# Performance Analysis of LEDBAT in the Presence of TCP-Newreno

Amuda James Abu and Steven Gordon
*Sirindhorn International Institute of Technology, Thammasat University,*
*Pathumthani 12000, Thailand.*
*james@ict.siit.tu.ac.th, steve@siit.tu.ac.th*

***Abstract*—Low Extra Delay Background Transport (LEDBAT) is a novel one-way delay Internet congestion control algorithm developed to react to congestion earlier than any of the loss-based TCP congestion control algorithms (e.g. TCP-NewReno). A LEDBAT source quickly reduces its sending rate when the queue delay experienced in its path is greater than a fixed pre-defined target value. This paper analyses LEDBAT when it is sharing a bottleneck link with TCP. Our analysis identifies the threshold of a bottleneck buffer size that leads LEDBAT to revert to a minimum congestion window of only 1 packet in the presence of TCP. That is, for some applications its throughput will be too low. In addition to the fact that intra-protocol unfairness among multiple LEDBAT sources may occur when using the fixed minimum LEDBAT congestion window to improve the limited LEDBAT throughput, we show that the average LEDBAT throughput is fixed even as the bottleneck link capacity increases as opposed to TCP throughput that increases proportionally. This therefore necessitates the need for a dynamic minimum congestion window in the LEDBAT algorithm.**

***Keywords*-LEDBAT, delay-based congestion control, low priority protocols, peer-to-peer file sharing, real-time applications**

## I. Introduction

Low Extra Delay Background Transport (LEDBAT) congestion control algorithm [1] is a one-way delay and window based algorithm designed for peer-to-peer (P2P) applications and other applications that use multiple TCP [2] connections for data transfer. The novel congestion control algorithm is motivated by the unfairness problem in TCP aggravated by applications that use multiple TCP connections. The design of LEDBAT is such that a source maximally utilizes available network bandwidth when no other traffic source exists, and yields quickly to newly arriving traffic. For a LEDBAT source to be TCP-friendly, its sending rate must not be increased faster than TCP. As with TCP, the sending rate of a LEDBAT source, and hence throughput, are directly proportional to the congestion window [3]. As part of the design of LEDBAT, a constant value of target queue delay in the path of LEDBAT is assumed by the source. The source adjusts its sending rate with respect to an estimated queue delay in its path so that the actual queue delay does not exceed the target. Micro Transport Protocol (uTP) [4] is an application layer congestion control protocol used by $\mu$Torrent (a widely used UDP-based BitTorrent protocol) and similar to LEDBAT [1].

LEDBAT is designed to react to congestion in a network earlier than the TCP loss-based algorithm, TCP NewReno [5], [6]. A LEDBAT source achieves this by reducing its sending rate whenever a measured queue delay is greater than the target. LEDBAT has been designed to provide less-than best effort service in the presence of other traffic (especially TCP). Several works have analysed LEDBAT in the presence of a TCP loss based congestion control algorihtm [7], [8], [9], [10], [11], our work is significant as it is the first to identify the threshold of a bottleneck buffer size that leads LEDBAT to revert to a minimum congestion window in the presence of TCP.

This paper analyses the performance of LEDBAT in the presence of TCP[1] under different conditions. We start by giving a formal explanation of how different bottleneck buffer sizes impact on LEDBAT congestion window and hence throughput in the presence of TCP. Our results from simulations show that LEDBAT in the presence of TCP reverts to its minimum congestion window ($w_{min}$) if the bottleneck buffer size is greater than a threshold and LEDBAT starts earlier than TCP. Otherwise, the LEDBAT congestion window oscillates between $w_{min}$ and some upper bounds. Although there is no minimum value stated in [1], a reasonable assumption is one maximum sized segment (MSS). We further show that if increasing $w_{min}$ would improve the limited LEDBAT throughput in the presence of TCP and $w_{min}$ was a user configurable parameter, then it could result in different amount of the bottleneck capacity yielded to TCP by different LEDBAT sources. This can lead to intra-protocol unfairness among the multiple LEDBAT sources when they share the same bottleneck link in the presence of TCP. We finally show that with fixed minimum congestion window in LEDBAT, increasing bottleneck capacity offers TCP an increasing throughput but nearly fixed for LEDBAT. This may be undesirable for LEDBAT users thus necessitating the need for a dynamic minimum congestion window in the LEDBAT algorithm.

The rest of this paper is organized as follows. Section II gives the basic operations of LEDBAT while Section III reviews recent works on LEDBAT. Our system model is given in Section IV and results from simulations are presented in

[1]In the remainder of this paper, TCP refers to TCP using NewReno congestion control algorithm

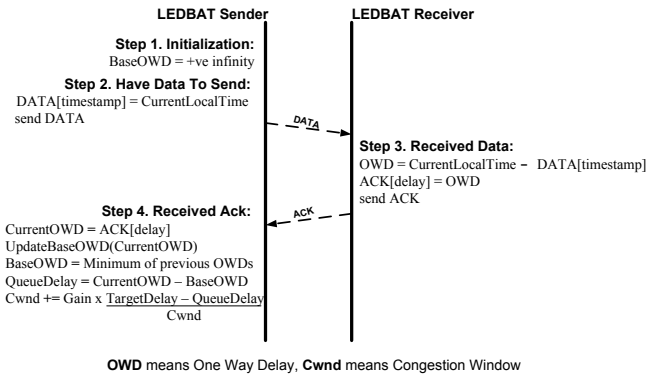


Figure 1. Pseudocode of LEDBAT congestion control at the sender and receiver

Section V. We conclude this paper in Section VI.

## II. LEDBAT Objectives and Operations

LEDBAT is designed for non-interactive applications to provide *lower-than-best-effort* service for end-users. The key objectives are [1]:

- To maximally utilize the bottleneck link capacity while keeping queue delay low when no other traffic is present in the network.
- To quickly yield to traffic sharing the same bottleneck queue that uses standard TCP congestion control or UDP (used by some real-time traffic).
- To contribute little to the queue delays induced by TCP traffic.
- To operate well in networks with FIFO queue with drop-tail queue discipline and to be deployable for common applications that currently dominate large portion of the Internet traffic.

Figure 1 illustrates the LEDBAT congestion control algorithm. The algorithm involves the source estimating the delay to the destination by placing a time stamp in data packets. The destination sends the measured one-way delay of the data packet in a delay field in the acknowledgement packet. Upon receiving the acknowledgement, the source uses the measured one-way delay to estimate the queue delay in the path. The source assumes the queue delay is the difference between the current one-way delay measurements and a base set of one-way delay measurements. The base one-way delay is taken as the minimum one-way delay from a list of previous one-way delay observations.

The LEDBAT source has a target queue delay: the source aims not to increase the queue delay above this target. The sender increases its sending rate as long as the estimated queue delay is less than the delay target. Otherwise, it reduces its sending rate before the access router buffer is full, in order to allow other applications to obtain a fair share of network resources and experience low queue delay.

LEDBAT uses a linear controller in its design to proportionally modulate the congestion window with the estimated queue delay. Equation (2) describes the controller where $w$ is the LEDBAT source congestion window, $w_{min}$ is the minimum congestion window, $\hat{d}_{que}$ is the queue delay estimated by the LEDBAT source, $G$ is a constant gain and $d_{tar}$ is the target queue delay. $d_{tar}$ and $G$ (both constants) are two key parameters that influence how well LEDBAT achieves its aims of saturating the bottleneck and yielding quickly to other traffic [10], [12].

$$cwnd = w(t) + \frac{G(d_{tar} - \hat{d}_{que}(t))}{w(t)} \quad (1)$$

$$w(t+1) = \begin{cases} \frac{1}{2}w(t) & \text{if packet loss} \\ w_{min} & \text{if } cwnd \leq w_{min} \\ cwnd & \text{otherwise} \end{cases} \quad (2)$$

LEDBAT aims to achieve friendliness with TCP by: 1) not increasing faster than TCP during start-up phase, 2) quickly yielding to TCP, and 3) halving its congestion window when a packet loss is detected in the path of LEDBAT flow. Carefully choosing a good value of gain is a step towards achieving TCP-friendliness in terms of *non-greater than TCP* ramp-up speed of LEDBAT.

## III. Related Work

Of the existing congestion control algorithms proposed over the last two decades, delay-based and low-priority algorithms have similar aims and mechanisms to LEDBAT. However, LEDBAT differs from many such algorithms including TCP-Vegas [13], TCP-NICE [14] and TCP-LP [15], in that it aims at minimizing queue delay in a network to a defined value that can be tolerated by voice, video and gaming applications. The work in [16] provides a survey of these and other low-priority congestion control algorithms. Several works in recent times have focused on the research and experimentation of LEDBAT operation. The rest of this section provides a review of these works on LEDBAT.

In addition to some potential issues with LEDBAT under certain conditions, it has been shown in [7], [8], [9], [10], [11], [12] that LEDBAT achieves some of its design objectives. In the performance analysis of LEDBAT in a controlled testbed and Internet experiment [7], it was found out that TCP traffic on the "unrelated" backward path is capable of causing LEDBAT to significantly underutilize the link capacity in the forward path. In [8] LEDBAT competes fairly with TCP in the worst case (i.e. LEDBAT misconfiguration). Potential intra-protocol fairness issues have been identified in LEDBAT [8] which can be fixed by using slow-start in the LEDBAT algorithm, random drops of LEDBAT sender window, and multiplicative decrease [9]. The proposed solutions are not without a performance

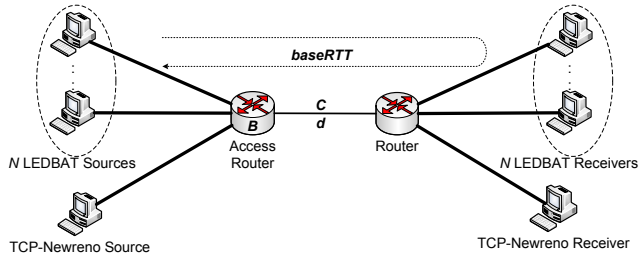


Figure 2. Network model

trade-off between link utilization and fairness [9]. LEDBAT achieves the lowest priority in the presence of TCP when compared to other low priority protocols (TCP-NICE and TCP-LP) [10]. Sensitivity analysis in [10] showed that unfairness exists between two LEDBAT flows with different delay targets or different network conditions. Our previous work in [12] anaylsed the impact of different values of gain on LEDBAT throughput and fairness. Based on the analysis, a dynamic gain algorithm for stabilising LEDBAT sending rate was proposed.

Although the works in [7], [8], [9], [10], [11] have analysed the performance of LEDBAT in the presence of TCP, our work is significant as it is the first to identify the threshold of a bottleneck buffer size that leads LEDBAT to revert to its minimum congestion window. As it will be shown later on in this paper, a bottleneck buffer size less than the threshold leads LEDBAT to oscillate between the minimum congestion window and some upper bounds.

## IV. System Model

This analysis is aimed at: 1) identifying the threshold of a bottleneck buffer size that leads LEDBAT to revert to its minimum congestion window in the presence of TCP; 2) quantifying the amount a bottleneck capacity that a LEDBAT flow yields to a newly arriving TCP flow in the same access network; 3) showing that LEDBAT throughput is nearly fixed when bottleneck capacity is increased unlike TCP throughput. This section presents the system model, which is based on the topology in Figure 2. The assumptions used in the analysis, as well as a formal explanation of how different cases of the bottleneck buffer size impact LEDBAT performance in the presence of TCP, are also given in this section.

The network topology assumes $N$ LEDBAT source(s) sharing a common path with a single TCP source. It is assumed that the TCP and LEDBAT sources always have data to send and their sending windows are not limited by the receiver advertised windows (but by the congestion window). All sources (LEDBAT and TCP) send fixed size packets of $P$ bytes. The RTT for all sources when there is no queue delay is $baseRTT$. The uplink in the access network is the path bottleneck and has capacity $C$, a typical case in the access networks of most ISP's networks [17], [1]. The capacities of all other links are assumed to be greater than $C$. The router uses a FIFO drop-tail queue with maximum size of $B$ packets. For the values of LEDBAT design parameters, we use 25ms and 40 respectively for $d_{tar}$ and $G$ [1].

LEDBAT throughput in the presence of TCP will depend on the bottleneck buffer size. This is because LEDBAT congestion window will only be increased or decreased if the estimated queue delay is less or greater than the target delay, respectively. During an active session of TCP, the TCP source halves its congestion window upon inferring a packet loss, thus reducing the queue delay in the path. When the queue delay is reduced, say to $d_{threshold}$, LEDBAT will increase its congestion window if $d_{threshold}$ is less than the target delay. Otherwise LEDBAT decreases its congestion window. Denoting the bottleneck buffer size that results in a queue delay of $d_{threshold}$ as $B_{threshold}$:

$$B_{threshold} = C \times (baseRTT + 2d_{tar}) \tag{3}$$

This is because a TCP source keeps $C \times baseRTT$ plus a number of backlogged packets (in the queue) in transit expecting to receive acknowledgements. For the TCP source to maintain $C \times baseRTT$ number of unacknowledged packets (excluding backlogged packets in the queue) in transit after halving its congestion window upon inferring a packet loss, the bottleneck buffer size $B$ must be equal to $C \times baseRTT$. However, if $B$ has an additional size of twice the product of $C$ and $d_{tar}$ then the TCP source will have $C \times (baseRTT + d_{tar})$ number of unacknowledged packets in transit after halving its congestion window, where $C \times d_{tar}$ represents the number of backlogged packets in the queue shortly after TCP halves its congestion window.

Therefore, in the presence of TCP, it follows that if $B < B_{threshold}$ and TCP halves its congestion window, LEDBAT will estimate the queue delay to be less than the target delay and consequently increase its sending rate. Otherwise, LEDBAT will estimate the queue delay to be greater than the target delay thus decreasing its sending rate to a pre-defined minimum value. We therefore express the congestion window performance of a LEDBAT source in Equation 4 where $w^*$ is LEDBAT congestion window in the presence of TCP and $[w_{min},\ w^*_{upper}]$ represents the oscillation of $w^*$ between $w_{min}$ and some upper bounds $w^*_{upper}$ of LEDBAT congestion window in the presence of TCP.

$$w^*(t) = \begin{cases} w_{min} & \text{if } B \geq B_{threshold} \\ [w_{min},\ w^*_{upper}] & \text{otherwise} \end{cases} \tag{4}$$

Therefore to analyse the scenario shown in Figure 2, bottleneck buffer sizes which satisfy the two conditions given in Equation 4 are chosen so that we can show simulation

Table I
SIMULATION PARAMETER VALUES

| Parameter | Value |
|---|---|
| C (Mb/s) | 2 |
| Other links capacity (Mb/s) | 10 |
| d (ms) | 25 |
| Other links delay (ms) | 5 |
| B (pkts) | 10, 20 |
| $w_{min}$ (pkts) | 1 |
| Traffic source | FTP |
| Packet size (B) | 1500 |
| Number of LEDBAT source | 1 |

results supporting our intuition in Equation 4. Buffer of these sizes are likely to be present in current access routers [18], [19]. A single TCP session can sufficiently fill a bottleneck buffer in a network as multiple TCP sessions, thus a single TCP session is considered in this analysis.

The path delay is $d$. Note that although $d$ is assigned to the bottleneck link in Figure 2, it in fact represents the delay across multiple links. As only small size acknowledgement packets are sent, it is assumed that the queue delay in the reverse direction is 0.

## V. PERFORMANCE ANALYSIS

In this section we present simulation results supporting our intuition described in Equation 4 and showing the limited and fixed LEDBAT throughput in the presence of TCP. Key performance metrics are LEDBAT congestion window, access router queue delay, percentage of bottleneck capacity yielded to TCP by LEDBAT, average LEDBAT throughput. We refer to the percentage of the bottleneck capacity yielded to TCP by LEDBAT to mean the percentage of $C$ allocated to TCP when TCP starts while LEDBAT is in steady state. We use this to quantify the impact of introducing a TCP flow on the LEDBAT throughput.

### A. Simulation Setup

We implemented the LEDBAT congestion control algorithm as a new variant of TCP congestion control mechanism in ns-2.34 [20], [12]. This is because the novel algorithm can be used with any of the existing transport protocols (e.g. TCP, UDP) [1]. TCP timestamping [21] is used so that the LEDBAT sender can determine the one-way delay. Simulation parameter values are listed in Table I while other parameters take their default value in ns-2.34.

We consider the impact of different values of $B$, $w_{min}$, and $C$ on the performance of LEDBAT in the presence of TCP. The following scenarios are simulated:

- LEDBAT starts at time 0 and last for 60 seconds, TCP arrives at time 10 seconds and stops at time 50 seconds.
- TCP starts at time 0 and stops at 50 seconds, LEDBAT arrives at time 10 seconds and last for the entire simulation time of 60 seconds.

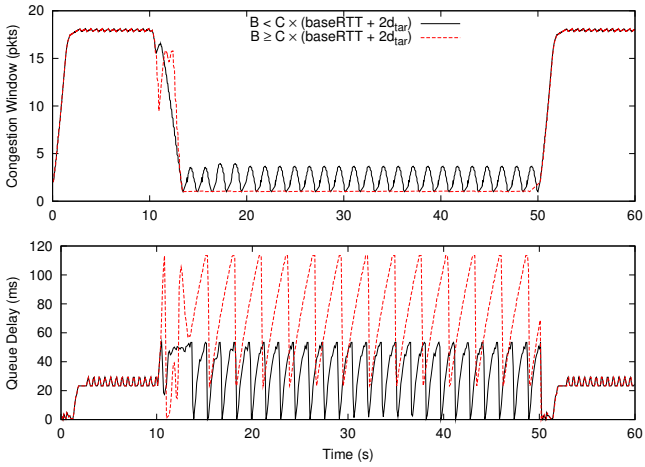


Figure 3. Evolution of the congestion window of LEDBAT and queue delay of packets when LEDBAT starts earlier than TCP for different values of the bottleneck buffer size.

- LEDBAT starts at time 0 while TCP arrives 150 seconds later. Both complete at 300 seconds

### B. LEDBAT Congestion Window and Access router Queue Delay Over Time

In this section the results of LEDBAT congestion window and access router queue delay over time are presented for the 60-second simulation for different values of $B$ and the relative arrival time of TCP to LEDBAT. Although LEDBAT yields to TCP in all the cases considered, different amounts of the bottleneck capacity are yielded to TCP for different values of $B$ and TCP arrival times relative to LEDBAT.

*1) LEDBAT Starting Earlier Than TCP:* As shown in Figure 3, LEDBAT spends most of the time in steady state in the first 10 seconds before TCP arrives when the bottleneck capacity is already saturated and queue delay is near 25ms.

As TCP arrives at 10s for the case where $B \geq B_{threshold}$ i.e. $B = 20$ packets, LEDBAT quickly yields by reducing its congestion window. This is because of the large increase in queue delay caused by the arrival of TCP packets during TCP slow start phase. However once the queue size reaches the maximum, packet loss is experienced by TCP which results in halving of the TCP congestion window. During this time, LEDBAT increases its congestion window after about 1s because queue delay is less than the LEDBAT target delay. As TCP enters congestion avoidance phase, the access router queue starts to build up (even beyond the target), leading to LEDBAT decreasing its congestion window until it reaches the pre-defined minimum congestion window of 1 packet ($w_{min} = 1$ by default in the simulation). As the queue delay of packets does not go below the target indicated by the non-increasing congestion window of LEDBAT from 1 packet in Figure 3, even when TCP reacts to packet loss, LEDBAT congestion window remains at the

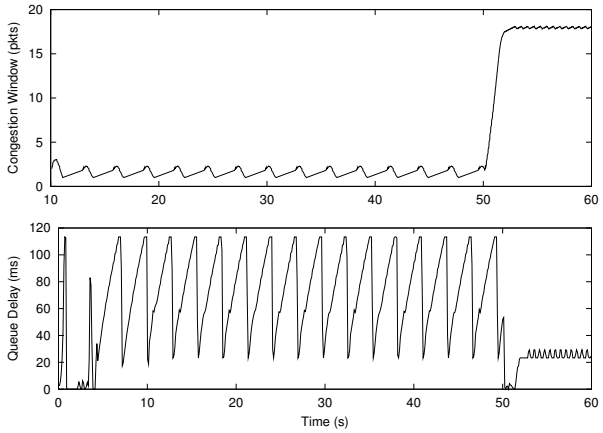


Figure 4. Evolution of the congestion window of LEDBAT and queue delay of packets when TCP starts earlier than LEDBAT

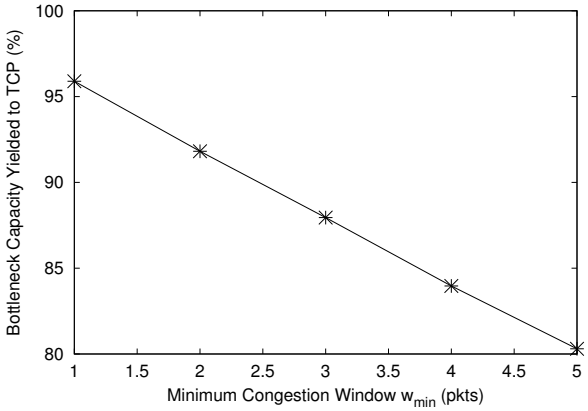


Figure 5. Percentage of the bottleneck link capacity yielded by LEDBAT to TCP as the minimum congestion window $w_{min}$ increases.

minimum level for the rest of the TCP session. Thus, the average congestion window and consequently the throughput of LEDBAT during TCP session in this scenario tends to the minimum congestion window of LEDBAT as the duration of the TCP congestion avoidance phase increases.

However for the case of $B < B_{threshold}$ i.e. $B = 10$ packets, similar observation to the case where $B = 20$ is observed before and shortly after the arrival of TCP at 10s. The difference becomes obvious as TCP enters its congestion avoidance phase. In this phase LEDBAT congestion window does reach 1 packet but increases to about 4 packets as shown in Figure 3. Thus, LEDBAT congestion window oscillates between 1 and 4 packets for the entire session of TCP. The end result is a higher throughput than the case where $B \geq B_{threshold}$.

When the TCP session completes at time 25s, the queue delay drops below the target and LEDBAT soon returns to steady state for all values of $B$.

*2) TCP Starting Earlier Than LEDBAT:* In this section results showing LEDBAT performance when TCP arrives 10s earlier than LEDBAT for $B \geq B_{threshold}$ are given in Figure 4. After 10s LEDBAT congestion window oscillates between 1 and 2 packets. This is because the LEDBAT source measures a base one-way delay as the *actual* base one-way delay plus the queue delay currently caused by TCP. The source then increases its sending rate until the target delay is reached. However when TCP halves its congestion window due to packet loss, LEDBAT estimates the queue delay to be less than target and increases its sending rate. Subsequent packet losses by TCP results in this process being repeated until the end of the TCP session as shown in Figure 4. The end result is a higher average LEDBAT throughput than when LEDBAT starts earlier than TCP.

Therefore, the case of when TCP starts after LEDBAT has reached steady state and when $B \geq B_{threshold}$ represents the worst case scenario for LEDBAT in the presence of TCP. Results in subsequent sections are obtained from the worst case scenario.

### C. Impact of Minimum Congestion Window

The results shown in Figure 5 are measured only when TCP traffic is present for the last 150 seconds in the 300-second simulation. Unsurprisingly, Figure 5 shows how the percentage of the bottleneck capacity obtained by TCP reduces as the minimum LEDBAT congestion window, $w_{min}$, increases. However, the results in Figure 5 rather suggest a potential intra-protocol unfairness among multiple LEDBAT sources sharing the same bottleneck link in the presence of TCP. This could occur if increasing $w_{min}$ would improve the limited LEDBAT throughput in the presence of TCP, the LEDBAT sources used different values of $w_{min}$, and $w_{min}$ was a configurable parameter.

### D. Fixed LEDBAT Throughput with Increasing Bottleneck Capacity

Using a similar setup to Section V-C, we run simulations for different values of $C$ with the default value of $w_{min}$. We set the capacity of all other links to 100Mb/s to ensure that $C$ remains the bottleneck. We set $B$ to 100 packets so that the condition $B \geq B_{threshold}$ still holds for all values of $C$ considered.

Results of the average value of LEDBAT throughput in Figure 6 show that increasing $C$ has no significant impact on LEDBAT throughput. This is because LEDBAT congestion window reverts to a fixed minimum congestion window of $w_{min}$ in the presence of TCP, thus limiting the throughput for the LEDBAT source to a nearly constant value. This is opposite to TCP that obtains an increasing absolute portion of the bottleneck capacity with increasing $C$ (see Figure 6). This may be undesirable for a LEDBAT user, thus necessitating the need for a dynamic minimum congestion

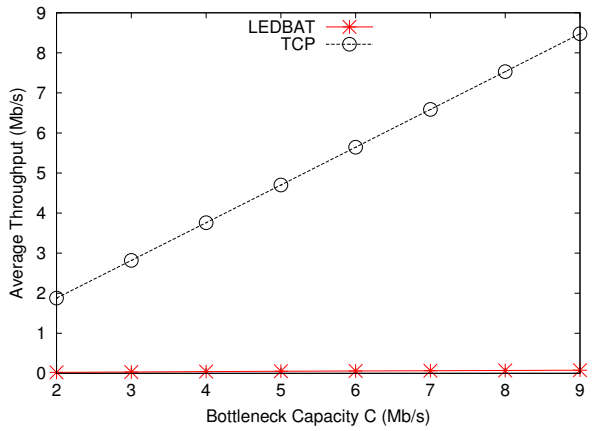


Figure 6. Fixed average LEDBAT throughput and non-fixed average TCP throughput as the bottleneck capacity $C$ increases.

window for LEDBAT that increases as the bottleneck capcity increases.

## VI. Conclusion

This paper has analysed the performance of LEDBAT in the presence of TCP with focus on how different cases of the bottleneck buffer size impact LEDBAT performance in the presence of TCP. From our analysis, we identify the threshold of the bottleneck buffer size that leads LEDBAT to revert to its minimum congestion window in the presence of TCP. Although LEDBAT achieves its objectives of yielding to TCP, for some applications LEDBAT may yield *too much*, hence reducing the usage of the protocol. Our results also show that the minimum congestion window can be used to increase LEDBAT throughput at the expense of reduced TCP throughput. Other disadvantages of LEDBAT's dependence on a fixed minimum congestion window in the presence of TCP are twofold. Firstly intra-protocol unfairness may occur among multiple LEDBAT sources using different values of the minimum congestion window and sharing the same bottleneck link. Secondly the average throughput for LEDBAT is nearly fixed when the bottleneck capacity increases as opposed to TCP that proportionally increases its throughput. The latter may be undesirable for LEDBAT users thus necessitating the need for a dynamic minimum congestion window in the LEDBAT algorithm which we will address in the future.

## References

[1] S. Shalunov, "Low Extra Delay Background Transport (LEDBAT)," IETF Internet Draft, (work-in-progress), Mar. 2010, http://tools.ietf.org/pdf/draft-ietf-ledbat-congestion-00.pdf.

[2] J. Postel, "Transmission control protocol (TCP)," IETF RFC 793, 1981.

[3] F. Kelly, "Mathematical modelling of the Internet," in *Proceedings of the Fourth International Congress on Industrial and Applied Mathematics*, Edinburgh, Scotland, 5–9 Jul. 1999, pp. 105–116.

[4] uTorrent, "Micro transport protocol (UTP)," http://www.utorrent.com/documentation/utp, accessed on 24th September 2010.

[5] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," IETF RFC 2582, Apr. 1999.

[6] M. Allman, V. Paxson, and W. R. Stevens, "TCP congestion control," IETF RFC 2581, Apr. 1999, http://www.ietf.org/rfc/rfc2581.txt.

[7] D. Rossi, C. Testa, and S. Valenti, "Yes, we LEDBAT: Playing with the new BitTorrent congestion control algorithm," in *Passive and Active Measurement (PAM)*, Zurich, Switzerland, Apr. 2010, pp. 31–40.

[8] D. Rossi, C. Testa, S. Valenti, and L. Muscariello, "LEDBAT: the new BitTorrent congestion control protocol," in *Proceedings of the International Conference on Computer Communication Networks*, Zurich, Switzerland, Aug. 2010, pp. 1–6.

[9] G. Carofiglio, L. Muscariello, D. Rossi, and S. Valenti, "The quest for LEDBAT fairness," in *Proceedings of IEEE Globecom*, Miami, FL, Dec. 2010, p. (to appear).

[10] G. Carofiglio, L. Muscariello, D. Rossi, and C. Testa, "A hands-on assessment of transport protocols with lower than best effort priority," in *Proceedings of the 35th IEEE Conference on Local Computer Networks*, Denver, CO, Oct. 2010, p. (to appear).

[11] M. I. Andreica, N. Tapus, and P. Johan, "Performance evaluation of a python implementation of the new LEDBAT congestion control algorithm," in *Proceedings of IEEE International Conference on Automation, Quality and Testing Robotics (AQTR)*, Cluj-Napoca,Romania, May 2010, pp. 1–6.

[12] A. J. Abu and S. Gordon, "A dynamic algorithm for stabilising LEDBAT congestion window," in *Proceedings of the International Conference on Computer and Network Technology*, Bangkok, Thailand, Apr. 2010, pp. 157–161.

[13] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[14] A. Venkataramani, R. Kokku, and M. Dahlin, "TCP Nice: A mechanism for background transfers," in *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, Boston, MA, 9–11 Dec. 2002, pp. 329–343.

[15] A. Kuzmanovic and E. W. Knightly, "TCP-LP: Low-priority service via end-point congestion control," *IEEE/ACM Transactions on Networking*, vol. 14, no. 4, pp. 739–752, Aug. 2006.

[16] M. Welzl, "A survey of lower-than-best effort transport protocols," IETF Internet Draft, (work-in-progress), Mar. 2010.

[17] A. Akella, S. Seshan, and A. Shaikh, "An empirical evaluation of wide-area Internet bottlenecks," in *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, San Diego, CA, USA, 27–29 Oct. 2003, pp. 316–317.

[18] Y. Ganjali and N. McKeown, "Update on buffer sizing in Internet routers," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 67–70, Oct. 2006.

[19] R. S. Prasad, C. Dovrolis, and M. Thottan, "Router buffer sizing for TCP traffic and the role of the output/input capacity ratio," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1645–1658, Oct. 2009.

[20] ns-2, "Network Simulator," http://www.isi.edu/nsnam.

[21] V. Jacobson, B. Braden, and D. Borman, "TCP extensions for high performance," IETF RFC 1323, May 1992.