Rethinking the Low Extra Delay Background Transport (LEDBAT) Protocol

Giovanna Carofiglio[b], Luca Muscariello[c], Dario Rossi[a,1], Claudio Testa[a], Silvio Valenti[a,2]

[a] *Telecom ParisTech, France*
[b] *Bell Labs, Alcatel-Lucent, France*
[c] *Orange Labs, France Telecom, France*

Abstract

BitTorrent has recently introduced LEDBAT, a novel application-layer congestion control protocol for data exchange. The protocol design assumes that network bottlenecks are at the access of the network, and that thus user traffic competes creating self-induced congestion. To relieve this phenomenon, LEDBAT is designed to quickly infer when self-induced congestion is approaching (by detecting relative changes of the one-way delay in the transmission path), and to react promptly by reducing the sending rate prior to the congestion occurrence. Previous work has however shown LEDBAT to be affected by a latecomer advantage, where newly arriving connections can starve already existing flows. In this work, we propose modifications to the congestion window update mechanism of LEDBAT that solve this issue, thus guaranteeing intra-protocol fairness and efficiency. Closed-form expressions for the stationary throughput and queue occupancy are provided via a fluid model, whose accuracy is confirmed by means of ns2 packet level simulations. Our results show that the proposed change can effectively solve the latecomer issue, furthermore without affecting the other original LEDBAT goals.

1. Introduction

As recently pointed out in [9], "Internet delays now are as common as they are maddening". The root cause for these delays can be identified with the excess buffering inside a network, which is nicknamed "bufferbloat". Though this is nothing new [10], the situation got worse in the latest years due to mainly two facts: (i) TCP loss-based design, that forces the bottleneck buffer to fill before the sender reduces his rate and (ii) the fact that low uplink capacities of widely deployed ADSL and Cable modems can translate into significant queuing delay (up to few seconds [24]).

Evidently, BitTorrent engineers were well aware of this fact. Indeed, the popular peer-to-peer file sharing system with hundreds of millions of daily users worldwide,

[1] Corresponding author `dario.rossi@enst.fr`
[2] Current affiliation: Google Inc.

has recently developed a novel application-layer congestion control protocol for data exchange. The novel insight in the widely explored congestion control landscape is in this case the reasonable assumption that the bottleneck is most likely at the access of the network (e.g., at the ADSL modem line), which means that congestion is therefore typically *self-induced* by concurrent traffic sessions generated by the same user (e.g., BitTorrent transfers in parallel with Skype call and Web browsing). The new protocol, named LEDBAT after *Low Extra Delay Background Transport*, is designed to solve this issue and targets (i) efficient but (ii) low priority transfers. When LEDBAT flows have the exclusive use of the bottleneck resources, they fully exploit the available capacity. When instead other transfers –such as VoIP, gaming, Web or other TCP flows– are ongoing, LEDBAT flows back off to avoid harming the performance of interactive traffic. To attain the efficiency aim, LEDBAT flows need to create queuing, as otherwise the capacity would not be fully utilized. At the same time, due to the low-priority aim, the amount of extra queuing delay induced by LEDBAT flows should be small enough to avoid hurting the interactive traffic – hence the protocol name.

LEDBAT[3] has been defined as an IETF draft [37] (which focuses more on the algorithmic aspects) and as a BitTorrent Enhancement Proposal BEP-29 [32] (that instead focuses more on the UDP framing). and has recently become the BitTorrent default congestion control protocol, replacing thus TCP for the data transfer.

While previous research [33, 34, 8, 7, 4, 13, 36, 39, 40] on LEDBAT has shown its potential for P2P transfers it has also highlighted some serious fairness deficiencies. In more details, LEDBAT has a good interplay with BitTorrent-like transmissions of short-lived chunk-long flows [39, 40]: hence, the definition of this protocol under a BEP document makes perfectly sense, as the performance improvement for BitTorrent is in this case coupled to a reduction of the self-induced bufferbloat. At the same time, LEDBAT is affected by a latecomer unfairness issue [34, 8] that arises in (the not so uncommon) case of backlogged flows: under this conditions, latecomer flow takes over the bottleneck resource, starving the first-comers. As such, the normalization of an ill-defined protocol can have potentially dramatic effects: while the phenomenon hardly ever happens in P2P swarm like content delivery, it can severely impact the performance of backups, photo uploads and, more generally, long lived uploads of home users towards their virtual storage (e.g. DropBox, Google, Windows Live, Apple iCloud, etc.).

With respect to the normalization of an IETF congestion control algorithm, whose scope goes beyond a specific application, though popular it may be, fairness is a significant property that should always be taken into account for the design of any experimental or deploy-able protocol. The lack of fairness is an indication of poor convergence properties, e.g. the algorithm is unstable because of an unwise choice of the parameters or, more tricky, the algorithm cannot be stable for any choice of parameter. We show in this paper that LEDBAT falls in this second category.

The main contribution of this work is to propose a modification to the LEDBAT congestion control that, leaving untouched the design goals, solves the fairness issue

[3]The protocol has been christened as LEDBAT in the IETF, and as uTP in the BitTorrent BEP community: to avoid ambiguity, we use its IETF name.

– therefore avoiding unwanted effects at the application-layer altogether. Throughout this paper, we make use of several complementary techniques to study our proposal. First, we use passive measurements to gauge the popularity of LEDBAT transfers in the real Internet, and exploit an active testbed methodology to show the fairness issue in current BitTorrent. We propose a modification to the original LEDBAT protocol, to jointly achieve fairness and efficiency, and develop a fluid model to describe the system dynamics. An analytical solution of the model proves the soundness of our design, while numerical solutions allow us to grasp the transient phase as well. Finally, we use extensive `ns2` packet-level simulations to evaluate the LEDBAT performance under several scenarios. Since we need to ensure that our proposal works as expected under any circumstances, we include the general case of backlogged transfers (e.g., two concurrent low priority backups). At the same time, we need to ensure that LEDBAT does not harm the experience of BitTorrent users, hence we include P2P-like scenarios involving multiple-flows and a heterogeneous network environment as well.

The remainder of this paper is organized as follows. Related work and motivations are covered in Sec. 2 and Sec. 3 respectively. The unfairness issue is introduced in Sec. 4, followed by our proposed modification in Sec. 5 along with the fluid model and its analytical solution. More complex network scenarios are tackled by means of packet-level simulations starting from Sec. 6, where we also compare the numerical solution of the fluid model with simulation results. We then proceed by studying the impact of the traffic model (e.g., backlogged vs chunk-based transfers) in Sec. 7, the sensitivity of the protocol to parameter changes in Sec. 8. Lastly we evaluate the performance of the protocol in P2P-like settings from a single-peer as well as a whole swarm perspective in Sec. 9, before Sec. 10 concludes the paper.

2. Related Work

Congestion control studies on the Internet date back to [21] and it is out-of-scope to provide a full review of the existing literature here. Still, a couple of references are worth citing as they share LEDBAT low-priority spirit [41, 25, 28, 23]. For instance, TCP-LP [25] and NICE [41] share the same goal as LEDBAT aiming at implementing a *Lower-than Best Effort (LBE)* service for background transfers. In more detail, NICE extends the delay-based behavior, typical of TCP Vegas, with a multiplicative decrease reaction to early congestion, which is actually detected when the number of packets experiencing a large delay in an RTT exceeds a given threshold. On the other hand, TCP-LP enhances the loss-based behavior of TCP NewReno with an early congestion detection based on the distance of the instantaneous One-Way Delay from a weighted moving average calculated on all observations. In case of congestion, the protocol halves the rate and enters in a inference phase, during which, if further congestion is detected, the congestion window is set to zero and normal TCP NewReno behavior is restarted. At the same time, TCP-LP [25], NICE [41] differ from LEDBAT in that the latter aims at introducing a *bounded* extra delay: i.e., when queuing delay reaches a given target, the LEDBAT protocol slows down its transmission rate to ensure the queuing delay target is not exceeded. Notice that this is especially important for VoIP, gaming, and all other interactive delay-sensitive applications.

Related work has already tackled the intra-protocol fairness issue affecting delay-based congestion control algorithms. In particular, regarding TCP Vegas [6] which is the first example of such a family of techniques, the problem was pointed out [31] in relation to (i) to route changes and (ii) to persistent congestion when multiple concurrent Vegas flows compete at the same bottleneck. The latter is the same malfunctioning we spotted in the original LEDBAT design: latecomer flows overestimate the base delay because of the queuing delay old flows have already imposed in the buffer. LEDBAT proposers clearly took advantage of this literature when designing the protocol, for instance when they adopted the same solution of [31] to the rerouting problem (i.e., by using only the recent history of delay observations for the base delay estimation), but they neglected the latecomer advantage.

To solve the fairness issue, researchers have followed various approaches: on the one hand, some works try to improve the estimation of the base RTT [13, 16, 44]; on the other hand, others propose techniques to achieve fairness in spite of the base delay estimation error [5, 27, 18, 26]. The first type of work usually relies on additional support from the network to correct the measurement: for instance, [44] uses an out-of-band priority packet which skips the queue and provides a good estimation of the base delay; [16] instead adapts its parameters according to the number of congested routers on the path, thus relying on their feedback. Authors of [5, 18] follow the opposite approach and prove that a particular choice of parameters allows flows to converge to a fair share of the available bandwidth. The delay based algorithm proposed in [27] is also shown capable of dealing with noisy delay measurement, thanks to the careful choice of the controller function. Finally, [26] proposes a new delay based AIMD algorithm and choose a backoff factor which avoids measurement errors. Our work is the first to study this issue for a *Lower-than Best Effort* protocol and to achieve together efficiency, fairness and lower-priority. The main contribution is the reintroduction of the multiplicative decrease component, as we correctly identify the root cause of unfairness in the addictive decrease component [8, 22] rather than in the measurement error.

Other related work concerns the BitTorrent application. Also in this case, BitTorrent has not only become a largely popular application among its users, but it has become a rather popular research subject as well. At the same time, only few works have, for the time being, focused on LEDBAT aspects [13, 33, 34, 8, 7, 4]. An experimental approach is used in [13, 33, 36] to investigate on LEDBAT operations. In [13] BitTorrent developers detail a specific aspect of their implementation: namely, an algorithm to solve the problem of the clock drift, to ameliorate the queuing delay estimation at the sender side. In [33] we present an experimental study of the protocol, exploiting a black-box approach, since at the time of the experiments the protocol was closed-source. Authors in [36] instead study LEDBAT in a local testbed, employing different real ADSL modems, focusing on the interaction of LEDBAT and Active Queue Management (AQM) techniques that are becoming commonplace in modern home gateways. Finally, our recent work on the impact of LEDBAT on the swarm completion time (using simulative [39] or experimental [40] approaches), has shown that reducing the queuing delay results in a faster propagation of control information (e.g., chunk availabilities), which is beneficial for the whole swarm. This work not only addresses P2P scenarios, but also considers more general settings where LEDBAT can be used

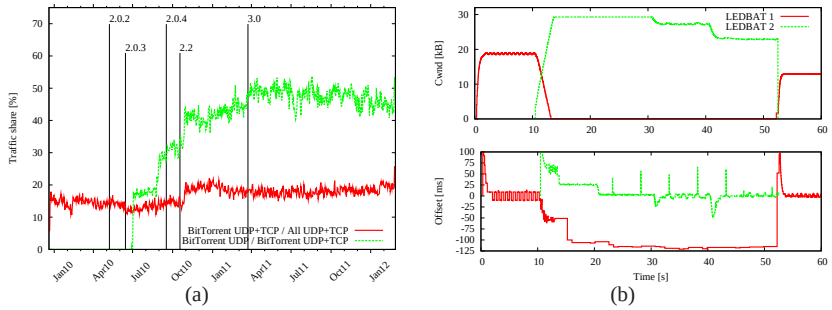

Figure 1: (a) Proportion of BitTorrent and BitTorrent LEDBAT traffic in the wild. (b) latecomer unfairness of the BitTorrent LEDBAT implementation in simple testbed experiment (congestion window, top; offset from target, bottom).

as a congestion control for potentially any Internet application (including backlogged transfers).

Most of the work on LEDBAT adopts a simulative approach [34, 8, 7, 4]. In our previous work [34], we performed a preliminary performance evaluation of LEDBAT considering the default parameter settings suggested in the IETF draft, and unveiling the latecomer issue. In [7] we instead focus on a comparison of low-priority protocols, contrasting LEDBAT with TCP-LP [25] and NICE [41], showing that LEDBAT has the lowest level of priority. Along similar lines, authors in [4] investigate the policies for dynamic parameter tuning. In [8] we focus closely on the fairness issue, and identify the late-comer advantage as an intrinsic drawback of Additive Increase Additive Decrease (AIAD) policy, as was already shown by Jain et al. in the 80s [22, 11]. In the case of LEDBAT we show in [8] that errors in the queuing delay measurement can further exacerbate the problem. In this paper we bring back to light the rather fundamental argument supporting the multiplicative decrease component of the sender window after congestion signals, in order to ensure fairness. In particular, we focus on the delay-based component of LEDBAT that comes into play for intra-protocol fairness. It is worth remarking that such an argument is valid for any kind of congestion indicator, delay or loss at the bottleneck link.

This work builds upon the knowledge gained in our previous effort [34, 8, 7], from which it significantly differ. First, while [34, 8, 7] only resorted to simulation, in this work we exploit a fluid approach, analytical techniques, simulations and experimental measurement. Also, scenarios investigated are more complex (e.g., heterogeneous delays, multiple flows P2P-like scenario, chunk-based transmission) than our previous work, providing a more complete picture of the protocol performance. Moreover, with the exception of [8], our previous work mainly focused on the evaluation of the LEDBAT as is, i.e., without attempting any *modification* to the draft proposal – which is instead the main aim of the current work. Finally, even though we proposed some simple solutions to the fairness issue in [8], these partly failed in meeting the efficiency goal – which we instead successfully address in this work.

3. Motivation

Despite recent research showing an increasing importance of video over the share of Internet traffic [3], BitTorrent still represents a significant portion of user generated data – and due to the recent shutdown of popular file-hosting services such as Megaupload/Megavideo [15], we can expect the Internet ratio of file-sharing to increase again. In Fig. 1(a), we depict, over the last few years, the BitTorrent traffic share (UDP and TCP traffic, overall traffic) averaged at the PoP of 5 European ISPs that we continuously monitor [14, 17]. The dashed line represents the relative percentage of BitTorrent traffic carried over UDP (hence, over the LEDBAT transport protocol), normalized over the total amount of BitTorrent traffic. Labels report a few BitTorrent application releases over the considered period[4]. The figure clearly shows, soon after the release of μTorrent 2.0.2, which first introduced data transport over LEDBAT, a steep increase of the percentage of BitTorrent traffic carried over UDP. Notice also that, while the overall share of BitTorrent is steady during the whole period[5], the percentage of data sent over UDP slowly increased and stabilized to about half of the BitTorrent traffic volume. Thus LEDBAT already represents a significant portion of Internet traffic nowadays. Moreover, as the IETF LEDBAT WG is now closing, and the draft its making its way to an IETF RFC, this proves the importance and timeliness of the present study.

To further highlight the relevance of this study, we point out that the latecomer unfairness we unveiled in [34] by simulation, also holds in practice, possibly leading to severe flow starvation. We show this by performing testbed experiments of the recently released BitTorrent open-source LEDBAT library [19]. As in [34], we consider two PCs connected by a $C = 10$ Mbps Ethernet bottleneck, where we emulate by means of `netem` [20] a $RTT = 50$ ms delay. The first flow starts at time $t = 0$ while we let the latecomer join (and spoil) the party at $t = 10$ s. Backlogged transfers are started using the source code provided in [19], instrumented to produce detailed application-level logs[6]. Results of the experiment are shown in Fig. 1(b), whose top portion reports the congestion window of the two flows over time. As soon as the first flow starts, it increases its congestion window until the target is reached, and then settles. However, when the latecomer kicks in at $t = 10$ s, the congestion window of the first-comer drops until starvation. The situation persists until $t = 50$ s, at which time we stop the latecomer transfer: right after, the first-comer opens its congestion window again, saturating the link.

This behavior can be explained considering that LEDBAT aims at introducing a *fixed target* amount of queuing delay in the bottleneck. The bottom plot of Fig. 1(b) reports the offset from the fixed queuing delay target[7] $\tau = 100$ ms measured by each LEDBAT flow over time. At $t = 0$ the queue is empty, so the queuing delay is null

[4] See "Announcement" thread from the μTorrent forum `http://forum.utorrent.com/viewforum.php?id=4`

[5] With an increase after Megaupload shutdown, though [14] already anticipated the P2P traffic raise.

[6] Packet level traces are also captured and post-processed as in [33] for cross-checking purposes: the results, which we are unable to report here for lack of space, are in agreement with the application logs.

[7] Early version of the IETF draft specified a mandatory value of $\tau = 25$ ms, while last versions suggest $\tau = 100$ ms, as in the BEP-29. Notice also that, as shown in [7, 36], heterogeneous targets τ can be cause of further unfairness.

and the offset sensed by the first flow equals the target. As the first flow grows its window and starts transmitting causing queuing delay, the offset shrinks, until the target is hit and the offset reaches zero: in this region, the congestion window settles and the capacity is efficiently exploited. However as soon as the second flow starts, its base delay measurement include a non-null queuing delay: more precisely, it senses a queuing delay equal to the target, which is caused by the first comer, to which it adds its own target $\tau = 100$ ms. The latecomer thus sets a target higher than the first one (double in this case), and aggressively starts climbing the bottleneck. In its turn, the first comer senses a growing queuing delay, which exceeds its own target (as a negative offset from target means that the target is exceeded) and so it slows down its own sending rate. This is an unfortunate situation, that can however be easily corrected as we show in the following sections.

4. Current LEDBAT fairness issues

According to the original draft proposal [37], LEDBAT maintains a minimum One-Way Delay (OWD) estimation D_{min}, which is used as base delay to infer the amount of delay due to queuing. LEDBAT flows have a target queuing delay τ, i.e., they aim at introducing a small, fixed, amount of delay in the queue of the bottleneck buffer. Flows monitor the variations of the queuing delay $q(t) - D_{min}$ to evaluate the distance $\Delta(t)$ from the target:

$$\Delta(t) = (q(t) - D_{min}) - \tau, \tag{1}$$

where $q(t)$ is the queueing delay measured at time t. The value of the offset $\Delta(t)$ is then used to drive the congestion window evolution, which is updated packet-by-packet at each acknowledgement reception as follows:

$$cwnd(t+1) = \begin{cases} cwnd(t) + \alpha \frac{\tau - \Delta(t)}{\tau} \frac{1}{cwnd(t)} & \text{if no loss ,} \\ \frac{1}{2} cwnd(t) & \text{if loss.} \end{cases} \tag{2}$$

where t is a discrete time variable that increments by 1 at each ACK arrival and $cwnd(t)$ is the congestion window at time t. The drawbacks of such a congestion window update mechanism have been outlined in [8] and mainly consist in the intra-protocol unfairness coupled with a poor calibration of the LEDBAT level of aggressiveness with respect to TCP.

4.1. Impact of additive decrease

We proved in [8] that the unfairness arising from two competing LEDBAT flows starting at different moments is due to the *additive decrease* component $\alpha \frac{\tau - \Delta(t)}{\tau}$ that intervenes when $\Delta(t) > \tau$. We thus argue that the additive decrease, rather than the measurement errors, is the main cause of unfairness in the LEDBAT protocol; in other words, the late-comer advantage is actually a fundamental drawback of the additive decrease term, meaning that the original design is currently misguided.

Without loss of generality, let us consider the case of N LEDBAT flows with the same round trip time $R(t)$, sharing the same link of capacity C and finite buffer size B.

Each flow $i \in \mathcal{N}$, with $\mathcal{N} = \{1, 2, \ldots, N\}$, starts at $t_i \geq 0$, with $t_1 \leq t_2 \leq \cdots \leq t_N$ and with an initial congestion window W_i. Given the packet-level congestion window dynamics in (2), we demonstrate the following statement.

Proposition 4.1 *If $N < \frac{B}{\tau C}$, and $d_{max}(t_N) \triangleq \max_{i,j \in \mathcal{N}}[W^i(t_N) - W^j(t_N)] > 0$, then the system is unfair, i.e., $\exists t^* \geq t_N$, such that $\forall t > t^*$ $d_{max}(t) > 0$.*

Proof 4.2 *Given (2), a simple fluid representation of the window dynamics of flow i, $W_i(t)$, in continuous time, is:*

$$\frac{dW_i(t)}{dt} = \frac{1}{R}\frac{\tau - Q(t)}{\tau}, \tag{3}$$

where we supposed for simplicity $R(t) \approx R$, which is true for large propagation delay (the proof can be easily extended to the case of variable round trip delays). Since the estimated queuing delay can be different for each flow, depending on its stored base delay, we replace $Q(t)$ by $Q_i(t)$, i.e., the queue occupancy measured by each sender, and simply observe that $Q_i(t)$ varies in the interval $(Q(t) - (N-1)\tau, Q(t))$. Indeed, the last flow makes the largest error in the estimation of the queuing delay, because it measures as base delay the actual propagation delay increased by $(N-1)\tau$, the sum of the target delay of all preceding flows. It follows that, $\forall i, j \in \mathcal{N}$:

$$W^i(t) - W^j(t) = W^i(t_N) - W^j(t_N) + \int_{t_N}^{t} \frac{Q_j(u) - Q_i(u)}{R\tau} du$$

where $|Q_j(t) - Q_i(t)|$ is bounded by $(N-1)\tau$. Hence, if we choose t^* equal to $t_N + \frac{W^{i*}(t_N) - W^{j*}(t_N)}{(N-1)/R}$, with $(i^*, j^*) = \arg\max_{i,j \in \mathcal{N}} W^i(t_N) - W^j(t_N)$, we have:

$$\begin{aligned}
d_{max}(t) &\triangleq \max_{i,j \in \mathcal{N}} W^i(t) - W^j(t) \\
&\geq \max_{i,j \in \mathcal{N}} W^i(t_N) - W^j(t_N) + \frac{(N-1)}{R}(t - t_N) \\
&= \frac{(N-1)}{R}\left((t - t_N) + \frac{W^i(t_N) - W^j(t_N)}{N-1}R\right) \\
&= \frac{(N-1)}{R}(t - t^*) > 0, \qquad \forall t > t^*.
\end{aligned}$$

Observation 4.3 *The fact that the system evolves towards an unfair state is strictly related to the fact that the dynamic equations, describing the state of the system are unstable. Besides equations (3) for the sources, we have:*

$$\frac{dQ(t)}{dt} = \sum_{i=1}^{N} \frac{W_i}{R} - C\mathbb{1}_{Q_t > 0}. \tag{4}$$

Equations (3),(4) define a linear system of ODEs with characteristic polynomial equal to $\lambda^{N-1}\left(\lambda^2 + NC\tau R\right)$. *The corresponding eigenvalues are* $\lambda_1 = 0$, $\lambda_{1,2} = \pm i\sqrt{\frac{N}{R\tau}}$, *and the matrix associated with the system of ODEs is easily shown to be diagonalizable by standard algebra. As the eigenvalues have zero real part, the system cannot consequently be asymptotically stable. Being the matrix diagonalizable, the solution is limited for every* t. *However, the dependence to the initial condition never vanishes because of the zero real part of the eigenvalues. In addition, the associated matrix cannot be inverted because of the zero eigenvalue, which implies that the solution of the system has an orbit around any* $(W_1, \ldots, W_N, Q)$ *such that* $\sum_i W_i = RC$, $Q = \tau$. *In other words, the linear response of LEDBAT is never able to make the stable point reachable from any initial condition: this is the root cause of the observed latecomer advantage phenomenon that we aim at solving in the following.*

5. Proposed LEDBAT modification

To address the latecomer issue, we propose to modify the delay-based decrease term and *to introduce a multiplicative decrease* continuously driven by the estimated distance from the target, $\Delta(t)$. Intuitively, the multiplicative window reduction response to congestion allows the source sending rate to slow down enough to make a stable (and fair) point always reachable. Clearly, to guarantee at the same time fairness and protocol efficiency, a proper choice of the decrease factor has to be made, so as to prevent significant (and unnecessary) drops in the congestion window. In addition, we observe that the additive increase term as in (2) makes LEDBAT flows slow down the increase factor until the target τ is reached, in which case the window increase completely stops. This clearly implies a smaller convergence to the target and hence a minor efficiency if compared to the case of a constant additive increase factor independent of $\Delta(t)$. Based on the above observation, we propose to modify the increase term as well, and *to introduce an additive increase* according to a constant factor α as in TCP Reno. In this way we expect to achieve better efficiency performance without violating the low priority requirements as expressed in the LEDBAT draft. Indeed, by selecting $\alpha \leq 1$ the additive increase component can be made at most as aggressive as TCP. Summarizing the observation from the previous section, we propose to modify the congestion window evolution as follows:

$$cwnd(t+1) = \begin{cases} cwnd(t) + \alpha\frac{1}{cwnd(t)} & \text{if no loss and } \Delta \leq 0, \\ cwnd(t) + \alpha\frac{1}{cwnd(t)} - \frac{\zeta}{\tau}\Delta & \text{if no loss and } \Delta > 0, \\ \frac{1}{2}cwnd(t) & \text{if loss.} \end{cases} \tag{5}$$

In the following sections we quantify the overall improvement deriving by such a congestion window update by means of both a *fluid model*, which provides a closed-form characterization of the stationary throughput and *simulations*, which allow the study of more complex scenarios. In the remainder of this paper, we refer to the modified version of LEDBAT as fair-LEDBAT (fLEDBAT).

Table 1: Notation

N	Number of fLEDBAT flows
C	Link capacity
$\{W^i(t)\}_{i=1,\ldots,N}$	Congestion windows at time t
$\{X^i(t)\}_{i=1,\ldots,N}$	Instantaneous rates at time t
Q_t	Queue occupancy at time t
α	Additive Increase factor
ζ	Multiplicative Decrease factor
R_t	Round trip time at time t
τ	Queuing delay target

5.1. Fluid model description

In this section we develop a fluid model of the congestion window and hence of the transmission rate of one or more fLEDBAT flows, aimed at capturing first order system dynamics. The congestion window is now a continuous variable both in time and in space, $W(t)$ (the remainder of the notation is summarized in Tab.1). We consider the case of N fLEDBAT flows sharing the same link of capacity C and experiencing the same[8] Round Trip Time R_t. In addition, we make the following assumptions:

- The round trip time R_t is defined by the sum of twice the propagation delay, R, transmission delay $1/C$ and queueing delay $q(t)$. We further assume that the propagation delay is predominant, i.e., $R_t \approx R$.
- The queueing delay $q(t)$ is defined as ratio of the queue occupancy Q_t at time t divided by the link capacity C, i.e., $q(t) = Q(t)/C$. Thus, we assume that the queuing delay information *instantaneously* propagates to the sender, neglecting thus the delay in the feedback loop.
- We assume that flows can correctly estimate the queuing delay, which is equivalent to $D_{min} = 0$.
- By Little's law, we assume that congestion windows and link rates are linked by $X_t^i = W_t^i/R_t, \quad \forall i = 1, ..., N$.

Still, the assumption that flows can correctly estimate the queuing delay may not hold in practice. As such, we expect that simulation results may show an offset with respect to the model predictions, which is due to this simplifying assumption. There are however two main reasons for which we believe this assumption, which makes the problem tractable, is also reasonable. First, additional mechanisms to enhance the delay estimation accuracy could be then adopted in order to ameliorate the overall protocol performance: this has been done in previous work [25], and is also part of the current BitTorrent effort [13] to reduce the measurement error and hence reinforcing our assumptions. Second, a more fundamental reason is that the characterization of

[8]Though the model generalizes to the case of heterogeneous RTT, for the sake of simplicity in this paper we focus on the homogeneous case.

protocol dynamics in absence of such estimation error is a necessary step in the fLEDBAT protocol design – as, even on simplistic settings, important properties of the protocol such as efficiency and fairness can be *proved* to hold with the help of a rigorous framework.

5.2. Fluid system dynamics

Let us consider the case of N fLEDBAT connections, whose congestion window evolves according to (5). The corresponding flow-level congestion window evolution is:

$$\frac{dW_i(t)}{dt} = \frac{\alpha}{R} - \frac{\zeta}{\tau}\left(\frac{Q(t)}{C} - \tau\right)\frac{W_i(t)}{R}\mathbb{1}_{W_i(t)\geq 0}\mathbb{1}_{Q(t)\geq C\tau}, \tag{6}$$

where we denote by $W_i(t)$ the instantaneous congestion window at time t for connection i in the fluid system. As we assume an approximately constant round trip delay, we replace R_t by R in (6). The instantaneous queue occupancy instead satisfies:

$$\frac{dQ(t)}{dt} = \sum_{i=1}^{N}\frac{W_i(t)}{R} - C\mathbb{1}_{Q(t)\geq 0}. \tag{7}$$

where, in other words, only the flow that exceeds the capacity creates queuing in the buffer. Thus, the instantaneous rate of connection i, $X_i(t)$, satisfies:

$$\frac{dX_i(t)}{dt} = \frac{\alpha}{R^2} - \frac{\zeta}{R\tau}\left(\frac{Q(t)}{C} - \tau\right)X_i(t)\mathbb{1}_{\{X(t)\geq 0\}}\mathbb{1}_{Q(t)\geq C\tau} \tag{8}$$

and (7) can be re-written as:

$$\frac{dQ(t)}{dt} = \sum_{i=1}^{N}X_i(t) - C\mathbb{1}_{Q(t)\geq 0}. \tag{9}$$

5.3. Main results

We now present the main results of this paper: namely, the existence of a unique and globally stable solution. We also express, with closed form formulæ, the performance of the protocol at the equilibrium, proving its *efficiency* and *fairness* – which was our initial goal. Let us start by proving that the system admits a unique solution.

Proposition 5.1 *The system of ODEs (8)-(9) admits the unique equilibrium* $P^* = (X_1^*, \ldots, X_N^*, Q^*)$

$$X_i^* = C/N,\ i = 1, \ldots, N \qquad\qquad Q^* = C\tau + \frac{N\alpha\tau}{\zeta R} \tag{10}$$

where X_i^* *and* Q^* *denotes the stationary values of* X_i *and* Q *respectively.*

Proof 5.2 *We consider the stationary regime by the condition* $(\dot{X}_i, \ldots, \dot{X}_N, \dot{Q}) = (0, \ldots, 0)$

$$\begin{aligned}
\dot{Q} = 0 \Leftrightarrow \sum_{i}^{N} X_i^* = C,\\
\dot{X}_i = 0 \Leftrightarrow 0 = \frac{\alpha}{R^2} - \frac{\zeta}{RC\tau}(Q^* - C\tau)X_i^*,\\
\Leftrightarrow 0 = \frac{\alpha}{R^2} - \frac{N\alpha}{CR^2}X_i^* \Leftrightarrow X_i^* = C/N,\ i = 1, \ldots, N. \qquad (11)
\end{aligned}$$

Then, the following proposition states that this unique equilibrium is also globally stable (see [42]).

Proposition 5.3 *The system of ODEs (8)-(9) is globally stable in* P^*.

Proof 5.4 *Let us write* $\mathbf{X} = (X_1, \ldots, X_N)$*, we consider the trajectories of the point* $(\mathbf{X}, Q) \in \mathbb{R}_+^{N+1}$ *driven by the ODEs (8)-(9). In the region* $A = \{\mathbf{x}, q : 0 < q < C\tau\}$*, the state equations simplify to:*

$$\begin{cases}
\dot{X}_i & = \frac{\alpha}{R^2} \Rightarrow X_i = X_i(0) + \frac{\alpha}{R^2}t,\ \ \forall i\\
\dot{Q} & = \sum_{i=1}^{N} X_i - C \Rightarrow\\
Q(t) & = Q(0) + (\sum_{i=1}^{N} X_i(0) - C)t + \frac{N\alpha}{2R^2}t^2
\end{cases} \qquad (12)$$

Clearly, for any $(\mathbf{X}(0), Q(0)) \in A$*, there exists a finite* $t \geq 0$ *such that* $(\mathbf{X}_t, Q_t) \notin A$*. This means that all points* $(\mathbf{X}_t, Q_t) \in A$ *are unstable. The unique equilibrium point* P^**, calculated in Prop.5.1, is outside* A*. For* $(\mathbf{X}, Q) \notin A$*, the state equations become:*

$$\begin{cases}
\dot{X}_i & = \frac{\alpha}{R^2} - \frac{\zeta}{CR\tau}(Q - C\tau)X_i\\
\dot{Q} & = \sum_{i=1}^{N} X_i - C
\end{cases}$$

We now use the technique of the Lyapunov function to show that P^* *is a stable point, i.e., we have to show that there exist a function* V_t *defined in a neighborhood of* P^**, positively defined for* $t \geq 0$*, with orbital derivative negatively semidefinite (in which case, the solution* P^* *is stable in the sense of Lyapunov, see [42] Theorems 8.1-8.3). Outside* A*, we define the Lyapunov function by:*

$$V(\mathbf{X}, Q) = \sum_{i=1}^{N}(X_i - X_i^*) - \log\left(\frac{X_i}{X_i^*}\right) + \frac{\zeta(Q - Q^*)^2}{2RC\tau} \qquad (13)$$

Clearly, $V(P^*) = 0$, $V(\mathbf{X}, Q) \geq 0 \; \forall (\mathbf{X}, Q) \notin A$, and

$$\dot{V}(\mathbf{X}, Q) = \sum_{i=1}^{N} (\dot{X}_i - \dot{X}_i \frac{X_i^*}{X_i}) + \dot{Q} \frac{\zeta(Q - Q^*)}{RC\tau} \tag{14}$$

$$= \sum_{i=1}^{N} \frac{\dot{X}_i}{X_i} (X_i - X_i^*) + (X_i - X_i^*) \left[\frac{\zeta(Q - Q^*)}{RC\tau} \right]$$

$$= \sum_{i=1}^{N} (X_i - X_i^*) \left[\frac{\alpha}{X_i R^2} - \frac{\zeta(Q - C\tau)}{RC\tau} + \frac{\zeta(Q - Q^*)}{RC\tau} \right]$$

$$= \sum_{i=1}^{N} (X_i - X_i^*) \left[\frac{\alpha}{X_i R^2} - \frac{N\alpha}{CR^2} \right]$$

$$= \frac{\alpha}{R^2} \sum_{i=1}^{N} (X_i - X_i^*) \left[\frac{1}{X_i} - \frac{1}{X_i^*} \right] = -\frac{\alpha}{R^2} \sum_{i=1}^{N} \frac{(X_i - X_i^*)^2}{X_i X_i^*}. \tag{15}$$

Therefore $\dot{V}(\mathbf{X}, Q)$ is negatively semidefinite for any ball including the equilibrium point P^*. This proves that P^* is an equilibrium globally stable as per [42].

Once we know that the system has a unique globally stable equilibrium, we want to show what the *convergence rate* of the system in a neighborhood of the equilibrium is. This can easily be evaluated considering *local stability* properties of the system.

Proposition 5.5 *The system of ODEs (8)-(9) is locally stable in the equilibrium $P^* = (X_1^*, \ldots, X_N^*, Q^*)$*

Proof 5.6 *We write $(\dot{X}_1, \ldots, \dot{X}_N, \dot{Q}) = (f_1, \ldots, f_N, g)$, for $X_i > 0$, $Q > 0$ where f_i and g are defined as follows:*

$$\begin{cases} f_i(X, Q) = & \frac{\alpha}{R^2} - \frac{\zeta}{C\tau R}(Q - C\tau) X_i \mathbb{1}_{Q(t) \geq C\tau} i = 1, ..., N \\ g(X, Q) = & \sum_{i=1}^{N} X_i - C \end{cases} \tag{16}$$

Linearizing the system of ODEs in P^*, and defining $\Delta X_i = X_i - X_i^*$, $\Delta Q = Q - Q^*$, and $\mathbf{Y} = (\Delta X_1, \ldots, \Delta X_N, \Delta Q)$ we obtain $(\tilde{f}_1, \ldots, \tilde{f}_N, \tilde{g}) = \dot{\mathbf{Y}} = A\mathbf{Y}$ where A is a $(N+1) \times (N+1)$ square real matrix defined as follows:

$$A = \begin{pmatrix} -\frac{\alpha}{CR^2} & 0 & \cdots & 0 & -\frac{\zeta}{C\tau R} \\ 0 & -\frac{\alpha}{CR^2} & \cdots & 0 & -\frac{\zeta}{C\tau R} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & \cdots & 1 & 0 \end{pmatrix}$$

The characteristic polynomial is then:

$$\left(\lambda + \frac{\alpha}{CR^2}\right)^{N-1} \left(\lambda^2 + \frac{\alpha}{CR^2}\lambda + N\frac{\zeta}{C\tau R}\right)$$

whose roots have all real part negative.

Proposition 5.7 *The solution of the system of ODEs (8)-(9) converges to the global stable equilibrium P^* at a rate $e^{-\Theta t}$ with,*

$$\Theta = \frac{\alpha}{CR^2}\left(\frac{1+\mathbb{1}_{\zeta\leq\zeta^*}\sqrt{1-\zeta/\zeta^*}}{2}\right)$$

and $\zeta^* = \frac{\alpha^2\tau}{4NCR^3}$.

Proof 5.8 *We calculate the dominant eigenvalue of the matrix A, i.e., the eigenvalue with the real part with the smallest absolute value.*

To conclude, we summarize our main findings in the following observation, expressing the results in terms of the expected performance of fLEDBAT.

Observation 5.9 *Prop. 5.1,5.3,5.5,5.7 prove that the designed protocol is efficient ($X^* = C$), and long term fair ($X_i^* = C/N$).*
In addition the queuing delay ($Q^*/C = \tau + \frac{N\alpha\tau}{C\zeta R}$) attains the target τ by an error of $\frac{N\alpha\tau}{C\zeta R}$.

Thus, our initial goals of an *efficient* and *fair* protocol are met. Clearly, a number of issues need further investigation (i.e., how the protocol performs in practice where not all modeling assumptions hold, what is the impact of parameters and of packet-level dynamics, how it performs against TCP, etc.) that we investigate in the next section by means of a thorough simulation campaign in a number of different scenarios.

6. Simulation overview

So far we have developed a mathematical model of our new proposed protocol in order to formally prove its properties. However, the model is based on a number of simplifying assumptions and it neglects some aspects due to packet-level quantization (i.e., queue length and congestion window in multiple of fixed-size packets as opposed to continuous rate in the fluid model). Hence, in the remainder of this work we carry out a thorough packet-level `ns2` [2] simulation campaign, to cope with scenarios where such assumptions do not hold. Our implementation is available as open-source at [1]: besides, we point out that our code can be used as a `ns2` module in simulation, or as a Linux kernel module for experimental studies[9].

Unless otherwise stated, we consider a reference scenario consisting of a bottleneck link of capacity $C = 10$ Mbps and buffer size $B = 100$ packets (about 4 times the LEDBAT target). For the sake of simplicity, we consider fixed packet size equal to $P = 1500$ Bytes. Data flows in a single direction, and ACKs are not delayed, dropped nor affected by cross-traffic on their return path (except in the P2P scenarios reported

[9] Since the LEDBAT module is implemented using *integer arithmetic* only, it can be run as a kernel module; instead, as fLEDBAT employs *floating-point arithmetic*, for the time being it can only be used as a `ns2` module.

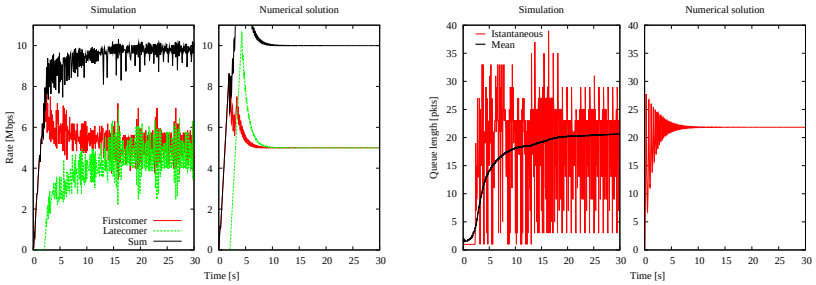

Figure 2: Comparison of (left) simulation and (right) numerical solution for (a) Rates and (b) Queue length. The similar average values in left and right plots confirm a good fit between packet level simulation results and flow level numerical results.

in Sec. 9.2). All flows have the same round trip time $RTT = 50$ ms, half of which is due to the propagation and transmission delay components of the bottleneck link (i.e., a one-way base delay of 25 ms), to which we add a jittering component uniformly distributed in [0,1] ms to avoid synchronization issues. We defer the study of more realistic scenarios, including heterogeneous delays, different access technologies, background traffic and P2P-like traffic models to Sec. 9. As far as TCP flows are concerned, we select the NewReno flavor, enabling the selective acknowledgement SACK option due to its growing widespread use [30]. By selecting the NewReno flavor, we gather conservative results since we expect more recent TCP variants implemented by default in Linux and Windows (respectively Cubic [35] and Compound [38]) operating systems to be more aggressive than traditional NewReno flows. Each simulation point reported in the following is the results of 10 simulation runs, over which we gather the average and standard deviation of the metrics under study.

However, we still need to provide evidence of the fluid model accuracy; we do so by comparing the numerical solution of the fluid model with `ns2` simulation results. We consider the simple network scenario described above, and two fLEDBAT flows with the same target [10] $\tau = 25$ ms. For the time being, we fix the decrease component by setting $\zeta = 0.1$ (and explore the impact of ζ later on). To recreate the conditions for the latecomer unfairness phenomenon, the two flows do not start at the same time, but their start times are separated by a gap (2 seconds in the figure). The system state over time is depicted in Fig. 2, which reports both the flow rates X_1, X_2 (top) and the buffer occupancy Q_t (bottom) gathered either by numerical solution (right) or `ns2` simulation (left). As a general comment, the numerical solution shows a good agreement with the simulation results (although as expected packet-level dynamic exhibits much wider fluctuations, while the fluid model gives an average behavior). Indeed, notice that the

[10] Notice that delay target τ was initially set to 25 ms in the IETF draft and to 100 ms in the BEP-29 specification. However, as shown in [7], provided that all flows have the *same target value*, which is furthermore set to a value *not exceeding the buffer size*, the actual value of τ is not critical. Hence, results shown in the following are valid for both target settings.

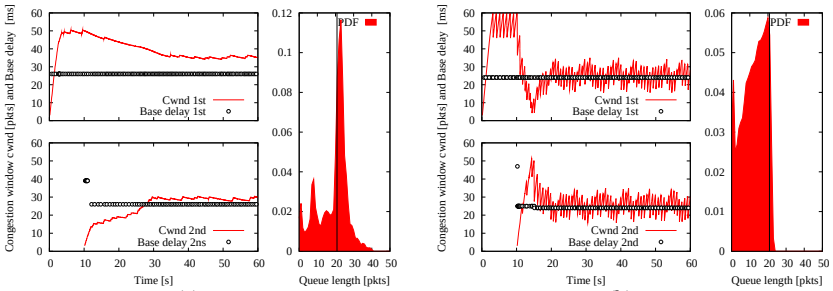

Figure 3: Time evolution of fLEDBAT dynamics with (a) chunk-based $\zeta = 0.01$ and (b) backlogged $\zeta = 5$ traffic models.

average of queue occupancy and flow rates yielded by the fluid system closely matches the simulation dynamics (for the sake of readability, we plot the *moving average* of the queue length gathered via simulation alongside the instantaneous occupancy). As expected, both numerical and simulation results show that the capacity is, after an initial transient phase, fairly shared among flows (i.e., $\overline{X}_i \approx C/2, \forall i$) and that furthermore the queuing delay target is reached (i.e., $\overline{Q}_t \approx C\tau$).

In the following we study several aspects of the LEDBAT protocol family, that we separately report to better isolate their effect. Sec. 7 addresses the impact of different *traffic models* on protocol performance, considering backlogged vs chunk-by-chunk transfers. Then, Sec. 8 addresses a *sensitivity analysis* of the protocol to ζ parameter variation. Finally, Sec. 9 compares LEDBAT and fLEDBAT under heterogeneous P2P-like scenarios.

7. Impact of traffic model

In this section we assess how the fLEDBAT protocol deals with different kinds of traffic. Besides the classical backlogged transfer, we simulate a chunk-based transfer, which mimics the behavior of a BitTorrent data exchange between two peers.

7.1. Chunk-by-chunk transfer

In this scenario, we consider sources that continuously transmit chunks of data, where each chunk has the typical BitTorrent size of $250\,\text{kB}$ (nearly 170 full payload packets). As soon as a chunk transmission ends (i.e., when the last acknowledgment for that chunk has been received at the sender side), a new chunk transmission is scheduled with the same peer. This traffic model, which emulates the dynamics of P2P traffic exchange, differs from backlogged transfers in that, after the last data packet of a chunk has been sent, the source peer stops transmitting for about RTT seconds until the matching acknowledgement is received, and a new chunk transmission can start (i.e., chunks transmission is *not* pipelined). Notice also that we keep the congestion window parameter across chunks (i.e., the congestion window is *not* reset between subsequent chunks exchanged with the same peer).

Fig. 3-(a) reports the time evolution of the system dynamics when $\zeta = 0.01$: in the left portion, congestion window and base delay estimation of the firstcomer (top) and latecomer (bottom) flows are reported, while the right portion shows the distribution of the queue length. We can see that, despite the latecomer initially has an incorrect view of the base delay (as in LEDBAT), the multiplicative decrease phase of the firstcomer allows the latter to correct its estimate, after which the performance share converges to an equitable state. Due to (i) the continuous adjustment of AI and MD dynamics and (ii) the fact that chunk transmission seldom pauses the transmission, the queue is no longer stable as for the standard LEDBAT case [34], but fluctuates around the occupancy value predicted by the model (represented by a solid vertical line).

7.2. Backlogged transfer

In the case of backlogged transmission, a latecomer phenomenon may still arise depending on the value of ζ: indeed, when ζ is too small, the multiplicative decrease component of the first flow is slower than the additive increase of the latecomer, which is thus unable to correct its wrong estimation. However, provided that ζ is large enough to let the queue flush, fLEDBAT can still reintroduce fairness.

Results for the backlogged scenario are reported in Fig. 3-(b) for $\zeta = 5$. Especially, the queue now seldom flushes, as it can be seen by the increased probability to have a null queue length shown by the PDF. In turn, this helps latecomers gain a correct view of the base delay. In this case though, the model slightly overestimates the queue size: indeed, due to larger ζ values, the congestion window fluctuations are now wider. Also, as the queue flushes, the protocol is less efficient with respect to the previous cases, because the capacity is not fully utilized all the time.

We point out that, since fLEDBAT is designed to be a low-priority protocol, slight inefficiency can be tolerable if they reintroduce correctness of operation in general settings –especially if otherwise this could have serious consequences, as in the case of a latecomer long-lived backup starving the firstcomer.

8. Sensitivity analysis

In this section we carry out further simulations to assess the impact of the choice of the parameter ζ on the protocol performance. In order to gather a complete sensitivity analysis of fLEDBAT parameters, we consider several scenarios: (i) a TCP NewReno flow competing with a fLEDBAT flow, (ii) a LEDBAT flow competing with a fLEDBAT flow, (iii) two or more fLEDBAT flows competing at the same bottleneck. All flows operate in chunk-by-chunk transmission mode.

As performance metrics, we consider *fairness*, *efficiency* and *protocol breakdown* of the data transfer. Specifically, we use Jain fairness index F, which is defined as $F = (\sum_{i=1}^{N} x_i)^2/(N \cdot \sum_{i=1}^{N} x_i^2)$ where x_i is the rate of the i-th flow. Fairness tops to $F = 1$ when bandwidth is perfectly shared among all flows, while it drops to a minimum of $F = 1/N$ when one flow monopolizes the bottleneck leaving the others $N - 1$ in starvation. Regarding the efficiency, we consider the link utilization metric η defined as the ratio of the overall throughput (including headers) over the link capacity C, i.e., $\eta = \sum_{i=1}^{N} x_i/C$. For the sake of illustration, we also consider the *protocol*

breakdown, defined as the percentage of traffic sent by fLEDBAT sources over the total traffic, which immediately conveys the level of low priority of fLEDBAT with respect to other protocols competing at the same bottleneck.

8.1. Observations on α, τ and low-priority level

Since a careful sensitivity analysis focused on gain α and target τ has already been carried out in [7], in the following we briefly summarize the main lessons as far as these two parameters are concerned, while we provide a thorough set of simulation results for the newly introduced parameter, i.e., the decrease factor ζ.

Let us consider the target parameter τ first. Already in the homogeneous case of several flow with equal settings, from [7] we gather that the performance of LEDBAT can not be easily controlled by tuning the target τ. Indeed, the low priority level can be changed only when the $C\tau$ product approaches the buffer size – however changes in the priority level are too steep for very small variations of τ. Moreover, there is no single value of τ that can adapt to both low-capacity and high-capacity links at the same time: e.g., a queuing delay target of 100 ms translates into a 12.5 KB buffer at 1 Mbps, but it requires an amount of buffer that may not be available at higher capacities (e.g., 12.5 MB at 1 Gbps). Finally, in the heterogeneous case of several flows with different settings, even a small difference between values of τ yield to extremely unfair situations, with flows having larger τ being more aggressive. For this reason we do not consider τ as a free parameter.

Let us now consider the gain parameter α: in this case, it is worth noting that the increase component of fLEDBAT differs from the one of LEDBAT. Indeed, in both protocols the growth is proportional to α, but while in LEDBAT the increase is also proportional to the offset from the target (which means the congestion window growth slows down when the estimated queuing delay approaches the target value), in fLEDBAT the growth is unrelated to the delay offset. Therefore, the value of $\alpha = 1$ is constrained in reason of the low-priority goal (so to match the 1-packet-per-RTT TCP growth in congestion avoidance).

Finally, due to the bounded target, fLEDBAT inherits from LEDBAT the lowest possible level of priority [7] compared to NICE and to TCP-LP (which we cannot elaborate further due to space constraints).

8.2. fLEDBAT vs TCP

Fig. 4(a) shows the efficiency and fairness performance when a single TCP and a single fLEDBAT flow share the bottleneck; first of all, we can see that low-priority goal is met, as TCP is enjoying the largest portion of the capacity (fLEDBAT breakdown goes to 0% and fairness drops to $1/N$). As expected, efficiency is high: as we already observed in [34] for LEDBAT, fLEDBAT is still able to push some bytes on the link, thereby increasing the overall link utilization with respect to the case where a single TCP Reno pass through the bottleneck.

With the exception of extremely low values of $\zeta < 10^{-3}$ (which soften the effect of the multiplicative decrease, and sharpen the impact of the Reno-like additive increase), the low-priority goal is therefore satisfied. Thus, selecting ζ is not a concern as far as heterogeneous fLEDBAT vs TCP scenarios are considered.

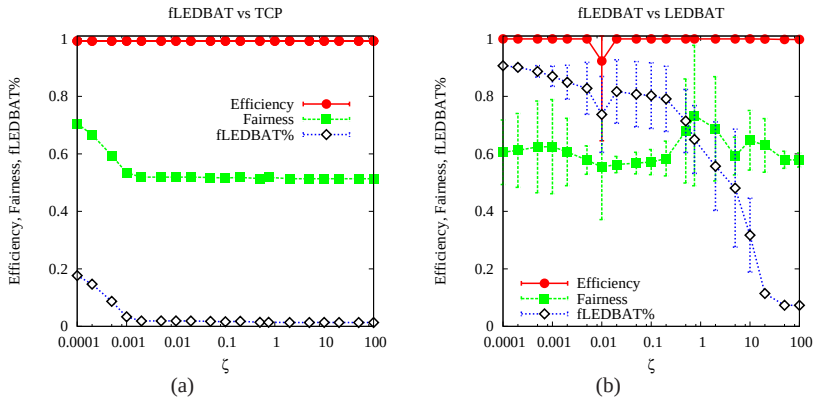

Figure 4: Sensitivity analysis to ζ: Efficiency, long-term fairness and protocol breakdown of a fLEDBAT flow sharing the bottleneck with (a) a TCP flow, (b) a LEDBAT flow.

8.3. fLEDBAT vs LEDBAT

Fig. 4(b) shows the efficiency and fairness performance when a single fLEDBAT flow and a LEDBAT share the bottleneck. We randomize the start time of both flows in the [0,10] sec interval, so that the latecomer can be either of the two protocols. In this case, we infer that fLEDBAT is generally more aggressive (due to the AI dynamic) until ζ grows too large, in which case the reverse happens (due to the MD dynamic). Specifically, less than 20% of the bottleneck is occupied by LEDBAT when $\zeta < 10^{-2}$. For larger values of ζ though, LEDBAT becomes increasingly competitive with fLEDBAT: the crossover happens at about $\zeta = 5$, after which fLEDBAT becomes even lower priority than LEDBAT.

In no case, however, the share is fair and a latecomer phenomenon may still arise. Consider that, when LEDBAT starts first and saturates the bottleneck, it induces a very steady queue. Therefore, when an fLEDBAT latecomer flow arrives on the bottleneck, it measures an incorrect base delay. However, as LEDBAT reacts with a *linear* decrease to the increasing delay, the fLEDBAT latecomer will not have the opportunity to correct its estimate – as it otherwise does whenever the firstcomer flow reacts with a *multiplicative* decrease to the increasing delay. Hence, when ζ is small, the fLEDBAT latecomer can starve the LEDBAT flow.

8.4. fLEDBAT vs fLEDBAT

We finally consider the intra-protocol scenario in which two fLEDBAT flows share the bottleneck. We set the start time of latecomer flow to $t = 10$ s, which was shown in [34] to represent a worst case scenario for the fairness index. Fig. 5(a) reports results for varying ζ, focusing on efficiency and fairness metrics. From the figure, it is clear that fLEDBAT is able to operate fairly and efficiently under a wide range of parameters. Overall, taking into account also the previous remark in the intra-protocol fLEDBAT vs TCP scenario, we have that any value of ζ in the gray shaded zone yields to an efficient, fair and low-priority system.

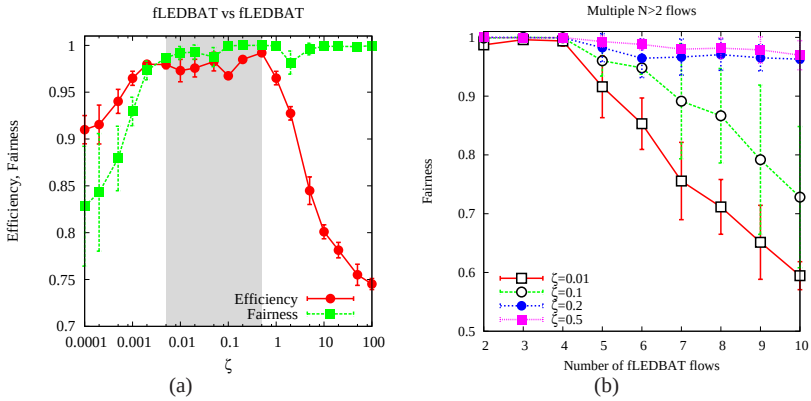

Figure 5: Sensitivity analysis to ζ: (a) Efficiency, long-term fairness and protocol breakdown of two fLEDBAT flows; (b) variation of the intra-fairness index of fLEDBAT, for different values of ζ and a growing number of flows.

8.5. Tuning ζ for multiple-flows scenario.

In this scenario we present results for a varying number of fLEDBAT flows competing at the bottleneck, with in $N \in [2, 10]$. Every k-th flow arrive at $t_k = k10$ s, and we evaluate the performance only after the N-th last flow has arrived at the bottleneck. Results are reported in Fig. 5(b), where we select a few values of $\zeta \in \{0.01, 0.1, 0.2, 0.5\}$ from the shaded gray zone of Fig. 5(a). As can be seen, it is always possible to find a value of ζ that guarantees fairness for the whole set of flows, with any values in the range providing good results for the number of flows that are typically concurrently active in BitTorrent. Moreover, the very same values of ζ that provide fair resource share, were already shown to provide efficient use of the resources for $N = 2$ flows in Fig. 5(a), which clearly holds for $N > 2$ (omitted to avoid cluttering the pictures).

9. P2P Scenarios

In order to compare fLEDBAT vs LEDBAT performance under other conditions, closer to the BitTorrent use-case, we finally consider a chunk-based scenario that (i) loosely mimics the behavior of BitTorrent peers, (ii) employs Internet-like heterogeneous delays and access rates, (iii) considers background traffic and coupled queues, (iv) addresses the impact of chunk sizes.

We consider two different scenarios: first a single BitTorrent peer and a single queue in isolation, using a more sophisticated traffic model and observing the effect of delay heterogeneity alone. Second, we study a BitTorrent swarm-like scenario, as close as possible to the actual target application scenario of LEDBAT, with 100 peers continuously exchanging data among them. In this case, the state of queues is no longer independent, as data traffic mixes with acknowledgement traffic in the reverse direction, causing a coupling of the congestion controllers as well. To gather even more

realistic results, we finally add HTTP-like background traffic to the swarm scenario, studying its impact on congestion controls dynamics.

Before presenting the results of our simulations, it is worth saying a few words on the traffic model we use to simulate a BitTorrent peer. Like a real BitTorrent source [12], our simulated peers open a number of connections to other N peers, but they actively exchange chunks of data with only a restricted number $M < N$ of the available peers at the same time (set to $M = 5$ and $N = 10$). We employed different chunk sizes, ranging from 250 KB - 4096 KB [29], using 250 KB chunks unless otherwise stated. At the end of each chunk transmission, the sender chooses the next destination peer as follows: with a *persistence probability* P_P, the sender will send another chunk to the same peer, keeping the congestion window settings; with probability $(1 - P_P)$, the sender will choose an inactive neighbor at random, resetting in this case the congestion window to 1. A detailed discussion of the meaning of different values of P_P is found later on.

We point out that the goal of this work is not on assessing the impact of LEDBAT on BitTorrent, for which we invite the reader to our previous work [39, 40]. Hence, we chose not to implement all the application-level details of a BitTorrent source (e.g., tit-for-tat, optimistic unchoking, signaling, etc.). Rather, our objective is assessing that fLEDBAT does not worsen flow-level performance with respect to LEDBAT: hence, we argue that this simpler traffic model is a good approximation of the actual peer interaction given our purpose.

As far as network conditions are concerned, we consider both *FTTH symmetric* access capacities ($C = 10$ Mbps, $B = 100$ packets, default unless otherwise stated) and *ADSL asymmetric* capacities ($C = 1$ Mbps uplink, $C = 8$ Mbps downlink $B = 100$ packets). To add further realism, we simulate both a *homogeneous* network setup (i.e., in which all peers have the same propagation delay $RTT = 50$ ms) as well as *heterogeneous* scenario (default unless otherwise stated) where the propagation delay of the access link of each peer is distributed according to realistic delay measurement [43], with mean equal to 37.9 ms.

9.1. Single peer perspective

Let us start by considering a single peer behind a FTTH connection, exchanging 250 KB chunks with peers chosen according to the algorithm described above: when a chunk transfer is completed, the source keeps the same destination peer with a probability P_P, and changes it with probability $(1 - P_P)$. fLEDBAT and LEDBAT are simulated separately, i.e., all peers either use the former or the latter protocol, and we consider both an homogeneous and heterogeneous delay scenario. For fLEDBAT we set the parameter $\zeta = 0.1$, which yielded good performance in the sensitivity analysis.

In our experiments we explore the full range of $P_P \in [0.1]$. As $P_P \to 0$, we expect the performance of the two protocols to be close: indeed, when connections are reset every 170-th packet (which corresponds to 250 KB chunks), the protocols are basically always in transient state and the target is not even likely reached during a chunk transfer. Conversely, differences are expected to arise in the more stable scenarios $P_P \to 1$, where congestion parameters are kept across chunks. Actually, we expect a real BitTorrent source to have a behavior similar to a sender with $P_P \geq 0.8$: in fact, one source normally tries to keep 4 out of the 5 "best" (i.e., higher capacity) peers while at the

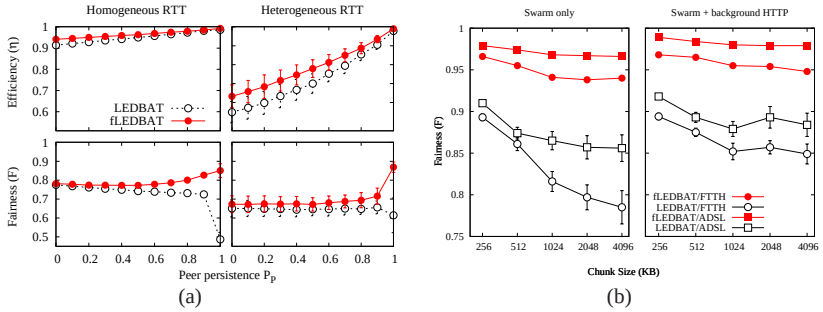

Figure 6: P2P-like scenarions: (a) Efficiency and fairness of fLEDBAT in the homogeneous vs heterogeneous RTT scenarios; (b) Fairness in FTTH and ADSL swarm-like scenarios, with and without background HTTP traffic

same time continuously discovering new, potentially "better", peers (i.e., by means of optimistic unchoking).

Results of the comparison are reported, in term of efficiency and fairness, in Fig. 6(a). Since flows are no longer backlogged, from now on we consider *short-term fairness*, measured at 1 Hz rate (i.e., corresponding to a good tradeoff between a minimum number of RTTs to have statistically meaningful results, and a maximum time lag, as flows may die out after a single chunk transfer). Notice that we expect short-term fairness to be harder to achieve than the long-term fairness considered in the previous sections (hence we expect lower F values).

At first glance, we remark that under all scenarios and P_P values, fLEDBAT is more efficient (due to AI) and fair (due to MD) with respect to LEDBAT, and the gain is more evident exactly in the operational range of BitTorrent (i.e., $P_P \geq 0.8$). In the homogeneous case depicted in the two plots on the left, as expected, the fairness gap exacerbates as $P_P \to 1$: in this case, fLEDBAT ability to correctly measure the base delay leads to an increase of the fairness metric. On the contrary, LEDBAT fairness decreases as P_P grows, due to the latecomer issue: the effect is stronger when $P_P = 1$, as in this case the unfair situation persists through the whole duration of the experiment and leads to a consistent drop of F. Similar considerations hold for the heterogeneous case (see plots on the right), although in this case the heterogeneous delays introduce RTT unfairness in both fLEDBAT and LEDBAT, reducing the absolute value of F (i.e., we do not attempt to account for the RTT bias in the fairness definition). However, RTT unfairness does not translate into serious issues (such as long time starvation), and fLEDBAT always guarantees a fairness higher than LEDBAT (as latecomer advantage disappears).

As far as efficiency η is concerned, when the congestion window parameters are reset at every chunk transmission ($P_P = 0$), the link capacity is not fully utilized even in the homogeneous case. The heterogeneous case further adds inefficiency, as flows with higher RTT increase their congestion window more slowly, hence further wasting link capacity. However, it is worth pointing out that the additive increase component of fLEDBAT makes it more efficient than LEDBAT under any circumstance, while the

multiplicative decrease component guarantees at the same time its lower-priority with respect to TCP.

9.2. Entire swarm perspective

We now consider an entire swarm, where 100 peers continuously exchange data among them using 5 parallel upload slots, as in BitTorrent: hence, the uplink queue of each peer contains a mix of (i) data packets being uploaded to the swarm and (ii) acknowledgement packets related to data being downloaded from the swarm. As this happens for each queue, P2P traffic interacts both in the forward and backward directions. Notice also that the end-to-end congestion controls of data transfers are tightly coupled in a non-trivial way, as in our model each peer maintains multiple connections to a set of other peers, which moreover dynamically evolve over time. We do not try to model all BitTorrent dynamics (e.g., chunk trading logic), but rather assumes that peers are always able to find content where they seek it (i.e., a large file where there is enough chunk diversity in the system). Therefore, we do not attempt at measuring application-level statistics, such as the torrent download time, but rather focus on the transport-layer fairness statistics.

Unlike the previous scenario, here we fix the persistence probability to $P_P = 0.8$. As shown in the former experiment, each interruption in data transmission favors LEDBAT for the consequent draining of the queue lets the protocol correct the base delay estimation. Under this consideration, since BitTorrent optimistic unchoking happens on 1 slot out of 5 at low rate (each 30 seconds), $P = 0.8$ corresponds to a lower bound (since the chunk transfer can be much shorter than 30 seconds) and gives an optimistic (thus, conservative) estimate for LEDBAT fairness. Moreover, our simulation model also introduces an *idle RTT* (i.e., time elapsed between the transmission of the last data packet of a chunk and the reception of its acknowledgment) before a new chunk is sent to a persistent peer, that actual BitTorrent systems avoid by means of request pipelining and that thus further simplifies the LEDBAT task. However, we can partly compensate for this effects by employing larger chunks.

Since homogeneous and heterogeneous RTT scenarios yielded qualitatively similar results, in the following we only consider heterogeneous delay settings for more realism, with either FTTH or ADSL access. Finally, we also simulate a scenario where peers browse the Web while participating in the swarm: in particular we add an HTTP source from which peers download Web-pages (files with a size uniformly distributed in the range $[0 - 512]$ KB) using traditional TCP. There are always 25 peers, selected at random, with active HTTP downloads, so that a quarter of the total swarm is always involved in short interactive background Web transfers.

Short-term fairness results are shown in Fig. 6(b): first, notice that the coupling of queues is more beneficial for the fairness index (for both LEDBAT and fLEDBAT) with respect to the single queue scenario shown in Fig. 6(a). At the same time, in all cases fLEDBAT consistently achieves higher fairness than LEDBAT. Larger chunk sizes, as expected, badly affect LEDBAT because they reduce the number of transmission interruptions (which helped in emptying the queues). Conversely, fLEDBAT appears almost insensitive to chunk size.

Comparing ADSL with FTTH scenarios, we see that in the latter case a further cause of unfairness may arise: indeed, as capacities are symmetric and one peer may

have several downloads ongoing (only upload slots are limited in number), downlink can become a bottleneck as well (e.g., in real systems this may happen in case of popular torrents with many seeds). The impact of background HTTP traffic is instead beneficial to the fairness of both fLEDBAT and (especially) LEDBAT: this is due to the fact that peers downloading HTTP data will send out a burst of acknowledgement packets, that possibly cause buffer overflows in the uplink queues (which assist in raising fairness in LEDBAT, as shown in [34]). Background traffic has instead a smaller impact on fLEDBAT, as the protocol achieves higher fairness without external help.

10. Conclusion

In this work, we proposed modifications to the LEDBAT congestion control algorithm that not only are able to achieve *low-priority inter-protocol* (i.e., against TCP) and *efficiency intra-protocol* (e.g., with other fLEDBAT flows), but also reintroduce *intra-protocol fairness*, solving thus the late-comer issues of the original LEDBAT proposal.

To model the protocols dynamic we used a fluid-model approach which allows, on the one hand, to detect the main issue at the base of LEDBAT unfairness (i.e., the additive decrease component) and on the other hand, to prove the correctness of fLEDBAT design. We also derived closed-form expressions for the average rate and queue length in the general case with N sources. Furthermore, by means of packet-level simulations, we further assess that fLEDBAT can safely operate under a number of scenarios (such as chunk-by-chunk and backlogged transmission) as it is not sensitive to parameter selection and operates reasonably well under a wide range of parameters. Finally, we tested the protocol in multiple-flows P2P-like realistic scenarios with heterogeneous RTTs and transfer emulating the operations of a BitTorrent peer, the original target application for LEDBAT.

Results show that fLEDBAT not only solves the fairness issue for backlogged flows, but maintains the same good properties of LEDBAT– that is, it yields to TCP while exploiting the spare bandwidth. Overall, we see that our proposed modifications lead to a demonstrable improvement in performance with respect to LEDBAT, in terms of both fairness and efficiency, especially for the case of backlogged connections. At the same time, our simulations confirm the robustness of fLEDBAT even under realistic heterogeneous network conditions on which BitTorrent can be expected to operate.

Apart from the intra-protocol unfairness, which was solved in the general case this work, in our opinion there is still a critical point in the LEDBAT algorithm definition that remains an open issue. This issue, described in [7, 36] and debated in the LEDBAT IETF working group mailing list, concerns the use of a *fixed* queuing delay target. Such fixed settings (which are referred to as “magic numbers” in the mailing list) are indeed not a good practice, as they may lead to undesirable behavior: as a matter of fact, not compliant implementation may set a higher target with respect to the mandatory standard values (i.e., malicious backlogged users can hence easily obtaining an unfair advantage over compliant users). Furthermore, a fixed queuing delay target is not scaling with the link capacity: i.e., while 100 ms queuing may be reasonable for today’s ADSL modem (with large buffer size relatively to their narrow uplink capacities), this will not scale when high capacity Fiber-to-the-Home (FTTH) access

will be ubiquitous. However, while fixed settings are not robust against malicious or misconfigured implementations, at the same time there is no obvious way of defining an *adaptive* target without loosing a bounded guarantees on the additional delay, that interactive applications would like to keep as small as possible.

References

[1] LEDBAT `ns2` code. http://perso.telecom-paristech.fr/ valenti/pmwiki/pmwiki.php?n=Main.LEDBAT.

[2] The Network Simulator `ns2`. http://www.isi.edu/nsnam/ns/.

[3] Cisco visual networking index: Forecast and methodology, 2008-2013. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/-ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html, June 2009.

[4] A.J. Abu and S. Gordon. A Dynamic Algorithm for Stabilising LEDBAT Congestion Window. In *2nd IEEE International Conference on Computer and Network Technology (ICCNT 2010)*, Bangkok, Thailand, Apr 2010.

[5] Catherine Boutremans and Le Boudec. Jean-Yves. A note on the fairness of TCP vegas. In *International Zurich Seminar on Broadband Communications*, pages 163–170, 2000.

[6] L.S. Brakmo, S.W. O'Malley, and L.L. Peterson. TCP Vegas: new techniques for congestion detection and avoidance. *ACM SIGCOMM Comp. Comm. Rev.*, 24(4):24–35, 1994.

[7] G. Carofiglio, L. Muscariello, D. Rossi, and C. Testa. A hands-on Assessment of Transport Protocols with Lower than Best Effort Priority. In *35th IEEE Local Computer Network (LCN 2010)*, Denver, CO, Oct 2010.

[8] G. Carofiglio, L. Muscariello, D. Rossi, and S. Valenti. The quest for LEDBAT fairness. In *IEEE Global Communication (GLOBECOM 2010)*, Miami, FL, Dec 2010.

[9] Vint Cerf, Van Jacobson, Nick Weaver, and Jim Gettys. Bufferbloat: what's wrong with the internet? *Commun. ACM*, 55(2):40–47, Feb 2012.

[10] Stuart Cheshire. It's the latency, stupid! `http://rescomp.stanford.edu/~cheshire/rants/Latency.html`, May 1996.

[11] D.M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN systems*, 17(1):1–14, 1989.

[12] B. Cohen. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer systems*, Cambridge, MA, Jun 2003.

[13] B. Cohen and A. Norberg. Correcting for clock drift in uTP and LEDBAT. In *Invited talk at 9th USENIX International Workshop on Peer-to-Peer Systems (IPTPS 2010)*, San Jose, CA, Apr 2010.

[14] A. Finamore, M. Mellia, M. Meo, M. Munafo, and D. Rossi. Experiences of internet traffic monitoring with tstat. *IEEE Network Magazine, Special Issue on Network Traffic Monitoring and Analysis*, May 2011.

[15] Franich, D. Megaupload (and megavideo) shut down by the feds. http://popwatch.ew.com/2012/01/19/megaupload-megavideo-shut-down-fbi/, 19 Jan 2012.

[16] Qing Gao and Qinghe Yin. Adaptive vegas: A solution of unfairness problem for tcp vegas. In Cheeha Kim, editor, *Information Networking. Convergence in Broadband and Mobile Networking*, volume 3391 of *Lecture Notes in Computer Science*, pages 132–141. Springer Berlin / Heidelberg, 2005.

[17] Garcia-Dorado, J. and Finamore, A. and Mellia, M. and Meo, M. and Munafo', M. Characterization of isp traffic: Trends, user habits, and access technology impact. *IEEE Transactions on Network and Service Management, to appear*.

[18] Go Hasegawa, Masayuki Murata, and Hideo Miyahara. Fairness and stability of congestion control mechanisms of tcp. *Telecommunication Systems*, 15:167–184, 2000.

[19] G. Hazel. uTorrent Transport Protocol library. http://github.com/bittorrent/libutp, May 2010.

[20] S. Hemminger et al. Network emulation with NetEm. In *Linux Conf Australia (LCA 2005)*, Canberra, Australia, Apr 2005.

[21] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM Comp. Comm. Rev.*, Stanford, CA, Aug 1988.

[22] R. Jain and K.K. Ramakrishnan. Congestion avoidance in computer networks with a connectionless network layer: concepts, goals and methodology. In *Computer Networking Symposium, 1988., Proceedings of the*, Apr 1988.

[23] P. Key, L. Massoulié, and B. Wang. Emulating low-priority transport at the application layer: a background transfer service. In *ACM SIGMETRICS*, New York City, NY, Jun 2004.

[24] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: Illuminating the edge network. In *ACM IMC*, pages 246–259, 2010.

[25] A. Kuzmanovic and E.W. Knightly. TCP-LP: low-priority service via end-point congestion control. *IEEE/ACM Transactions on Networking (TON)*, 14(4):752, 2006.

[26] D.J. Leith, R.N. Shorten, G. McCullagh, L. Dunn, and F. Baker. Making Available Base-RTT for Use in Congestion Control Applications. *IEEE Communications Letters*, 12:429–431, Jun 2008.

[27] S. Liu, T. Basar, and R. Srikant. TCP-Illinois: A loss-and delay-based congestion control algorithm for high-speed networks. *ACM Performance Evaluation*, 65(6-7):417–440, 2008.

[28] S. Liu, M. Vojnovic, and D. Gunawardena. 4cp: Competitive and considerate congestion control protocol. In *ACM SIGCOMM*, Pisa, Italy, Sep 2006.

[29] Pawel Marciniak, Nikitas Liogkas, Arnaud Legout, and Eddie Kohler. Small Is Not Always Beautiful. In *IPTPS'2008*, Tampa Bay, Florida United States, 2008.

[30] M. Mellia, M. Meo, L. Muscariello, and D. Rossi. Passive analysis of tcp anomalies. *Elsevier Computer Networks*, 52(14), October 2008.

[31] Jeonghoon Mo, Richard J. La, Venkat Anantharam, and Jean Walrand. Analysis and comparison of tcp reno and vegas. In *In Proceedings of IEEE Infocom*, New York, NY, USA, Mar 1999.

[32] A. Norberg. BitTorrent Enhancement Proposals on uTorrent transport protocol. http://www.bittorrent.org/beps/bep_0029.html, 2009.

[33] D. Rossi, C. Testa, and S. Valenti. Yes, we LEDBAT: Playing with the new BitTorrent congestion control algorithm. In *Passive and Active Measurement (PAM 2010)*, Zurich, Switzerland, Apr 2010.

[34] D. Rossi, C. Testa, S. Valenti, and L. Muscariello. LEDBAT: the new BitTorrent congestion control protocol. In *19th IEEE International Conference on Computer Communications and Networks (ICCCN 2010)*, Zurich, Switzerland, Aug 2010.

[35] I. S. Ha, Rhee and L. Xu. CUBIC: A new TCP-friendly high-speed TCP variant. In *ACM SIGOPS Operating System Review*, New York, NY, Jul 2008.

[36] Joscha Schneider, Joerg Wagner, Rolf Winter, and Hans-Joerg Kolbe. Out of my way – evaluating low extra delay background transport in an adsl access network. In *22nd International Teletraffic Congress (ITC22)*, 2010.

[37] S Shalunov. Low Extra Delay Background Transport (LEDBAT). IETF Draft, Mar 2010.

[38] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A compound TCP approach for high-speed and long distance networks. In *25th IEEE Conference on Computer Communications (INFOCOM 2006)*, Barcelona, Spain, Apr 2006.

[39] C. Testa and D. Rossi. The impact of utp on bittorrent completion time. In *IEEE Peer to Peer (P2P'11)*, Kyoto, Japan, September 2011.

[40] C. Testa, D. Rossi, A. Rao, and A. Legout. Experimental assessment of bittorrent completion time in heterogeneous tcp/utp swarms. In *Traffic Measurement and Analysis (TMA) Workshop at Passive and Act ive Measurement (PAM)*, Wien, AU, March 12-14 2012.

[41] A. Venkataramani, R. Kokku, and M. Dahlin. TCP Nice: A mechanism for background transfers. In *8th USENIX Symposium on Operating Systems Design and Implementation (OSDI'02)*, Boston, MA, Dec 2002.

[42] Ferdinand Verhulst. *Nonlinear differential equations and dynamical systems*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[43] B. Wong, A. Slivkins, and E.G. Sirer. Meridian: A lightweight network location service without virtual coordinates. *ACM SIGCOMM Comp. Comm. Rev.*, 35(4):96, 2005.

[44] Yong Xia, David Harrison, Shivkumar Kalyanaraman, Kishore Ramachandran, and Arvind Venkatesan. Accumulation-based congestion control. *IEEE/ACM Trans. Netw.*, 13:69–80, February 2005.