BBRv3: Algorithm Overview and Google's Public Internet Deployment

Google TCP BBR team: Neal Cardwell, Yuchung Cheng, Kevin Yang, David Morley
Soheil Hassas Yeganeh, Priyaranjan Jha, Yousuk Seung
Van Jacobson

Google QUIC BBR team: Ian Swett, Bin Wu, Victor Vasiliev

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

IETF 119: Brisbane

Mar 21, 2024
Outline

- BBR algorithm high-level overview
- BBR deployment status at Google
- BBR code status and open source release plans

Goals for this talk:

- Responding to requests from CCWG chairs for BBR refresher/overview
- Inviting the community to...
  - Offer feedback on suggested plan, if any, for BBR with respect to IETF drafts/RFCs
  - Read the drafts and offer editorial feedback
  - Share algorithm or code fixes or enhancements
  - Share test results
  - Post bug reports
BBRv3 CC in a nutshell

- **Design goals:**
  - High throughput with up to a targeted level of random packet loss
  - Bounded queuing delay, even with bloated bottleneck buffers
  - Usable coexistence when sharing Reno and CUBIC congestion control

- **Mechanisms:**
  - Model-based: dynamically probes and models the network path
    - Models max bandwidth, min RTT, max aggregation, max inflight
  - Signals:
    - Bandwidth, RTT
    - ECN (like DCTCP, L4S)
    - Loss (explicit loss rate cap of 2%)
BBRv3 congestion control: the big picture

Input signals: throughput, RTT, loss, ECN

Model

BW

RTT

...·

BBR

Probing State Machine

Rate quantum

cwnd

Pacing Engine

Increases / Decreases inflight around target inflight

inflight

target inflight = est. BDP

time

Data

Paced Data
A summary of BBR versions:

- **BBRv1**: [obsolete/deprecated]
  - Bandwidth, RTT as signals primary signals; loss used over short time scales

- **BBRv2**: [obsolete/deprecated]
  - BBRv1 + using ECN, loss as signals

- **BBRv3**: [discussed at IETF 117: slide link]
  - BBRv2 + bug fixes and performance tuning

- **BBR.Swift**: [discussed at IETF 109: slide link]
  - BBRv3 + using network_RTT (excluding receiver delay) as primary CC signal
### A quick comparison of CC algorithms

<table>
<thead>
<tr>
<th></th>
<th>CUBIC</th>
<th>BBRv1</th>
<th>BBRv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model parameters to the state machine</td>
<td>N/A</td>
<td>Max throughput, Min RTT</td>
<td>Max throughput, Min RTT, Max aggregation, Max inflight</td>
</tr>
<tr>
<td><strong>Loss</strong></td>
<td>Reduce cwnd by 30% on window with any fast recovery loss</td>
<td>N/A</td>
<td>Explicit loss rate cap (2%)</td>
</tr>
<tr>
<td><strong>Startup</strong></td>
<td>Slow-start until RTT rises (Hystart) or any loss</td>
<td>Slow-start until tput plateaus</td>
<td>Slow-start until tput plateaus or ECN/loss rate &gt; target</td>
</tr>
</tbody>
</table>

A quick comparison of CC algorithms

<table>
<thead>
<tr>
<th></th>
<th>CUBIC</th>
<th>BBRv1</th>
<th>BBRv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model parameters to the state machine</td>
<td>N/A</td>
<td>Max throughput, Min RTT</td>
<td>Max throughput, Min RTT, Max aggregation, Max inflight</td>
</tr>
<tr>
<td><strong>Loss</strong></td>
<td>Reduce cwnd by 30% on window with any fast recovery loss</td>
<td>N/A</td>
<td>Explicit loss rate cap (2%)</td>
</tr>
<tr>
<td><strong>Startup</strong></td>
<td>Slow-start until RTT rises (Hystart) or any loss</td>
<td>Slow-start until tput plateaus</td>
<td>Slow-start until tput plateaus or ECN/loss rate &gt; target</td>
</tr>
</tbody>
</table>
BBR deployment status at Google

- **Google-internal traffic:**
  - **BBRv3** is TCP congestion control for all internal **WAN traffic**
  - **BBR.Swift** is TCP congestion control used **within a datacenter**

- **Google-external traffic:**
  - **BBRv3** is TCP CC for all **Google.com** and [new!] **YouTube** public Internet traffic
  - A/B experiments: BBRv3 vs v1 for small % of users for:
    - QUIC for google.com and YouTube
TCP BBRv3 release:
- Linux TCP BBRv3 is open source (dual GPLv2/BSD), available for review/testing:
  - Plan to email patches to propose inclusion in mainline Linux TCP ASAP
- BBRv1 code in Linux TCP "bbr" module will be upgraded to BBRv3
- Why upgrade BBRv1->BBRv3 in place rather than a separate module? BBRv3 has...
  - Better coexistence with Reno/CUBIC, vs v1
  - Lower loss rates, vs v1
  - Lower latency for short web requests (from google.com, YouTube data), vs v1
  - Throughput similar to v1 (within 1% of v1 on YouTube)
Conclusion

● Next:
  ○ Plan on submitting BBRv3 for inclusion in mainline Linux TCP ASAP
  ○ Internet Drafts cover BBRv2; plan to update them to cover BBRv3 ASAP:
    ■ Delivery rate estimation: draft-cheng-iccrg-delivery-rate-estimation
    ■ BBR Congestion control: draft-cardwell-iccrg-bbr-congestion-control
● We invite the community to share…
  ○ Feedback on the algorithm, code, or drafts
  ○ Test results, issues, patches, or ideas
  ○ Feedback on where to go from here with drafts
● Thanks!
https://groups.google.com/d/forum/bbr-dev

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