A hands-on Assessment of Transport Protocols with Lower than Best Effort Priority*

Giovanna Carofiglio [ø], Luca Muscariello [†], Dario Rossi [‡], Claudio Testa [‡]
[ø] Bell Labs, Alcatel-Lucent, France, giovanna.carofiglio@alcatel-lucent.com
[†] Orange Labs, France, luca.muscariello@orange-ftgroup.com
[‡] Telecom ParisTech, France, firstname.lastname@enst.fr

*This is the author version of the paper accepted to be presented to the 35th IEEE Conference on Local Computer Networks (LCN'10) in Denver, CO, on October 11-14, 2010.

***Abstract*—Last year, the official BitTorrent client switched to LEDBAT, a new congestion control algorithm targeting a lower-than Best Effort transport service. In this paper, we study this new protocol through packet-level simulations, with a special focus on a performance comparison with other lower-than Best Effort protocols such as TCP-LP and TCP-NICE: our aim is indeed to quantify and relatively weight the level of Low-priority provided by such protocols.**

Our results show that LEDBAT transport generally achieves the lowest possible level of priority, with the default configurations of TCP-NICE and TCP-LP representing increasing levels of aggressiveness. In addition, we perform a careful sensitivity analysis of LEDBAT performance, by tuning its main parameters in both an inter-protocol (against TCP) and intra-protocol (against LEDBAT itself) scenarios. In the inter-protocol case, even in case of misconfiguration LEDBAT competes as aggressively as TCP, but we show that it is not possible to achieve an arbitrary level of low-priority by merely tuning its parameters. In the intra-protocol case, we show that coexistence of legacy flows with slightly dissimilar settings, or experiencing different network conditions, can result in significant unfairness.

I. Introduction

BitTorrent, undoubtedly one of the most successful P2P file-sharing applications nowadays, has recently adopted a new closed-loop congestion control algorithm, namely Low Extra Delay Background Transport (LEDBAT) [1], which is implemented at the application-layer and exploits UDP at the transport layer. The aim of the new protocol is "to not disrupt Internet connections, while still utilizing the unused bandwidth fully" [2] or in other words to delivery data with a lower priority in respect to general Best Effort, and thus TCP, traffic.

Lower than Best-Effort (LBE) priority is achieved in LEDBAT by reacting earlier than TCP to congestion notification, and reducing its transmission rate so to avoid harming TCP traffic: while TCP Reno infers congestion from packet losses, LEDBAT infers congestion from increasing buffering delay, hence prior than losses occur.

Thus, a first important contribution of LEDBAT is that it constitutes a relief for operators, as they no longer need to throttle the now gentle P2P traffic [3]. Moreover it relieves self-induced congestion when the bottleneck is placed at the user access link (e.g., DSL or cable). Self-induced congestion arises in this case when users run several applications having different QoS constraints in parallel (e.g., Web browsing, gaming, VoIP, file-sharing, backup): as the bottleneck is at the access, users are themselves generating competing traffic, but at the same time they would likely not want large background transfer to interfere with foreground interactive applications. In this context, LBE is a promising end-to-end technique that does not require coordination among applications, nor complex queuing policies or IP table rules to be setup by the end-users on their own PC [4].

However, many other services beside P2P file-sharing may successfully exploit a LBE transport protocol e.g., the class of high volume data exchange, data mirroring and pre-fetching, network backups, etc. As such, LEDBAT is not the sole example of LBE transport that has been proposed in the literature: other notable protocols are for instance TCP-LP [5], TCP-NICE [6], 4CP [7] and Microsoft BITS [8]. Despite the relevance of the above scenario, to the best of our knowledge no comparison attempt has been made yet between the different low-priority protocols: this situation is unlike the high-speed data transfer scenario, where several works [9], [10], [11] that compare different flavors of TCP exists, showing their relative merits and disadvantages.

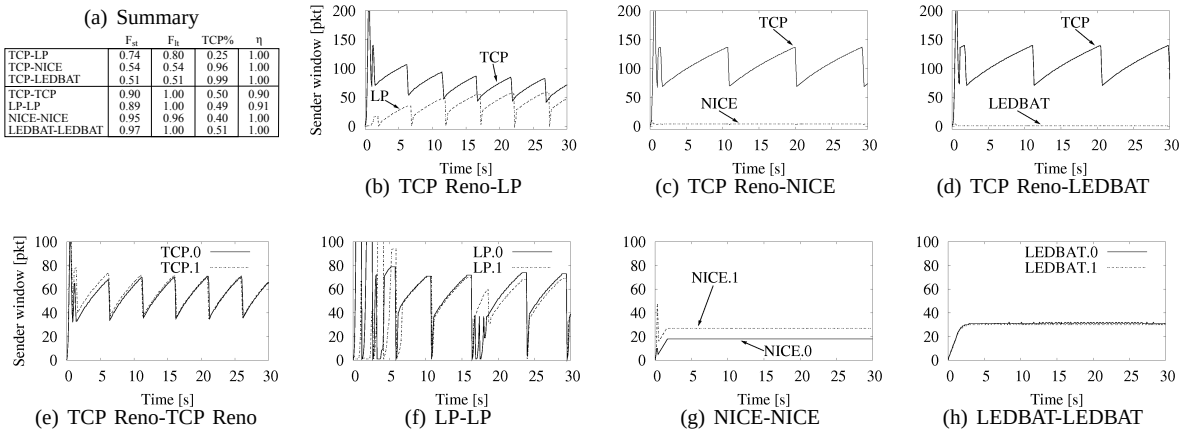
In this work, we carry out a comparison of LBE protocols by means of `ns2` simulations aiming at quantifying and ranking the relative level of priority. Thanks to a systematic evaluation of the fairness and the efficiency, we investigate on three different LBE protocols, namely the new BitTorrent protocol LEDBAT [1], LP [5] and NICE [6]. Notice that only LP implementation is available as open source, so we implement both NICE and LEDBAT in the simulator, and made it available at [12]. As a scenario for the comparison, we consider the typical situation with many concurrent P2P flows sharing an access bottleneck link with other higher-priority traffic. Our results show that (i) LEDBAT transport achieves the lowest level of priority, while NICE and LP represent increasing levels of aggressiveness. Moreover, we find that (ii) the level of low priority in LEDBAT cannot be easily set by tweaking the protocol parameters, and that (iii) in the case of legacy LEDBAT implementations sharing the same bottleneck, even small differences in parameter settings (e.g., target delay) or network conditions (e.g., RTT delay) can result in significant unfairness.

II. Lower than Best Effort transport Protocols

This section provides an overview of the relevant related work. On the one hand, we have literature on BitTorrent that for the most part has however focused on other aspects than LEDBAT, such as modeling BitTorrent performance [13], studying incentive mechanism [14] and locality-aware peer selection strategies [15], or analyzing torrent popularity [16]. Only recently attention has grown on LEDBAT, as in [17] in which we carry out a measurement study on the protocol implemented in the official client, and in [18], [19], in which we perform a first evaluation by means of simulations and then propose some solutions to the latecomer issue unveiled before.

On the other hand, we have studies that focus on congestion control: as the literature dates back to the late 80s [20] it is thus very wide. However we will focus our attention on the above mentioned LBE protocols, providing a detailed introduction that will be instrumental to the comparison. With this regard, we just point out that although work exists that, in similar a spirit to ours compares high-speed TCP versions [9], [10], [11], to the best of our knowledge such a comparison effort has not been done for lower-than best effort protocols.

To achieve their goals, all LBE protocols need to detect congestion earlier than standard loss-based TCP. As the latter detects congestion by inferring that a packet loss occurred (e.g., by expiration of a timer, or by reception of duplicated acknowledgement), LBE protocols need to rely on a finer measure of congestion: typically, they equate increasing delay



	F_a	F_b	TCP%	η
TCP-LP	0.74	0.80	0.25	1.00
TCP-NICE	0.54	0.54	0.96	1.00
TCP-LEDBAT	0.51	0.51	0.99	1.00
TCP-TCP	0.90	1.00	0.50	0.90
LP-LP	0.89	1.00	0.49	0.91
NICE-NICE	0.95	0.96	0.40	1.00
LEDBAT-LEDBAT	0.97	1.00	0.51	1.00

Fig. 1. Low priority at a glance: Inter (top) and Intra (bottom) protocol interaction on a simple bottleneck

with incipient congestion. In other words, LBE protocols perform some delay measurement $D(t)$, and infer from the increase of $D(t)$ that congestion is building up, accounting the delay growth to some amount of queuing in the bottleneck link along the path. As we will see, the specific delay measurement $D(t)$ and the rule to decide that variations in $D(t)$ are actually due to congestion, vary from protocol to protocol.

Then, once congestion has been (early) detected, this triggers a congestion-relief reaction, which again differs across protocols. It is however possible that congestion is not detected in a timely fashion, causing a packet loss of the LBE protocol: in this case, the reaction of all protocols falls back to standard TCP timeout mechanism, i.e. a drastic reduction of the congestion window $cwnd$.

In the remainder of this section, we provide further details concerning each of the LBE protocols under consideration. To facilitate their comparison, we also report simple simulation results in Fig. 1, so to better visually highlight the relevant characteristics of each protocol. The top row of Fig. 1 reports the heterogeneous case where two flows employing different congestion control protocols are compared; the bottom plot Fig. 1 shows time evolution of two flows employing the same LBE protocol assuming similar network conditions.

More precisely, for each LBE$\in \{LP, NICE, LEDBAT\}$ protocol, Fig. 1 depicts the temporal evolution of the $cwnd$ of different flows in two scenarios of a simple bottleneck topology. In the inter-protocol case (top row, labeled as TCP-LBE), low-priority protocols compete against a standard TCP flow, while in the intra-protocol case (bottom row, labeled as LBE-LBE) two LBE flows compete against each other. In the figure, link capacity is set to C=10 Mbps, round-trip delay to RTT=50 ms and the buffer is B=100 MTU sized packets.

A. TCP-LP

TCP-LP (or LP *tout court*) measures one-way packet delays and employs a simple delay threshold-based method for early inference of congestion. More specifically, LP estimates the minimum D_{min} and maximum one-way delay D_{max}, filtering the instantaneous measure of the delay $D(t)$ by means of an exponentially weighted moving average $\tilde{D}(t)$ with smoothing parameter α, updated packet-by-packet. The smoothed average $\tilde{D}(t)$ and the condition for early congestion detection are:

$$\tilde{D}(t) = (1-\alpha)\tilde{D}(t-1) + \alpha D(t) \quad (1)$$

$$\tilde{D}(t) > D_{min} + (D_{max} - D_{min})\delta \quad (2)$$

where $\delta \in (0,1)$ is a custom threshold parameter. Throughout this paper, we use the values $\alpha = 1/8$, $\delta = 0.15$ that are selected in [5] by means of simulation experiments.

In the absence of early-congestion indication, LP behaves like standard TCP Reno, i.e., performing an additive increase of the congestion window $cwnd$ as shown in Fig. 1-(b) and (f). Whenever an early congestion is detected, according to the rules outlined above, LP halves the congestion window and enters an inference phase by starting an inference timeout timer. During this period, LP only observes responses from the network and avoids increasing the congestion window. After this phase, if congestion persists it reduces the congestion window to zero and restarts the TCP Reno congestion avoidance scheme. Finally, in case of losses, LP behaves like TCP Reno.

B. TCP-NICE

TCP-NICE (or NICE *tout court*) instead maintains a minimum round trip delay RTT_{min} and maximum RTT_{max} of the RTT delay. Congestion is detected when more than a given fraction ϕ of packets during the same RTT experiences a delay exceeding:

$$RTT > RTT_{min} + (RTT_{max} - RTT_{min})\delta \quad (3)$$

where δ and ϕ are protocol parameters set to $\delta = 0.2$ and $\phi = 0.5$ as in [6]. Notice that (3) is the same formula of LP (1), but computed on the RTT variable, and using the fraction-trick instead of a moving average.

In the absence of congestion, NICE behaves like TCP-Vegas [21], whose congestion window dynamics are delay-based (and thus rather different from loss-based dynamics). Whenever early-congestion is signaled, NICE simply halves

its congestion windows and sending rate, practically reintroducing the multiplicative decrease behavior. Finally, when a loss is detected it instead reacts like TCP Reno.

The fact that NICE inherits its congestion control behavior from Vegas [21] rather than from TCP Reno has profound impact on the $cwnd$ evolution: as observed in Fig. 1-(c) and (g), NICE shows a much smoother behavior as its throughput stabilizes around the effective link capacity. We point out that NICE allows $cwnd$ to be a fraction of 1 by sending one packet after waiting for the appropriate number of RTTs: the use of fractional values for $cwnd$ guarantees non-intrusiveness even in the case of many NICE flows sharing the same bottleneck.

C. LEDBAT

Finally, LEDBAT maintains a minimum one-way delay 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 variations of the queuing delay $D(t) - D_{min}$ to evaluate the distance $\Delta(t)$ from the target as in (4):

$$\Delta(t) = \tau - (D(t) - D_{min}) \quad (4)$$

$$cwnd(t+1) = cwnd(t) + \gamma\Delta(t)/cwnd(t) \quad (5)$$

where τ, γ are protocols parameters that we study later on.

In the absence of early-congestion indication, i.e., when the target τ has not been reached yet, $\Delta(t) > 0$ in (4) and thus $cwnd$ grows as defined by (5). Notice that when the target is reached, $\Delta(t) = 0$ thus $cwnd$ settles.

Values of $\Delta(t) < 0$ are perceived as early-congestion indication (i.e., other traffic is increasing the queuing delay $D(t) - D_{min}$), to which LEDBAT reacts by reducing $cwnd$ proportionally to the offset from the target according to (5). Finally, in case of losses, it behaves like TCP Reno.

Overall, LEDBAT shares similarities with, and exhibit differences from, the other LBE protocols: (i) as LP, it relies on one-way delay estimation to detect congestion, but unlike LP it does not employ a smoothing average; (ii) as NICE, its congestion controller is based on the delay, but unlike NICE it employs a PID controller in order to reach (or deviate from) the target delay. As we can see from Fig. 1, the behavior of LEDBAT is however closer to NICE than to LP.

III. Methodology

A. Simulation scenarios

We employ `ns2` simulations in order to compare the LBE protocols. While TCP Reno and LP protocols are already implemented, we implement both NICE and LEDBAT congestion control protocols in the network simulator. The code of the latter used for our work can be found at [12]. As reference network scenario, we use a dumbell topology where the capacity of the bottleneck is fixed to $C = 10$ Mbps, the one-way propagation delay equals 25 ms (thus round trip delay is equal to $RTT = 50$ ms), and the buffer size is set to $B_{max} = 100$ packets. We consider backlogged sources[1], that use a fixed packet size equal to S =1500 Bytes. All TCP and LBE sources start simultaneously, so that we avoid potential late-comer issues [18], and last for 120 seconds.

In this work we first focus on a sensitivity analysis of LEDBAT, to assess the impact of parameters τ and γ on the system performance. We carry out this analysis in both (i) an inter-protocol case, where a TCP Reno flow and a LEDBAT flow share the bottleneck and (ii) an intra-protocol case, where a two LEDBAT flows compete against each other. The aim of (i) is to determine whether τ and γ offer the chance to tune the level of LBE priority in LEDBAT, while (ii) aims at verifying whether unfairness may arise among legacy LEDBAT implementations (e.g., different releases of the same code, different implementations or parameter settings, etc.).

We then focus on a comparison of TCP and LBE protocols, again considering two cases: (iii) a single TCP flow shares the bottleneck with a varying number of homogeneous LBE flows (i.e., same LBE protocol) and (iv) several heterogeneous LBE flows compete against each other. In both cases, our aim is to evaluate the level of low priority of LBE protocols. Finally, we consider more realistic scenarios in (v) by taking into account the impact of RTT heterogeneity on LBE performance.

B. Evaluation metrics

Performance evaluation is carried out considering different metrics, that relate to either network-centric (e.g., efficiency, average queue size) or user-centric performance (e.g., fairness, packet loss rate). To illustrate this, Fig. 1-(a) summarizes the performance of flows in corresponding scenarios in terms of some of these metrics (i.e. efficiency, fairness, and breakdown).

Bottleneck link efficiency (η) is the primary network-centric metric, and expresses the link utilization as the ratio between the sum of the throughput values x_i achieved by all flows over the available capacity $\eta = \sum_i x_i/C$.

Average queue occupancy index (B) is computed averaging buffer occupancy during the simulation (measured at each enqueue event in the buffer), and normalizing the value over the buffer size $B = E[B]/B_{max}$ for convenience.

Whenever the buffer overruns and packets are dropped, all protocols drastically reduce their sending window: *packet loss probability (P_l)* therefore relates to user-performance, and is computed as the ratio of the dropped packets over the total number of packets sent on the link.

We further express the system performance using two metrics apt at describing how the link resources are shared among flows. To gauge the impact of LBE on TCP, we define *TCP breakdown ($TCP_{\%}$)* as the TCP Reno traffic share percentage over the total amount of data exchanged on the link, i.e., $TCP_{\%} = \sum_{j \in TCP} x_j / \sum_i x_i$.

We further describe the capacity share in terms of *Jain fairness index (F)*, defined as $F = (\sum_{i=1}^{N} x_i)^2/(N \cdot \sum_{i=1}^{N} x_i^2)$

[1]As we consider backlogged sources only, dynamics of LEDBAT are well described by means of (5) only; in case of non-backlogged sources, the dynamics changes slightly to avoid cwnd increase indefinitely [1]

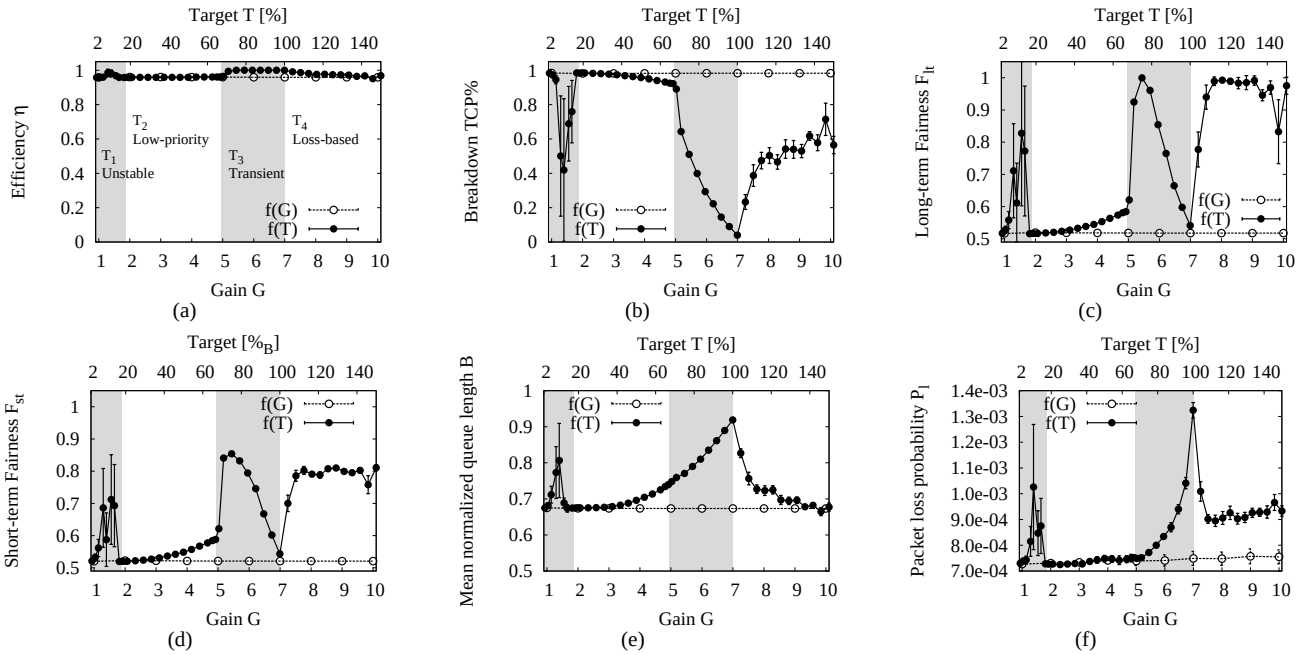


Fig. 2. LEDBAT vs TCP Reno: Inter-protocol sensitivity analysis, for varying LEDBAT target T and gain G parameters

where N is the number of considered flows and x_i is the rate of the $i-th$ flow. In the best case, when all flows get a fair share of the resources, F is equal to one, while in the worst one, namely when a single flow exploits all the link, it is equal to $1/N$.

We compute the fairness index over both the whole flow duration and over a smaller time scales (considering a temporal window of 20 RTT, or equivalently 1 s): we refer to *long-term fairness* (F_{lt}) in the first case, and to *short-term fairness* (F_{st}) in the latter one. Notice that the ability to achieve short-term (vs long-term) fairness may have rather different implications, e.g., if we consider the case of several P2P flows measuring throughput to perform peer selection (as long-term fairness may not be sufficient and significantly biases peer decisions).

IV. LEDBAT Sensitivity Analysis

A. Inter-protocol: LEDBAT vs TCP

We start our sensitivity analysis by considering two flows, a standard TCP Reno and a LEDBAT one, that start simultaneously and vary the values of parameters τ and γ one at a time. Notice that the standardization draft *does not specify any value* for the gain parameter γ. Conversely, the draft *specifies a mandatory value* for the target parameter equal to $\tau = 25$ ms. This choice of τ is somewhat arbitrary, and based on experimental observations (whose results are however unreported so far) or motivated by practical constraints (e.g., today's limits in the precision of the delay measurement, etc.), rather than being based on concrete foundations. As such, τ is often referred to as "magic number" with a deprecatory sense in LEDBAT WG discussion [22]: therefore, we believe that a thorough exploration of the impact of the above parameters is necessary, which we carry out by simulation.

For convenience, we re-express the gain parameter γ as multiples of the τ, i.e. $\gamma = G/\tau$, and explore the range $G \in [1, 10]$ For convenience, we re-express the target delay parameter τ in terms of buffer percentage as $T = \tau C/(SB_{max})$, and explore the range $T \in [2, 150]\%$, corresponding to $\tau \in [2.4, 180]$ ms. For reference purposes, notice that the mandatory draft value $\tau = 25$ ms correspond to $T = 20\%$, while a full buffer occupancy $T = 100\%$ is attained when $\tau = 120$ ms.

Fig. 2 reports the simulation results for each of the metric $f \in \{\eta, TCP_{\%}, F_{st}, F_{lt}, B, P_l\}$ described early in Sec. III-B, arranged as one per plot. In each plot, we report two curves, namely $f(G)$ and $f(T)$. The $f(G)$ curve reports how $f(\cdot)$ varies as a function of the gain $G \in [1, 10]$ (on the bottom x-axis), when target is fixed to $\tau = 25$ ms. The $f(T)$ curve instead reports how $f(\cdot)$ varies as a function of the target $T \in [2, 150]\%$ (on the top x-axis), when gain is fixed to $G = 1$.

From all the subplots we can see that, for all metrics, the $f(G)$ curve is roughly flat, i.e., the gain parameter only minimally affects the behaviour of the LEDBAT protocol in this case. This can be explained by the fact that, as LEDBAT is designed to yield to TCP, it will yield irrespectively of G. The gain value thus only affects the speed at which LEDBAT will yield, which quickly happens for any value of G.

Therefore, from now on we restrict our attention to the impact of the target parameter, and analyze the behavior of the $f(T)$ curves. In Fig. 2-(a) we can see that the efficiency η is only slightly influenced by the variation of the target and remains always close to the total link capacity. This is expected, as even if the target is misconfigured, either LEDBAT or TCP Reno can take advantage of the unused bandwidth, which result in an overall efficient use of the link capacity.

Considering instead the $TCP_{\%}$ reported in Fig. 2-(b), we can identify four working regions. When the target is very small $T_1 \in [2, 20]\%$ the LEDBAT protocol is not always able to reach the target delay, which leads to shaky $TCP_{\%}$ behavior. In a second region $T_2 \in [20, 65]\%$, LEDBAT completely yields to the TCP Reno flows, working in low-priority mode and thus attaining its goal. In a third region $T_3 \in [65, 100]\%$, LEDBAT aggressively starts to erode bandwidth to the TCP Reno flow: this causes losses in the TCP Reno flow, which progressively backs off; as a consequence, the $TCP_{\%}$ starts decreasing until LEDBAT has the monopoly of the buffer $T = 100\%$ and TCP Reno starves ($TCP_{\%} \simeq 0\%$). Finally, in the fourth region $T_4 > 100\%$ the target exceeds the buffer size: in this case, LEDBAT falls back in the TCP Reno-like loss-based behavior, increasing the sending rate until a loss occurs, which immediately drop down its rate. As a consequence, the breakdown is now more similar ($TCP_{\%} \simeq 50\%$)

Similar considerations can be gathered by looking at the long-term F_{lt} or short-term F_{st} fairness plots shown in Fig. 2-(c) and Fig. 2-(d) respectively: indeed, an even breakdown corresponds to maximum fairness ($F_{st} \simeq 1$) while to an uneven breakdown, favoring either TCP Reno ($TCP_{\%} \simeq 100\%$) or LEDBAT ($TCP_{\%} \simeq 0\%$), always corresponds to minimum fairness values ($F_{st} \simeq 1/2$). We also notice that, although as expected short-term fairness is more difficult to achieve ($F_{st} < F_{lt}$), the same qualitative behavior holds for both fairness timescales.

From Fig. 2-(e) and Fig. 2-(f) we see that, as expected, increasing the target the average buffer occupancy rises, as the increased traffic due to LEDBAT sums up with the TCP Reno one. Losses increase as well reaching a peak for $T = 100\%$, corresponding to LEDBAT maximum aggressiveness; afterward LEDBAT is in loss-mode, and the scenario degenerates into two TCP Reno flows sharing a bottleneck.

Overall, the sensitivity analysis suggests that, although LEDBAT spans a wide range of low-priority levels (especially in the third region), its tuning is highly impractical. Indeed, the support of target values $T_3 \in [65, 100]\%$ is very small, meaning that a small variation of T lead to completely different scenarios, where either LEDBAT or TCP Reno exhibit starvation. Moreover, the actual values of τ yielding to a specific level of low-priority depends on network parameter (e.g., capacity C, buffer size B) and are likely affected from other factors as well (e.g., number of TCP Reno flows, heterogeneous RTT, etc.)

B. Intra-protocol: LEDBAT vs LEDBAT

We pursue our sensitivity analysis by considering two LEDBAT flows with heterogeneous settings sharing the same bottleneck link. We perform two sets of experiments, varying either (i) the gain ratio G_1/G_2 of the two flows when $G_2 = 1, \tau = 25$, or (ii) the target ratio T_1/T_2 when $T_2 = 20\%, \gamma = 1/\tau$. In both cases, the ratio varies in the $[1, 10]$ range. Results of the sensitivity analysis are reported in Fig. 3, which depicts the packet loss rate P_l (right y-axis), the average buffer size B, the efficiency η and the fairness F_{lt}

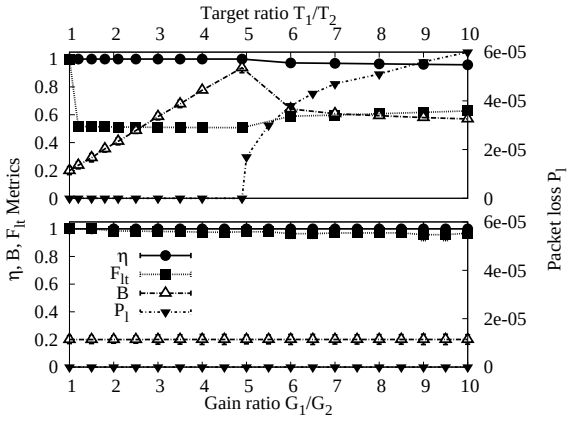


Fig. 3. LEDBAT vs LEDBAT: Intra-protocol sensitivity analysis, for varying LEDBAT target T_1/T_2 and gain G_1/G_2 ratios

(left y-axis) as a function of the T_1/T_2 target ratio (top plot) and G_1/G_2 gain ratio (bottom plot).

As in the previous case, the impact of the gain is very modest, even in the case of a 10-fold factor. This phenomenon has an intuitive explanation. Consider indeed, that a flow with the largest gain will start moving faster that the other flow toward the target. However, after the first flow increases its window, the convergence speed toward target will slow down, since the differences between the target and the measured delay is now smaller for the first flow than for the second. In other words, the difference in the delay offset in (5) compensates for differences in the gain factor γ.

Conversely, even slight differences in the target settings may have strong consequences as it can be seen in the top plot of Fig. 3. Indeed, as soon as $T_1/T_2 > 1$ it can be seen that the fairness immediately drops to its minimum value $F_{lt} = 0.5$. This is due to the fact that flows with higher-target are always more greedy than their lower-target counterpart. As a matter of fact, if both flows start at the same time, they both measure the same base delay, and the higher-target flow converges faster to its target and stabilizes: as the amount of queuing is now larger than that of the less aggressive flow, the latter backs off and starves. This holds until $T_1 + T_2 > 100\%$ which happens when $T_1/T_2 = 5$ given that $T_2 = 20\%$, in which case both LEDBAT flows may experience packet drops: nevertheless, higher-target flow will always be advantaged prior than losses occur, and so the unfairness persists.

Overall, we see that gain and target parameters have rather different effects: on the one hand, provided that LEDBAT flows have the same target, differences in gain do not entail any unfairness among flows. On the other hand, even a small difference in targets produces an extremely unfair situation: this is a delicate point, which we believe deserves further attention in the future.

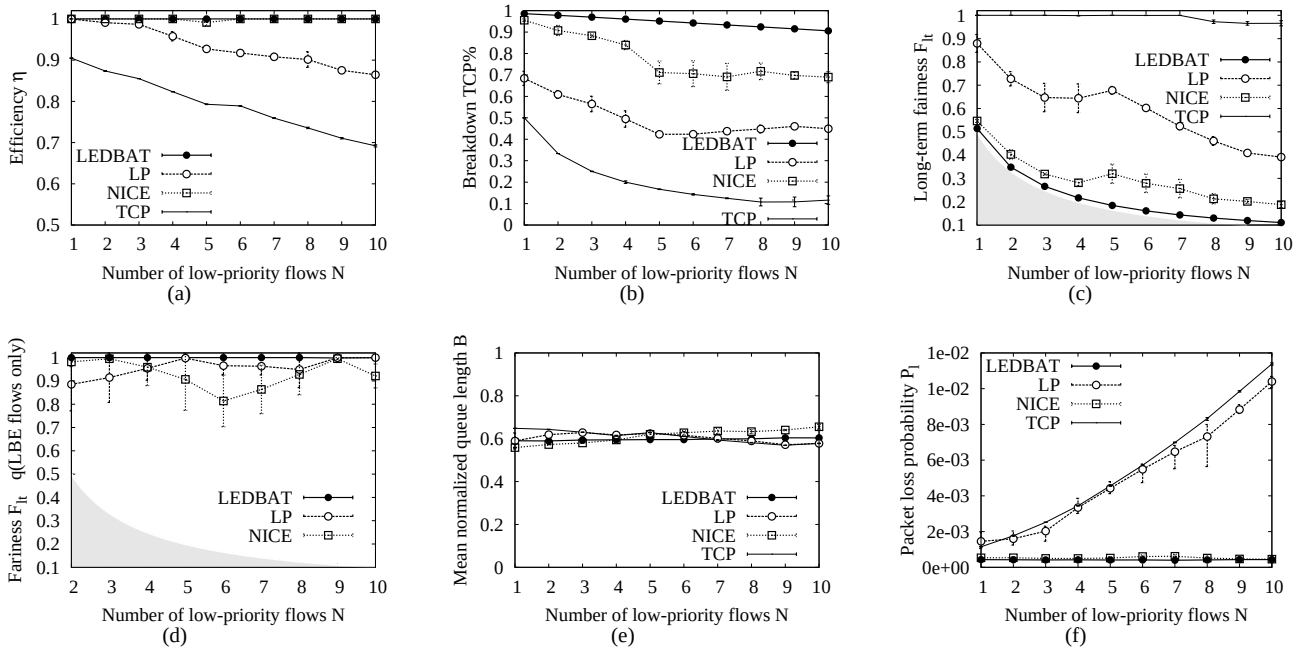


Fig. 4. LBE against one TCP flow: Impact of the number LBE flows on the system performance

V. LBE PROTOCOLS COMPARISON

A. LBE against TCP

We now fix LEDBAT parameters and consider a larger set of LBE protocols in the comparison. Following our sensitivity analysis, we know that the selection of γ, rather than τ, is less critical: we set then $\gamma = 1/\tau$, and use the mandatory value $\tau = 25\,\text{ms}$ which we verified to be a robust choice.

We consider a typical scenario where N, ($N \in [1, 10]$) low-priority flows (e.g., due to P2P or other services) share the same bottleneck with a single TCP Reno connection, (representative of a generic high-priority service), for a total of $N+1$ flows. We perform several sets of simulations separately, considering each time a different LBE protocol. For reference purpose, we also simulate the case where $N + 1$ TCP Reno flows share the same bottleneck.

Results for the common set of metrics are reported in Fig. 4. Considering efficiency η in Fig. 4-(a), we see that delay-based NICE and LEDBAT are able to fully utilize the spare bandwidth left by TCP Reno. Conversely, in the LP or TCP Reno cases, losses entail a reduction in efficiency.

Breakdown $TCP_{\%}$ reported in Fig. 4-(b), states that e.g., in the $N = 10$ LEDBAT case, TCP Reno consumes about 90% of the link capacity (since $\eta \simeq 1$), leaving thus the $N = 10$ LEDBAT flows a mere 1% of the capacity each. Comparing this result with NICE (about 3% each) or LP (about 5% each) under the same $N = 10$ settings, we gather that LEDBAT achieves the lowest priority, closely followed by NICE. This is further exacerbated from the long-term fairness plot of Fig. 4-(c), showing that in the LEDBAT and NICE cases fairness approaches the minimum possible value (i.e., the shaded region indicates values that fairness cannot achieve since they are smaller than the lower bound $\frac{1}{N+1}$).

The plot in Fig. 4-(d) depicts instead the long-term fairness F_{lt} evaluated over the N LBE flows only. It can be seen that fairness is always high, meaning that generally the excess that remains after the TCP breakdown of Fig. 4-(b), is evenly shared among LBE flows. Notice that fairness among LBE flows is however lower in the case of NICE, where apparently some LBE flow opportunistically take advantage of the others.

Finally, average occupancy index B and packet loss P_l are reported in Fig. 4-(e) and (f) respectively. Again, delay-based versus loss-based congestion control principles are remarkably different, which is especially true in the case of the loss curve: interestingly, despite its low priority aim, the amount of loss induced by LP is strikingly similar to that of classic TCP Reno. Delay-based versus loss-based difference, although less evident, also reflects on the queue size: indeed, TCP Reno and LP average queue size decrease when number of flows and losses increase; conversely, queue occupancy in the NICE case slowly rises for increasing N, and is practically unaffected by N in the LEDBAT case.

B. LBE against LBE

In order to investigate the mutual interaction of the different lower-priority protocols, we define a heterogeneous scenario in which several LEDBAT, LP and NICE flows contend the same bottleneck link. We perform different tests in which an increasing number of flows is considered, from 1 to 5 for each flavor (which corresponds to a total of 3 to 15 flows). As reference, we perform also the corresponding experiment with the same number of TCP Reno flows only (i.e., 3 to 15 TCP Reno flows). We choose for all the LEDBAT flows, the standard parameters values, namely $\tau = 25\,ms$ and $\gamma = 1$. We

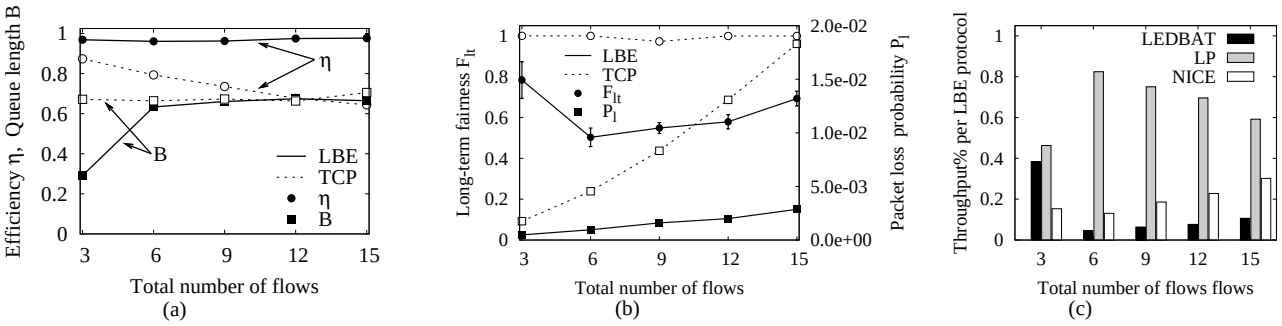


Fig. 5. LBE against LBE: LP, LEDBAT and NICE competing for the same bottleneck, compared with the same number of TCP Reno only flows.

point out that qualitatively similar results can be gathered using different parameter settings, which we are however unable to report for lack of space.

Let us start by examining the efficiency η and average normalized buffer length B, which are reported in Fig. 5-(a). Looking at the efficiency, we can see that in the heterogeneous LBE scenario, flows are able to utilize the available resource fully, with η always close to its maximum. On the contrary, the efficiency in the case of all TCP Reno flows progressively decreases as long as the number of competing flow increases, due to the typical synchronization behavior of the protocol after loss. Looking at the normalized average queue size we can notice that the average $B \simeq 2/3$ is not affected by the number of flows in the TCP Reno case. In the LBE case instead, average queue size approaches that of TCP Reno only when at least two flows per protocol insist on the bottleneck. When only a total of three LBE flows are competing for the resource, a rather unexpected phenomenon arises: in this case, LEDBAT often forces LP in low-priority mode and is thus able to exploit a significant part of the resource. As a consequence, the average queue size B reflects the LEDBAT target τ, plus the contributions due to LP and NICE. When more than two LP flows are instead present on the bottleneck, their behavior synchronizes and is perceived as more aggressive by LEDBAT: in this case, it is rarer that *both* LP flows enter into inference mode at exactly the same time, thus LEDBAT has fewer opportunities to profit from the resource.

Packet loss probability P_l and long-term fairness F_{lt} are reported in Fig. 5-(b). Concerning packet loss, since 2/3 of the total flow number consists of delay-based protocols, the loss rate is clearly lower than the TCP Reno reference case. Long-term fairness performance shown in Fig. 5-(b) is instead better understood by considering also the throughput breakdown reported in Fig. 5-(c), in which each bar represents the percentage of traffic due to a particular LBE protocol. As expected, fairness between heterogeneous LBE flows is lower than that of homogeneous TCP Reno connections, but is however higher than the LBE-TCP Reno performance earlier reported in Fig. 4-(c). In particular, maximum LBE fairness is achieved when only one flow per each LBE flavor is considered: from Fig. 5-(c) we see that LP and LEDBAT performance are very close in this case, which raises the fairness metric. When the number of low-priority flows increases, the fairness instead decreases due to a higher aggressiveness of the LP protocol.

C. Impact of RTT heterogeneity

Finally, we report on the impact of RTT heterogeneity in Fig. 6. We consider two flows, of the same protocol type (LBE or TCP) sharing the same bottleneck, that have a different round trip delay expressed by the RTT_1/RTT_2 ratio. We perform simulations separately for each protocol, exploring the $RTT_1/RTT_2 \in [1, 10]$ range; RTT_1 is increased by adding propagation delay to the return path, so that one-way delay estimation on the forward path is not affected. The top plot of Fig. 6 reports the long term fairness F_{lt}, while bottom plot reports the efficiency η as a function of the RTT ratio.

An interesting remark to make concerning the fairness metric is that only NICE, by virtue of its inheritance of Vegas [21] congestion control, provides fairness in the case of heterogeneous RTT settings. However, this comes at the price of a reduced efficiency, since in order to be fair, the more aggressive small-RTT flow has to slow down its rate to match that of the large-RTT flow. Efficiency loss happens also in the case of LP and TCP Reno, despite their inability to offer fairness. Finally, LEDBAT realized a perfectly efficient system, which comes at the price of a totally unfair share of the resources. In fact the small-RTT flow is able to reach its target first, due to the faster feedback, whereas the second flow will see a queuing delay (due to the small-RTT flow) equals to its target, and will thus settle in a starvation state.

VI. Conclusion

This paper analyzes different Lower-than Best Effort (LBE) transport protocols behavior. By means of simulation, we carried out a thorough comparison of LEDBAT, LP and NICE, studying the impact they have on TCP Reno traffic, as well as their mutual impact. The sensitivity analysis of LEDBAT reveals the difficulty in tuning its behavior, and especially its level of priority with respect to TCP Reno by means of a simple adjustment of its gain γ and target τ parameters. Indeed, the gain has practically no influence, while the impact of target can not be controlled, as changes in the system performance are too steep. Also, we see that gain and target

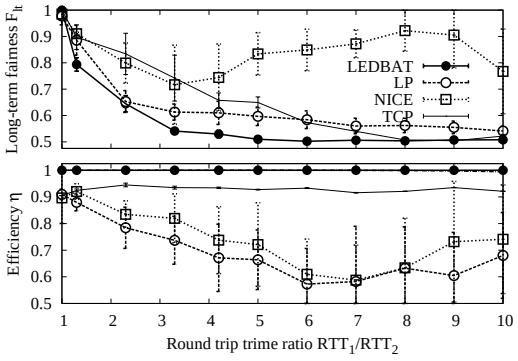


Fig. 6. Impact of the RTT heterogeneity

parameters have rather different effects if we consider the coexistence of legacy LEDBAT flows with heterogeneous settings: on the one hand, provided that LEDBAT flows have the same target, differences in gain do not entail any unfairness among flows; on the other hand, even a small difference in targets results in an extremely unfair situation. We conclude that tuning LEDBAT is thus a delicate point, which deserves further attention in the future, which holds true even when heterogeneous network settings are considered.

Our comparison study shows that LEDBAT achieves the lowest possible priority with respect to NICE and LP. Moreover, we find that LP inherits from its loss-based design a higher aggressiveness than the delay-based NICE whose degree of low-priority sits thus in between LEDBAT and LP. Interestingly, we point out that there are also limit cases (e.g., only an LP, LEDBAT and NICE flows sharing the same bottleneck) in which the low-priority degree can exhibit unexpected behavior (i.e., as LEDBAT is in this case as aggressive as LP).

We believe this work to be a first important step in understanding, comparing and ranking several LBE protocols. At the same time, an important question remains open: namely, how a different degree of low-priority can be achieved in a robust, tunable fashion, which our future research aims at answering.

ACKNOWLEDGEMENT

This work was supported by Celtic Project TRANS.

REFERENCES

[1] S. Shalunov. (2010) Low Extra Delay Background Transport (LEDBAT). [Online]. Available: http://tools.ietf.org/id/draft-ietf-ledbat-congestion-01.txt

[2] A. Norberg. uTorrent transport protocol. BitTorrent Enhancement Proposals. [Online]. Available: http://www.bittorrent.org/beps/bep_0029.html

[3] "Comcast throttles bittorrent traffic, seeding impossible," August 2007. [Online]. Available: http://torrentfreak.com/comcast-throttles-bittorrent-traffic-seeding-impossible/

[4] (2007) Setting up bittorrent with qos. [Online]. Available: http://forums.whirlpool.net.au/forum-replies-archive.cfm/627285.html

[5] A. Kuzmanovic and E. Knightly, "TCP-LP: low-priority service via end-point congestion control," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. 4, p. 752, 2006.

[6] A. Venkataramani, R. Kokku, and M. Dahlin, "TCP Nice: A mechanism for background transfers," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, p. 343, 2002.

[7] S. Liu, M. Vojnovic, and D. Gunawardena, "4CP: Competitive and considerate congestion control protocol," 2006.

[8] Microsoft background intelligent transfer service (bits). [Online]. Available: http://msdn.microsoft.com/en-us/library/aa363167.aspx

[9] E. Altman, K. Avrachenkov, and B. Prabhu, "Fairness in MIMD congestion control algorithms," *Telecommunication Systems*, vol. 30, no. 4, pp. 387–415, 2005.

[10] Y. Li, D. Leith, and R. Shorten, "Experimental evaluation of TCP protocols for high-speed networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 15, no. 5, p. 1122, 2007.

[11] S. Molnár, B. Sonkoly, and T. Trinh, "A comprehensive TCP fairness analysis in high speed networks," *Computer Communications*, vol. 32, no. 13-14, pp. 1460–1484, 2009.

[12] Ledbat implementation for ns2. [Online]. Available: http://perso.telecom-paristech.fr/~valenti/pmwiki/pmwiki.php?n=Main.LEDBAT

[13] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *ACM SIGCOMM'04*, Portland, Oregon, USA, Aug 2004.

[14] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and Improving a BitTorrent Performance Mechanisms," in *IEEE INFOCOM'06*, Barcelona, Spain, Apr 2006.

[15] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving Traffic Locality in BitTorrent via Biased Neighbor Selection," in *IEEE ICDCS '06*, Lisboa, Portugal, Jul 2006.

[16] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. Al Hamra, and L. Garcés-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime," in *In PAM'04*, Antibes, France, Apr 2004.

[17] D. Rossi, C. Testa, and S. Valenti, "Yes, we LEDBAT: Playing with the new BitTorrent congestion control algorithm," in *PAM'10*, Zurich, Switzerland, Apr 2010.

[18] D. Rossi, C. Testa, S. Valenti, and L. Muscariello, "LEDBAT: the new BitTorrent congestion control protocol," in *ICCCN'10*, Zurich, Switzerland, Aug 2010.

[19] G. Carofiglio, L. Muscariello, D. Rossi, and S. Valenti, "The quest for LEDBAT fairness," in *IEEE Globecom 2010*, Miami, FL, USA, Dec 2010.

[20] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM*, vol. 25, no. 1, 1988.

[21] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, pp. 24–35, 1994.

[22] LEDBAT Mailing List Archives. [Online]. Available: http://www.ietf.org/mail-archive/web/ledbat