BBR Congestion Control: Fundamentals and Updates

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

Aug 29, 2023

Google

Outline

- BBRv1
- BBRv2
- BBRv3
- Comparing Reno, CUBIC, BBRv3
- Links for further information
- Conclusion

BBRv1

Problems with loss-based congestion control

2011: many reported excessive buffering and delays on the Internet (a.k.a. bufferbloat)

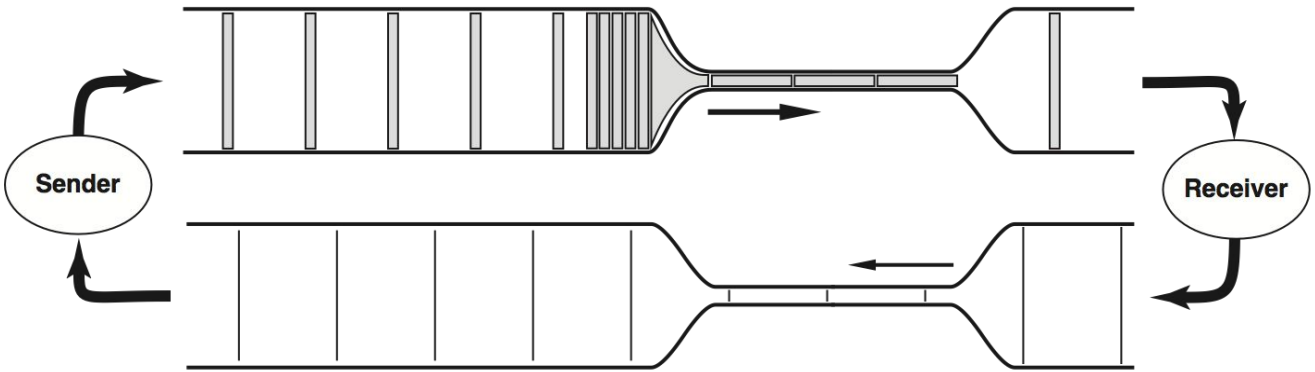
2012: single-connection HTTP/2 was much slower than multi-conn HTTP/1 on lossy links

2013: poor TCP throughput on WANs w/ commodity shallow-buffer switches

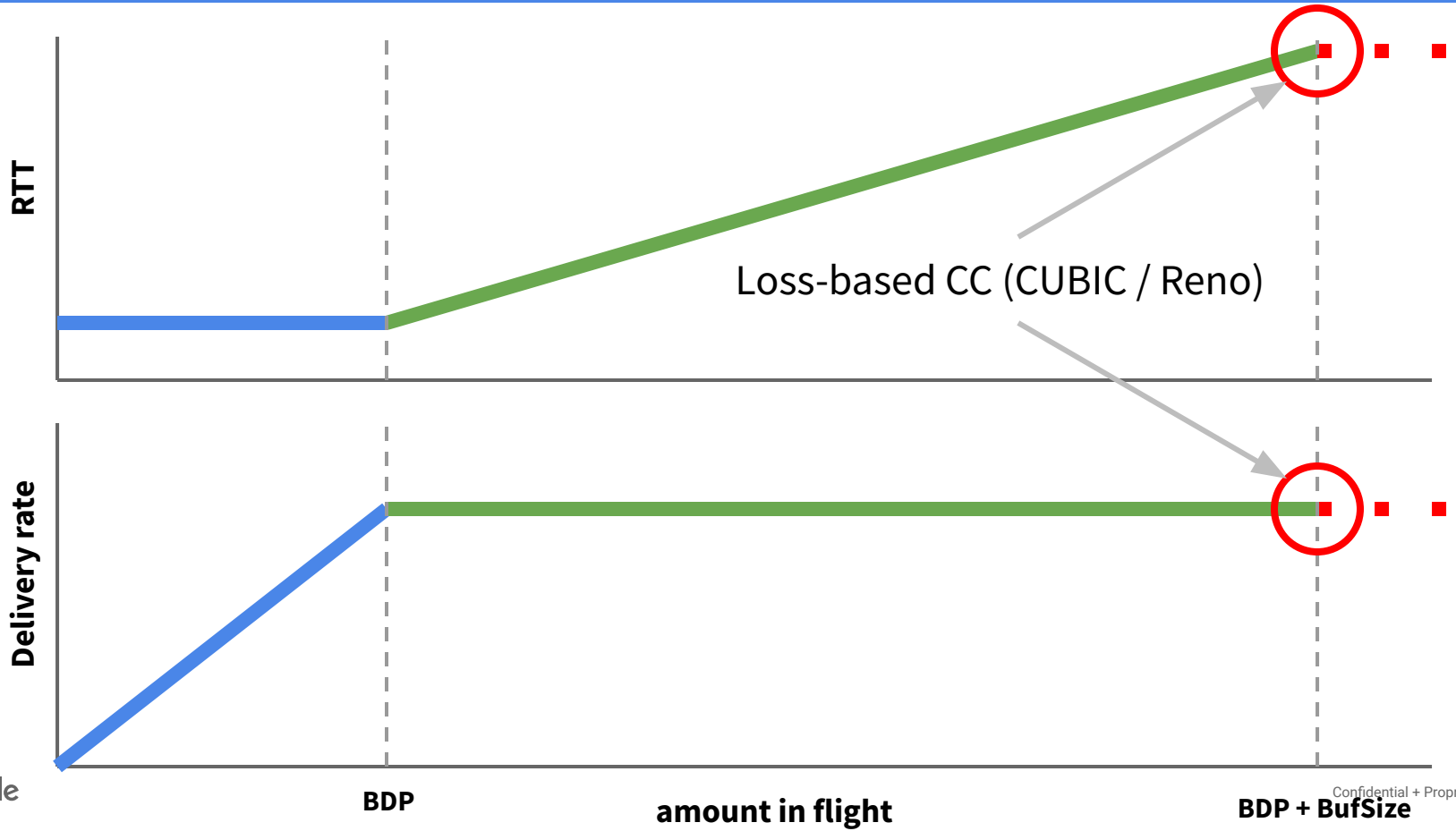
Culprit: loss-based congestion control (CC) (Reno, then CUBIC)

- Packet loss alone is **not** a good proxy to detect congestion
- Loss-based CC is overly sensitive to losses that come **before** congestion
  - 10Gbps over 100ms RTT needs <0.000003% packet loss (infeasible)
  - 1% loss (feasible) over 100ms RTT gets only 3Mbps
- Loss-based CC bloats buffers if loss comes **after** congestion

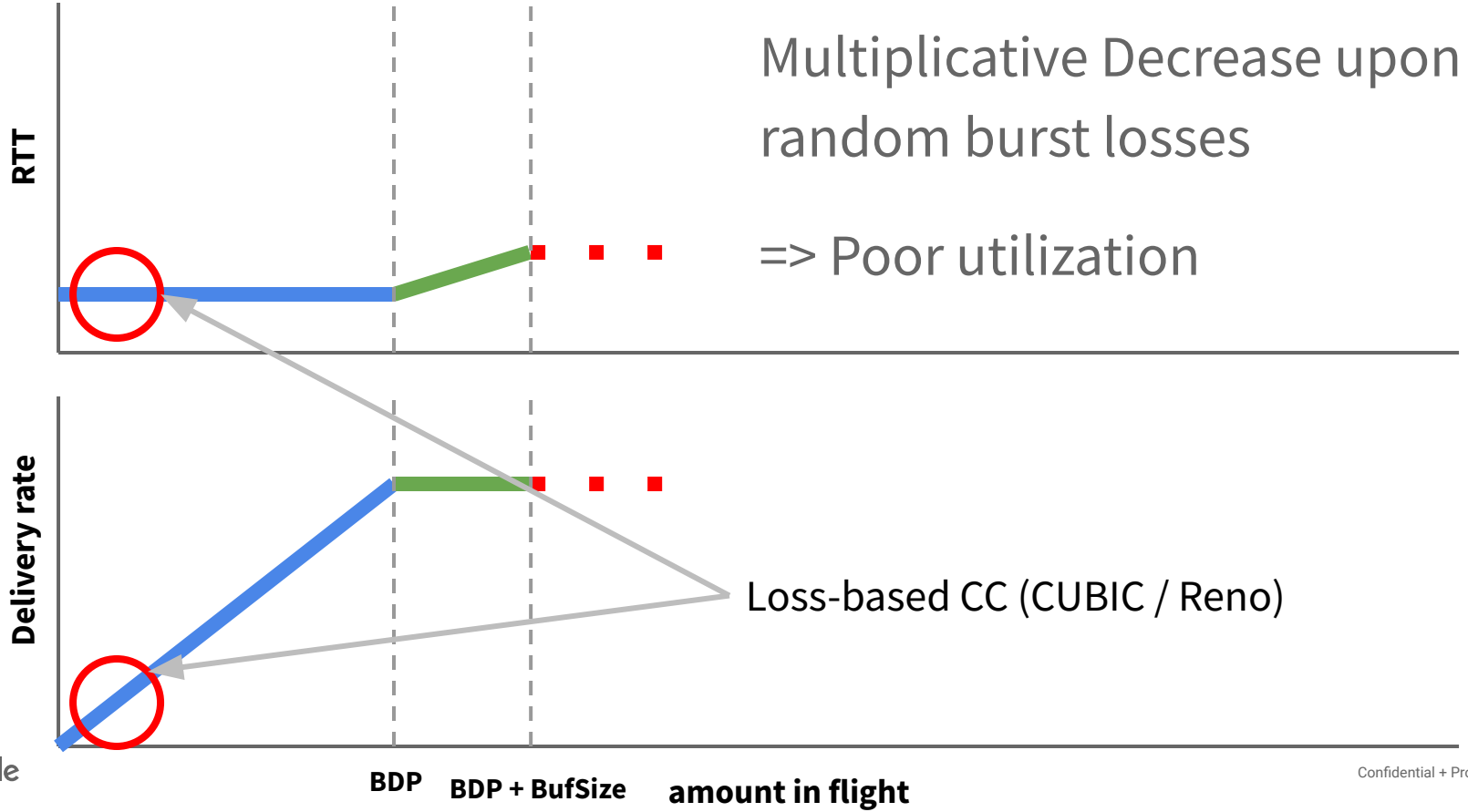
Network congestion and bottlenecks



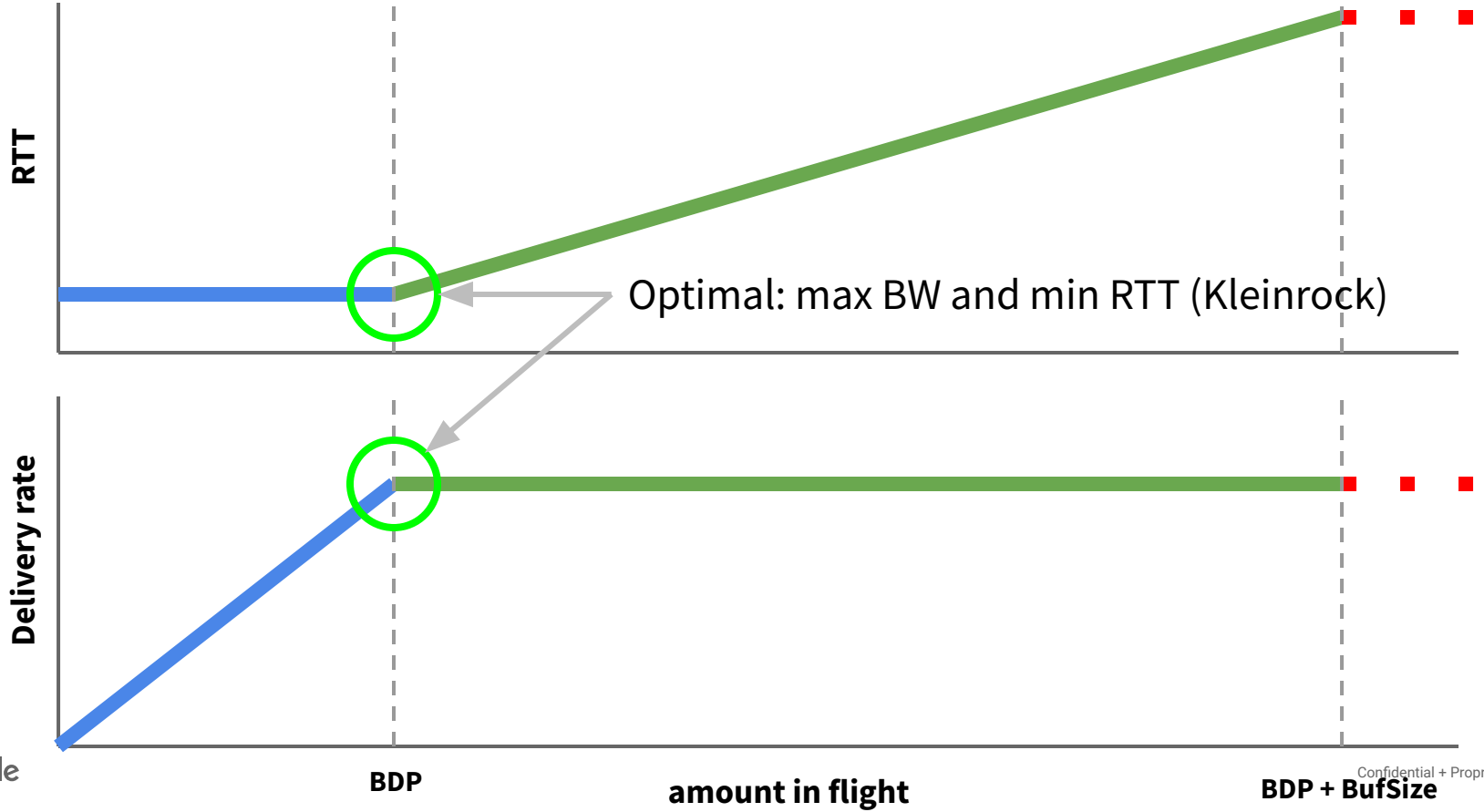
Loss-based congestion control in deep buffers



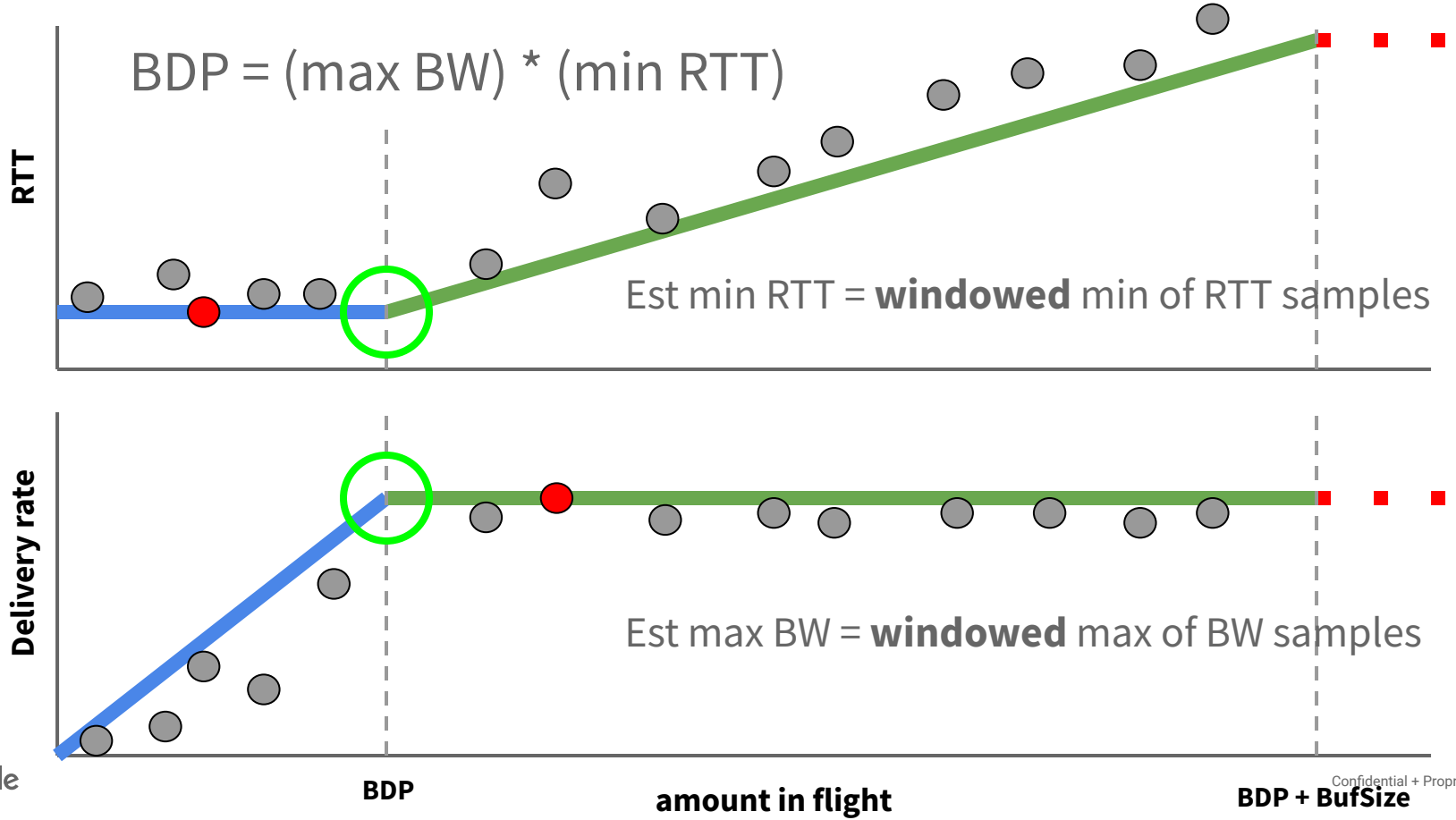
Loss-based congestion control in shallow buffers



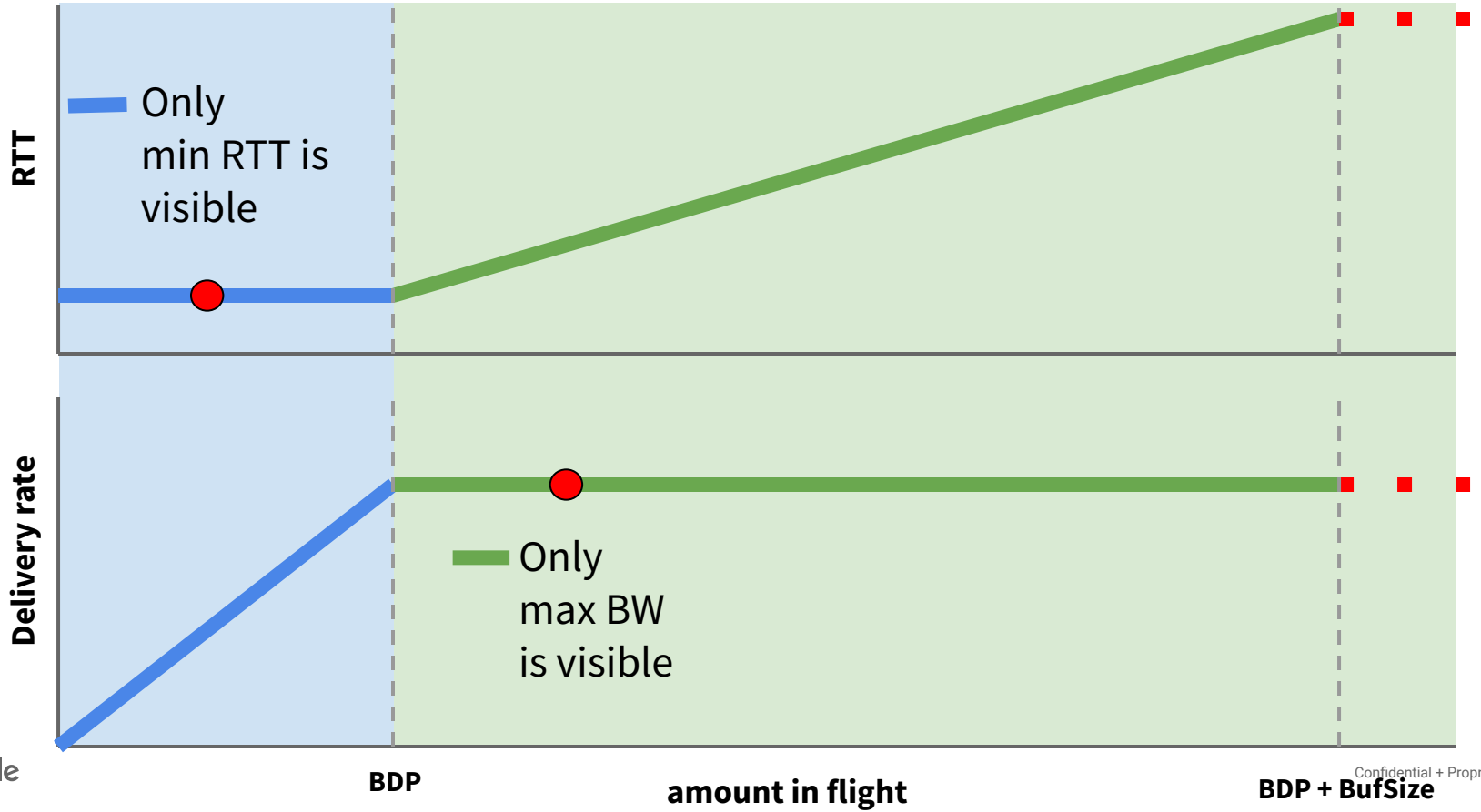
Optimal operating point



Estimating optimal point (max BW, min RTT)



To see max BW, min RTT: probe both sides of BDP



BBRv1: Design

BBR = **B**ottleneck **B**andwidth and **R**ound-trip propagation time

- Model network path
  - **Dynamically** estimate windowed max BW and min RTT on each ACK
- Control sending based on the model, to...
  - **Sequentially** probe max BW and min RTT, to feed the model samples
  - Pace near estimated BW
  - Vary pacing rate to keep inflight near BDP
- Seek high throughput with a small queue
  - Approaches maximum available throughput for random losses up to 15%
  - Maintains small queue independent of buffer depth

Evolution of BBR

BBR v2 [2019]: what's new?

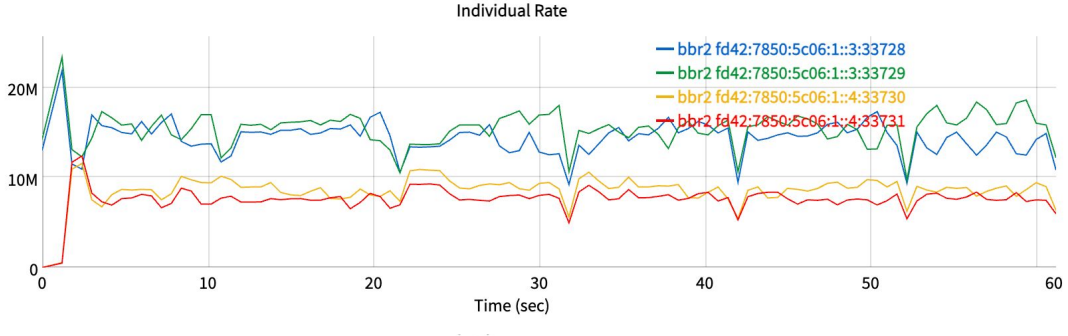
- Properties maintained between BBR v1 and BBR v2:
  - High throughput with a targeted level of random packet loss
  - Bounded queuing delay, despite bloated buffers
- Improvements from BBR v1 to BBR v2:
  - Improved coexistence when sharing bottleneck with Reno/CUBIC
  - Much lower loss rates for cases where bottleneck buffer < 1.5*BDP
  - High throughput for paths with high degrees of aggregation (e.g. wifi)
  - Responds to DCTCP/L4S-style ECN signals
  - Vastly reduced the throughput reduction in PROBE_RTT
- Following are a few tests, to illustrate the core properties maintained and improved...
  - Metrics we're evaluating in these:
    - throughput, queuing latency, retransmit rate, fairness

BBR v3 [2023]: what's new?

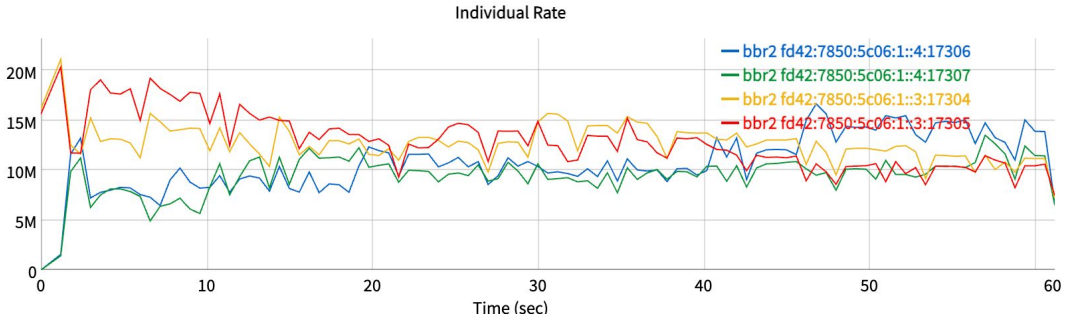
- BBR v3 is a minor evolution of BBR v2, with two areas of improvement:
  - 1: Bug fixes
    - Bandwidth convergence **with** loss and/or ECN marks
    - Bandwidth convergence **without** loss or ECN marks
  - 2: Performance tuning

BBR v3 bug fix 1: fix bw convergence *with* loss/ECN

Before bug fix 1:



After bug fix 1:



Example test results from:

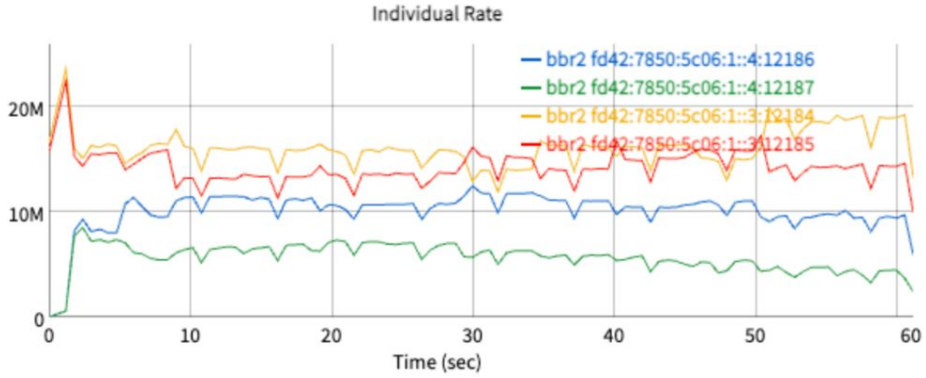
transperf bulk TCP transfer test with 4 TCP BBRv3 flows with

bottleneck_bw=50Mbps, min_rtt=40ms, **buffer=1*BDP**

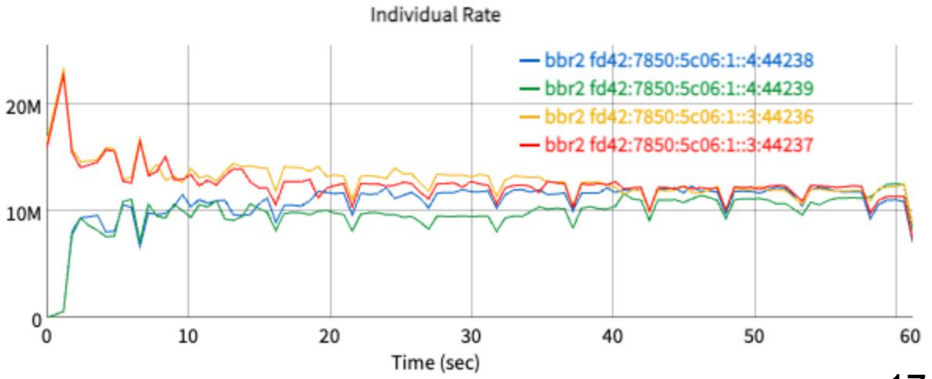
(at t=0s flows 0, 1 start; at t=1s flows 2, 3 start)

BBR v3 bug fix 2: fix bw convergence *without* loss/ECN

Before bug fix 2:



After bug fix 2:



Example test results from:

transperf bulk TCP transfer test with 4 TCP BBRv3 flows with

bottleneck_bw=50Mbps, min_rtt=40ms, **buffer=100*BDP**

(at t=0s flows 0, 1 start; at t=1s flows 2, 3 start)

BBR v3 performance tuning

- Performance tuning changes:
  - STARTUP cwnd gain: 2.89 => 2.0 [analytic derivation]
  - STARTUP pacing gain: 2.89 => 2.77 [analytic derivation]
  - When exiting STARTUP, set inflight_hi based on:
    - max(estimated BDP, max number of packets delivered in last round trip)
  - To trigger exit of STARTUP based on packet loss...
    - Require fewer loss events in a single round trip (6 rather than 8)
- Primary impact of these changes:
  - Lower queuing delays and packet loss rates during and shortly after STARTUP

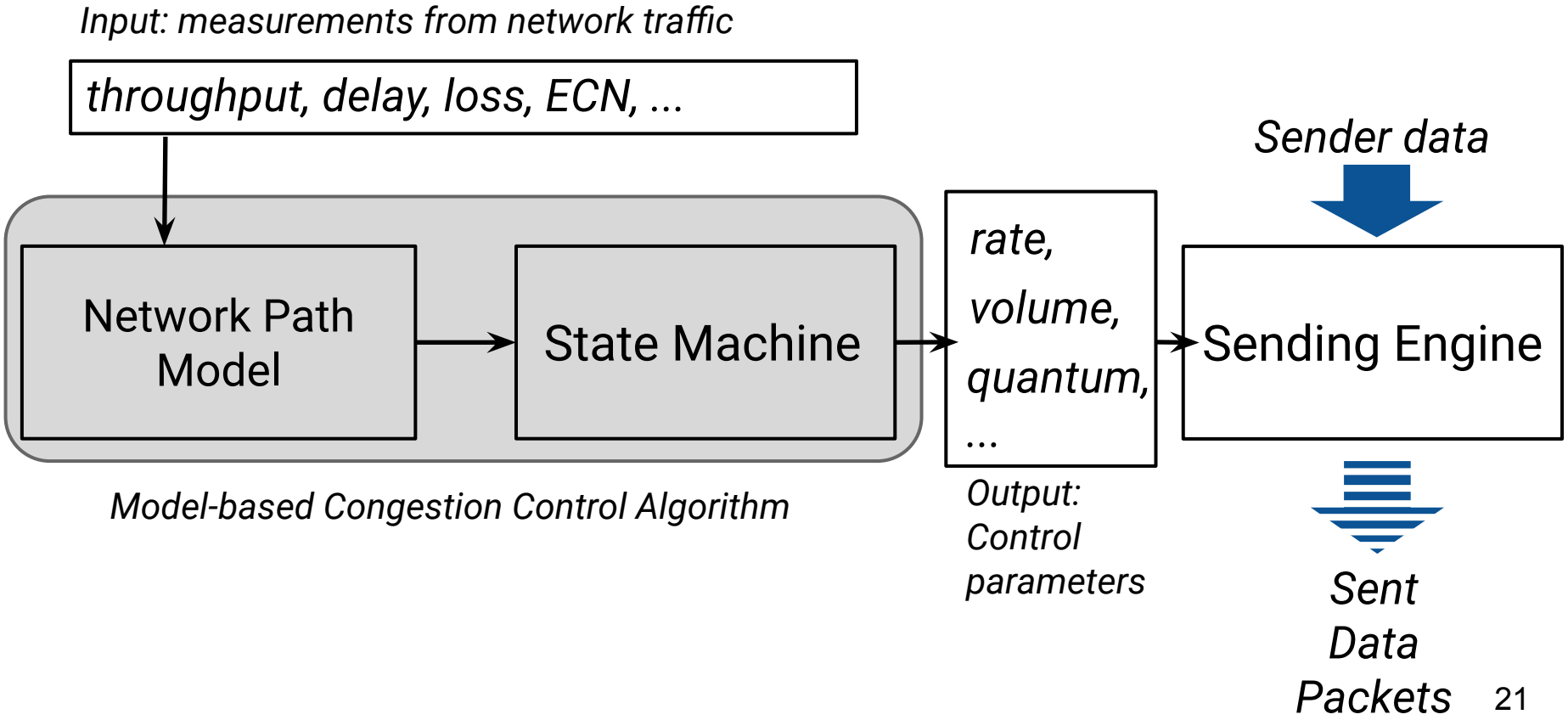
BBR v3 Properties

- BBR v3 properties:
  - Full throughput, with up to 1% random loss
  - Low queue delay, despite bloated buffers of any depth
  - Low queuing latency and loss using DCTCP/L4S-style ECN signals
  - Coexistence with usable throughput for CUBIC/Reno in the most common Internet and Datacenter scenarios

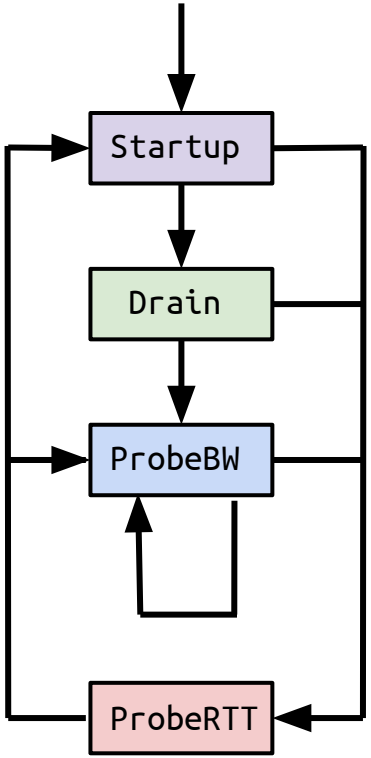
Evolution of BBR: a summary

	CUBIC	BBR v1	BBR v3
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 ceiling of 2%
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 throughput plateaus or ECN/loss rate > target

BBR v3 congestion control: the big picture

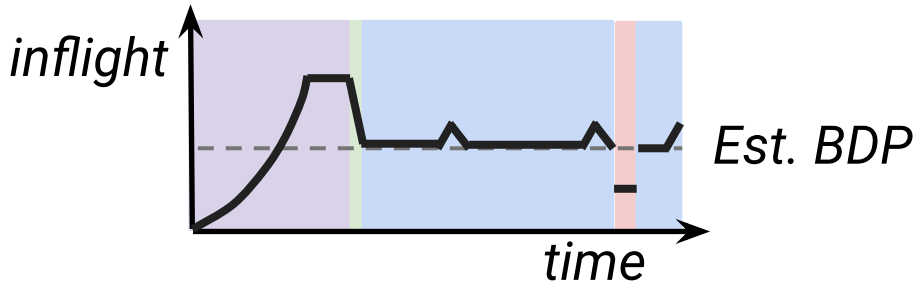


BBR: the state machine



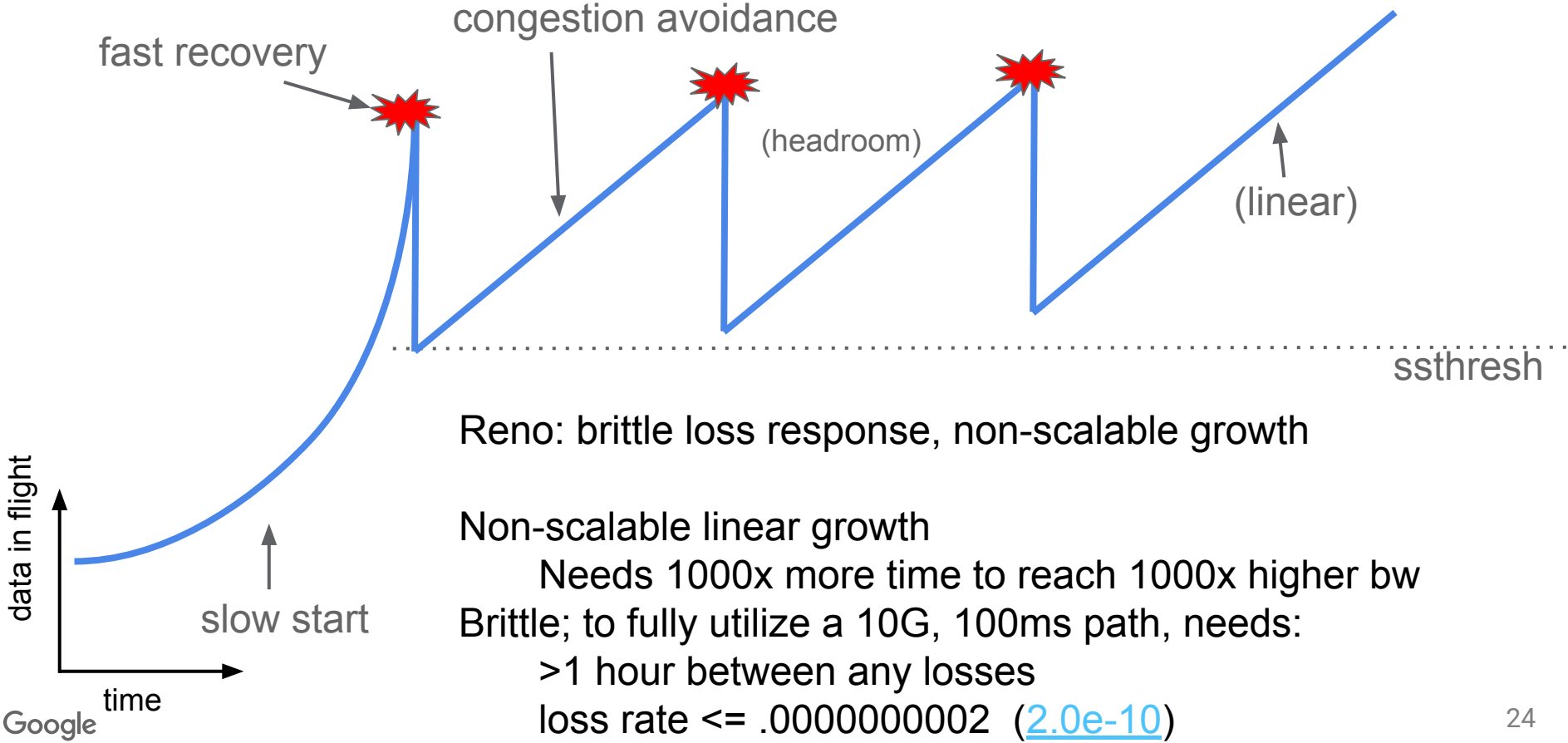
State machine uses 2-phase sequential probing of bw, RTT

- 1: raise inflight to probe BtlBw, get high throughput
- 2: lower inflight to probe RTprop, 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



Congestion Control algorithms: a comparison

Reno



Reno: brittle loss response, non-scalable growth

Non-scalable linear growth

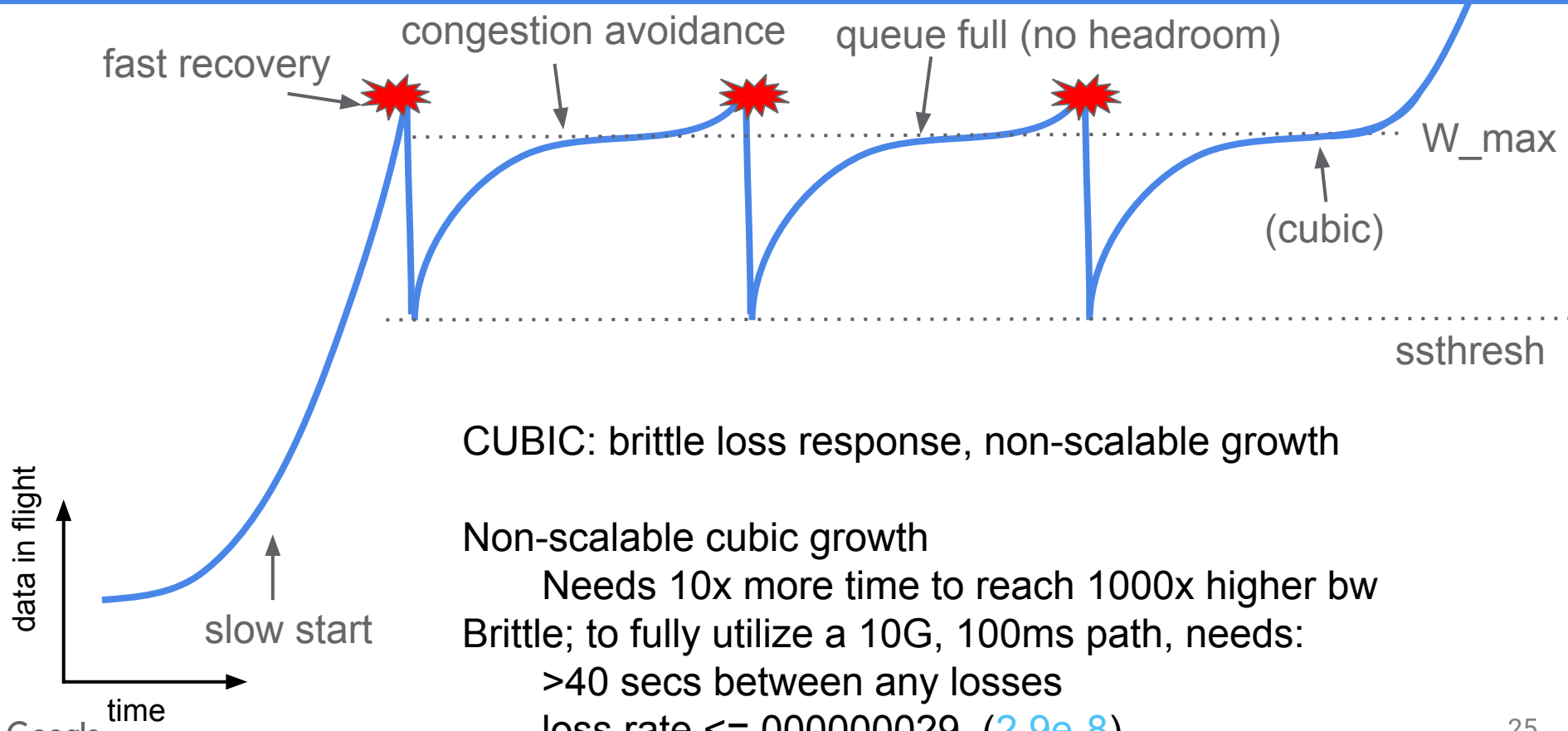
Needs 1000x more time to reach 1000x higher bw

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

>1 hour between any losses

loss rate <= .0000000002 (2.0e-10)

CUBIC



CUBIC: brittle loss response, non-scalable growth

Non-scalable cubic growth

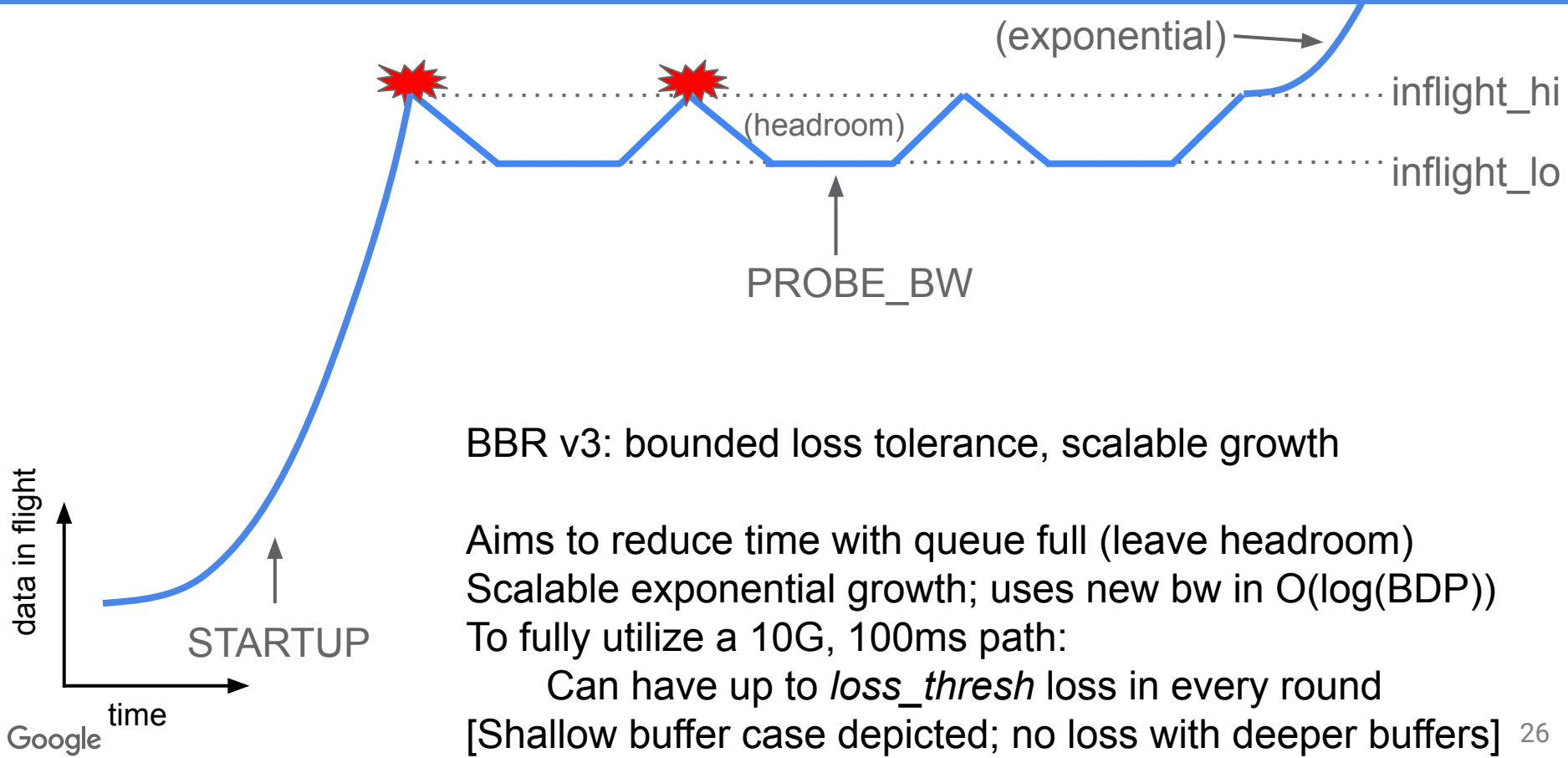
Needs 10x more time to reach 1000x higher bw

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

>40 secs between any losses

loss rate <=.000000029 (2.9e-8)

BBR v3



BBR v3: bounded loss tolerance, scalable growth

Aims to reduce time with queue full (leave headroom)
Scalable exponential growth; uses new bw in O(log(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]

Current Status of BBR

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 congestion control for all Google.com public Internet traffic
  - A/B experiments: BBRv3 vs BBRv1 for small % of users for:
    - TCP for YouTube
    - QUIC for google.com and YouTube

BBRv3 performance impact for public Internet traffic

- Impact of BBRv3 vs BBRv1 on Google.com and YouTube TCP public Internet traffic:
  - Lower retransmit rate (12% reduction)
  - Slight latency improvement (0.2% reduction) for:
    - Google.com web search
    - Starting YouTube video playback
  - Latency wins seem to be from lower loss rate (less/faster loss recovery)

BBR Open Source Code

- TCP BBRv3 release:
  - Linux TCP (dual GPLv2/BSD): github.com/google/bbr/blob/v3/README.md
  - Main updates: the bug fixes described earlier in this presentation
  - TCP BBR v3 release is open source (dual GPL/BSD), available for review/testing
  - Plan to email patches to propose inclusion in mainline Linux TCP
- 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)

How to Experiment with Linux TCP BBR

- How to enable BBR:
  - To enable manually for one-shot experimentation:
    - `sysctl net.ipv4.tcp_congestion_control=bbr`
  - To enable every time a machine boots:
    - Add to /etc/sysctl.conf (Ubuntu, Debian, RedHat, CentOS):
      - `net.ipv4.tcp_congestion_control=bbr`
      - `net.core.default_qdisc=fq`
- BBRv1 for TCP:
  - In mainline Linux (since v4.9 in Dec 2016)
- BBRv3 for TCP:
  - On github: **github.com/google/bbr/blob/v3/README.md**
- Pacing options:
  - Preferred: **fq** qdisc: implements pacing and fair queuing
  - If **fq** is not present, BBR uses TCP-layer pacing (usable since v4.20 in Dec 2018)

Conclusion

- Summary:
  - Open sourced BBRv3 on github with significant bug fixes vs BBRv2
  - BBRv3 used for all TCP for Google.com public Internet and internal WAN traffic
  - BBRv3 under A/B testing for YouTube TCP, YouTube and Google.com QUIC
- Next:
  - Plan on submitting BBRv3 for inclusion in mainline Linux TCP
  - Will update BBR Internet Drafts to cover BBRv3:
    - 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
- Thanks!

Q & A

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

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

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