically, becoming close to the LPCC share. Additionally, while reprioritization generally equalizes the priority of LPCC and TCP, we also find that some AQM settings may actually lead best-effort TCP to starvation – as these are however non-systematic, we believe the starvation issue to be of less practical relevance than reprioritization.

Reprioritization is a fairly general phenomenon, as it holds for any combination of AQM technique, LPCC protocol and network scenario. This is testified by our thorough sensitivity analysis, where we confirmed the phenomenon for varying network parameters such as buffer size Q_{max}, link capacity C, heterogeneous RTT delay, flows number and flow duty cycle for over 3,000 simulations.

Reprioritization is a real world phenomenon, easily replicable on testbeds and the real Internet, which cannot be easily solved via AQM tuning. Hence, we advocate that explicit collaboration is needed from LPCC (e.g., openly advertise low priority) to assist AQM in taking decisions that maintain the desired level of priority.

Acknowledgement

This work was carried out at LINCS `http://www.lincs.fr`. The research leading to these results has received funding from the European Union under the FP7 Grant Agreement n. 318627 (Integrated Project "mPlane").

References

[1] J. Gettys, K. Nichols, Bufferbloat: Dark buffers in the Internet, Communications of the ACM 55 (1) (2012) 57–65. doi:10.1145/2063176.2063196.

[2] S. Cheshire, It's the Latency, Stupid!, `http://rescomp.stanford.edu/~cheshire/rants/Latency.html` (1996).

[3] C. Kreibich, N. Weaver, B. Nechaev, V. Paxson, Netalyzr: Illuminating the Edge Network, in: ACM SIGCOMM Internet Measurement Conference (IMC), 2010, pp. 246–259. doi:10.1145/1879141.1879173.

[4] J. Gettys, jg's Ramblings, `http://gettys.wordpress.com`.

[5] B. Cohen, TCP Sucks, `http://bramcohen.com/2012/05/07/tcp-sucks` (2012).

[6] S. Floyd, V. Jacobson, Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Transactions on Networking 1 (1993) 397–413. doi:10.1109/90.251892.

[7] P. E. Mckenney, Stochastic Fairness Queueing, in: IEEE INFOCOM, 1990, pp. 733–740. doi:10.1109/INFCOM.1990.91316.

[8] M. Shreedhar, G. Varghese, Efficient Fair Queueing Using Deficit Round Robin, ACM SIGCOMM Computer Communication Review (CCR) 25 (1995) 231–242. doi:10.1145/217382.217453.

[9] R. Pan, B. Prabhakar, K. Psounis, CHOKE, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation, in: IEEE INFOCOM, 2000, pp. 942–951. doi:10.1109/INFCOM.2000.832269.

[10] G. Carofiglio, L. Muscariello, On the Impact of TCP and Per-flow Scheduling on Internet Performance, in: IEEE INFOCOM, 2010, pp. 1–9. doi:10.1109/INFCOM.2010.5461973.

[11] K. Nichols, V. Jacobson, Controlling Queue Delay, Communications of the ACM (2012) 42–50doi:10.1145/2209249.2209264.

[12] M. Christiansen, K. Jeffay, D. Ott, F. D. Smith, Tuning RED for Web Traffic, ACM SIGCOMM Computer Communication Review (CCR) 30 (2000) 139–150. doi:10.1145/347059.347418.

[13] ADUF - Historique firmware, `http://88.191.250.12/viewtopic.php?t=164746&view=previous`, 2008-01-20 (2008).

[14] A. Venkataramani, R. Kokku, M. Dahlin, TCP Nice: A Mechanism for Background Transfers, ACM SIGOPS Operating Systems Review (2002) 329–343doi:10.1145/1060289.1060320.

[15] A. Kuzmanovic, E. W. Knightly, TCP-LP: A Distributed Algorithm for Low Priority Data Transfer, in: IEEE INFOCOM, Vol. 3, 2003, pp. 1691–1701. doi:10.1109/INFCOM.2003.1209192.

[16] M. Kuehlewind, G. Hazel, S. Shalunov, J. Iyengar, Low Extra Delay Background Transport (LEDBAT), Internet Engineering Task Force (IETF), RFC6817 (2012).

[17] A. Norberg, BitTorrent Enhancement Proposals on uTorrent transport protocol, BEP29 (2009).

[18] G. Carofiglio, L. Muscariello, D. Rossi, C. Testa, S. Valenti, Rethinking Low Extra Delay Background Transport Protocols, Elsevier Computer Networks 57 (2013) 1838–1852. doi:10.1016/j.comnet.2013.02.020.

[19] B. Cohen, How has BitTorrent as a protocol evolved over time, `http://www.quora.com/BitTorrent-protocol-company/How-has-BitTorrent-as-a-protocol-evolved-over-time` (2011).

[20] Sandvine, Global Internet Phenomena Report (1H 2013), `https://www.sandvine.com/downloads/general/global-internet-phenomena/2013/sandvine-global-internet-phenomena-report-1h-2013.pdf` (2013).

[21] `http://www.infres.enst.fr/~drossi/index.php?n=Dataset.LEDBATAQM`.

[22] A. Eshete, Y. Jiang, Approximate Fairness through Limited Flow List, in: International Teletraffic Congress (ITC), 2011, pp. 198–205.

[23] J. hwan Kim, H. Yoon, I. Yeom, Active Queue Management for Flow Fairness and Stable Queue Length, IEEE Transactions on Parallel and Distributed Systems 22 (4) (2011) 571–579. doi:10.1109/TPDS.2010.104.

[24] D. M. Divakaran, A Spike-detecting AQM to Deal with Elephants, Computer Networks 56 (13) (2012) 3087–3098. doi:10.1016/j.comnet.2012.04.025.

[25] A. Eshete, Y. Jiang, L. Landmark, Fairness among High Speed and Traditional TCP under Different Queue Management Mechanisms, in: Proceedings of the ACM SIGCOMM Asian Internet Engineeering Conference (AINTEC), 2012, pp. 39–46. doi:10.1145/2402599.2402605.

[26] W. Eddy, [aqm] floating a draft charter, `http://www.ietf.org/mail-archive/web/aqm/current/msg00127.html` (2013).

[27] S. Liu, M. Vojnovic, D. Gunawardena, Competitive and considerate congestion control for bulk data transfers, in: IEEE International Workshop on Quality of Service (IWQoS), 2007, pp. 1–9. doi:10.1109/IWQOS.2007.376542.

[28] P. B. Key, L. Massoulié, B. Wang, Emulating Low-priority Transport at the Application Layer: a Background Transfer Ser-

vice, in: ACM SIGMETRICS/Performance, 2004, pp. 118–129. doi:10.1145/1005686.1005703.
[29] D. Ros, M. Welzl, Less-than-Best-Effort Service: A Survey of End-to-End Approaches, Communications Surveys & Tutorials 15 (2) (2013) 898–908. doi:10.1109/SURV.2012.060912.00176.
[30] G. Carofiglio, L. Muscariello, D. Rossi, C. Testa, A Hands-on Assessment of Transport Protocols with Lower than Best Effort Priority, in: IEEE Conference on Local Computer Networks (LCN), 2010, pp. 8–15. doi:10.1109/LCN.2010.5735831.
[31] J. Schneider, J. Wagner, R. Winter, H.-J. J. Kolbe, Out of My Way-Evaluating Low Extra Delay Background Transport in an ADSL Access Network, in: International Teletraffic Congress (ITC), 2010, pp. 1–8. doi:10.1109/ITC.2010.5608714.
[32] Y. Gong, D. Rossi, E. Leonardi, Modeling the Interdependency of Low-priority Congestion Control and Active Queue Management, in: International Teletraffic Congress (ITC), 2013, pp. 1–9. doi:10.1109/ITC.2013.6662942.
[33] Y. Gong, D. Rossi, C. Testa, S. Valenti, D. Taht, Fighting the Bufferbloat: on the Coexistence of AQM and Low Priority Congestion Control, in: IEEE INFOCOM Workshop on Traffic Measurement and Analysis (TMA), 2013, pp. 3291–3296. doi:10.1109/INFCOM.2013.6567153.
[34] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, N. McKeown, Reproducible Network Experiments Using Container-based Emulation, in: ACM CoNEXT, 2012, pp. 253–264. doi:10.1145/2413176.2413206.
[35] J. Costa-Roberts, C. Case, Seeing RED, `http://reproducingnetworkresearch.wordpress.com/2012/06/05/seeing-red` (2012).
[36] `http://www.enst.fr/~drossi/ledbat`.
[37] M. May, J. Bolot, C. Diot, B. Lyles, Reasons Not to Deploy RED, in: International Workshop on Quality of Service (IWQoS), 1999, pp. 260–262. doi:10.1109/IWQOS.1999.766502.
[38] D. Rossi, C. Testa, S. Valenti, L. Muscariello, LEDBAT: the New BitTorrent Congestion Control Protocol, in: International Conference on Computer Communications and Networks (ICCCN), 2010, pp. 1–6. doi:10.1109/ICCCN.2010.5560080.
[39] G. Carofiglio, L. Muscariello, D. Rossi, S. Valenti, The Quest for LEDBAT Fairness, in: IEEE GLOBECOM, 2010, pp. 1–6. doi:10.1109/GLOCOM.2010.5683559.
[40] The uTorrent Transport Protocol library, `http://github.com/bittorrent/libutp` (2010).
[41] D. Rossi, C. Testa, S. Valenti, Yes, We LEDBAT: Playing with the New BitTorrent Congestion Control Algorithm, in: Passive and Active Measurements (PAM), 2010, pp. 31–40. doi:10.1007/978-3-642-12334-4_4.
[42] I.-T. S. G. 12, One-way transmission time, International Telecommunication Union, ITU G.114 (2003).
[43] The Codel Archives, `https://lists.bufferbloat.net/pipermail/codel`.

Vitae

Yixi Gong received a B.Sc. in Information Security from Fudan University (Shanghai, China) in 2009 and an Engineering Degree in information system from Telecom ParisTech (Paris, France) in 2012. He is currently pursuing a Ph.D. at the Computer Science and Networking (INFRES) department of Telecom ParisTech (Paris, France) under the supervision of Prof. Dario Rossi. His research focuses on Internet traffic measurement.

Dario Rossi is a Professor at the Computer Science and Networking (INFRES) department of Telecom ParisTech (Paris, France). He received his M.Sc. and Ph.D. degrees from Politecnico di Torino in 2001 and 2005 respectively, and his HDR degree from Universite Pierre et Marie Curie (UPMC) in 2010. During 2003-2004, he held a visiting researcher position in the Computer Science division at University of California, Berkeley. He has coauthored over 100 papers in leading conferences and journals, holds 6 patents and he participated in the program committees of over 40 conferences including ACM CoNEXT and IEEE INFOCOM. His research interests include Internet traffic measurement, information centric networks, green networking, peer-2-peer and traffic engineering.

Claudio Testa received a M.Sc. degree in Computer and Communication Networks Engineering from the Politecnico di Torino (Torino, Italy), with a Master Thesis in the network traffic analysis field. He received a Ph.D. degree at the Computer Science and Networking (INFRES) department of Telecom ParisTech (Paris, France) under the supervision of Prof. Dario Rossi in November 2012. His research focused in the field of low-priority congestion control algorithms, peer-to-peer services and content distribution applications. Since July 2013, he is working in the Content Distribution Network team at Orange.

Silvio Valenti received a M.Sc. degree in Computer Science Engineering at Politecnico di Torino (Italy), in February 2008 and received a Ph.D. at INFRES department of TELECOM ParisTech (Paris, France) in 2012. His research interests are relative to peer-2-peer networking, Internet traffic classification and high-speed packet processing. Since November 2012, he is working in the Gmail team at Google.

Dave Taht works at Bufferbloat.net. Singer, pianist, guitarist, writer, experimenter, theorist, hacker, maker, faker - in no particular order, on any given day. - Cheerleader. Gimme an I!