BBR Congestion Control:

IETF 100 Update: BBR in shallow buffers

Neal Cardwell, Yuchung Cheng,

C. Stephen Gunn, Soheil Hassas Yeganeh

Ian Swett, Jana Iyengar, Victor Vasiliev

Van Jacobson

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

IETF 100: Singapore, Nov 13, 2017

Outline

- Quick review of BBR v1.0
- BBR v2.0
  - Summary of recent and upcoming BBR work at Google
  - Quick snapshot: Improving BBR behavior in shallow buffers: before and after
- Conclusion

The problem: loss-based congestion control

- BBR motivated by problems with loss-based congestion control (Reno, CUBIC)
- Packet loss alone is **not** a good proxy to detect congestion
- If loss comes **before** congestion, loss-based CC gets low throughput
  - 10Gbps over 100ms RTT needs <0.000003% packet loss (infeasible)
  - 1% loss (feasible) over 100ms RTT gets 3Mbps
- If loss comes **after** congestion, loss-based CC bloats buffers, suffers high delays

BBR (Bottleneck BW and RTT)

- **Model** network path: track windowed max BW and min RTT on each ACK
- Control sending rate based on the model
- **Sequentially** probe max BW and min RTT, to feed the model samples
- Seek high throughput with a small queue
  - Approaches maximum available throughput for random losses up to 15%
  - Maintains small, bounded queue independent of buffer depth

BBR v1.0: the story so far

- BBR milestones already mentioned at the IETF:
  - BBR is used for TCP and QUIC on Google.com, YouTube
  - All Google/YouTube servers and datacenter WAN backbone connections use BBR
    - Better performance than CUBIC for web, video, RPC traffic
  - Code is available as open source in Linux TCP (dual GPLv2/BSD), QUIC (BSD)
  - Active work under way for BBR in FreeBSD TCP @ NetFlix
  - BBR Internet Drafts are out and ready for review/comments:
    - Delivery rate estimation: draft-cheng-iccrg-delivery-rate-estimation
    - BBR congestion control: draft-cardwell-iccrg-bbr-congestion-control
  - IETF presentations: IETF 97 | IETF 98 | IETF 99
  - Overview in Feb 2017 CACM

BBR v2.0: current areas of research focus at Google

- Reducing loss rate in shallow buffers
  - Further tuning for handling both deterministic and stochastic loss
  - Faster exit of Startup mode
- Reducing queuing delay
  - "Drain to target": pacing at sub-unity gain to keep inflight closer to available BDP
- Improving fairness
  - Detailed update on progress for this issue at a future IETF
- Improving throughput on wifi, cellular, cable networks with widespread ACK aggregation
  - Improving bandwidth estimation
  - Provisioning enough data in flight by modeling ACK aggregation
    - Latest wifi LAN testbed results increase BBR bw from 40 Mbps to 270 Mbps
- Reducing queuing and loss in datacenter networks with large numbers of flows
- Plan is to use BBR for all Google TCP and QUIC traffic: datacenter, WAN, public Internet

BBR v2.0: changes recently deployed at Google

- Goal: reducing queuing/losses on shallow-buffered networks and/or with cross traffic
- Changes July-Oct 2017:
  - Gentler PRR-inspired packet scheduling during loss recovery
  - Refined cwnd provisioning for TSO quantization
  - Refined bandwidth probing for app-limited traffic

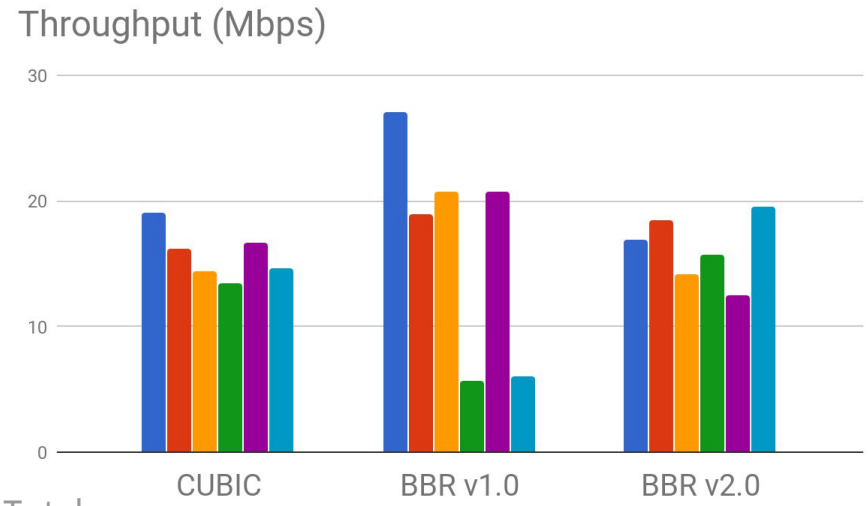
BBR v1.0: behavior in shallow buffers

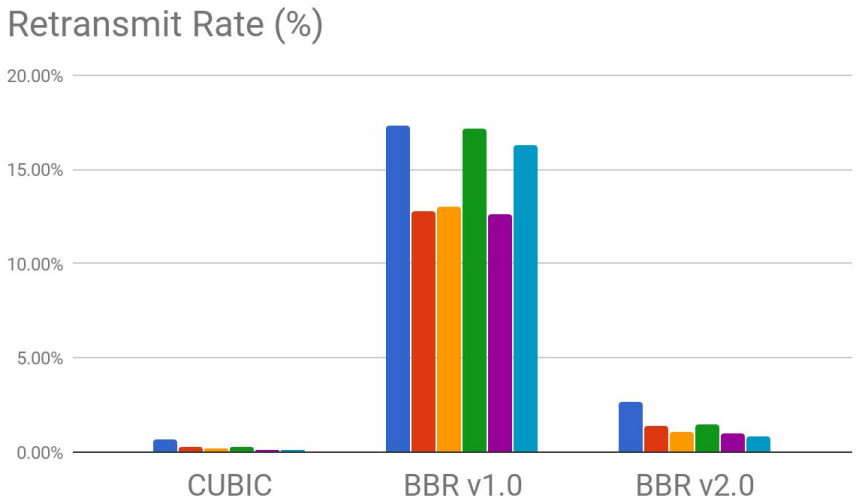
- BBR v1.0 has known issues in shallow buffers (previously discussed in IETF, bbr-dev)
  - Competing bulk BBR flows tend toward ~1*BDP of data in the queue
  - Thus, if buffer is smaller => high packet loss
- Root cause: BBR v1.0: bandwidth probing and estimation dynamics
  - Mainly: BW probing based on simple static proportions of model parameters
    - Probes at 1.25x estimated bandwidth once every 8 round trips
  - Based on trade-offs among competing real-world design considerations:
    - Cell systems with dynamic bw allocation need significant backlog
    - Shallow buffers need compensation for stochastic loss
  - BBR v1.0 used a simple "one size fits all" static bw-probing gain and frequency
  - BBR v2.0 will use a dynamically adaptive approach...

BBR v2.0: changes related to shallow buffers

- Goal: reduce queuing delay & packet loss, allow loss-based CC to maintain higher rates
- Generalized and simplified the long-term bandwidth estimator
  - Previously only targeted at policers
  - Now applied from start of any fast recovery until next bandwidth probe phase
  - Estimates long_term_bw = windowed average bandwidth over last 2-3 round trips
- New algorithm parameters to adapt to shallow buffers:
  - Max safe volume of inflight data (before we seem to fill the buffer and cause loss)
  - Volume of data with which to probe (probing starts at 1 packet, doubles upon success)
- New "full pipe+buffer" estimator uses loss rate signal to adapt to shallow buffers
  - Triggers if loss rate (over scale of round-trip) > 5%
  - Upon "full pipe+buffer" trigger event:
    - Set estimate of max safe volume of inflight data to current flight size
    - Multiplicative decrease (0.85x) for scalable/fast fairness
    - Before re-probing BW, scalable wait (1-4sec, RTT-fair) as a function of estimated BW
- WIP: further work under way...

BBR in shallow buffers: before (v1.0) and after (v2.0)

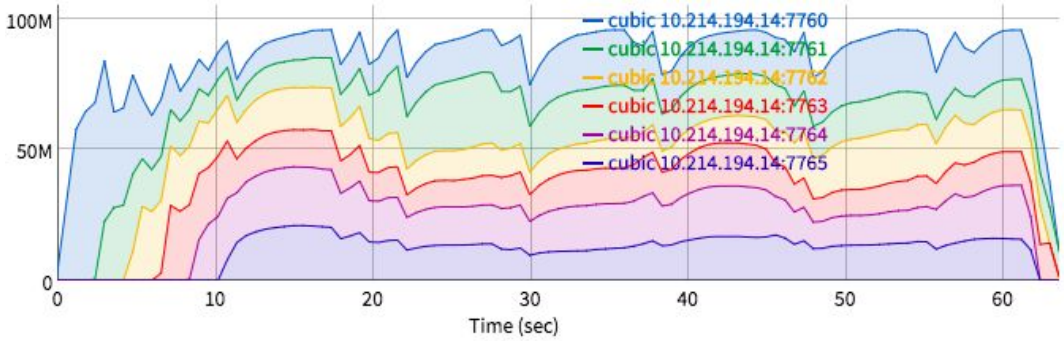




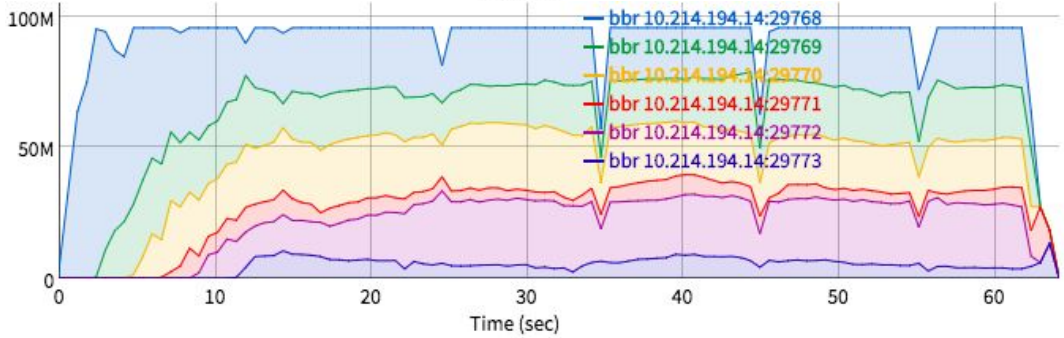
	CUBIC	BBR v1.0	BBR v2.0
Total:	88.9	93.5	89.9
t=20-60s:	90.4	93.8	92.0

CUBIC	BBR v1.0	BBR v2.0
0.3%	15%	1.4%
0.06%	14%	1.3%

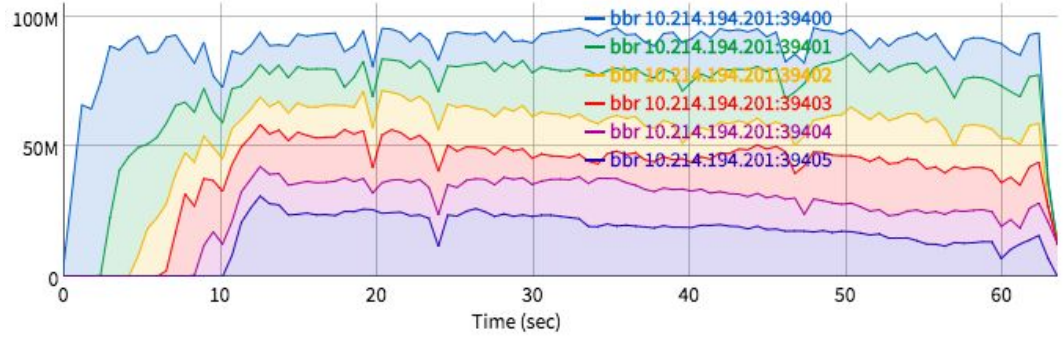
60-sec bulk TCP netperf, 6 flows (t=0,2,4,6,8,10), bw = 100Mbps, RTT 100ms, buffer = 5% of BDP (41 packets)



← CUBIC



← BBR v1.0



← BBR v2.0

Conclusion

- Status of BBR v1.0
  - Deployed widely at Google
  - Open source for Linux TCP and QUIC
  - Documented in IETF Internet Drafts
- Actively working on BBR v2.0
  - Linux TCP and QUIC at Google
  - Work under way for BBR in FreeBSD TCP @ NetFlix
  - Always happy to hear test results or look at packet traces...

Q & A

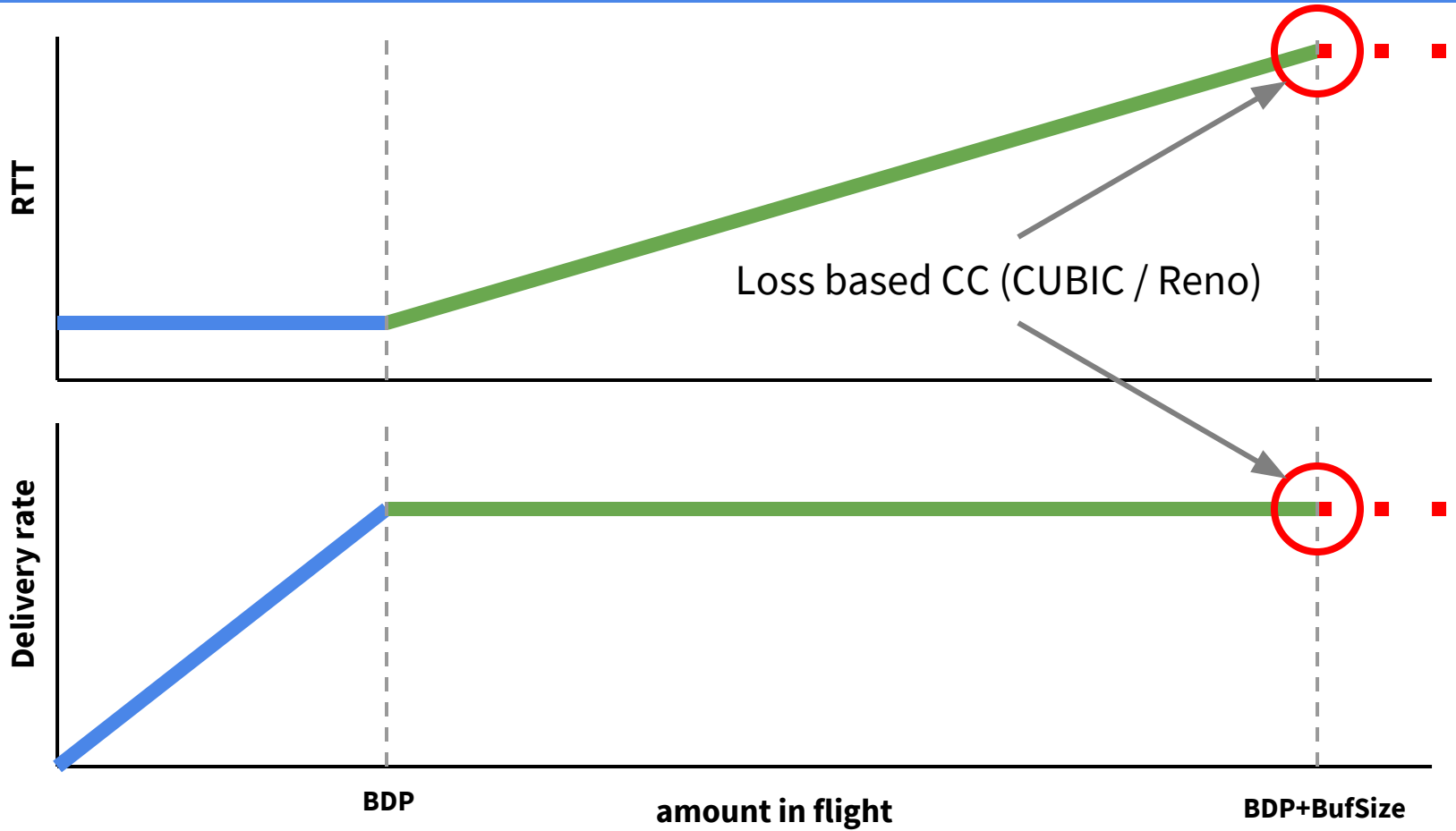
https://groups.google.com/d/forum/bbr-dev

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

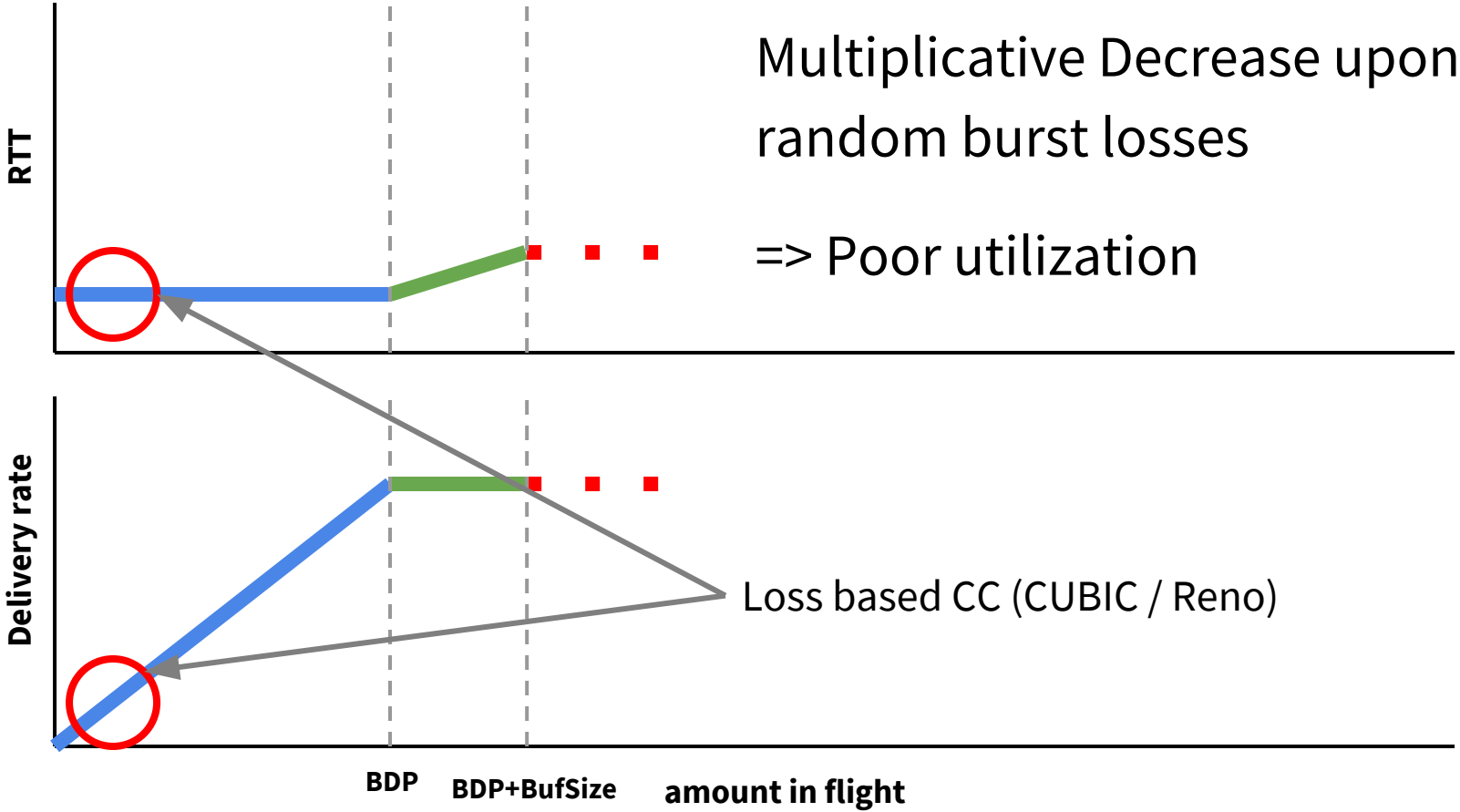
Special thanks to Eric Dumazet, Nandita Dukkipati, Pawel Jurczyk, Biren Roy, David Wetherall, Amin Vahdat, Leonidas Kontothanassis, and {YouTube, google.com, SRE, BWE} teams.

Backup slides from previous BBR talks...

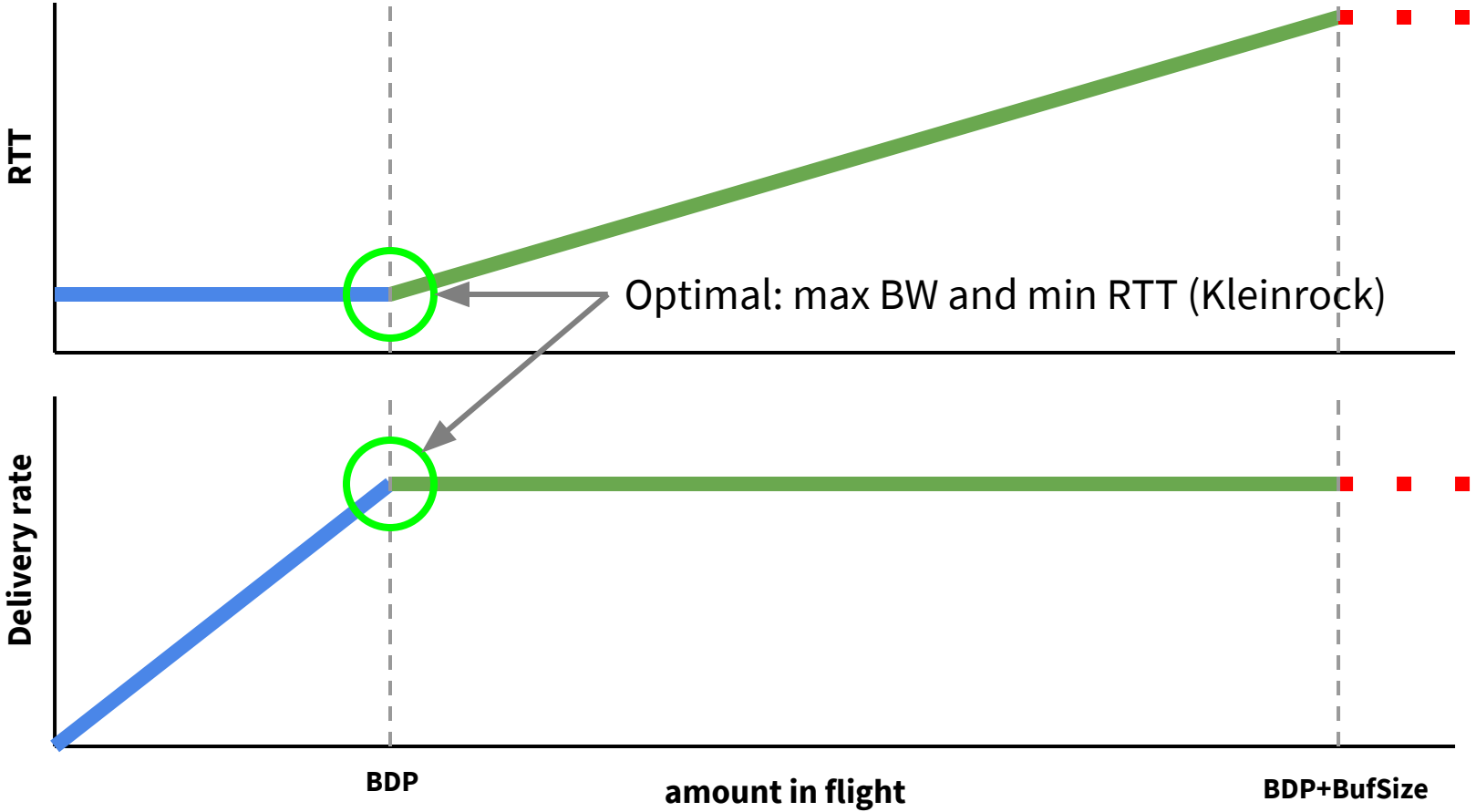
Loss based congestion control in deep buffers



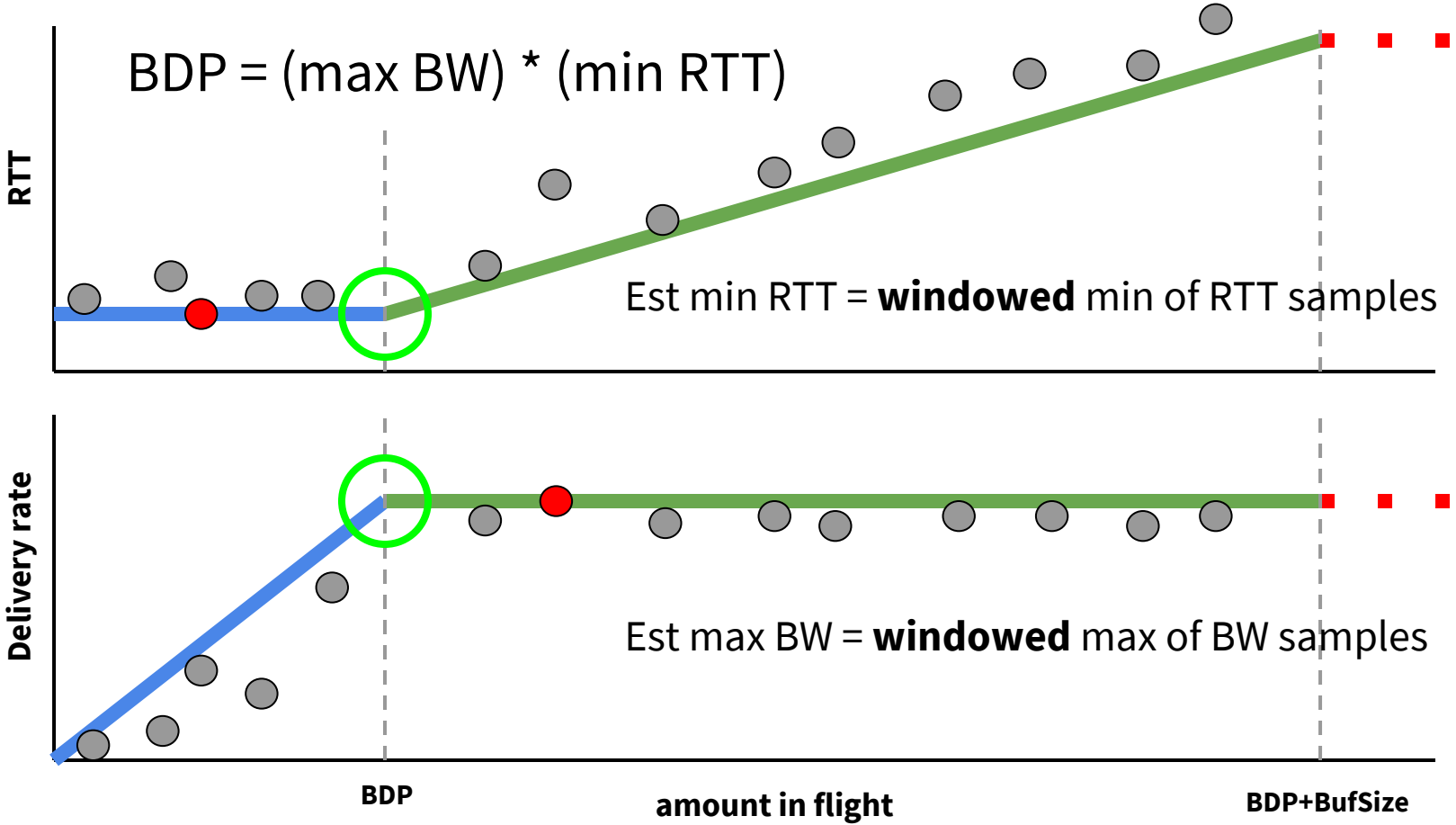
Loss based congestion control in shallow buffers



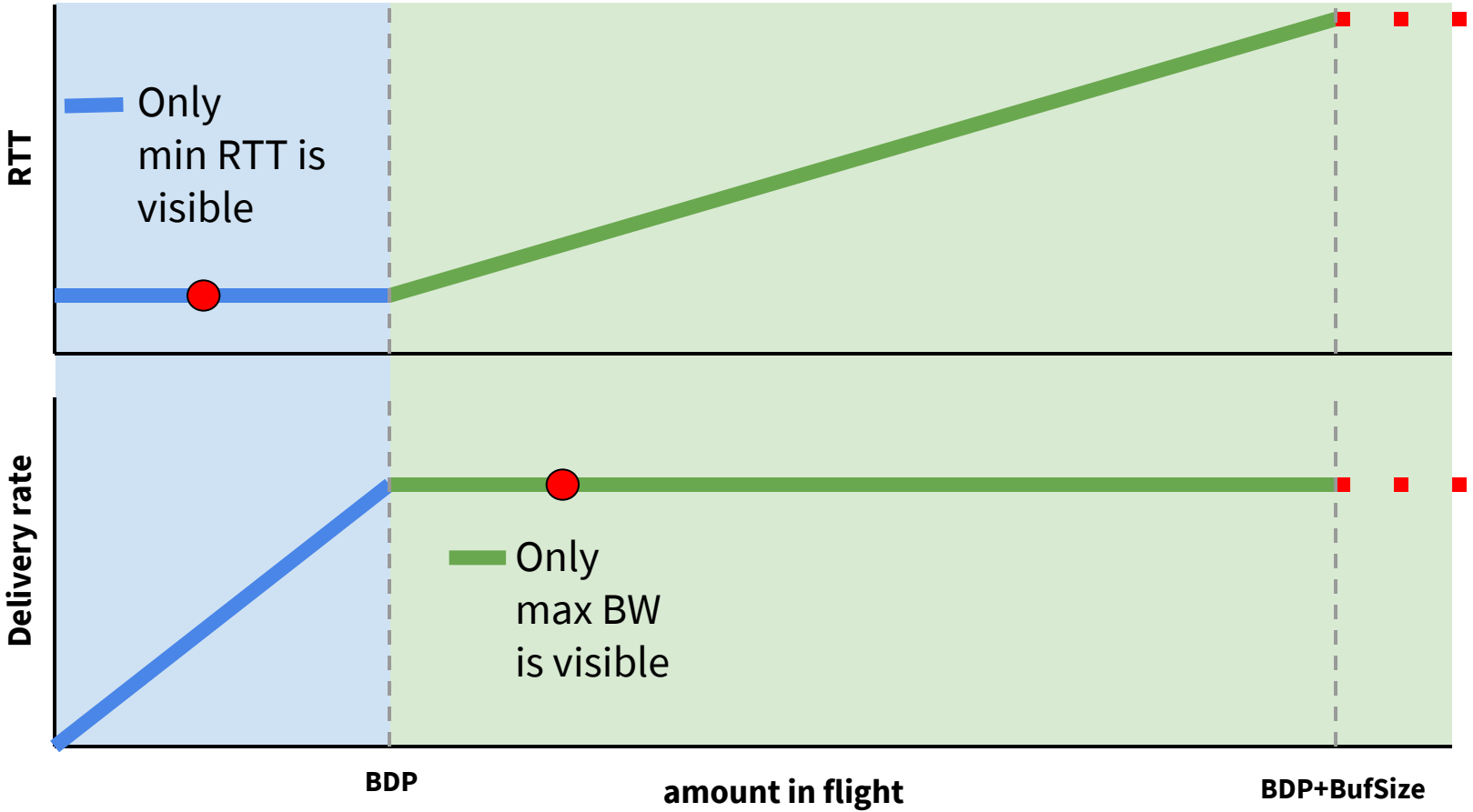
Optimal operating point



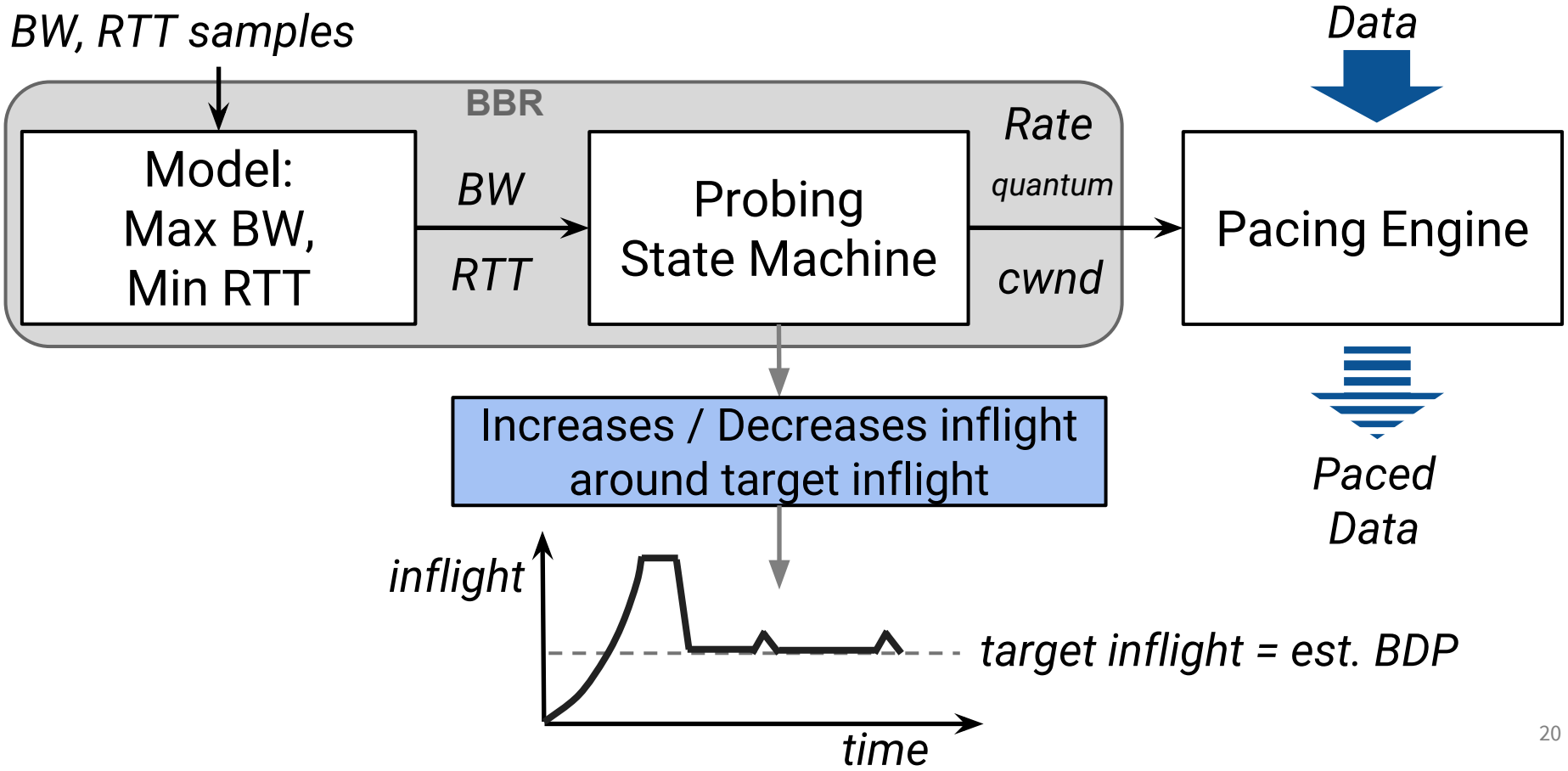
Estimating optimal point (max BW, min RTT)



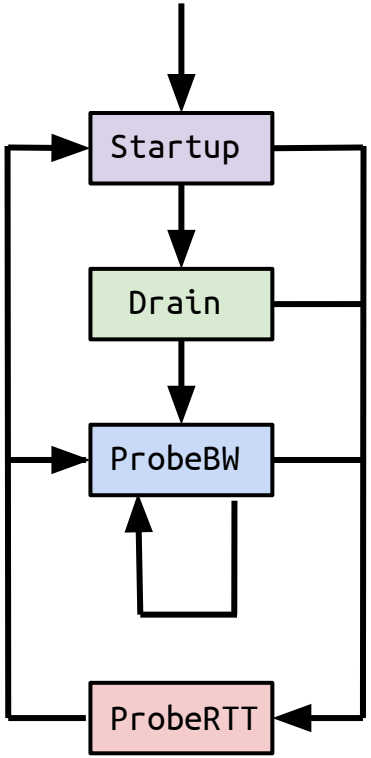
To see max BW, min RTT: probe both sides of BDP



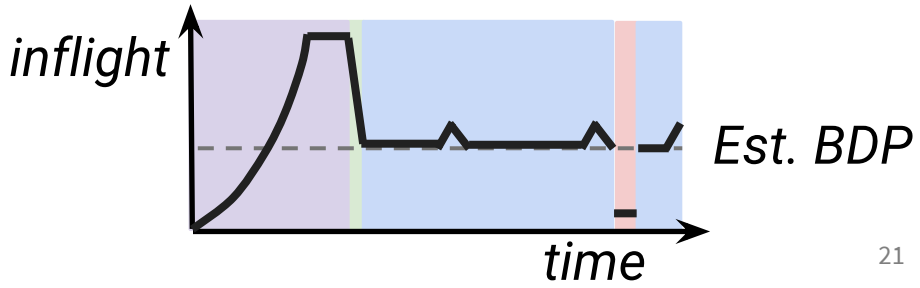
BBR congestion control: the big picture



BBR: probing state machine



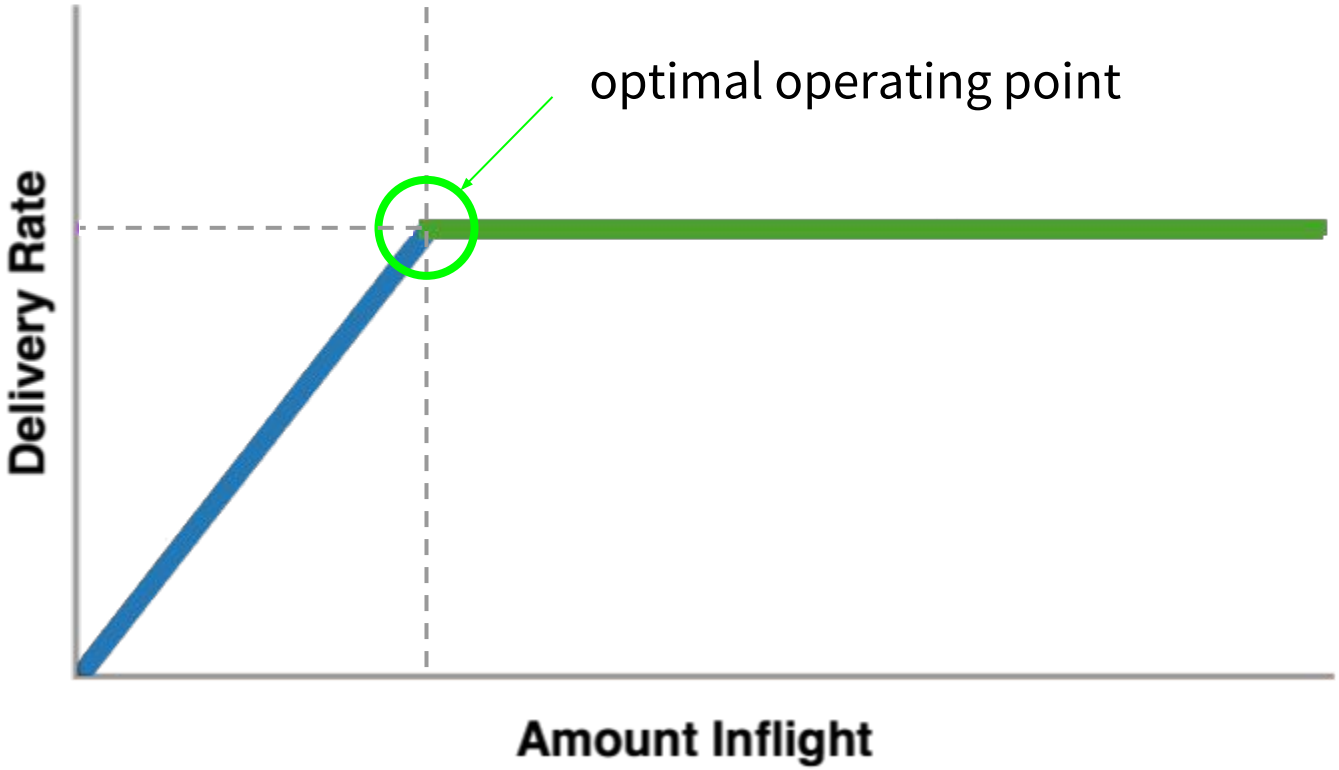
- State machine for 2-phase sequential probing:
  - 1: raise inflight to probe BtlBw, get high throughput
  - 2: lower inflight to probe RTprop, get low delay
  - At two different time scales: warm-up, steady state...
- Warm-up:
  - Startup: ramp up quickly until we estimate pipe is full
  - Drain: drain 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



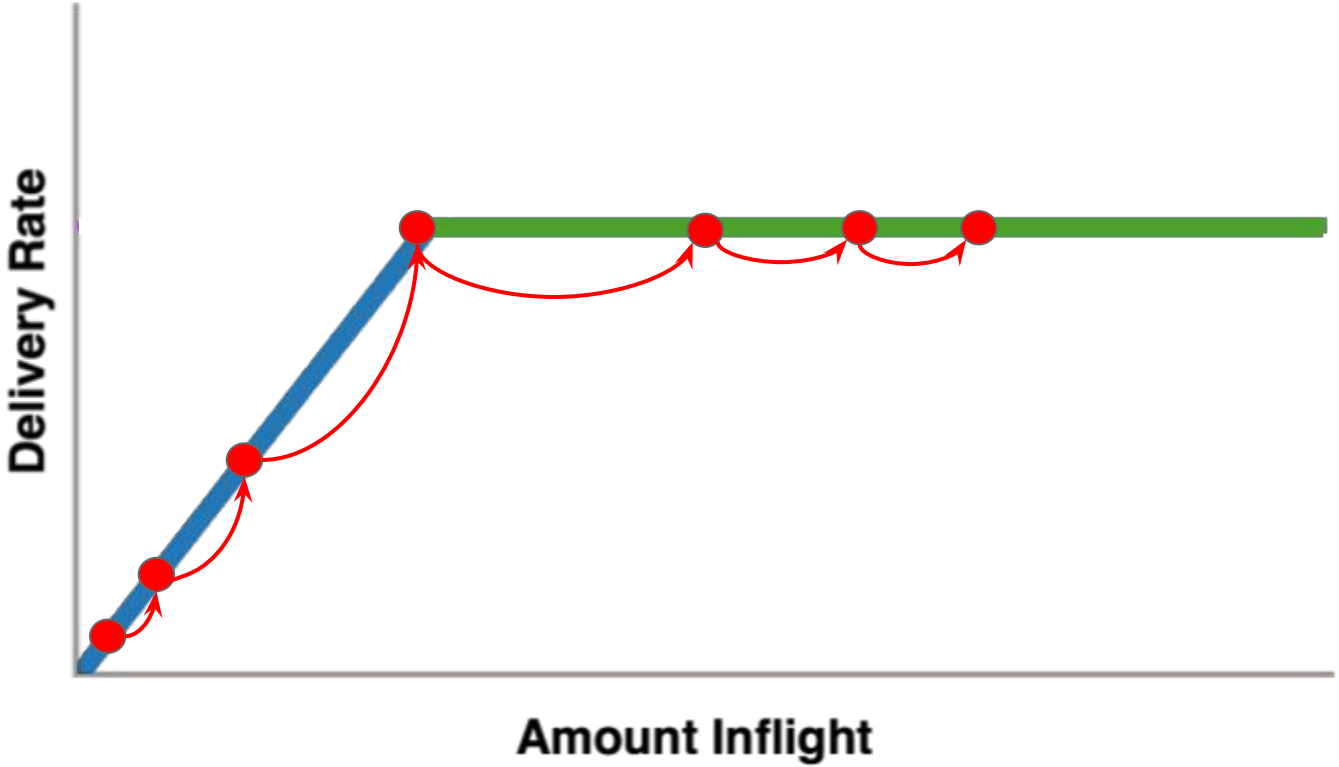
BBR congestion control algorithm: Internet Draft

- draft-cardwell-iccrg-bbr-congestion-control
- Network path model
  - BtlBw: estimated bottleneck bw available to the flow, from windowed max bw
  - RTprop: estimated two-way propagation delay of path, from windowed min RTT
- Target operating point
  - Rate balance: to match available bottleneck bw, pace at or near estimated bw
  - Full pipe: to keep inflight near BDP, vary pacing rate
- Control parameters
  - Pacing rate: max rate at which BBR sends data (primary control)
  - Send quantum: max size of a data aggregate scheduled for send (e.g. TSO chunk)
  - Cwnd: max volume of data allowed in-flight in the network
- Probing state machine
  - Using the model, dial the control parameters to try to reach target operating point

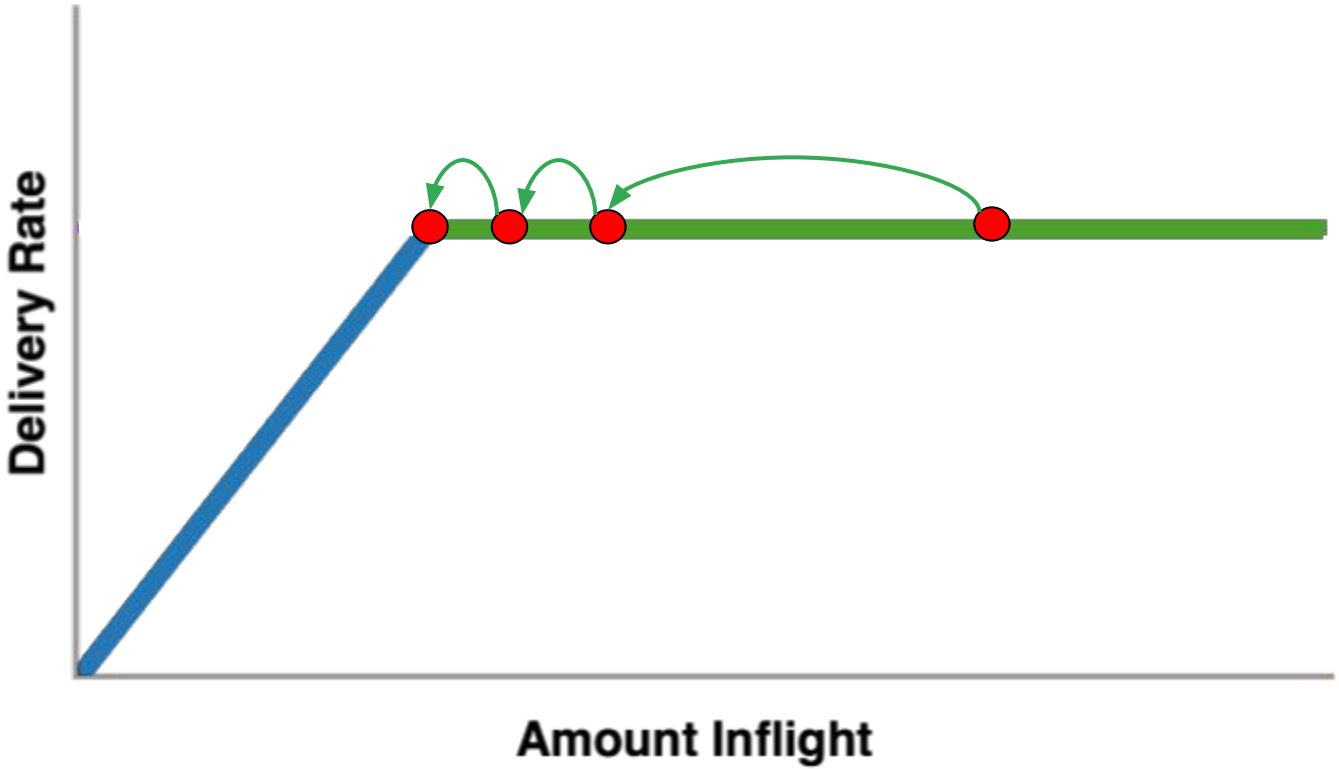
BBR: model based walk toward max BW, min RTT



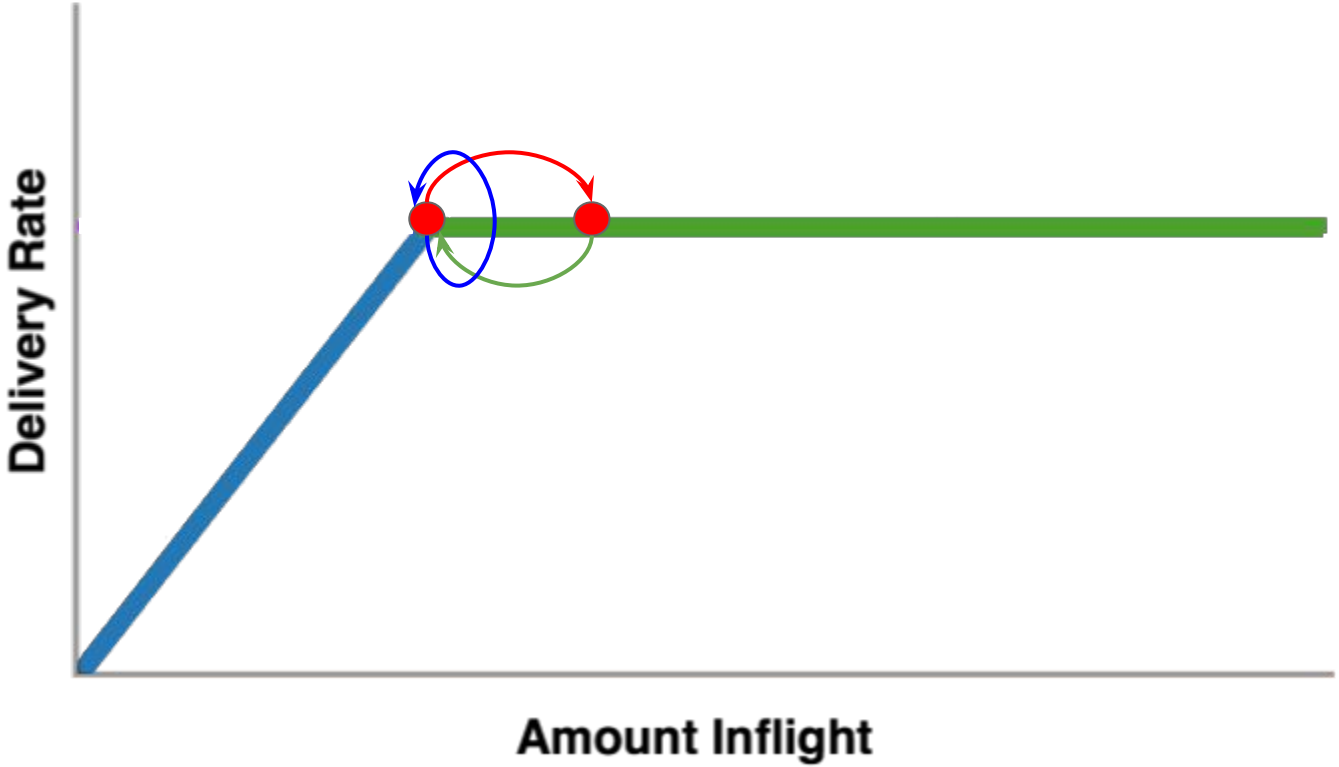
STARTUP: exponential BW search



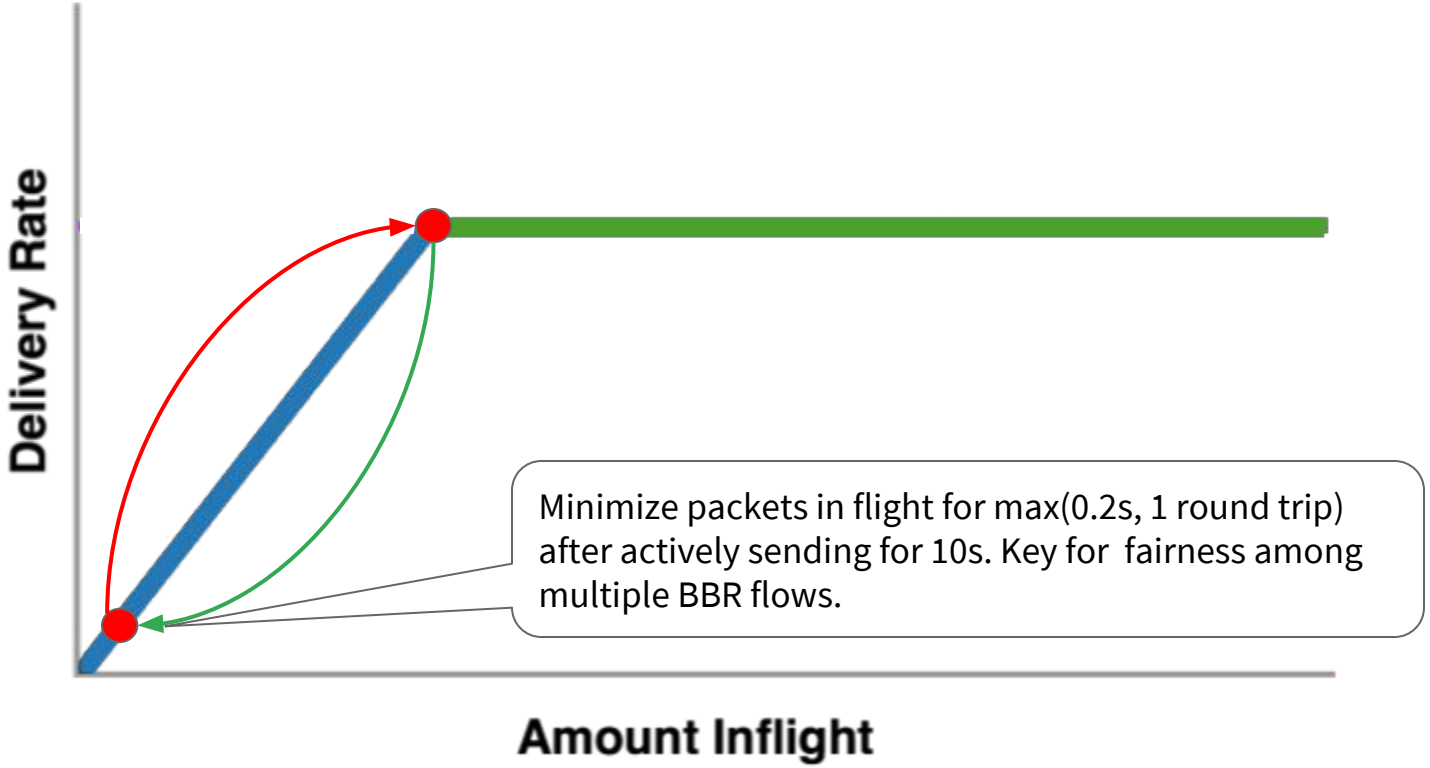
DRAIN: drain the queue created during STARTUP

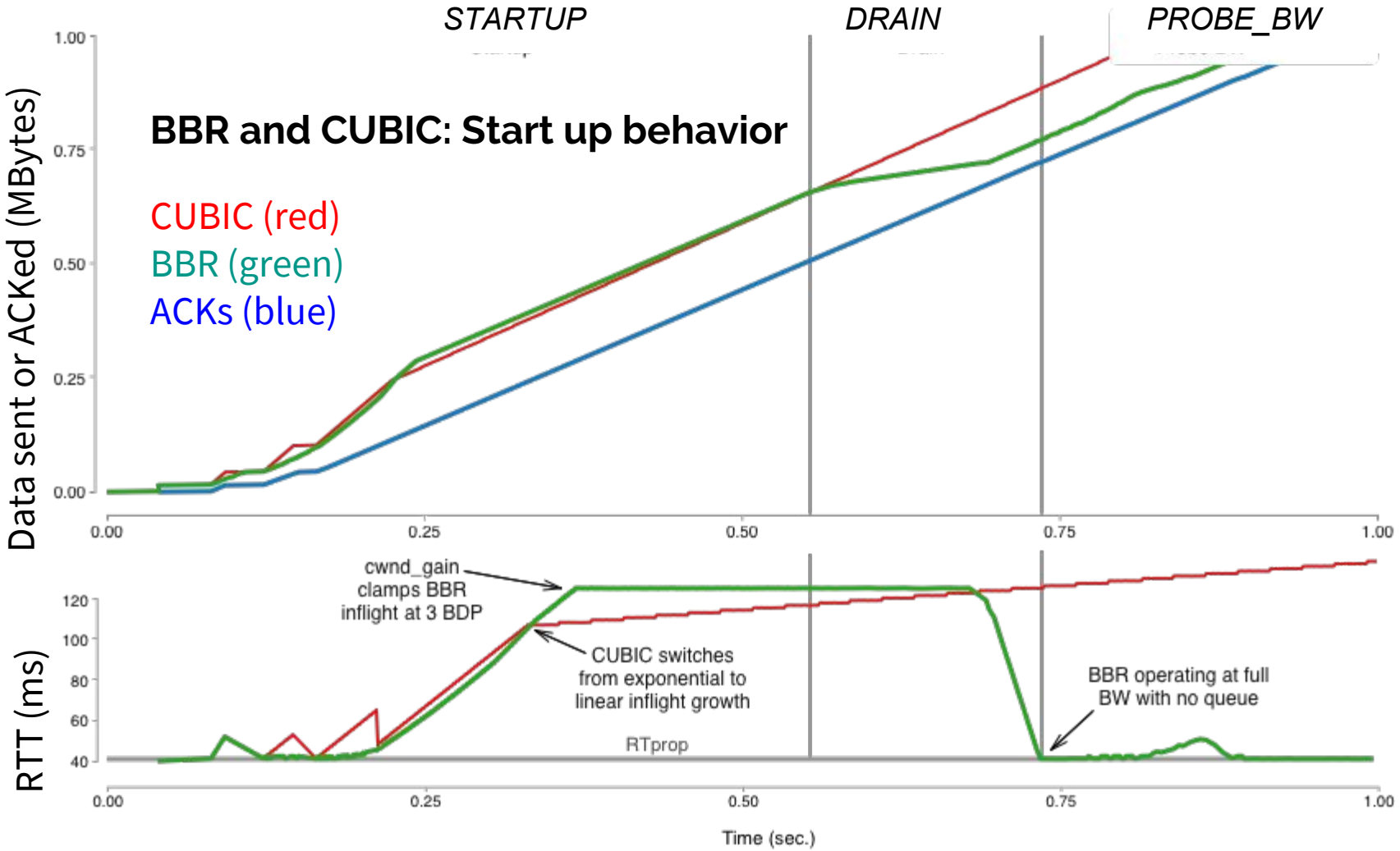


PROBE_BW: explore max BW, drain queue, cruise



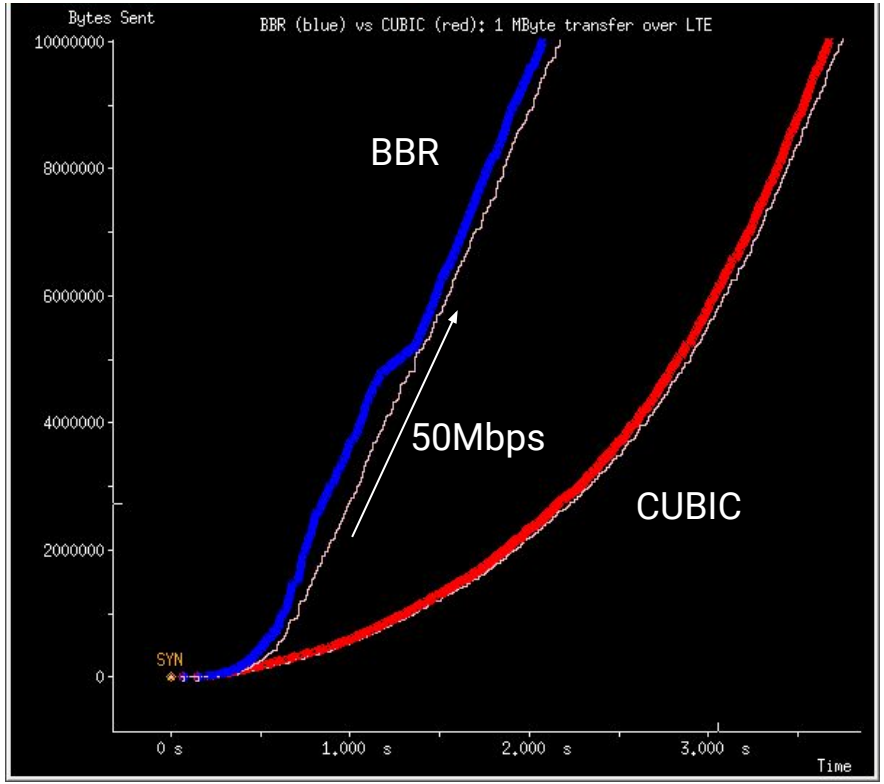
PROBE_RTT: drains queue to refresh min RTT




STARTUP
DRAIN
PROBE_BW
BBR and CUBIC: Start up behavior
CUBIC (red)
BBR (green)
ACKs (blue)
Data sent or ACKed (MBytes)
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
cwnd_gain
clamps BBR
inflight at 3 BDP
CUBIC switches
from exponential to
linear inflight growth
RTprop
BBR operating at full
BW with no queue
RTT (ms)
120
100
80
60
40
Time (sec.)

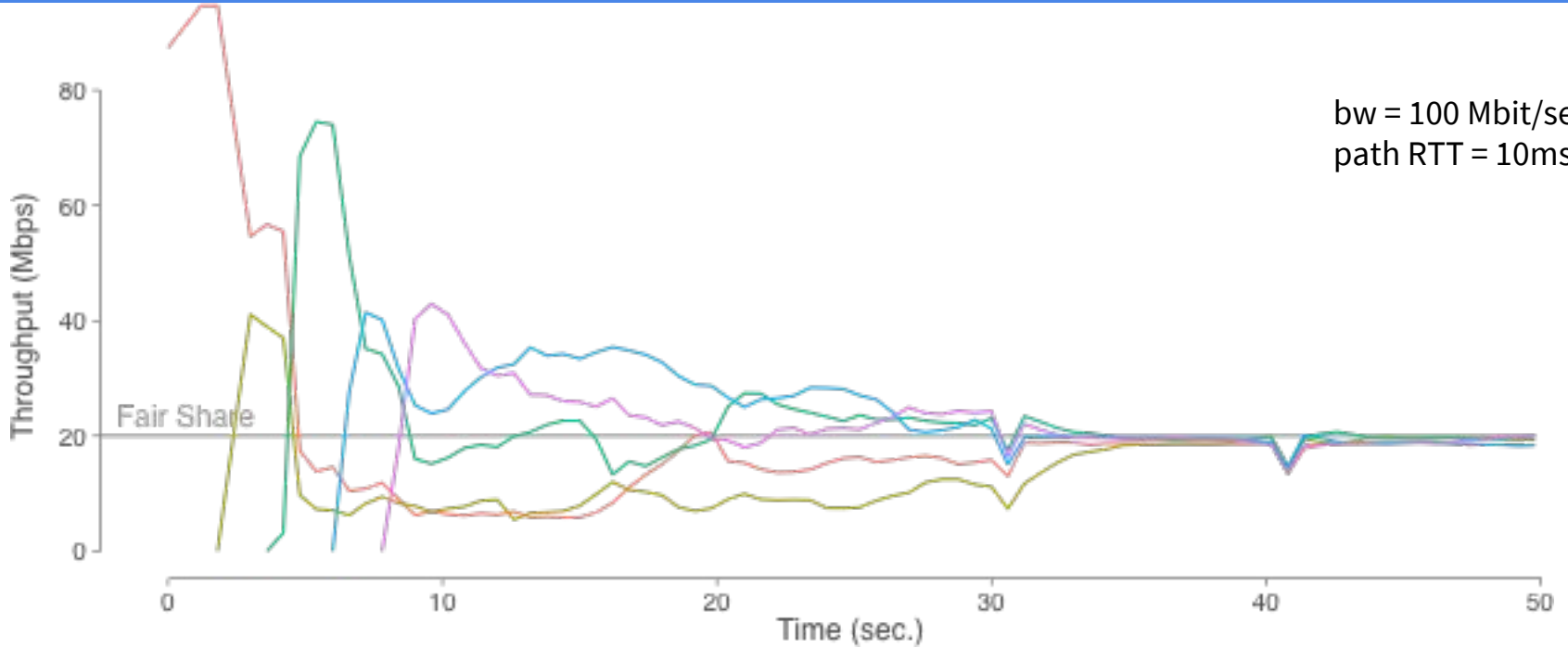
BBR: faster for short flows, too

	Cubic (Hystart)	BBR
Initial rate	10 packets / RTT	
Acceleration	2x per round trip	
Exit acceleration	A packet loss or significant RTT increase	Delivery rate plateaus



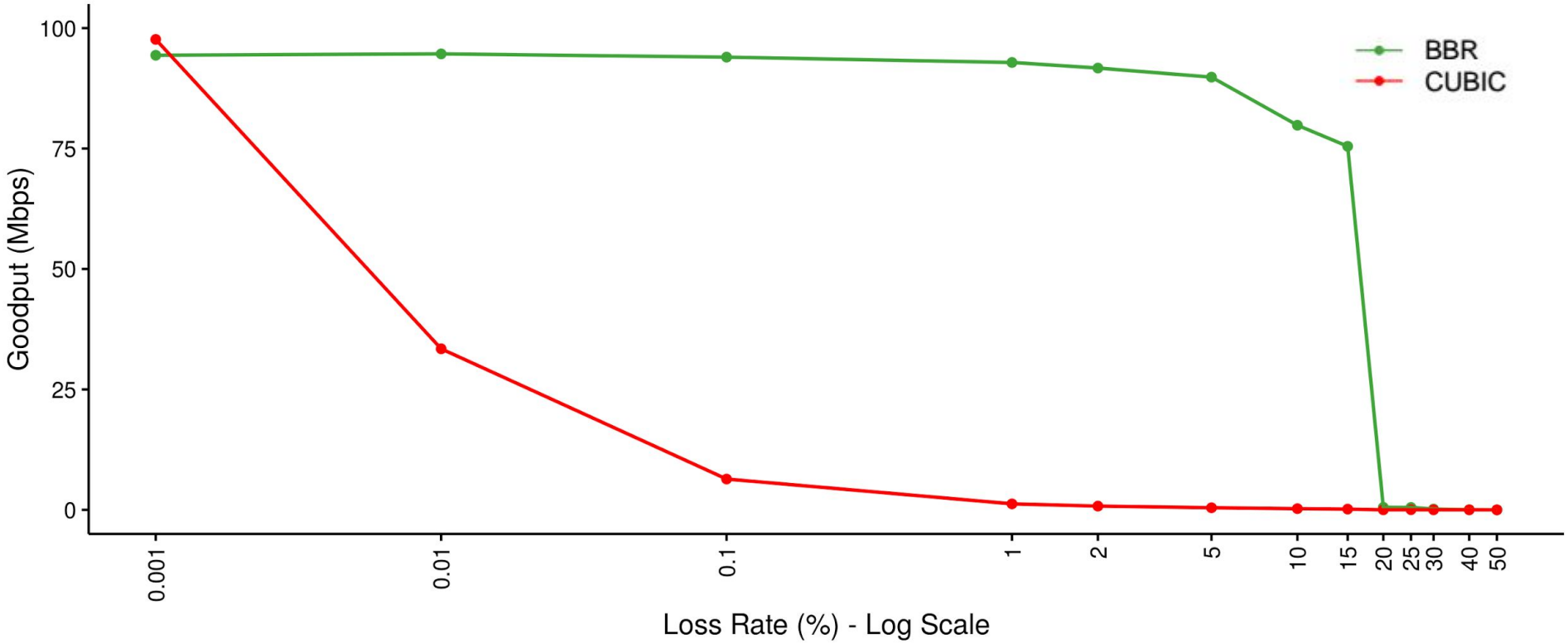
BBR and Cubic time series overlaid. BBR downloads 1MB 44% faster than Cubic. Trials produced over LTE on Neal's phone in New York

BBR multi flow convergence dynamics



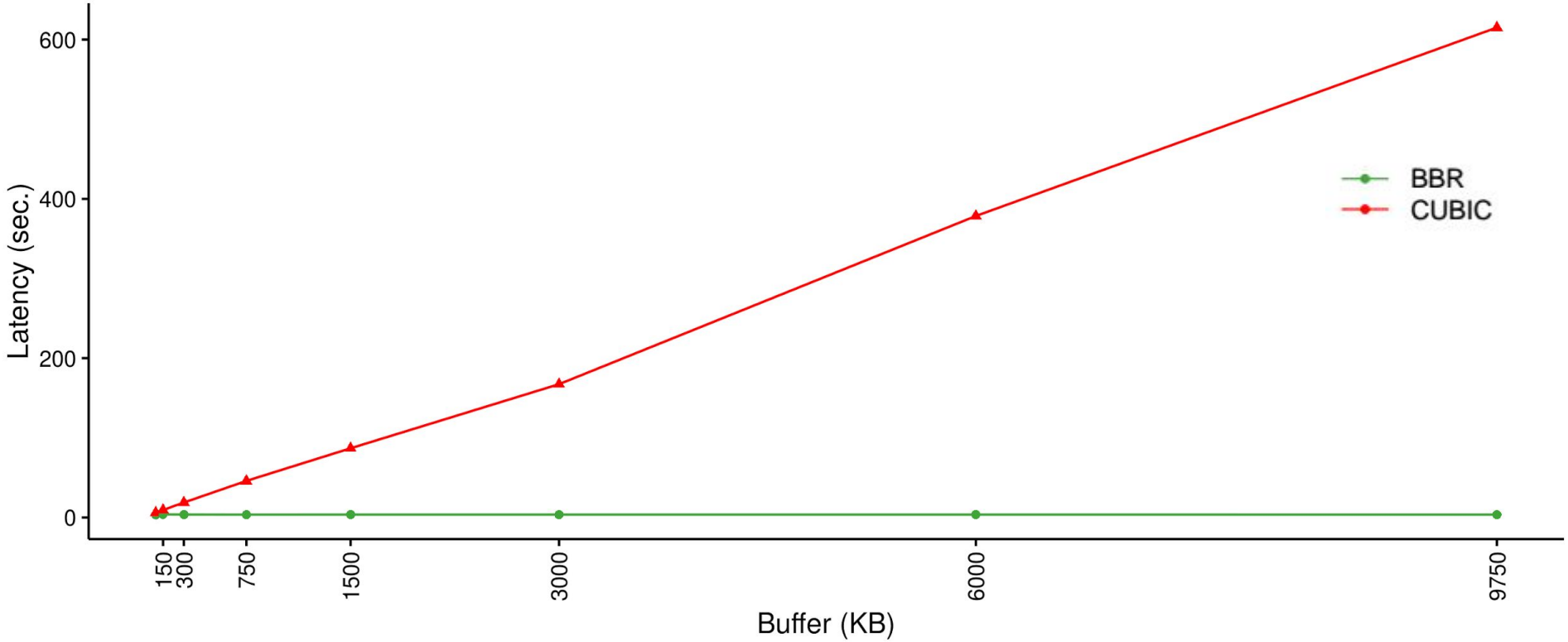
1. Flow 1 briefly slows down to reduce its queue every 10s (PROBE_RTT mode)
2. Flow 2 notices the queue reduction via its RTT measurements
3. Flow 2 schedules to enterslow down 10 secs later (PROBE_RTT mode)
4. Flow 1 and Flow 2 gradually converge to share BW fairly

BBR: fully use bandwidth, despite high packet loss



BBR vs CUBIC: synthetic bulk TCP test with 1 flow, bottleneck_bw 100Mbps, RTT 100ms

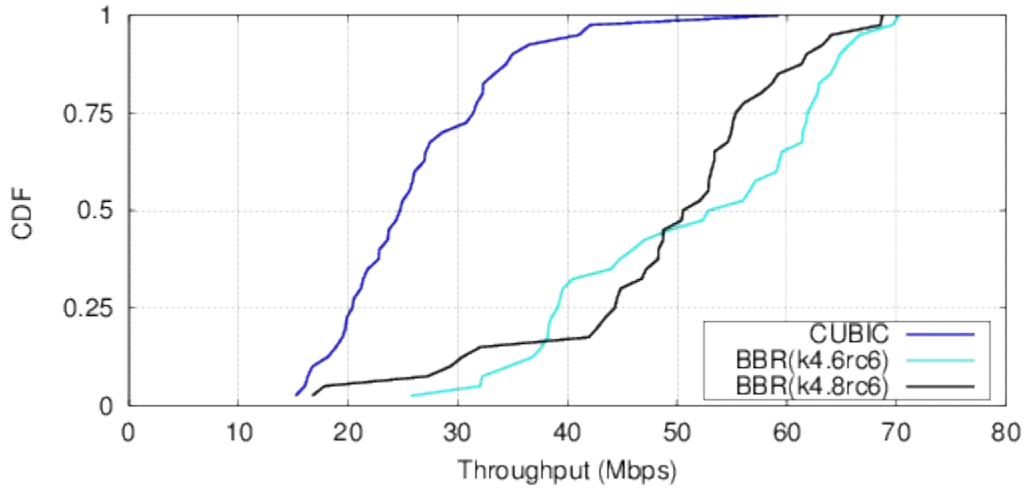
BBR: low queue delay, despite bloated buffers



BBR vs CUBIC: synthetic bulk TCP test with 8 flows, bottleneck_bw=128kbps, RTT=40ms

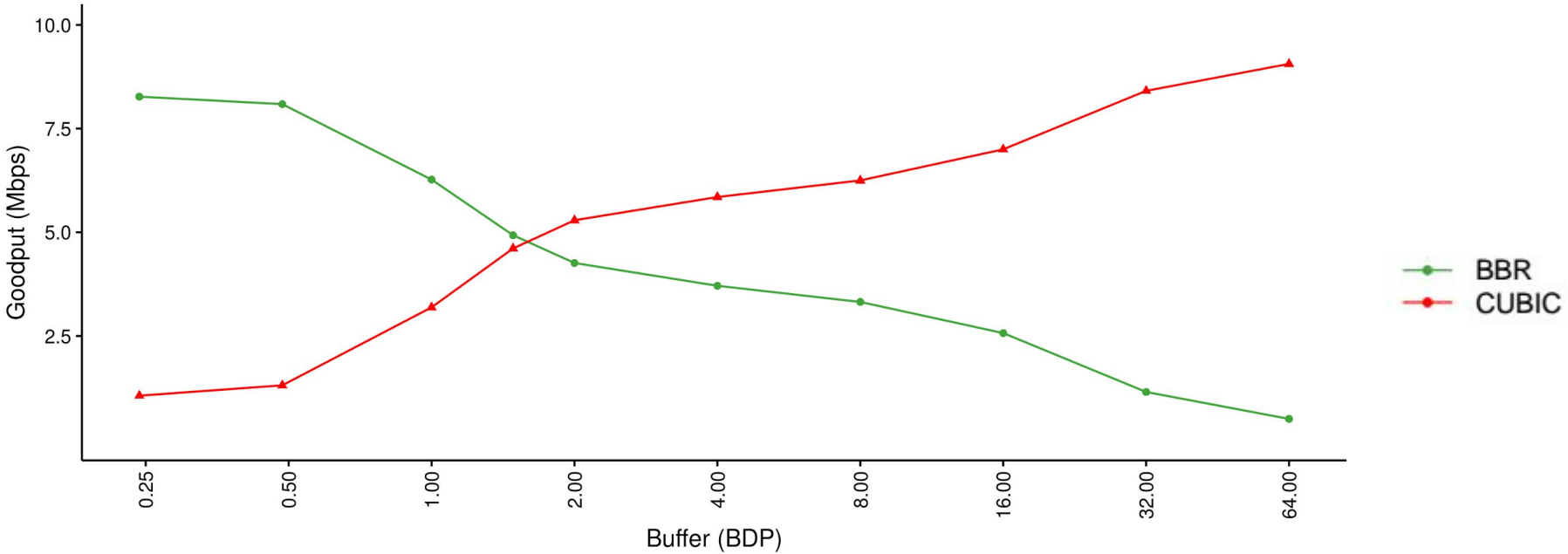
BBR: robust detection of full pipes > faster start up

- **BBR** STARTUP: estimate reached full BW if **BW** stops increasing significantly
- **CUBIC** Hystart: estimate reached full BW if **RTT** increases significantly
- But delay (RTT) can increase significantly well before full BW is reached!
  - Shared media links (cellular, wifi, cable modem) use slotting, aggregation
- e.g.: 20 MByte transfers over LTE (source: post by Fung Lee on bbr dev list, 2016/9/22):

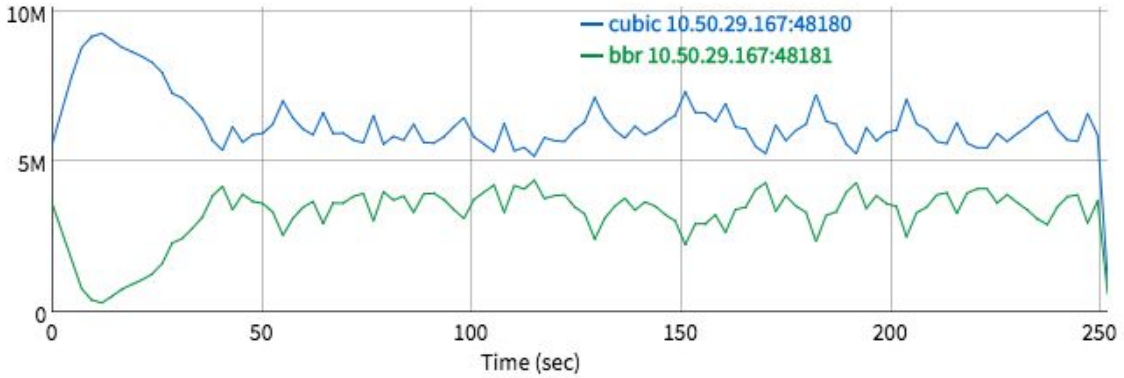


Improving dynamics w/ with loss based CC

1xCUBIC v 1xBBR goodput: bw=10Mbps, RTT=40ms, 4min transfer, varying buffer sizes



BBR and loss based CC in deep buffers: an example



At first CUBIC/Reno gains an advantage by filling deep buffers

But BBR does not collapse; it adapts: BBR's bw and RTT probing tends to drive system toward fairness

Deep buffer data point: 8*BDP case: bw = 10Mbps, RTT = 40ms, buffer = 8 * BDP

> CUBIC: 6.31 Mbps vs BBR: 3.26 Mbps

Improving BBR

BBR can be even better:

- Smaller queues: lower delays, less loss, more fair with Reno/CUBIC
  - Potential: cut RTT and loss rate in half for bulk flows
- Higher throughput with wifi/cellular/DOCSIS
  - Potential: 10 20% higher throughput for some paths
- Lower tail latency by adapting magnitude of PROBE_RTT
  - Potential: usually PROBE_RTT with cwnd = 0.75*BDP instead of cwnd=4

End goal: improve BBR to enable it to be the default congestion control for the Internet

We have some ideas for tackling these challenges

We also encourage the research community to dive in and improve BBR!

Following are some open research areas, places where BBR can be improved...