Network Working Group
Internet-Draft
Intended status: Informational
Expires: February 26, 2021

P. Balasubramanian
O. Ertugay
D. Havey
Microsoft

August 25, 2020

LEDBAT++: Congestion Control for Background Traffic
draft-irtf-iccrg-ledbat-plus-plus-01

Abstract

This informational memo describes LEDBAT++, a set of enhancements to the LEDBAT (Low Extra Delay Background Transport) congestion control algorithm for background traffic. The LEDBAT congestion control algorithm has several shortcomings that prevent it from working effectively in practice. LEDBAT++ extends LEDBAT by adding a set of improvements, including reduced congestion window gain, modified slow-start, multiplicative decrease and periodic slowdowns. This set of improvement mitigates the known issues with the LEDBAT algorithm, such as latency drift, latecomer advantage and inter-LEDBAT fairness. LEDBAT++ has been implemented as a TCP congestion control algorithm in the Windows operating system. LEDBAT++ has been deployed in production at scale on a variety of networks and been experimentally verified to achieve the original stated goals of LEDBAT.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 26, 2021.

Copyright Notice

Table of Contents

1. Introduction

Operating systems and applications use background connections for a variety of tasks, such as software updates, large media downloads, telemetry, or error reporting. These connections should operate without affecting the general usability of the system. Usability is measured in terms of available network bandwidth and network latency. LEDBAT [RFC6817] is designed to minimize the impact of lower than best effort connections on the latency and bandwidth of other connections. To achieve that, each LEDBAT connection monitors the transmission delay of packets, and compares them to the minimum delay observed on the connection. The difference between the transmission delay and the minimum delay is used as an estimate of the queuing

delay. If the queuing delay is above a target, LEDBAT directs the connection to reduce its bandwidth. If the queuing delay is below the target, the connection is allowed to increase its transmission rate. The bandwidth increase and decrease are proportional to the difference between the observed values and the target. LEDBAT reacts to packet losses and other congestion signals in the same way as standard TCP.

However, there are a few issues that plague LEDBAT, some previously documented, and some discovered by experiments. LEDBAT++ adds additional mechanisms on top of (and in some cases deviates from) LEDBAT to overcome these problems. The remaining sections describe the problems and the mechanisms in detail. The objective of this informational RFC is to document LEDBAT++ enhancements on top of a base LEDBAT implementation in the Windows operating system. encourage its use so the algorithm can be further verified and improved.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. LEDBAT Issues

This section lists some known LEDBAT issues from existing literature and also list some new problems observed as a result of experimentation with an implementation of [RFC6817].

3.1. Latecomer advantage

Delay based congestion control protocols like LEDBAT are known to suffer from a latecomer advantage. When the newcomer establishes a connection, the transmission delay that it encounters incorporates queuing delay caused by the existing connections. The newcomer considers this large delay the minimum, and thereby increases its transmission rate while other LEDBAT connections slow down. Eventually, the latecomer will end up using the entire bandwidth of the connection. Standard TCP congestion control as described in [RFC0793] and [RFC5681], causes some queuing, the LEDBAT delay measurements incorporate that queuing, and the base delay as measured by the connection is thus set to a larger value than the actual minimum. As a result, the queues remain mostly full. In some cases, this queuing persists even after the closing of the competing TCP connection. This phenomenon was already known during the design of LEDBAT, but there is no mitigation in the LEDBAT design. The designers of the protocol relied instead on the inherent burstiness of network traffic. Small gaps in transmission schedules would allow

the latecomer to measure the true delay of the connection. This reasoning is not satisfactory because workloads can upload large amount of data, and would not always see such gaps.

3.2. Inter-LEDBAT fairness

The latecomer advantage is caused by the improper evaluation of the base delay, with the latecomer using a larger value than the preexisting connections. However, even when all competing connections have a correct evaluation of the base delay, some of them will receive a larger share of resource. The reason for that persistent unfairness is explained in [RethinkLEDBAT]. LEDBAT specifies proportional feedback based on a ratio between the measured queuing delay and a target. Proportional feedback uses both additive increases and additive decreases. This does stabilize the queue sizes, but it does not guarantee fair sharing between the competing connections.

3.3. Latency drift

LEDBAT estimates the base delay of a connection as the minimum of all observed transmission delays over a 10-minute interval. It uses an interval rather than a measurement over the whole duration of the connection, because network conditions may change over time. For example, an existing connection may be transparently rerouted over a longer path, with a longer transmission delay. Keeping the old estimate would then cause LEDBAT to unnecessarily reduce the connection throughput. However experiments show that this causes a ratcheting effect when LEDBAT connections are allowed to operate for a long time. The delay feedback in LEDBAT causes the queuing delay to stabilize just below the target. After an initial interval, all new measurements are equal to the initial transmission delay plus a fraction of the target. Every 10 minutes, the measured base delay increases by that fraction of the target queuing delay, leading to potentially large values over time.

3.4. Low latency competition

LEDBAT compares the observed queuing delays to a fixed target. The target value cannot be set too low, because that would cause poor operation on slow networks. In practice, it is set to 60ms, a value that allows proper operation of latency sensitive applications like VoIP. But if the bottleneck buffer is small such that the queuing delay will never reach the target, then the LEDBAT connection behaves just like an ordinary connection. It competes aggressively, and obtains the same share of the bandwidth as regular TCP connections. On high speed links the problem is exacerbated.

3.5. Dependency on one-way delay measurements

The LEDBAT algorithm requires use of one-way delay measurements. This makes it harder to use with transport protocols like TCP that have no reliable way to obtain one way delay measurements. TCP timestamps do not standardize clock frequency, and the endpoints will need to rely on heuristics to guess the clock frequency of the remote peer to detect and correct for clock skew. TCP timestamps do not include clock synchronization, and would need some non-standard invention to compensate for clock skew. Any such mechanism is very fragile.

4. LEDBAT++ Mechanisms

4.1. Modified slow start

Traditional initial slow start can cause spikes in bandwidth usage. However skipping exponential congestion window increase results in really poor performance on long delay links. LEDBAT++ applies the dynamic GAIN parameter to the congestion window increases. In standard TCP operation, the congestion window increases for every ACK by exactly the amount of bytes acknowledged. A LEDBAT++ sender increases the congestion window by that number multiplied by the dynamic GAIN value. In low latency links, this ensures that LEDBAT++ connections ramp up slower than regular connections. LEDBAT++ sender limits the initial window to 2 packets. LEDBAT++ sender monitors the transmission delays during the slow start period. If the queuing delay is larger than 3/4ths of the target delay, exit slow start and immediately move to the congestion avoidance phase. After initial slow start, the increase of congestion window is bounded by the SSTHRESH estimate acquired during congestion avoidance, and the risk of creating congestion spikes is very low. Exit slow start in on excessive delay SHOULD be applied only during the initial slow start.

4.2. Slower than Reno increase

When the queuing delays are below the target delay, LEDBAT behaves like standard TCP [RFC0793]. LEDBAT introduces a GAIN parameter which can be set between 0 and 1. In order to solve the low latency competition problem, LEDBAT++ makes the GAIN parameter dynamic. When standard and reduced connections share the same bottleneck, they experience the same packet drop rate. The GAIN value ensures that the throughput of the LEDBAT connection will be a fraction (1/SQRT(1/GAIN)) of the throughput of the regular connections. Small values of GAIN work well when the base delay is small, and ensure that the LEDBAT connection will yield to regular connections in these networks. However, large values of GAIN do not work well on long delay links. In the absence of competing traffic, combining large

base delays with small GAIN values causes the connection bandwidth to remain well under capacity for a long time. In LEDBAT++, GAIN is a function of the ratio between the base delay and the target delay:

```
GAIN = 1 / (min (16, CEIL (2*TARGET/base)))
```

where CEIL(X) is defined as the smallest integer larger than X. Implementations MAY experiment with the constant value 16 as a tradeoff between responsiveness and performance.

4.3. Multiplicative decrease

[RethinkLEDBAT] suggests combining additive increases and multiplicative decreases in order to solve the Inter-LEDBAT fairness problem. It proposes to change the way LEDBAT increases and decreases the congestion window based on the ratio between the observed delay and the target. Assuming that the congestion window is changed once per roundtrip measurement. In standard LEDBAT, the per RTT window when delay is less than target is:

```
W += GAIN * (1 - delay/target)
```

In LEDBAT++, with multiplicative decrease, the per RTT window when delay is less than target is:

```
W += GAIN
```

Similarly in standard LEDBAT, the per RTT window when the delay is higher than target is:

```
W -= GAIN * (delay/target - 1)
```

In LEDBAT++, with multiplicative decrease, the per RTT window delay is higher than target is:

```
W += max( (GAIN - Constant * W * (delay/target - 1)), -W/2) )
```

It is RECOMMENDED that the Constant be set to 1. Implementations MAY experiment with this value. If the connections have different estimates of the base delay, capping the multiplicate decrease to at most W/2 is required. Otherwise, spikes in delay can cause the window to immediately drop to its minimal value. LEDBAT++ sender MUST also ensure that the congestion window never decreases below 2 packets, in order to avoid completely starving the connection.

4.4. Initial and periodic slowdown

The LEDBAT specification assumes that there will be natural gaps in traffic, and that during those gaps the observed delay corresponds to a state where the queues are empty. However, there are workloads where the traffic is sustained for long periods. This causes base delay estimates to be inaccurate and is one of the major reasons behind latency drift as well as the lack of inter-LEDBAT fairness. To ensure stability, LEDBAT++ forces these gaps, or slow down periods. A slowdown is an interval during which the LEDBAT++ connection voluntarily reduces its traffic, allowing queues to drain and transmission delay measurements to converge to the base delay. The slowdown works as follows:

- Upon entering slowdown, set SSTHRESH to the current version of the congestion window CWND, and then reduce CWND to 2 packets.

- Keep CWND frozen at 2 packets for 2 RTT.

- After 2 RTT, ramp up the congestion window according to the slow start algorithm, until the congestion window reaches SSTHRESH.

Keeping the CWND frozen at 2 packets for 2 RTT allows the queues to drain, and is key to obtaining accurate delay measurements. The initial slowdown starts shortly after the connection completes the initial slow start phase; 2 RTT after the initial slow start completes. After the initial slowdown, LEDBAT++ sender performs periodic slowdowns. The interval between slowdown is computed so that slowdown does not cause more than a 10% drop in the utilization of the bottleneck. LEDBAT++ sender measures the duration of the slowdown, from the time of entry to the time at which the congestion window regrows to the previous SSTHRESH value. The next slowdown is then scheduled to occur at 9 times this duration after the exit point. The combination of initial and periodic slowdowns allows competing LEDBAT connections to obtain good estimates of the base delay, and when combined with multiplicative decrease solves both the latecomer advantage and the Inter-LEDBAT fairness problems.

4.5. Use of Round Trip Time instead of one way delay

LEDBAT++ uses Round Trip Time measurements instead of one way delay. One possible shortcoming of round trip delay measurements is that they incorporate queuing delays in both directions. This can lead to unnecessary slowdowns, such as slowing down an upload connection because a download is saturating the downlink but in practice this seems to benefit the workloads because bottleneck link can carry ACK traffic in the other direction for the competing flows. Round trip measurements also include the delay at the receiver between receiving

a packet and sending the corresponding acknowledgement. These delays are normally quite small, except when the delayed acknowledgment logic kicks in. Effect of delayed ACK can be particularly acute when the congestion window only includes a few packets, for example at the beginning of the connection.

The problems of using one way delay are mitigated through a set of implementation choices. First, LEDBAT++ sender enables the TCP Timestamp option, in order to obtain RTT samples with each acknowledgement. A LEDBAT++ sender SHOULD filter the round trip measurements by using the minimum of the 4 most recent delay samples, as suggested in the LEDBAT specification. Finally, the queueing delay target is set larger than the typical TCP maximum acknowledgement delay. This avoids over reacting to a single delayed ACK measurement. LEDBAT++ default delay target of 60ms is different from the 100ms value recommended in [RFC6817].

5. Deployment Issues

LEDBAT++ is a sender-side algorithmic improvement. This implies that for many workloads it requires changes to the servers serving content. It does not address workloads or scenarios where the only entities that can be updated are clients.

Transparent proxies prevent measurement of end-to-end delay and might interfere with the effective operation of LEDBAT++.

The interaction between Active Queue Management (AQM) and LEDBAT++ is an area of research.

6. Security Considerations

LEDBAT++ enhances LEDBAT and inherits the general security considerations discussed in [RFC6817].

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgements

The LEDBAT++ algorithm was designed and implemented by Osman Ertugay, Christian Huitema, Praveen Balasubramanian, and Daniel Havey.

9. References

9.1. Normative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/info/rfc5681>.

[RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, DOI 10.17487/RFC6817, December 2012, <https://www.rfc-editor.org/info/rfc6817>.

9.2. Informative References

[RethinkLEDBAT] Carofiglios, G., Muscariello, L., Rossi, D., Testa, C., and S. Valenti, "Rethinking the Low Extra Delay Background Transport (LEDBAT) Protocol", Computer Networks, Volume 57, Issue 8, 4 June 2013, Pages 1838-1852, 2013, <http://perso.telecom-paristech.fr/~drossi/paper/rossi13comnet.pdf>.

Authors' Addresses

Praveen Balasubramanian
Microsoft
One Microsoft Way
Redmond, WA 98052
USA

Phone: +1 425 538 2782
Email: pravb@microsoft.com

Osman Ertugay
Microsoft

Phone: +1 425 706 2684
Email: osmaner@microsoft.com

Daniel Havey
Microsoft

Phone: +1 425 538 5871
Email: dahavey@microsoft.com