BBR TCP (Bottleneck Bandwidth and RTT)

BBR is a new algorithm for TCP Congestion Control. It was tested in Google’s data center networks as well as on some of their public-facing Web servers including Google.com and YouTube. It strives to optimize both throughput and latency/RTT by estimating the bottleneck bandwidth and RTT to compute a pacing rate. One goal — one that sets it apart from most traditional TCP variants — is to avoid filling up the bottleneck buffer, which might induce Bufferbloat.

The BBR team developed an improved version, BBRv2, to address some shortcomings with the original version of BBR. BBRv2 should no longer starve instances of other TCP implementations (e.g. Reno or CUBIC) sharing the same bottleneck, or induce high loss rates when the queue at the bottleneck is limited. BBRv2 also supports some forms of ECN signals; not the "classical" version where each "congestion experienced" bit is supposed to be interpreted like a lost packet for congestion control, but the newer “DCTCP-style” ECN variants, including Prague TCP/L4S. Unlike BBRv1, BBRv2 interprets loss as a signal, and tries to respect a (configurable) target loss rate ceiling.

Operation

A description of the BBR (v1) algorithm was published in the September/October 2016 issue of ACM Queue. An implementation in the Linux kernel has been proposed as a patch. Dave Taht posted a preliminary evaluation (“a quick look...”) on his blog. An good description of the BBR’s motivation and approach is included in the proposed kernel patch (see below).

In the STARTUP phase, BBR tries to quickly approximate the bottleneck bandwidth. It does so by increasing the sending rate until the estimated bottleneck bandwidth stops growing.

Bottleneck bandwidth is estimated from the amount of data ACKed over a given period, filtered through a "windowed max-filter".

In the DRAIN phase, the sending (pacing) rate is reduced to get rid of the queue that BBR estimates to have created while probing the bottleneck bandwidth during STARTUP.

In steady state, BBR will pace to the estimated bottleneck bandwidth. Periodically it tries to improve its network model by doing probing:

PROBE_BW mode: BBR probes for a bandwidth increase at the bottleneck by increasing the pacing rate, then decreasing the rate to remove temporary queuing in case the bottleneck bandwidth hasn’t grown.

PROBE_RTT mode: RTT is filtered through a windowed min-filter. Sometimes the algorithm will reduce the pacing rate to better approximate the base RTT in case queueing ("bufferbloat") is in effect.

State Diagram

A state diagram has been included in the Linux kernel source as a patch. I quote:

Here is a state transition diagram for BBR:

```
  | V
+----> STARTUP ----+
  | V
  | DRAIN ----+
  | V
+----> PROBE_BW ----+
  | V
  | PROBE_RTT <---
```

A BBR flow starts in STARTUP, and ramps up its sending rate quickly. When it estimates the pipe is full, it enters DRAIN to drain the queue. In steady state a BBR flow only uses PROBE_BW and PROBE_RTT.

A long-lived BBR flow spends the vast majority of its time remaining (repeatedly) in PROBE_BW, fully probing and utilizing the pipe’s bandwidth in a fair manner, with a small, bounded queue. If a flow has been continuously sending for the entire min_rtt window, and hasn’t seen an RTT sample that matches or decreases its min_rtt estimate for 10 seconds, then it briefly enters PROBE_RTT to cut inflight to a minimum value to re-probe the path’s two-way propagation delay (min_rtt). When exiting PROBE_RTT, if we estimated that we reached the full bw of the pipe then we enter PROBE_BW; otherwise we enter STARTUP to try to fill the pipe.

BBRv2 is based on the same state machine. The new behavior is illustrated in slides 20–26 of the IETF-104 presentation, see under “References” below.
Implementation

Linux kernel TCP implementation

Linux 4.9 and above: An implementation for the Linux kernel was submitted and merged in September 2016. This initial kernel implementation relied on a scheduler that is capable of pacing, such as the fq scheduler.

Linux 4.13 and above: In May 2017, Éric Dumazet submitted a patch to implement pacing in TCP itself, removing the dependency on the fq scheduler. This makes BBR simpler to enable, and allows its use together with other schedulers (such as the popular fq_codel).

TCP_CC_INFO instrumentation

BBR exports its bandwidth and RTT estimates using the getsockopt (TCP_CC_INFO) interface, see struct tcp_bbr_info. User-space applications can use this information as hints for adapting their use of a given connection, e.g. by selecting appropriate audio or video encodings.

BBRv2

An "alpha/preview" version of BBRv2 is available under a "v2alpha" branch on GitHub. An implementation of BBRv2 for QUIC is included in the Chromium source as bbr2_sender.cc and bbr2_sender.h.

Other implementations

Netflix is working on an implementation of BBR for FreeBSD. BBR has also been implemented for QUIC (including BBRv2, see above), and is in active use on Google's servers. Mark Claypool has written an implementation of BBR for ns-3, a popular network simulator.

History/Related Work

There is another instance of "BBR TCP", namely Bufferbloat Resistant TCP proposed by Michael Fitz Nowlan in his 2014 Ph.D. thesis (Yale library catalog entry, PDF). It shares the basic goals of (Google's) BBR TCP, namely to detect bottleneck rate and to avoid bufferbloat. But it uses a delay-based measurement approach rather than direct measurement of the amount of ACKed data per time, and doesn't seem to make use of pacing. Despite these differences, the link between the two BBR TCPs seems more than just a coincidental acronym collision: Fitz Nowlan is acknowledged in the Queue/CACM papers, and conversely acknowledges several of the "Google" BBR authors in his thesis.

References

- BBR Congestion Control Development, Google Groups, last accessed February 2019, mailing list with overview page full of pointers
- BBR: Congestion-Based Congestion Control, ACM Queue, Vol. 14 No. 5, September-October 2016 (HTML/PDF)
- tcp_bbr_add_bbr_congestion_control, N. Cardwell, V. Jacobson, Y. Cheng, N. Dukkipati, E. Dumazet, S. Yeganeh, commit 06872eae14974e930261b0550cc041e4f3ed0b76, Linus Torvalds's Linux Git tree, September 2016
- draft-cardwell-iccrg-bbr-congestion-control-00, BBR Congestion Control, Neal Cardwell, Yuchung Cheng, Soheil Hassas Yeganeh, Van Jacobson, July 2017
- draft-cheng-icrg-delivery-rate-estimation-00, Delivery Rate Estimation, Yuchung Cheng, Neal Cardwell, Soheil Hassas Yeganeh, Van Jacobson, July 2017
- A Quick Look at TCP BBR, D. Taht, blog post, 19 September 2016
- Making Linux TCP Fast, Y. Cheng, N. Cardwell, netdevconf 1.2, October 2016, (abstract, video)
- TCP BBR Quick-Start: Building and Running TCP BBR on Google Compute Engine, N. Cardwell, October 2016 (last accessed)
- BBR Congestion Control, N. Cardwell, Y. Cheng, Presentation to ICCRG at IETF 97, Seoul, November 2016 (video, starting ~57’00”, slides)
- BBR congestion control, J. Corbet, LWN.net, September 2016
- BBR Congestion Control: An Update, N. Cardwell, Y. Cheng, Presentation to ICCRG at IETF 98, Chicago, March 2017 (video, starting ~1h35’28”, slides)
- Driving Linux TCP Congestion Control algorithms around the LTE network Highway, Jae Chung, Feng Li, and Xiaoxiao Jiang, NetDev 2.1, April 2017 (abstract, slides)
- BBR TCP Opportunities, Matt Mathis, Presentation at the Quilt, 19 October 2016 (PDF slides)
- BBR TCP, Geoff Huston, May 2017, Potatoo blog/RIPE Labs
- CS 244 17: ReBBR, Reproducing BBR Performance In Losty Networks, Luke Hsiao and Jervis Muiind, June 2017 CS244 blog
- CS 244 17: Congestion-Based Congestion Control With BBR, Brad Girardeau and Samantha Steele, June 2017 CS244 blog
- BBRx: Extending BBR for Customized TCP Performance, Jae Won Chung, Feng Li, Boemjun Kim, NetDev 0x12, Montréal, July 2018 (abstract, video, slides, paper)