LEDBAT++: Low priority TCP Congestion Control in Windows

Praveen Balasubramanian
pravb@microsoft.com
Background

• Software updates, telemetry, or error reporting
• Should not interfere with Skype call, interactive web browsing, etc.
• Four solutions to “lower than best effort” service:
  • Delay based transport protocols
  • Congestion window update algorithm designed to be “less aggressive”
  • Application level solutions such as BITS, monitor network usage
  • Network assisted solutions flag the packets as low priority
• In 2016, started exploring LEDBAT
  • Easier to deploy and maintain than app layer alternatives e.g. BITS
  • Versus waiting for hypothetical universal support in the network
• Literature survey and our own experiments found issues with LEDBAT as described in the RFC
LEDBAT (RFC 6817) Brief Recap

• Low Extra Delay Background Transport
• Minimize the impact of “lower than best effort” connections on the latency and bandwidth of other connections
• Compare measured delay with the base delay
• If delay is less than target, additive increase
• If delay is higher than target, additive decrease
• No strict requirements on slow start (with a suggestion to avoid)
• React to packet loss and ECN like standard TCP
Problems with LEDBAT

• One-way delay measurements are hard with TCP
  • No standard clock frequency or synchronization
  • Clock skew

• Latecomer advantage
  • Reliance on inherent burstiness of network traffic to detect base delay

• Inter-LEDBAT fairness
  • Proportional feedback uses both additive increases and decreases, stable queue but no fair sharing
  • Carofiglio, G. et al. “Rethinking the LEDBAT Protocol”

• Somewhat vague recommendations regarding slow start

• Latency drift
  • Impacts long running LEDBAT connections

• Low latency competition
  • If bandwidth is large, queueing delay never exceeds the fixed target
Introducing LEDBAT++

• LEDBAT++ comprises of the following
  • Round trip latency measurements
  • Slower than Reno cwnd increase with adaptive gain factor
  • Multiplicative cwnd decrease with adaptive reduction factor
  • Modified slow start
  • Initial and periodic slowdown

• Part of Windows 10 since Anniversary Update
• Internal API currently in use by WER (Windows Error Reporting) and Windows Update Delivery Optimization
• Working on making the API and config public in future releases
Round trip latency

• Advantages
  • Already available in TCP
  • No need for clock synchronization

• Disadvantages
  • Incorporates queuing delay in both directions
  • Receiver delays and delayed ACKs

• Mitigations
  • Erring on the side of higher latency estimation is acceptable
  • Enable TCP timestamp option implicitly for LEDBAT connections
  • Filter the RTT samples (minimum of the 4 most recent samples)
  • Use a TARGET delay of 60 ms
    • Larger than typical* server ACK delay (50ms)
    • 100 msec consumes 2/3rd of budget for 150 msec maximum acceptable delay for VoIP
Slower than Reno

• Reno
  • On packet loss: $W \leftarrow W/2$
  • On packet acknowledgement: $W \leftarrow 1/W$

• Introduce a reduction factor $F$:
  • On packet loss: $W \leftarrow W/2$
  • On packet acknowledgement: $W \leftarrow 1/(F \times W)$

• Throughput of LEDBAT++ connection will be a fraction $(1/\sqrt{F})$ of the throughput of regular TCP connection

• Based on experimentation we picked an adaptive scheme for $F$
  • $F = \min (16, \lceil 2 \times \text{TARGET}/\text{base} \rceil)$
  • 16 is a good tradeoff between responsiveness and performance

• Solves low latency competition problem
Multiplicative Decrease

- Carofiglio, G. et al “Rethinking the Low Extra Delay Background Transport (LEDBAT) Protocol” suggest multiplicative decrease

<table>
<thead>
<tr>
<th></th>
<th>Standard LEDBAT, per RTT</th>
<th>Multiplicative decrease, per RTT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay lower than target</td>
<td>( W += \text{Gain} \times (1 - \text{delay/target}) )</td>
<td>( W += \text{Gain} )</td>
</tr>
<tr>
<td>Delay larger than target</td>
<td>( W -= \text{Gain} \times (\text{delay/target} - 1) )</td>
<td>( W += \text{Gain} - \text{Constant} \times W \times (\text{delay/target} - 1) )</td>
</tr>
</tbody>
</table>

- Only works when all connections measure same base delay, so
  - Use constant value of 1 and cap the multiplicative decrease coefficient to be at least 0.5
  - Ensure that cwnd never decreases below 2 packets

- Solves the Inter-LEDBAT fairness problem
Modified slow start

• Skipping slow start results in really poor performance on long delay links

• Slower than Reno ramp up
  • Apply the adaptive reduction factor F to the congestion window increases
  • Limit the initial cwnd to 2 packets

• If queuing delay is larger than $3/4^{th}$s of the TARGET, exit slow start
  • Immediately move to the “congestion avoidance” phase

• Only apply the “exit on excessive delay” during the initial slow start
  • Subsequent slow starts capped by recorded ssthresh
Initial and periodic slowdown

• Traffic is sustained for long periods
  • Inaccurate base delay estimates
  • Causes latency drift as well as the lack of inter-LEDBAT fairness

• Force gaps for measuring base delay, or “slowdown” periods
  • “slowdown” is an interval during which the LEDBAT++ connection voluntarily reduces its traffic
  • Upon entering slowdown, set ssthresh = cwnd, and reduce cwnd to 2 packets
  • Keep CWND frozen at 2 packets for 2 RTT
  • After 2 RTT, ramp up according to “slow start” until cwnd reaches ssthresh

• Initial slowdown 2*RTT after first slow start exit

• Periodic slowdown – not more than 10% drop in throughput
  • Measure duration of slowdown from entry to ramp up to ssthresh
  • Schedule next slowdown 9 times this duration

• Solves the latency drift problem
Bandwidth sharing with normal priority traffic

Blue: Standard TCP
Purple: Short flows
Red: LEDBAT++
Purple: Short flows
Reduced latency impact of LEDBAT++
Handling latency drift

- Standard LEDBAT
- Mult. Decrease
- Mult. Decrease + slowdowns
Latecomer advantage & Inter-LEDBAT fairness

![Graphs showing comparison between Standard LEDBAT and LEDBAT++]

1. Bytes Sent per Second at Time (Aggregation: Sum)
2. Ack Delay per Second at Time (Aggregation: Min)
Handling low latency competition

Blue: Standard TCP
Red: LEDBAT++, Fixed F=1

Blue: Standard TCP
Red: LEDBAT++, adaptive F=16
Conclusion

• We found several shortcomings of LEDBAT as a solution for background connections
• LEDBAT++ is an attempt to overcome these problems
• Experiments show that LEDBAT++ addresses the shortcomings
• LEDBAT++ is already deployed and used on millions of systems
• Working on making API and knob public
• Working on a draft submission
  • Should it be to iccrg or tcpm?
References and Q&A

- LEDBAT working group page: [https://datatracker.ietf.org/wg/ledbat/charter/](https://datatracker.ietf.org/wg/ledbat/charter/)

Q&A