

BBR v2

A Model-based Congestion Control

Neal Cardwell, Yuchung Cheng,

Soheil Hassas Yeganeh, Ian Swett, Victor Vasiliev,

Priyaranjan Jha, Yousuk Seung, Matt Mathis

Van Jacobson

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

Outline

- BBR v2 research update
 - Improvements between v1 and v2
 - Status and recent results
 - Overview of current BBR v2 design
- Deployment status and links for further info
- Conclusion

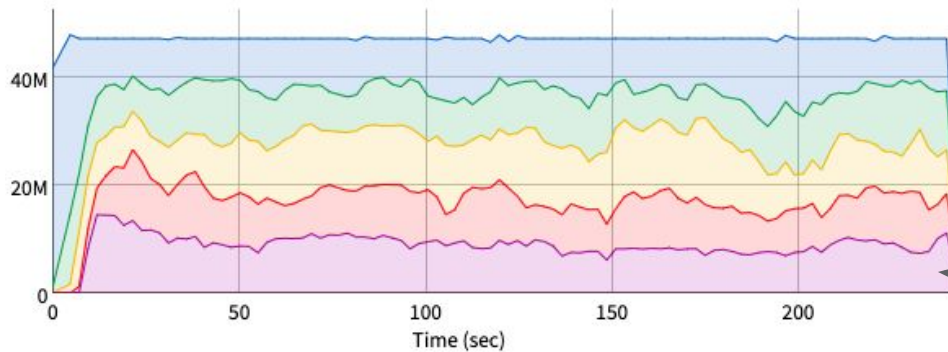
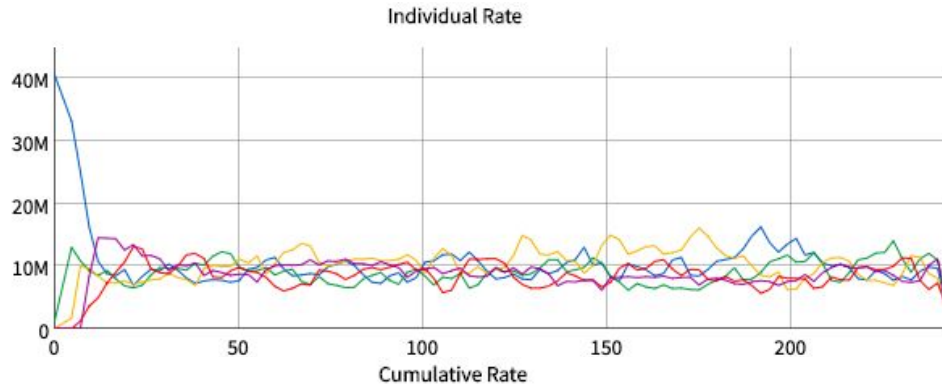
Issues with initial (v1) version of BBR

- Low throughput for Reno/CUBIC flows sharing a bottleneck with bulk BBR flows
- Loss-agnostic; high packet loss rates if bottleneck queue $< 1.5 \cdot \text{BDP}$
- ECN-agnostic
- Low throughput for paths with high degrees of aggregation (e.g. wifi)
- Throughput variation due to low cwnd in PROBE_RTT

BBR v2 tackles all of these...

BBR v2 improvements: coexistence with Reno/CUBIC

- BBR v1: low throughput for Reno/CUBIC flows sharing some paths
- BBR v2: adapts bandwidth probing for better coexistence with Reno/CUBIC [IETF [102](#)]

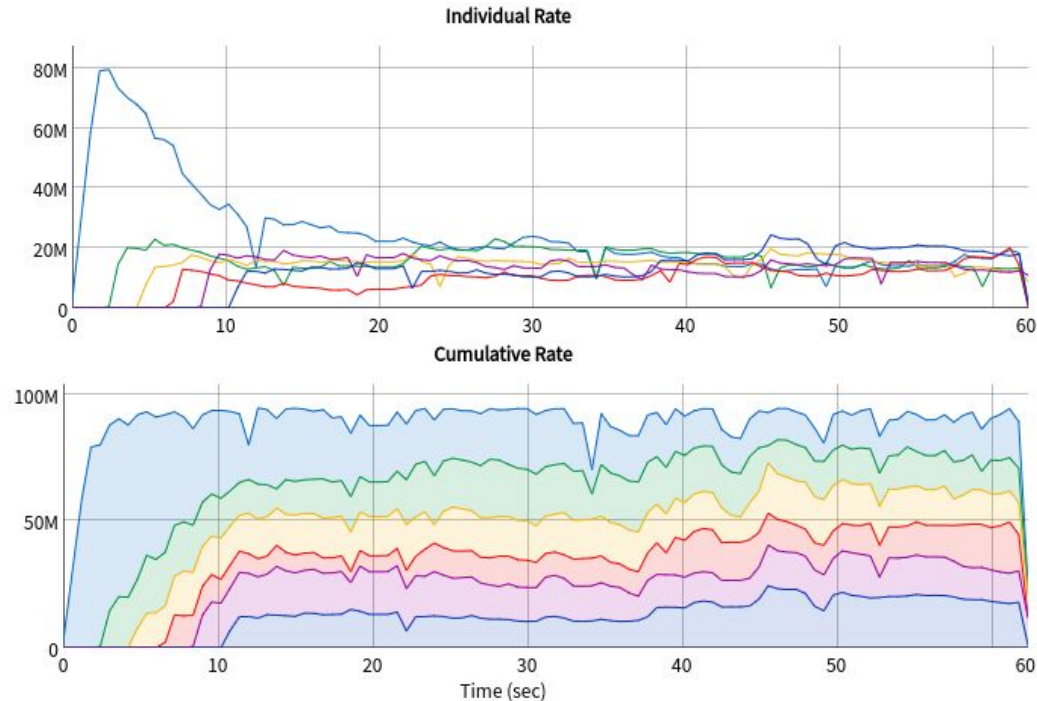


4 CUBIC, 1 BBR v2, 50M, 40ms,
buffer = 1xBDP
start time {0, 2, 4, 6, 8} secs

| | bw | retrans |
|---------|--------|---------|
| CUBIC 1 | 10.5 M | 0.3% |
| CUBIC 2 | 9.1 M | 0.1% |
| CUBIC 3 | 10.4 M | 0.1% |
| CUBIC 4 | 8.7 M | 0.1% |
| BBR v2 | 9.3 M | 0.1% |

BBR v2 improvements: using packet loss as a signal

- BBR v1: agnostic to loss, susceptible to high loss rates if bottleneck queue $< 1.5 \cdot \text{BDP}$
- BBR v2: uses loss as an explicit signal, with an explicit target loss rate ceiling [IETF [102](#)]

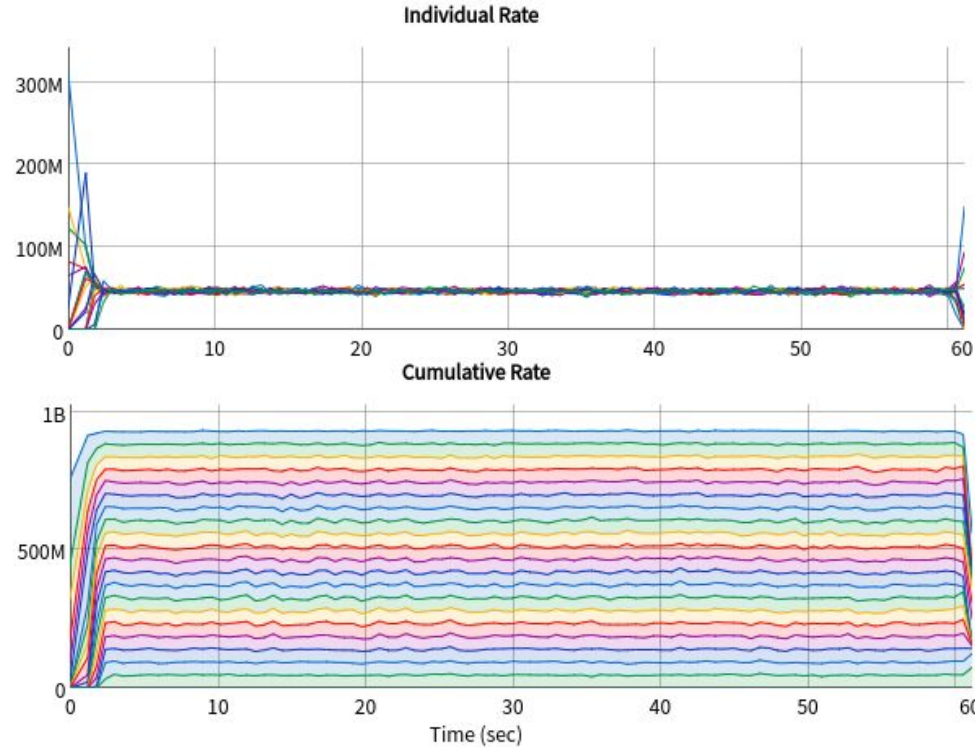


6 BBR v2, $t=\{0, 2, 4, 6, 8, 10\}$, 100M, 100ms, buffer = 5% of BDP (41 packets);

| | bw | retrans |
|-------|--------|---------|
| BBR 1 | 24.4 M | 2.7% |
| BBR 2 | 16.2 M | 0.8% |
| BBR 3 | 15.0 M | 0.8% |
| BBR 4 | 11.0 M | 0.7% |
| BBR 5 | 14.4 M | 0.8% |
| BBR 6 | 15.2 M | 0.7% |

BBR v2 improvements: using ECN as a signal

- BBR v1: does not use ECN
- BBR v2: uses DCTCP-style ECN, if available, to help keep queues short



20 BBR v2, starting each 100ms, 1G, 1ms,
Linux codel with ECN ce_threshold at
242us sojourn time.

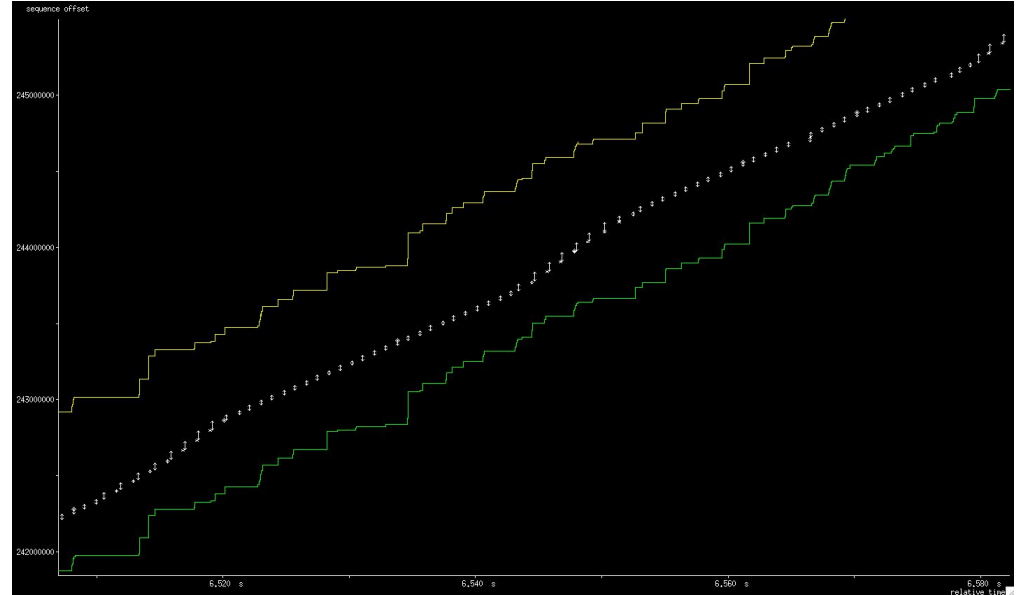
0 retransmits

flow throughput = [46.8 .. 51.1] Mbps

| | RTT (ms) |
|-----|----------|
| 50% | 1.278 |
| 95% | 1.499 |
| max | 2.854 |

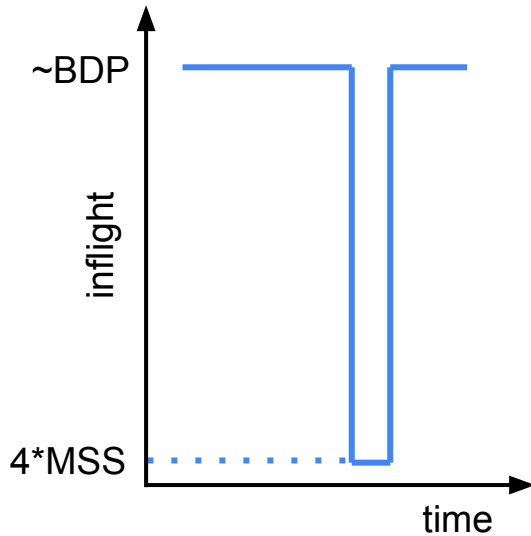
BBR v2 improvements: high throughput for wifi

- BBR v1: low throughput with aggregation > 1*BDP of data or ACKS (e.g. wifi LAN)
 - BBR v2: estimates recent degree of aggregation to match CUBIC throughput [IETF [101](#)]
-
- On YouTube, BBR v2 matches CUBIC throughput for users on wifi links
 - In controlled tests, BBR v2 utilizes wifi links well (at right, path is POP->Internet->DOCSIS->wifi):
 - Aggregation modeling code available in QUIC and now in Linux 5.1 [[commit](#)]

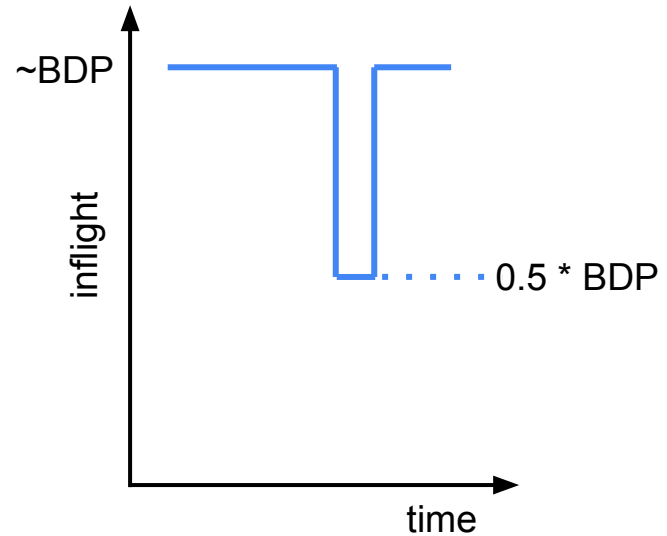


BBR v2 improvements: reducing throughput variation

- BBR v1: cuts cwnd to 4 packets if entering PROBE_RTT
- BBR v2: cuts cwnd to 50% of BDP if entering PROBE_RTT



BBR v1



BBR v2

What's new in BBR v2: a summary

| | CUBIC | BBR v1 | BBR v2 |
|---------------------------------------|--|--------------------------------|--|
| Model parameters to the state machine | N/A | Throughput, RTT | Throughput, RTT, max aggregation, max inflight |
| Loss | Reduce cwnd by 30% on window with any loss | N/A | Explicit loss rate target |
| ECN | RFC3168 (Classic ECN) | N/A | DCTCP-inspired ECN |
| Startup | Slow-start until RTT rises (Hystart) or any loss | Slow-start until tput plateaus | Slow-start until tput plateaus or ECN/loss rate > target |

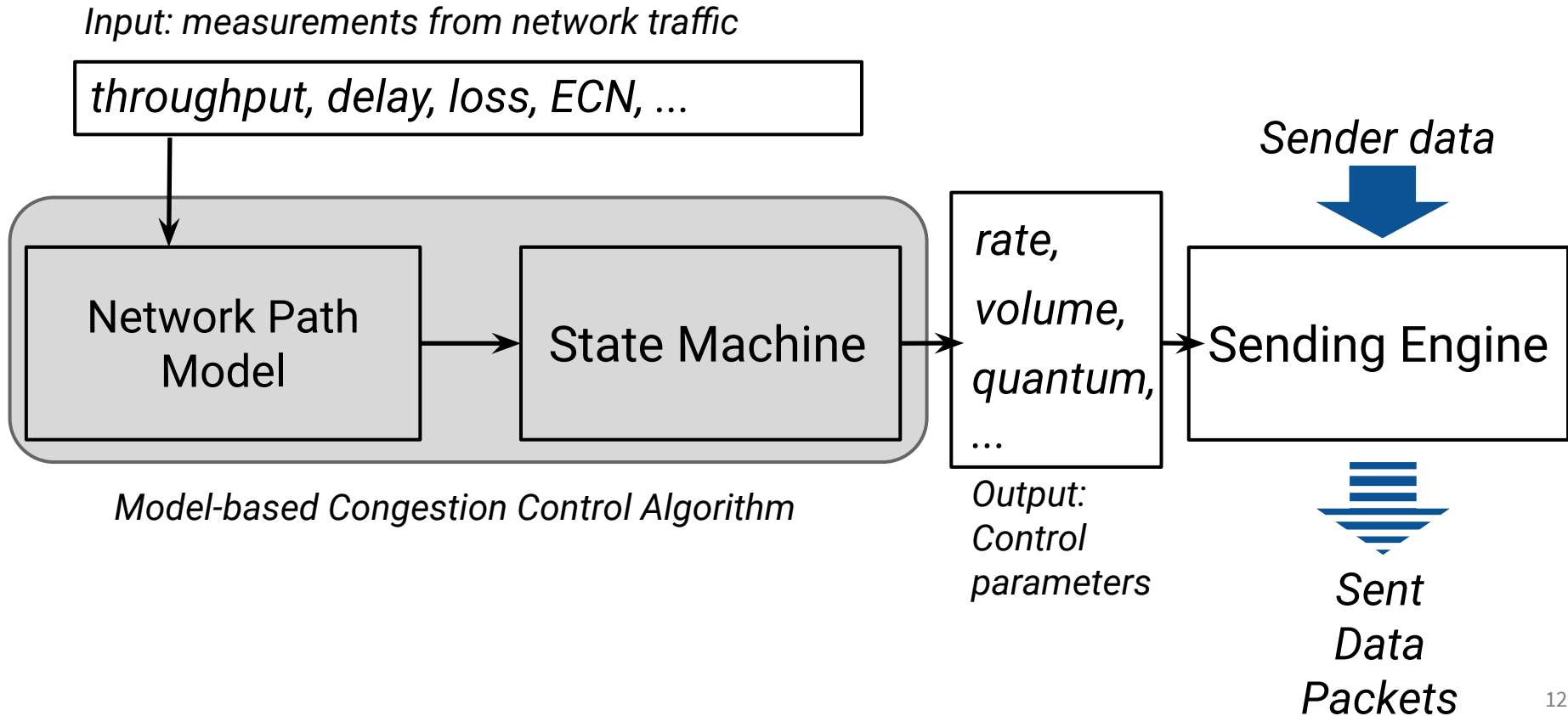
BBR v2: status

Testing BBR v2 with YouTube TCP traffic

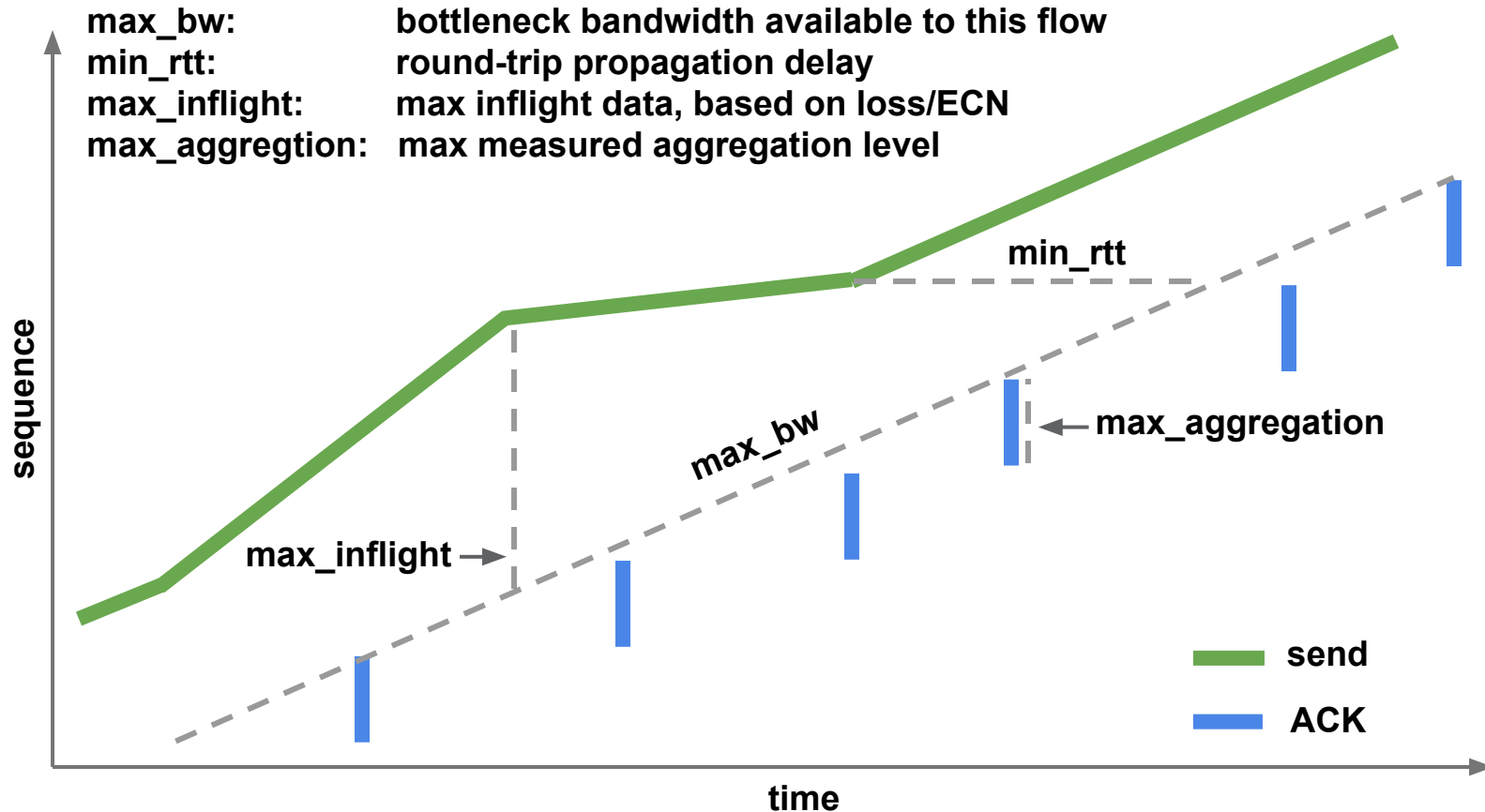
- Kernel with initial version of TCP BBR v2 on all YouTube machines
- Running experiments using BBR v2 variants on small fraction of global YouTube traffic
- Results so far:
 - Reduced queuing delays: RTTs lower than BBR v1 and CUBIC
 - Reduced packet loss: loss rates closer to CUBIC than BBR v1
 - Throughput matches CUBIC
- Continuing to iterate using production and lab tests
- Preparing for open source release

Overview of BBR v2 design...


BBR congestion control: the big picture



BBR v2: the network path model



BBR v2: the network path model



| | |
|-------------------------|--|
| max_bw: | bottleneck bandwidth available to this flow |
| min_rtt: | round-trip propagation delay |
| max_inflight: | max inflight data based on loss/ECN |
| max_aggregation: | max measured aggregation level |

These max bw and inflight parameters can be highly dynamic
Based on traffic levels

Thus BBR v2 maintains both short-term and long-term estimates for each
Analogous to CUBIC's short-term (ssthresh) and long-term (W_max) parameters

BBR v2: model adaptation

BBR v2 model estimates (bw, inflight) over the short term and long term

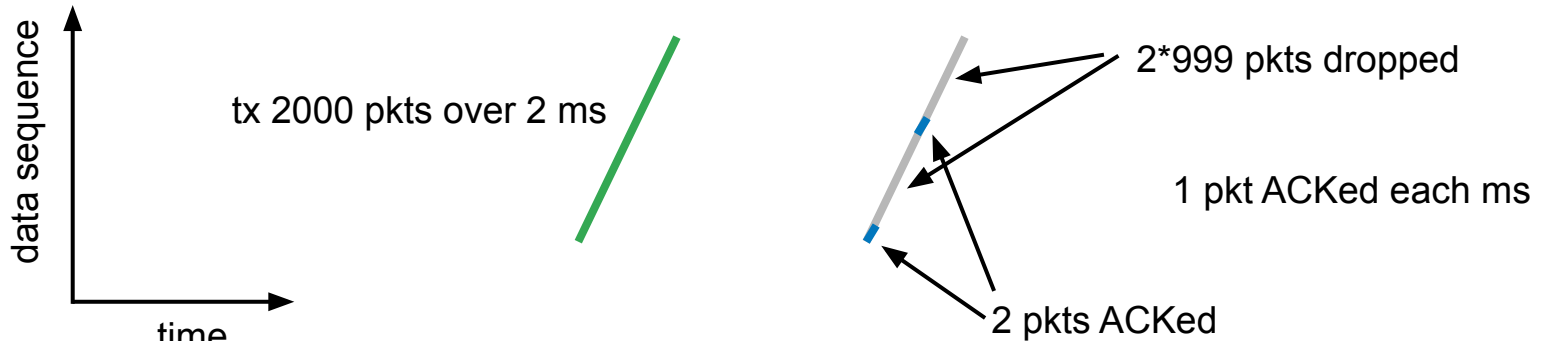
- Goals:
 - High throughput, even when experiencing moderate random loss
 - Low queue pressure (low queuing delays, low packet loss rates)

- Mostly adapt quickly to maintain flow balance and leaves headroom
 - $\{bw, inflight\}_{lo}$: short-term bounds using latest delivery/loss/ECN signals
 - Intuition: what is the (bw, inflight) delivery process the flow is measuring now?
- Periodically probe beyond flow balance to probe robustly for higher bw, inflight
 - $\{bw, inflight\}_{hi}$: long-term max measured before signals of congestion (loss, ECN)
 - Intuition: what is the max (bw, inflight) observed to be consistent with the network's desired loss rate and ECN mark rate SLO (service level objective)?

BBR v2 adaptation: motivating example (part 1)

How to adapt to a loss signal, given the ambiguities?

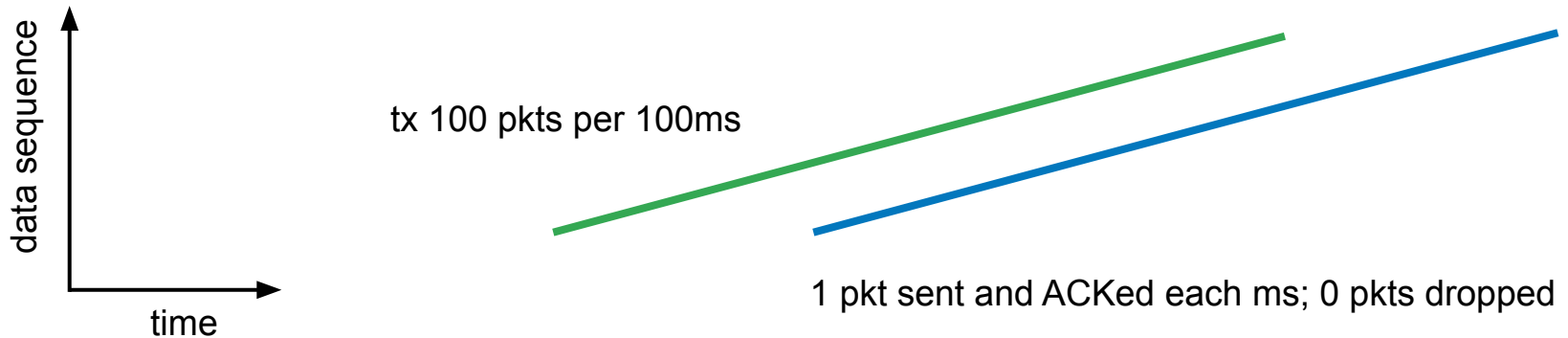
- Consider a shallow-buffered high-speed WAN with $RTT=100\text{ms}$
- An app does occasional 2000-pkt writes
- Available bw drops from 12 Gbit/sec \rightarrow 12 Mbit/sec (from 1000 pkt/ms to 1 pkt/ms)
 - On first write after bw drops, there will likely be very high loss
- Ambiguity: low tput may be due to lack of data, bursty traffic, or sustained cut in bw



BBR v2 adaptation: motivating example (part 2)

To fully utilize bottlenecks, adapt both max sending rate *and* max inflight

- In this example, should we simply cut cwnd to 2, due to 2 pkts being delivered?
 - No; if we pace at available bandwidth, can deliver a vastly higher volume of data...
- If the pattern in this example happens repeatedly, probably we should:
 - Gradually reduce sending rate to 1 pkt/ms (12 Mbit/sec)
 - Converge to a target inflight at new BDP of $1 \text{ pkt/ms} * 100\text{ms} = 100 \text{ pkt}$ (not 2 pkts)



BBR v2: short-term model adaptation: algorithm

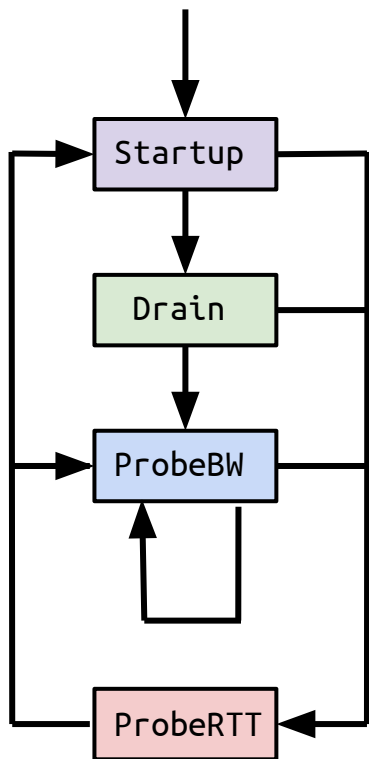
Strategy: gradually adapt to measured delivery process (bw, inflight) in the path

- Applies generally:
 - In fast recovery or RTO recovery
 - Whether application-limited or not
- Maintain 1-round-trip max of delivered bandwidth (rs->delivered/rs->interval_us)
 - $bw_latest = \text{windowed_max}(\text{delivered_bw_sample}, 1 \text{ round trip})$
- Maintain 1-round-trip max of delivered volume of data (rs->delivered):
 - $\text{inflight_latest} = \text{windowed_max}(\text{delivered_data_sample}, 1 \text{ round trip})$
- Upon ACK at end of each round that had a newly-marked loss ($\beta = 0.3$):

$$bw_lo = \max(bw_latest, (1 - \beta) * bw_lo)$$

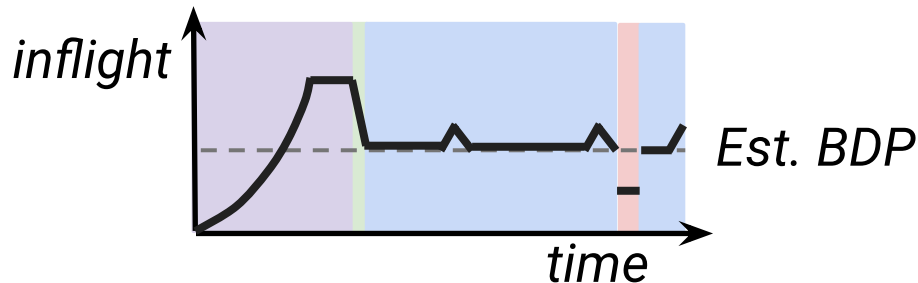
$$\text{inflight_lo} = \max(\text{inflight_latest}, (1 - \beta) * \text{inflight_lo})$$

BBR: the state machine



State machine uses 2-phase sequential probing of bw, RTT

- 1: raise inflight to probe BtlBw, get high throughput
- 2: lower inflight to probe RTTprop, get low delay
- At two different time scales: warm-up, steady state...
- Warm-up:
 - Startup: ramp up quickly until we estimate pipe is full
 - Drain: drain the estimated queue from the bottleneck
- Steady-state:
 - ProbeBW: cycle pacing rate to vary inflight, probe BW
 - ProbeRTT: if needed, a coordinated dip to probe RTT

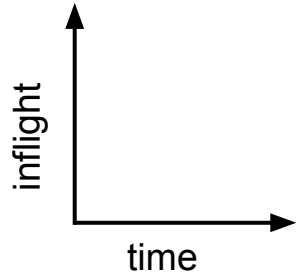


BBR v2 flow life cycle

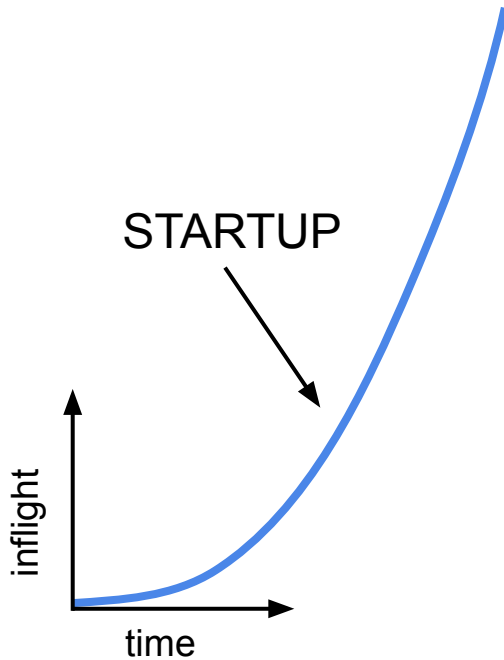
BBR v2 has a state machine similar to v1, at a high level

- But many of the mechanism details are new...

(**bold** = new in v2)



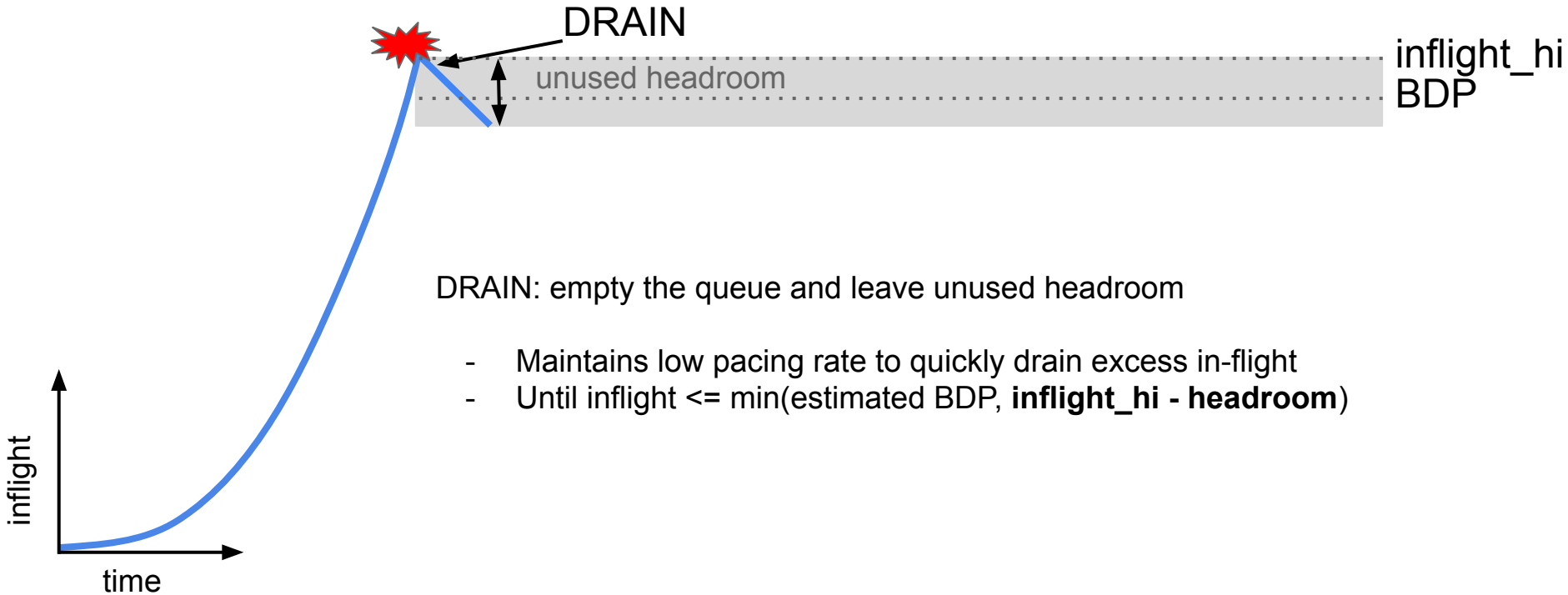
BBR v2 flow life cycle



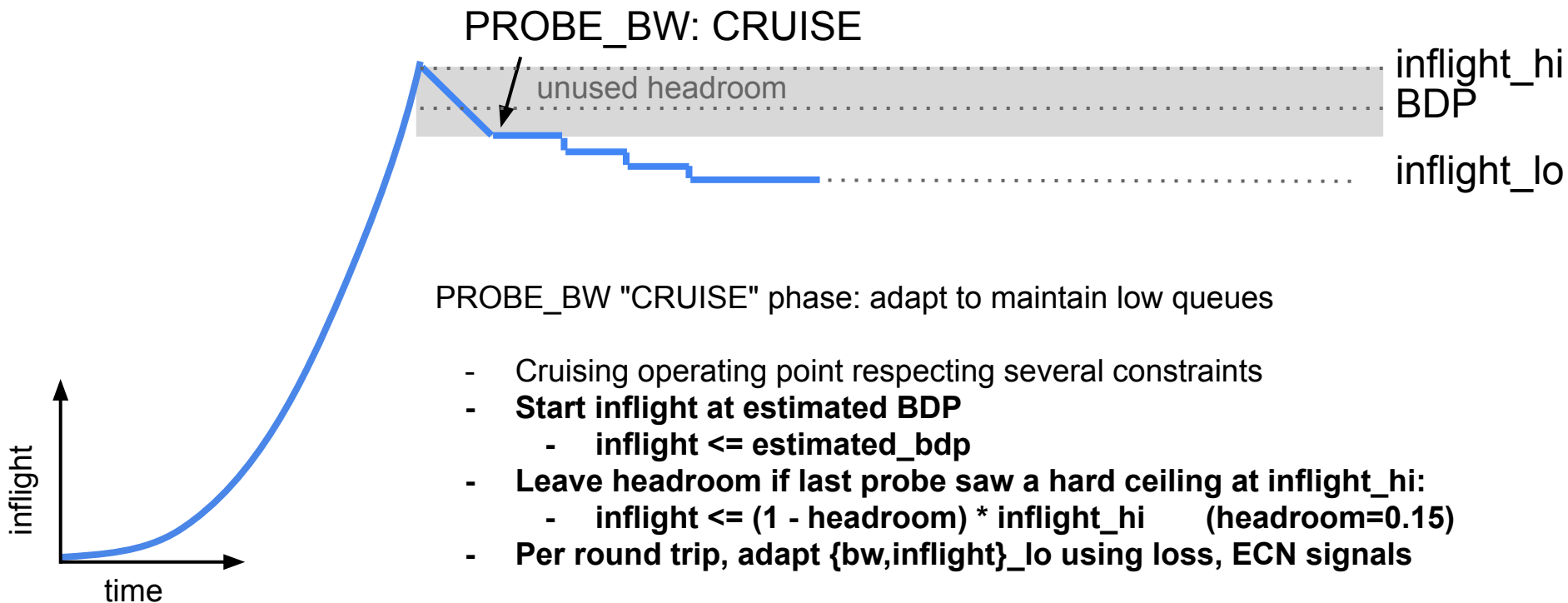
STARTUP: rapidly discover available bandwidth

- Doubles sending rate and inflight
- **Sets inflight_hi to estimated max safe in-flight volume if:**
 - **Filtered loss rate is too high**
- Exits when either:
 - Bandwidth samples plateau
 - **inflight_hi is set**

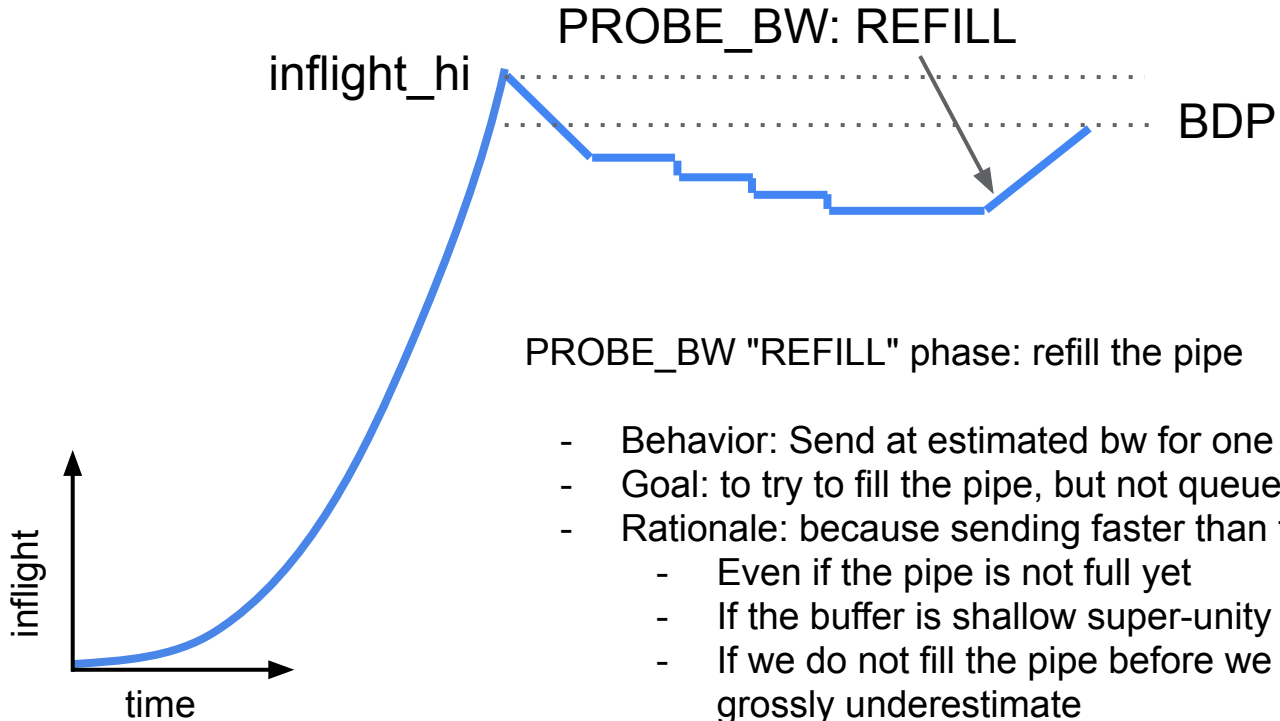
BBR v2 flow life cycle



BBR v2 flow life cycle



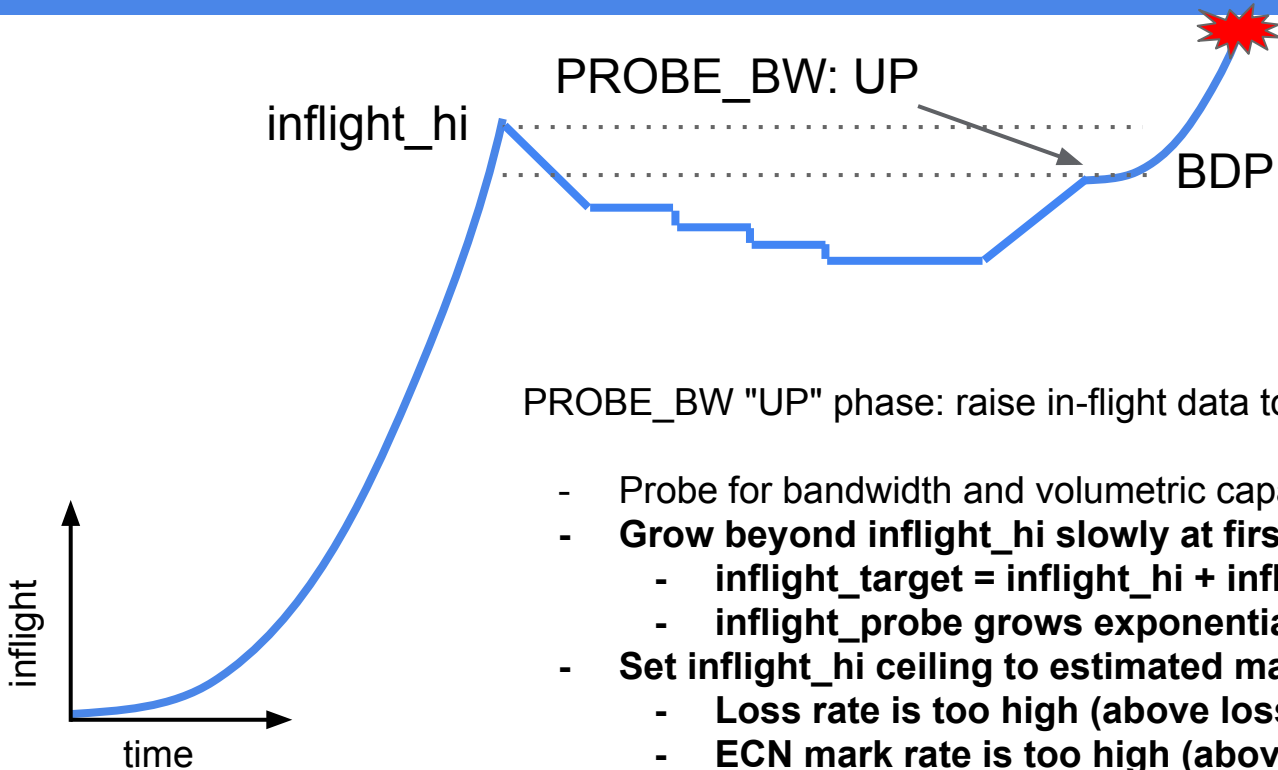
BBR v2 flow life cycle



PROBE_BW "REFILL" phase: refill the pipe

- Behavior: Send at estimated bw for one packet-timed round trip
- Goal: to try to fill the pipe, but not queue (yet)
- Rationale: because sending faster than the available bw builds a queue
 - Even if the pipe is not full yet
 - If the buffer is shallow super-unity can cause loss very quickly
 - If we do not fill the pipe before we cause this loss, then our model will grossly underestimate

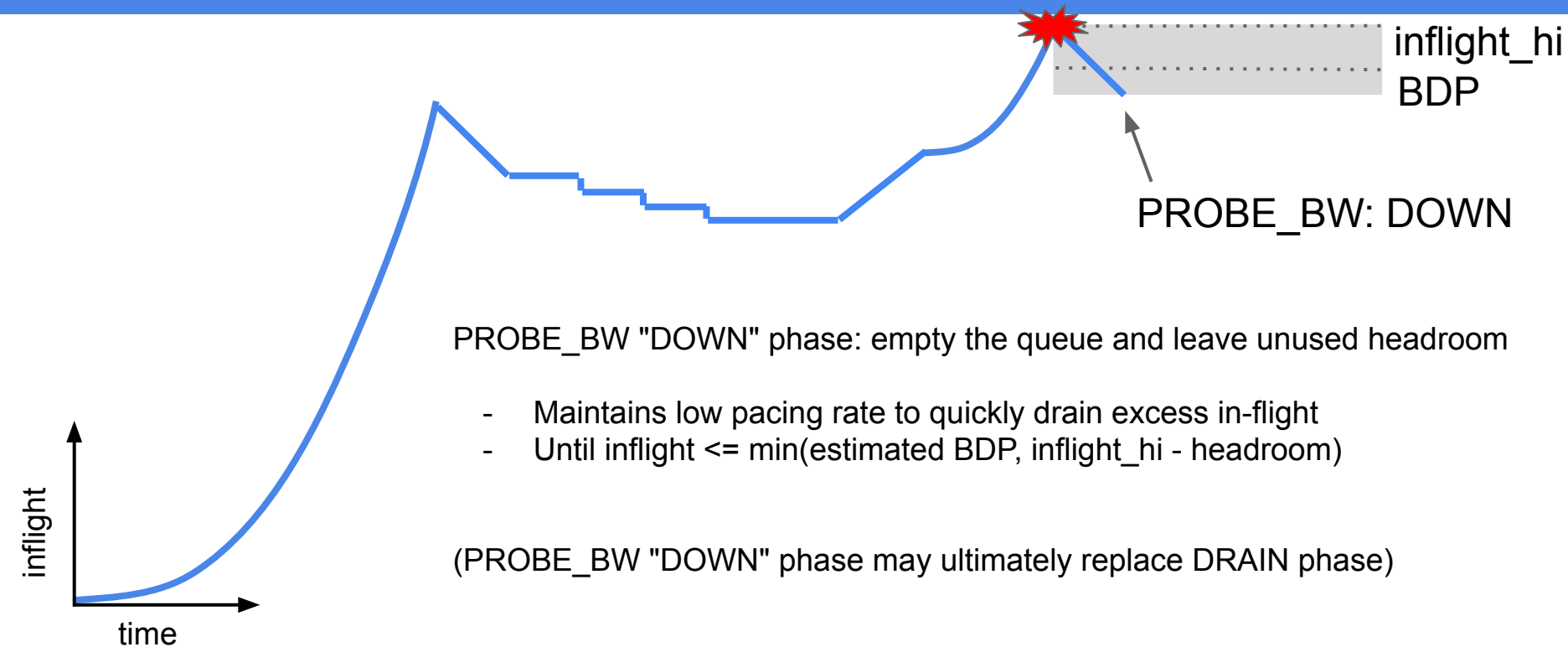
BBR v2 flow life cycle



PROBE_BW "UP" phase: raise in-flight data to probe for bandwidth

- Probe for bandwidth and volumetric capacity
- **Grow beyond inflight_hi slowly at first, then rapidly**
 - $\text{inflight_target} = \text{inflight_hi} + \text{inflight_probe}$
 - inflight_probe grows exponentially per round: 1, 2, 4, 8... packets
- **Set inflight_hi ceiling to estimated max safe in-flight volume if:**
 - Loss rate is too high (above loss SLO threshold)
 - ECN mark rate is too high (above ECN SLO threshold)
- **Terminate probing upon any of:**
 - Estimated queue is high enough ($\text{inflight} > 1.25 * \text{estimated_bdp}$)
 - Set inflight_hi ceiling based on loss or ECN

BBR v2 flow life cycle



PROBE_BW "DOWN" phase: empty the queue and leave unused headroom

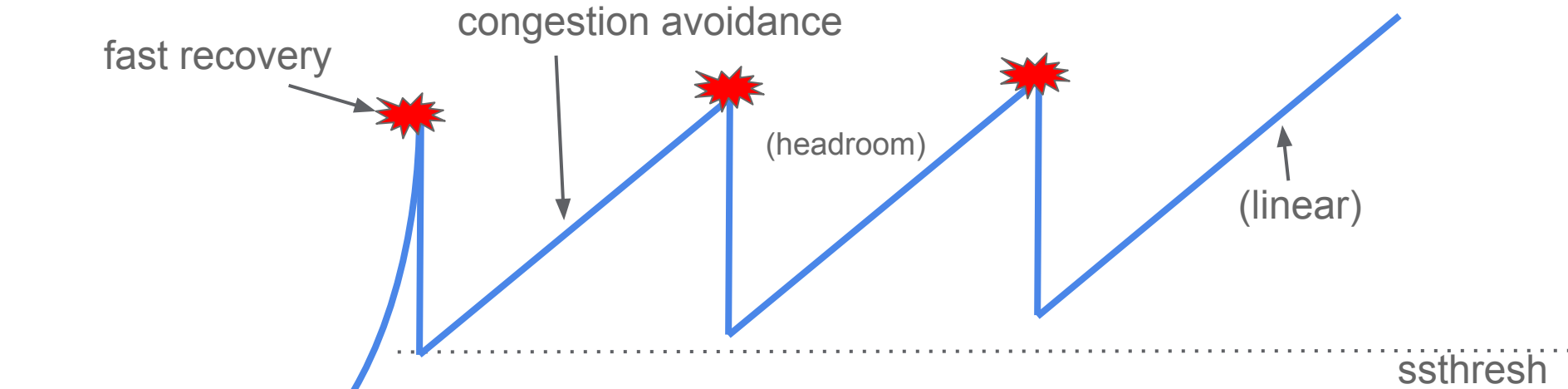
- Maintains low pacing rate to quickly drain excess in-flight
- Until $\text{inflight} \leq \min(\text{estimated BDP}, \text{inflight_hi} - \text{headroom})$

(PROBE_BW "DOWN" phase may ultimately replace DRAIN phase)

Comparing congestion control algorithms...

To summarize, let's review the macroscopic behavior of Reno, CUBIC, BBR v2...

Reno



Reno: brittle loss response, non-scalable growth

Non-scalable linear growth

Needs 1000x more time to reach 1000x higher bw

Brittle; to fully utilize a 10G, 100ms path, needs:

>1 hour between any losses

loss rate \leq .0000000002 ([2.0e-10](#))

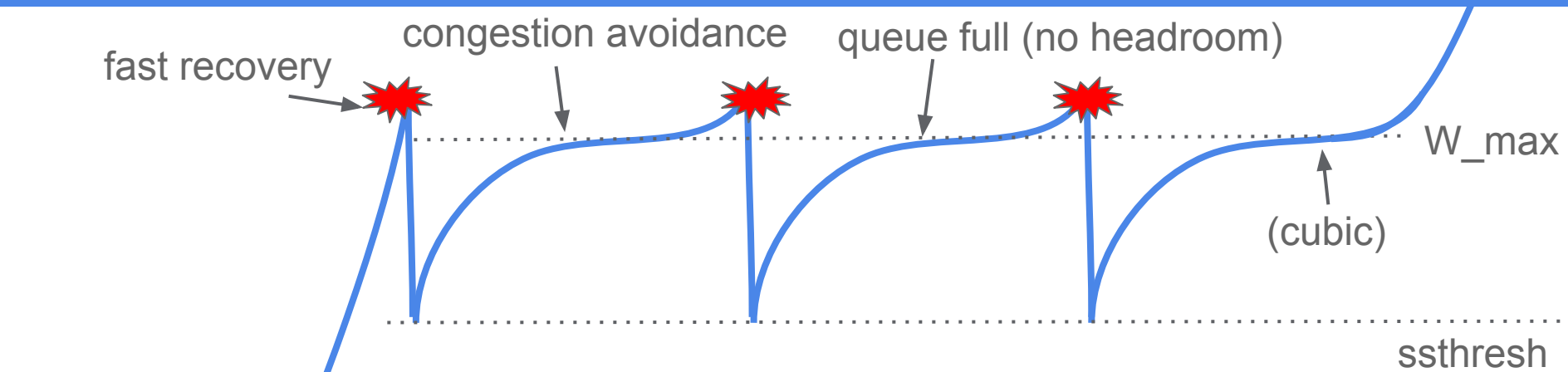
data in flight

slow start

time

Google

CUBIC



CUBIC: brittle loss response, non-scalable growth

Non-scalable cubic growth

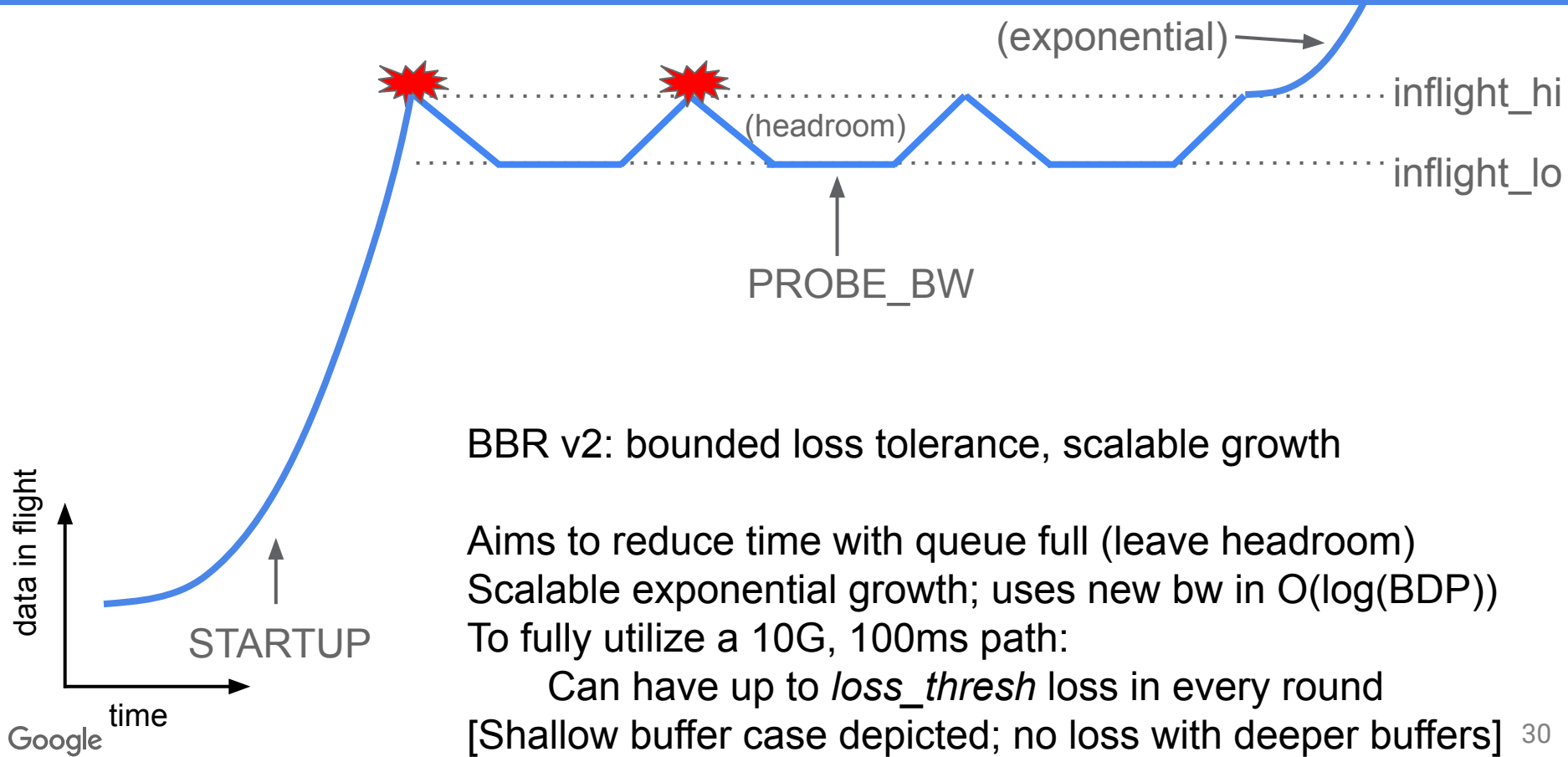
Needs 10x more time to reach 1000x higher bw

Brittle; to fully utilize a 10G, 100ms path, needs:

>40 secs between any losses

loss rate $\leq .000000029$ ([2.9e-8](#))

BBR v2



BBR v2: bounded loss tolerance, scalable growth

Aims to reduce time with queue full (leave headroom)

Scalable exponential growth; uses new bw in $O(\log(\text{BDP}))$

To fully utilize a 10G, 100ms path:

Can have up to *loss_thresh* loss in every round

[Shallow buffer case depicted; no loss with deeper buffers] 30

BBR status: deployment, release, documentation

- BBR v1 used for TCP/QUIC on Google.com/YouTube, Google WAN backbone
 - Better performance than CUBIC for web, video, RPC traffic
- BBR v2 running in experiments for a small percentage of YouTube users
- BBR v1 Code available as open source in [Linux TCP](#) (dual GPLv2/BSD), [QUIC](#) (BSD)
- Active BBR work under way for BBR in FreeBSD TCP @ NetFlix
- BBR v1 Internet Drafts are out and ready for review/comments:
 - Delivery rate estimation: [draft-cheng-iccr-g-delivery-rate-estimation](#)
 - BBR congestion control: [draft-cardwell-iccr-g-bbr-congestion-control](#)
- IETF presentations: [97](#) | [98](#) | [99](#) | [100](#) | [101](#) | [102](#)
- BBR v1 Overview in [Feb 2017 CACM](#)

Conclusion

Actively working on BBR v2

- Linux TCP and QUIC at Google; current focus areas:
 - Reducing queue pressure using packet loss and ECN signals
 - Coexistence with loss-based congestion control
- Work under way for BBR in FreeBSD TCP @ NetFlix
- Always happy to see patches, hear test results, or look at packet traces...

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

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

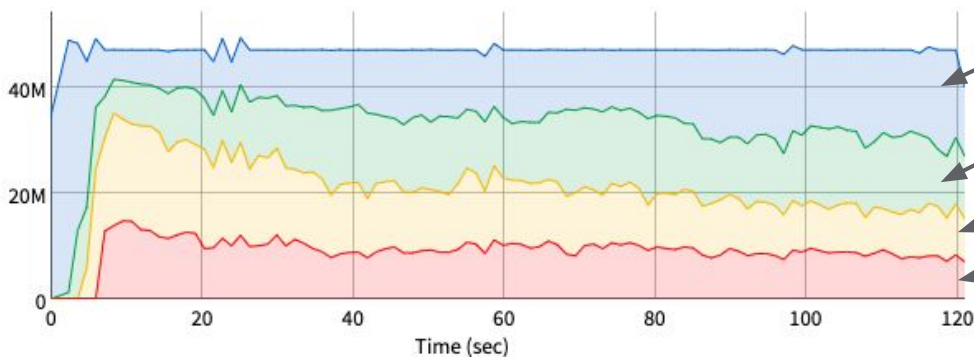
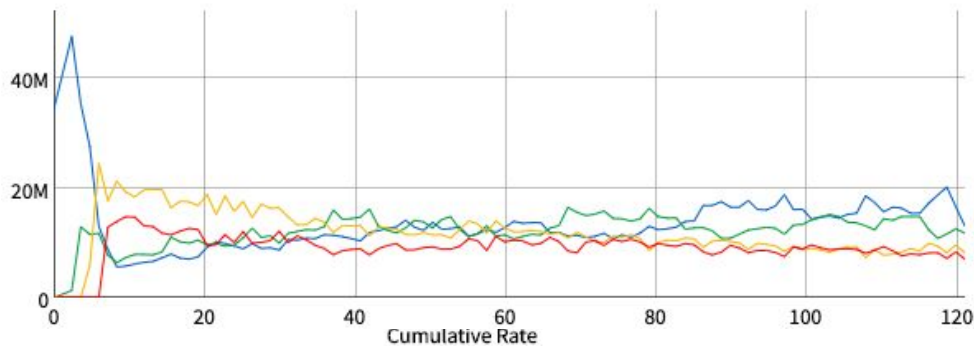
Special thanks to Eric Dumazet, Nandita Dukkipati, C. Stephen Gunn, Kevin Yang, Jana Iyengar, Pawel Jurczyk, Biren Roy, David Wetherall, Amin Vahdat, Leonidas Kontothanassis, and {YouTube, google.com, SRE, BWE} teams.

Backup slides...

BBR v2 improvements: coexistence with Reno/CUBIC

- BBR v1: low throughput for Reno/CUBIC flows sharing some paths
- BBR v2: adapts bandwidth probing for better coexistence with Reno/CUBIC [IETF [102](#)]

Individual Rate



2 CUBIC, 2 BBR v2, 50M, 40ms,
buffer = 1xBDP
start time {0, 2, 4, 6} secs

| | bw | retrans |
|---------|--------|---------|
| CUBIC 1 | 13.6 M | 0.4% |
| CUBIC 2 | 12.5 M | 0.1% |
| BBR 1 | 12.4 M | 0.3% |
| BBR 2 | 9.8 M | 0.2% |

BBR v2 design principles in a nutshell

- **Leave headroom: leave space for entering flows to grab**
- **React quickly: using loss/ECN, adapt to delivery process *now* to maintain flow balance**
- Don't overreact: don't do a multiplicative decrease on every round trip with loss/ECN
- **Probe differentially: probe on a time scale to allow coexistence with Reno/CUBIC**
- Probe robustly: try to probe beyond estimated max bw, max volume before we cut est.
- **Avoid overshooting: start probing at an inflight measured to be tolerable**
- Grow scalably: **start probing at 1 extra packet**; grow exponentially to use free capacity

(**bold** = new in v2)