BBR Congestion Control:

IETF 102 Update: BBR Startup

Ian Swett

https://groups.google.com/d/forum/bbr-dev

QUIC implementation

IETF 102: Montreal, July 19, 2018

Google

Issues and Motivation

Web flows

- A large portion of flows never exit STARTUP
- Flows on policed connections experience huge losses in STARTUP(up to 50%)
- Sometimes ack aggregation limits the rate of STARTUP growth or causes early exit

Quartc

- Realtime applications are very sensitive to large changes in bandwidth and bufferbloat
- The small pacing rate and long time spent in DRAIN was consistently problematic

Idea: Make BBR Startup more sensitive to loss

Problem: BBR already reacts to loss in STARTUP by entering packet conservation, but that doesn't decrease the time in STARTUP much, sometimes it increases it.

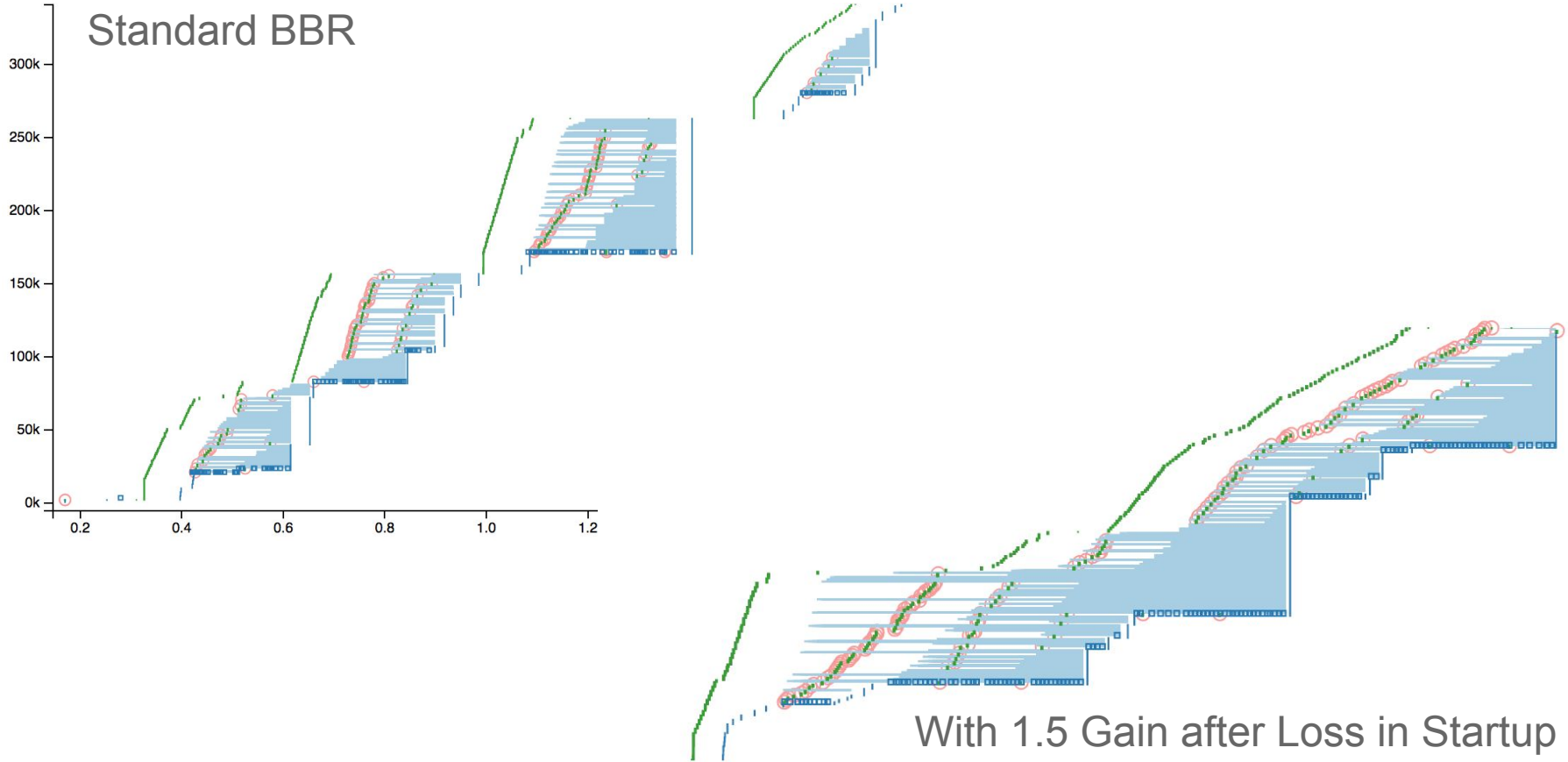
Proposal: Use a 1.5x pacing gain during STARTUP after both a loss has been detected AND a non app-limited sample has been taken

Reasoning: 1.5x should be enough to stay in STARTUP until max BW is reached

Result: Slight latency degradation in fraction of flows. Retransmit rates decrease >5% for video playbacks. Traces look a lot nicer.

Next Step: Increase gain back to 2.89 if the losses were spurious

Idea: Make BBR Startup more sensitive to loss (cont'd)



Idea: CWND gain of 2 in Startup

BBR is designed to be bandwidth based, BUT exiting STARTUP always requires BBR to become CWND limited.

Why: BBR exits STARTUP when the bandwidth hasn't increased more than 25%.

=> Consistently sending at 2.89x the max bandwidth builds a queue

Proposal: Derive the minimum CWND gain necessary to achieve a doubling in bandwidth every RTT.

=> Unsurprisingly it's **2** (assuming continuous delivery)

=> >40% less queue buildup exiting STARTUP (if it works…)

Recent Pacing Gain Derivation

Proposal: Ack Aggregation in Startup

Problems:

The 2x cwnd_gain derivation assumes continuous delivery, which isn't enough

=> CWND needs to be increased by at least 2 MSS to deal with delayed ACKs

If ACKs are aggregated, growth may be slow early even with 2.89 CWND gain

Goals:

Ensure there is sufficient CWND to double every RTT, regardless of aggregation.

Minimize the CWND when entering DRAIN

Proposal: Ack Aggregation in Startup (cont'd)

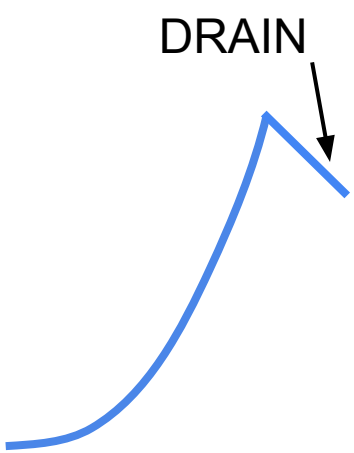
Proposal: Add the most recent excess acked to CWND. As you approach Max BW, this decreases because bandwidth stops increasing.

By not using a multi-RTT max filter, you avoid using an old excess delivered when the new measured bandwidth is much higher.

Results: Simulations show less time in STARTUP and lower SRTT when combined with a CWND gain of 2. Application experimental data coming soon.

Excess Acked Intro: slides-101-iccrg-an-update-on-bbr-work-at-google-00

Proposal: Increase DRAIN pacing gain to 0.75



Currently DRAIN pacing_gain is 0.34 (1 / 2.89)

Goal: Quickly drain excess 1-2x BDP of inflight after STARTUP

Problem: WebRTC has difficulty adapting to >2x bandwidth reductions

Proposal: Increase DRAIN pacing_gain to 0.75

If the CWND gain is reduced to 2, then the total time in DRAIN should be similar to today

Allows BBR to remove DRAIN mode and instead enter low-gain PROBE_BW when combined with drain_to_target.

Conclusion

In real-world flows, flow startup is a huge problem, and flows spend a lot of time in slow-start(Reno/Cubic) and STARTUP(BBR)

Real-time applications are particularly sensitive to STARTUP behavior and exit

It may be possible to improve BBR's STARTUP in ways that decrease retransmit rates and decrease bandwidth fluctuations without degrading application metrics

Experimental QUIC "Connection Options"(aka COPT):

- BBRS - Reduce pacing to 1.5x pacing_gain after loss
- BBQ2 - Reduce cwnd_gain to 2x during STARTUP
- BBQ3 - Enable ack aggregation compensation in STARTUP
- BBQ4 - Increase pacing_gain to 0.75x in DRAIN

Q & A

https://groups.google.com/d/forum/bbr-dev

Internet Drafts, paper, code, mailing list, talks, etc.

Special thanks to Neal Cardwell, Yuchung Cheng, Soheil Yeganeh, Van Jacobson, and Victor Vasiliev for their ideas and feedback.