Inferring the buffering delay of remote BitTorrent peers under LEDBAT vs TCP

C. Chirichella[1,3], D. Rossi[1], C. Testa[1], T. Friedman[2], A. Pescape'[3]
[1] Telecom ParisTech, first.last@enst.fr
[2] UPMC Sorbonne Universite, timur.friedman@upmc.fr
[3] Univ. Federico II, Napoli, first.last@unina.it

***Abstract*—Nowadays, due to excessive queuing, Internet delays grow sometimes as large as the propagation delay from moon to earth – for which the bufferbloat term was recently coined. Some points to active queue management (AQM) as its solution, others propose end-to-end congestion control techniques – like BitTorrent that recently replaced TCP with the LEDBAT transport protocol.**

In this demo, we implement a methodology to *monitor the upstream queuing delay experienced by remote hosts*, both those using LEDBAT, through LEDBAT's native one-way delay measurements, and those using TCP, through the timestamp option. By actively taking part into torrent downloads as leechers, our software is able to infer (and visualize) the amount of access delay suffered by the remote peers.

I. Motivations

As recently pointed out in [1], "Internet delays now are as common as they are maddening". The root cause for these delays can be identified with the excess buffering inside a network, which is nicknamed "bufferbloat". Though this is nothing new [2], the situation got worse in the latest years due to mainly two facts: (i) TCP loss-based design, that forces the bottleneck buffer to fill before the sender reduces his rate and (ii) relatively large memories in front of low-capacity ADSL and Cable uplink that translate into significant queuing delay (up to few seconds [4]).

While [1] points out *local active queue management* (AQM) techniques (e.g., affecting the scheduling and discard of packets in the buffer differently from a traditional FIFO discipline) as the ultimate solution to reduce queuing delay, it forgets however another important orthogonal direction: namely, the engineering of *end-to-end flow and congestion control* techniques alternative to TCP. Congestion control may have different goals, such as controlling the streaming rate over TCP connections as done by YouTube or Netflix, or aggressively protecting user QoE as done by Skype over UDP, or to provide bulk transfers service such as Picasa background upload option, Dropbox synchronization or Microsoft Background Intelligent Transfer Service (BITS).

The latest addition to the congestion control field is represented by BitTorrent "Low Extra Delay Background Transport" (LEDBAT), that focuses on bounding the maximum bufferbloat induced in the network, while permitting an efficient utilization of link resources at the same time. Shortly, LEDBAT is a delay-driven congestion control protocol, where the growth and shrink of the congestion window depends on the distance of the estimated queuing delay from a maximum *target* queuing delay. This target queuing delay is by default 100 ms, and is thus well lower than the maximum queuing delay caused by TCP bufferbloat. The protocol, which is defined as an IETF draft [7] (focused on the algorithmic aspects) and as a BEP [5] (focused on the UDP framing), *has recently become BitTorrent default congestion control protocol, replacing thus TCP.* According to a post by Brahm Cohen, and to our own measurements, *about half of the BitTorrent traffic is now carried over LEDBAT.*

Overall, while the direction taken by LEDBAT is helpful in reducing the bufferbloat problem, it is unclear whether a partial LEDBAT deployment can suffices to releave the bufferbloat: i.e., as only half of BitTorrent traffic goes over LEDBAT, the remaining half still goes over TCP, which can lead to bufferbloat anyway. Besides, not all user traffic is carried over BitTorrent, so that other data-intensive application using TCP can still force bufferbloat to happen.

II. Demo aim

Our demo software infers the queuing delays of remote LEDBAT and TCP hosts. As Fig. 1 depicts, the software is based on passive analysis of BitTorrent traffic running at a local monitor peer m. The local peer participates into torrents as a regular leecher, and exchanges with remote peers r_i data over either LEDBAT (when available) or TCP (legacy clients). In a nutshell, we reconstruct, at the local receiver, the state of the buffer as the remote sender would do (more detail in Sec. III). Users can interact with the demo by controlling one of the peers c participating into the torrent, e.g., by injecting TCP traffic to a controlled server s: this crafted traffic competes with the BitTorrent traffic, actively creating a bufferbloat that m can gauge.

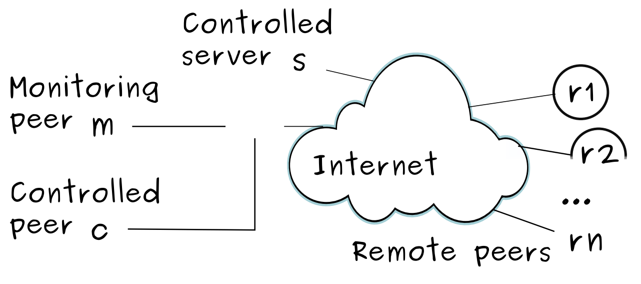

Fig. 1. Demo synopsis.

The demo visualizes then bufferbloat delay statistics, both per-peer and in aggregate form. We point out that upstream access delay is an important metric for all delay sensitive applications: this includes not only gaming and VoIP (that are severely affected by delay and jitter), but also Web browsing (as upstream access delay influences the duration of DNS resolution, TCP three-way handshake, HTTP GET, etc.).

We point out that while previous work on LEDBAT exists, it mostly adopt a simulative approach and are thus out of scope. Experimental work on BitTorrent and LEDBAT such as [3], [6], [8] has instead so far addressed different problems. More precisely, [3] proposes algorithms to improve OWD estimate against clock skew and drifts, while in [6] we performed testbed experiments on early versions of the LEDBAT protocol [6], and in [8] we focus on the impact of LEDBAT vs TCP on the swarm completion time.

This work is the first to study LEDBAT from a novel, exciting, bufferbloat perspective. We point out that while [8] focuses on the primary quality of experience metrics for the BitTorrent application (i.e., download time), in the methodology shown in this demo is able to we gauge an important quality of service metric (i.e., access delay) addressing thus a complementary aspect to [8] – i.e., the impact of BitTorrent on the quality of other, delay-sensitive, applications.

III. Methodology

The methodology we propose is based, as the LEDBAT protocol itself, on One Way Delay (OWD) measurement. OWD is composed by propagation, transmission, processing and queuing delays. Neglecting the processing delay, propagation and transmission delays are constant components, while the only variable component is the *queuing* delay. Intuitively, a packet which finds the queue empty (i.e., null queuing delay) will accurately estimate the constant portion of the OWD (i.e., the sum of propagation and transmission delays). This measure yields a minimum of the delay, that will be stored as a reference (or "base delay" in LEDBAT terminology). Queuing delay can then be estimated as the difference between the current delay samples and the base delay.

We now sketch (a simplified view of) the methodology to measure the queuing delay in the LEDBAT case, where the OWD is directly available in the packet header. In the TCP case, not shown for lack of space, queuing delay can be estimated by exploiting the timestamp option.

As Fig. 2 shows (employing the terminology of the LEDBAT draft [7]), delay measurements are performed collaboratively by the transceivers. In particular, the sender timestamps all sent packets with its local clock, using a specific header field. On reception of any new packet, the receiver calculates the OWD as the difference between its own local clock and the timestamp carried by the LEDBAT header, and sends it back to the sender (using another field of the LEDBAT header). In this way, any acknowledgement packet carries the estimated OWD suffered by the last received data packet in the opposite direction.

In practice, by merely monitoring a stream of LEDBAT packets, it is possible to infer the amount of queuing delay of

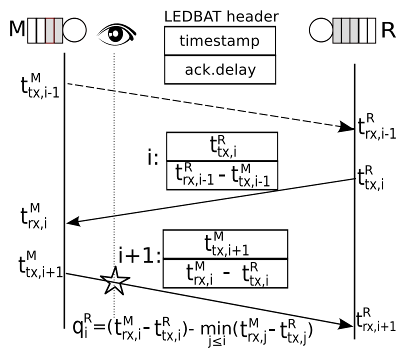

Fig. 2. The monitor M can infer the queue size of the remote peer R.

both the local and the remote host. To infers the queue size of a remote host R, a monitor co-located (or close to a probe host) M simply need to perform the same state update as if he was the intended destination of the sniffed packet,

- it first updates the base delay β^R as the minimum over all OWD samples received from that peer, i.e., $\beta^R = \min(\beta, t^M_{rx,i} - t^R_{tx,i})$
- it then evaluate the queuing delay q^R_i incurred by the i-th packet sent from R by subtracting the base delay β_R from the timestamp difference carried in the $(i+1)$-th packet in the opposite direction, i.e., $q^R_i = (t^M_{rx,i} - t^R_{tx,i}) - \beta^R$ (notice that the *OWD difference* cancels the clock offset between M and R; additionally, clock drift could be corrected as in [3]).

With a similar technique, we can compute delays in case of TCP peers as well. Individual samples are then collected, aggregated and visualized by the demo software.

At the same time, while the demo is able to show the status of remote peer bufferbloats on live torrents, we are aware that a larger set of experiments is necessary in order to build a picture representative of an Internet-wide situation. As such, we have not only validated our methodology in a local testbed (against different ground truths) but are also currently running a wide measurement campaign involving several torrents (which is however well beyond the aim of this demo).

References

[1] V. Cerf, V. Jacobson, N. Weaver, and J. Gettys. Bufferbloat: what's wrong with the internet? *Commun. ACM*, 55(2):40–47, Feb. 2012.

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

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

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

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

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

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

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