BBR TCP Opportunities

Matt Mathis

based on work by

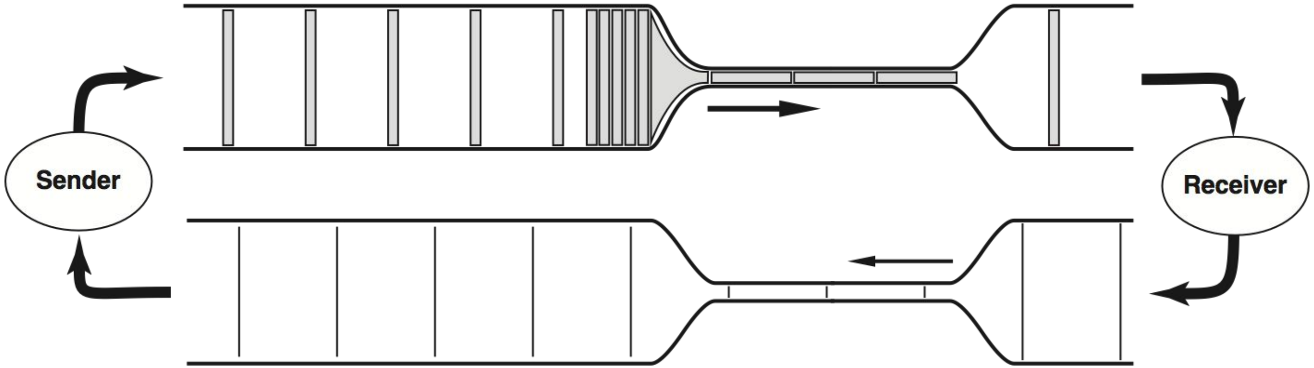
Neal Cardwell, Yuchung Cheng, C. Stephen Gunn,
Soheil Hassas Yeganeh, Van Jacobson

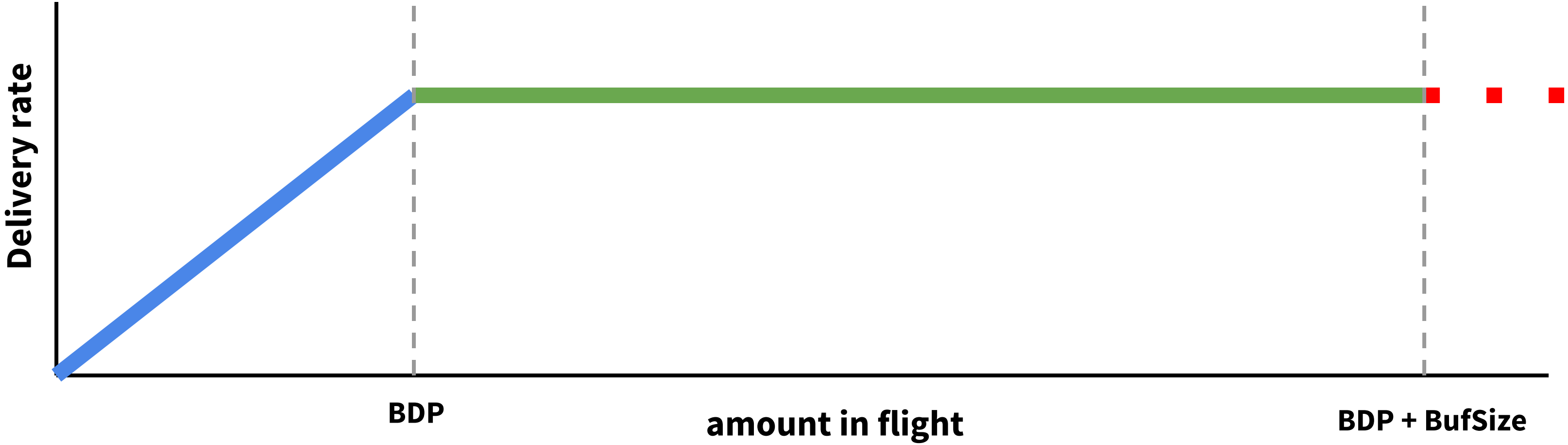
Bottleneck Bandwidth and RTT

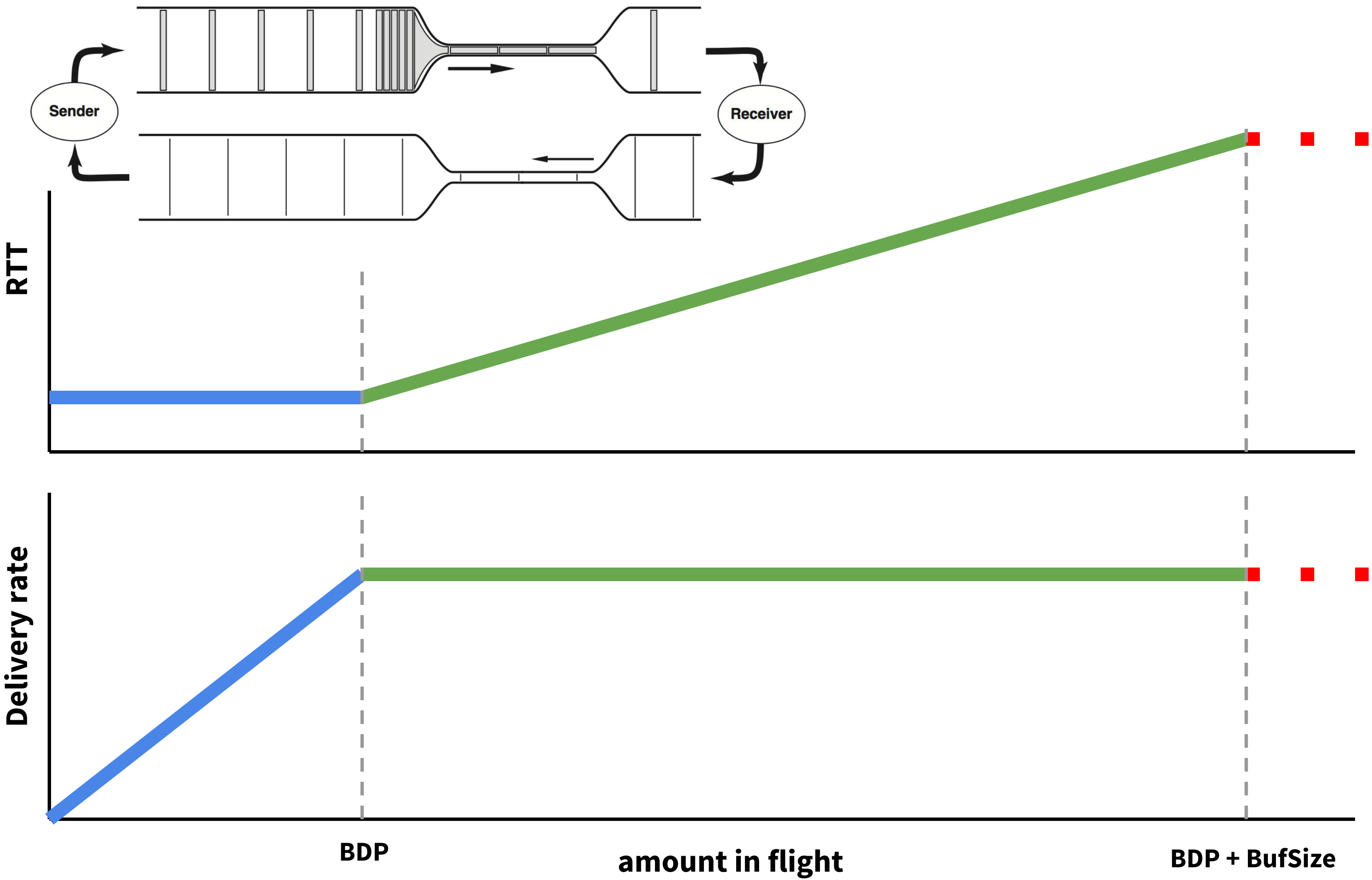
- New TCP congestion control framework
- Awesome results
- Reverts much prior work
    - Jacobson 88
    - TCP Friendly Rate Control
    - Nearly 3 decades of research
- Many research opportunities

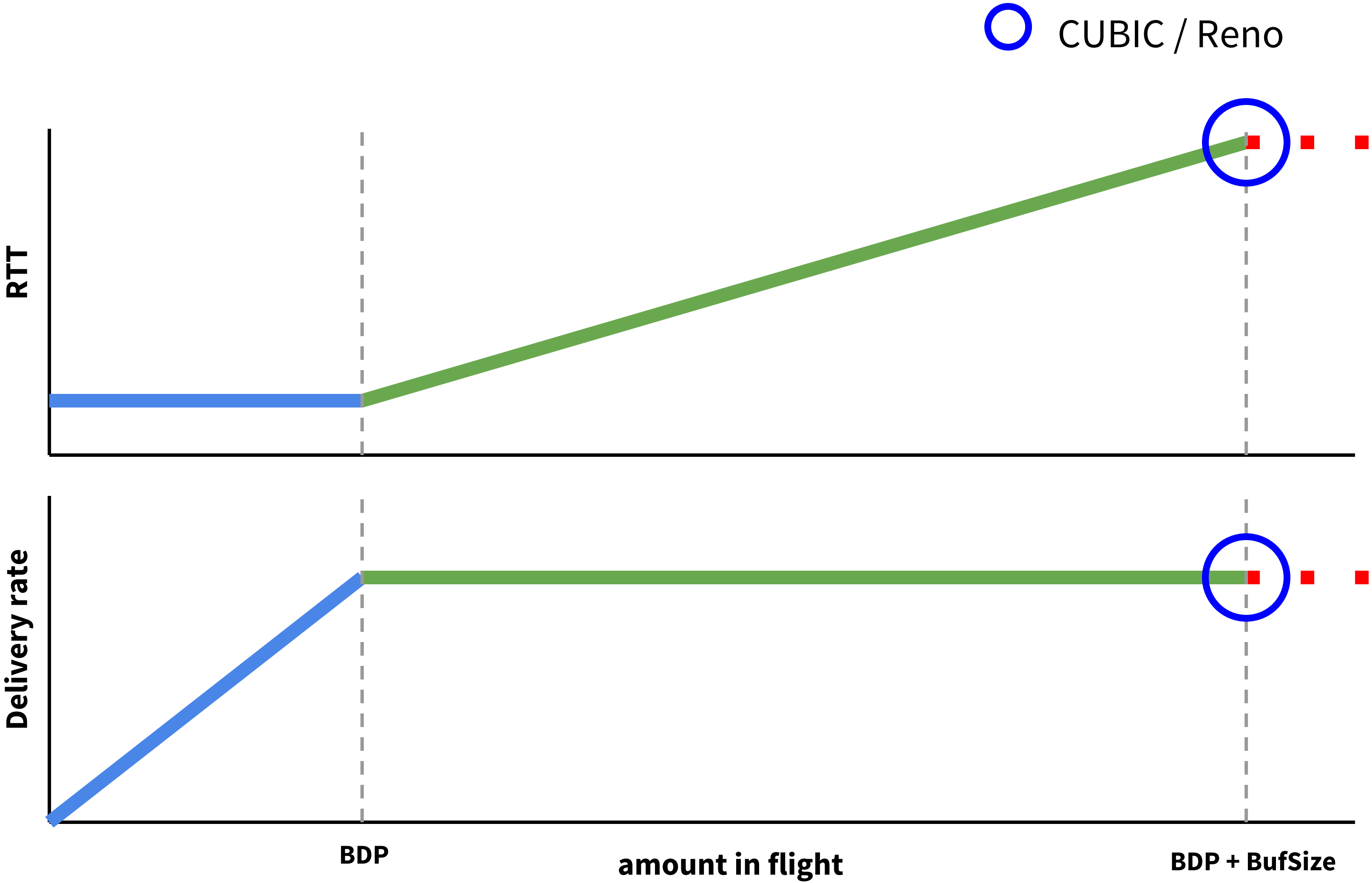
BBR TCP

Congestion and bottlenecks

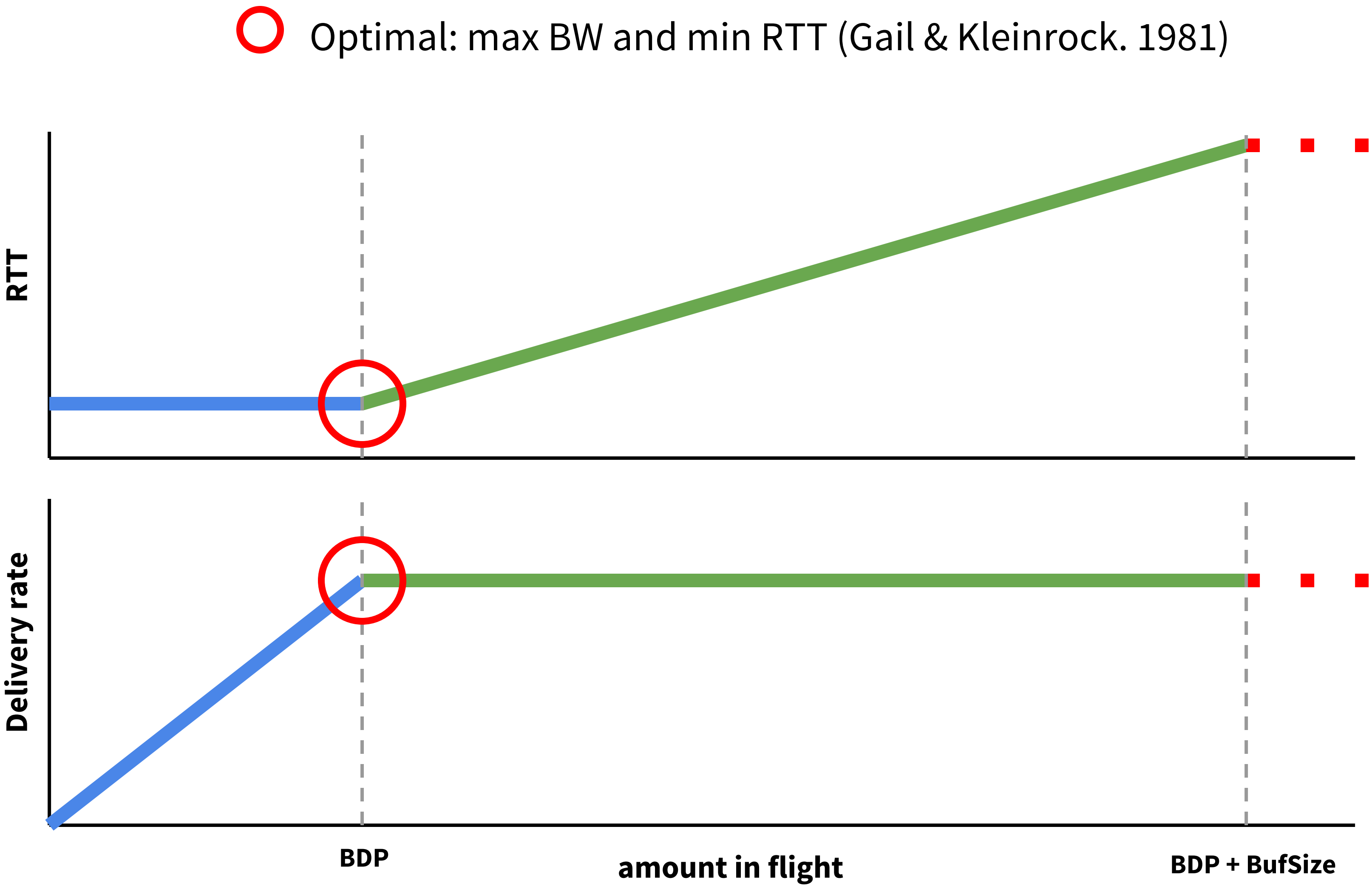




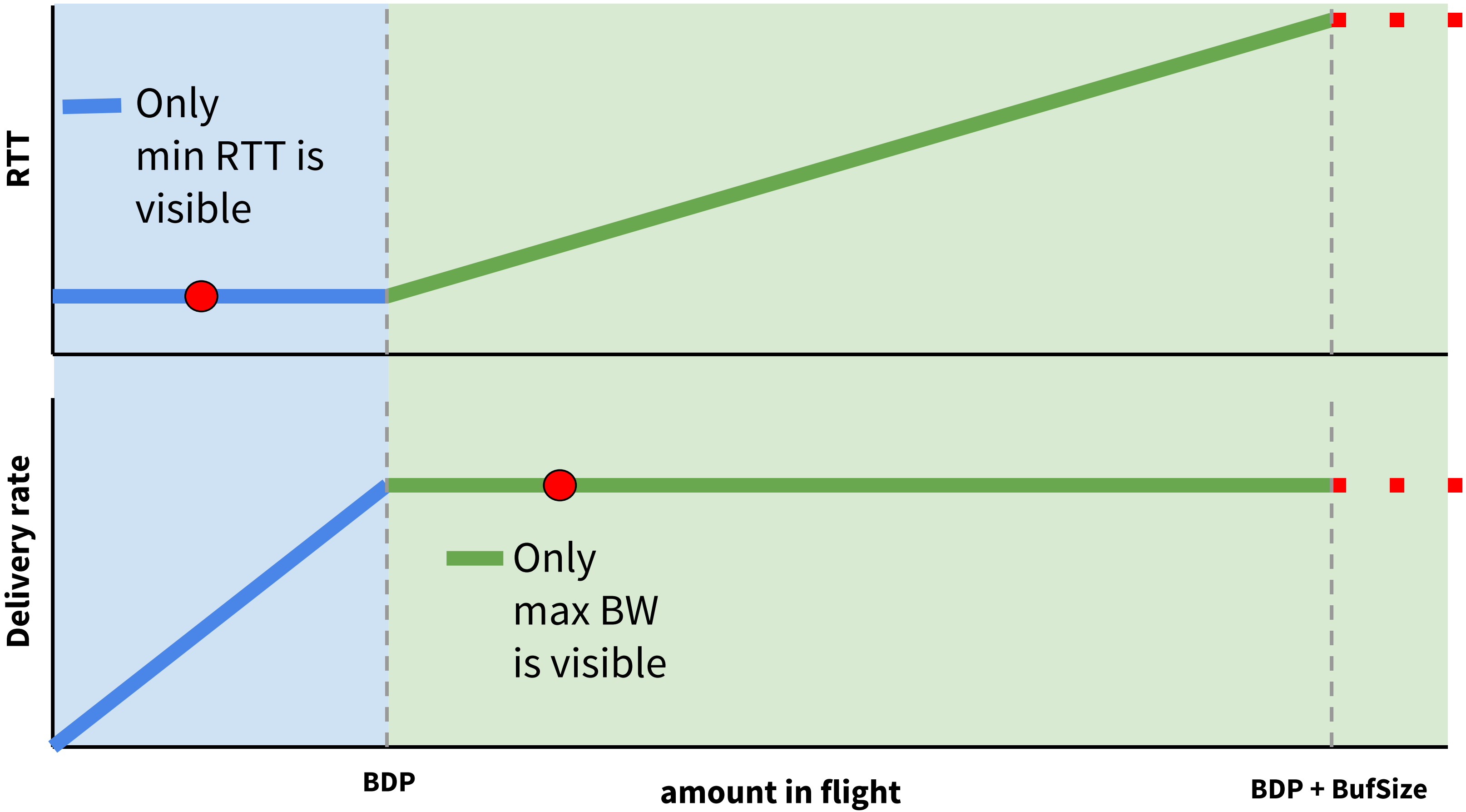

Sender
Receiver
RTT
Delivery rate
BDP
amount in flight
BDP + BufSize


CUBIC / Reno
RTT
Delivery rate
BDP
amount in flight
BDP + BufSize

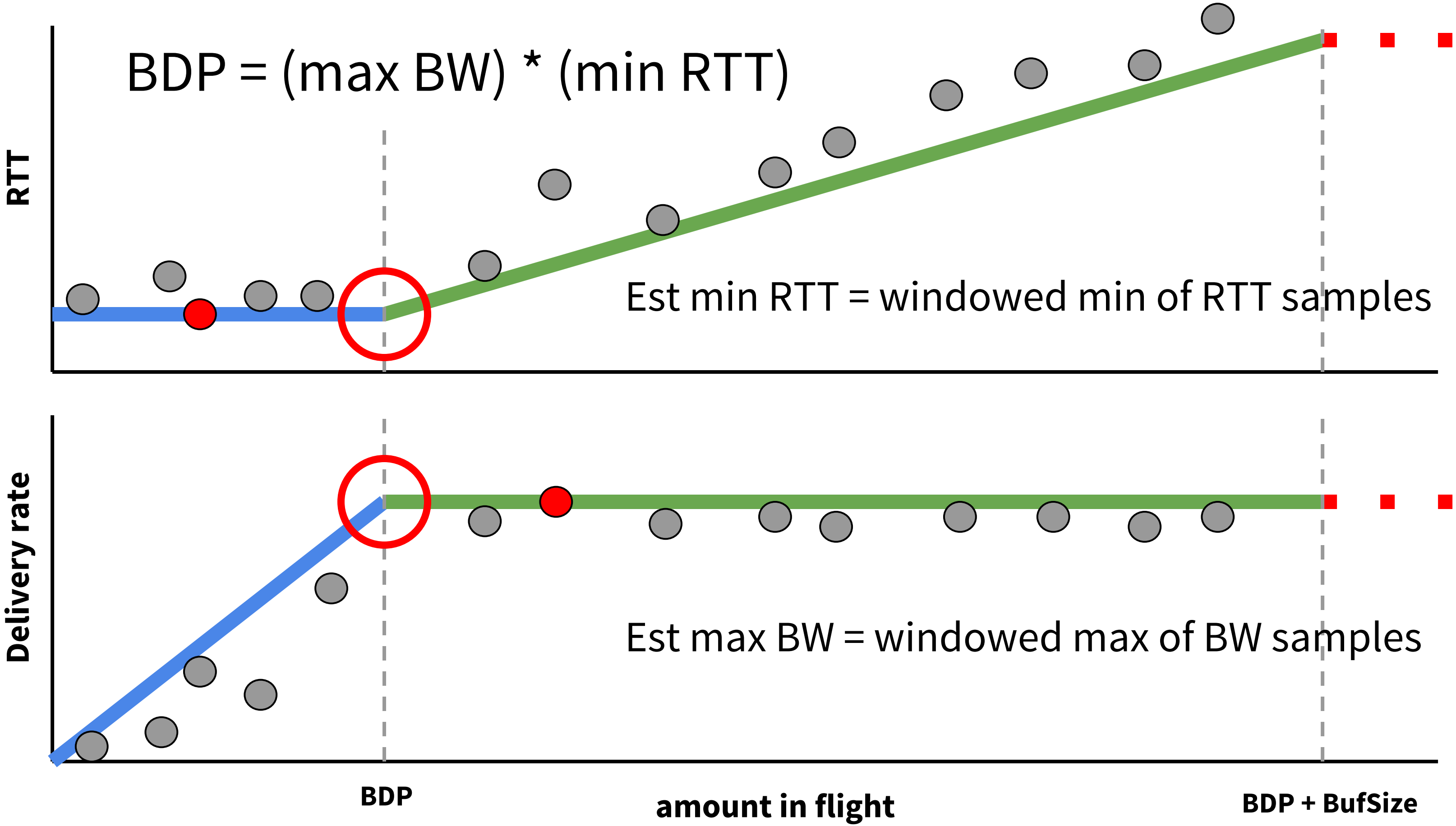
Optimal: max BW and min RTT (Gail & Kleinrock. 1981)



But to see both max BW and min RTT, must probe on both sides of BDP...



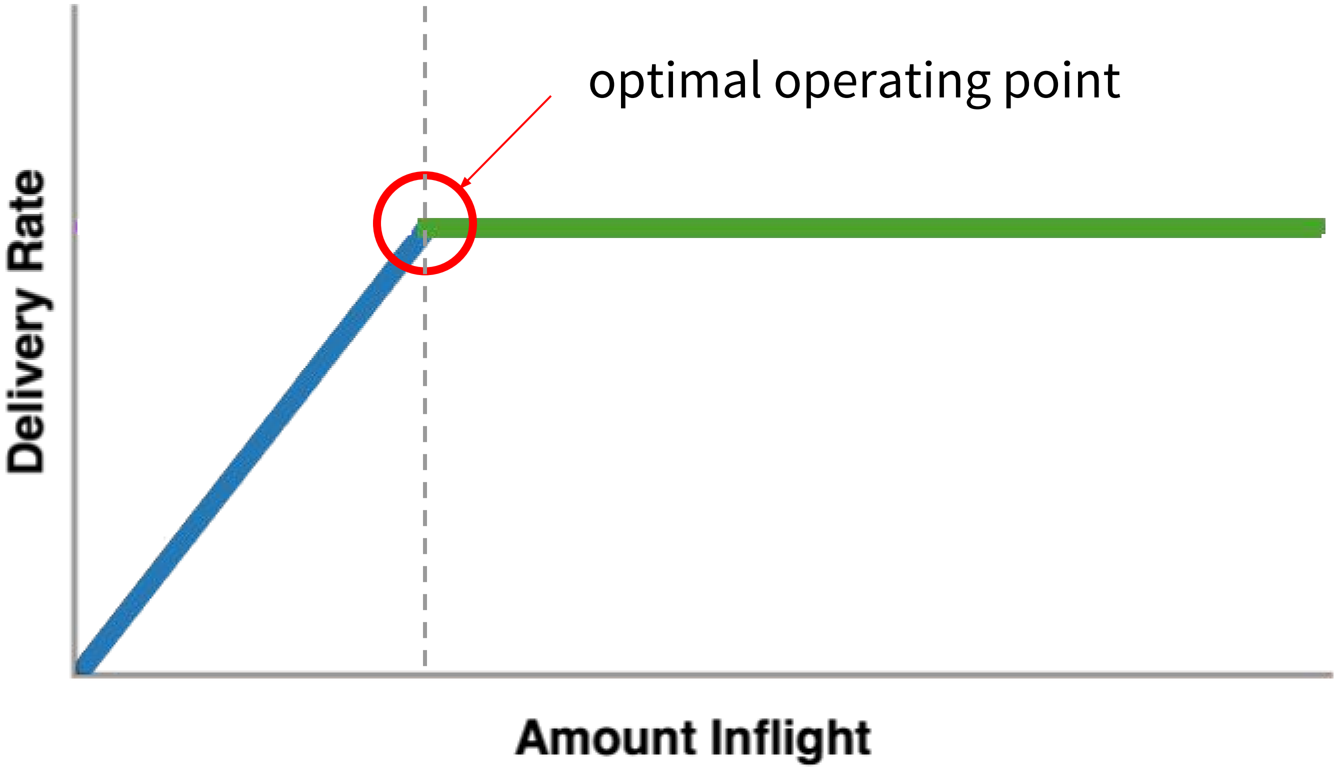
Estimating optimal point (max BW, min RTT)



BBR Summary

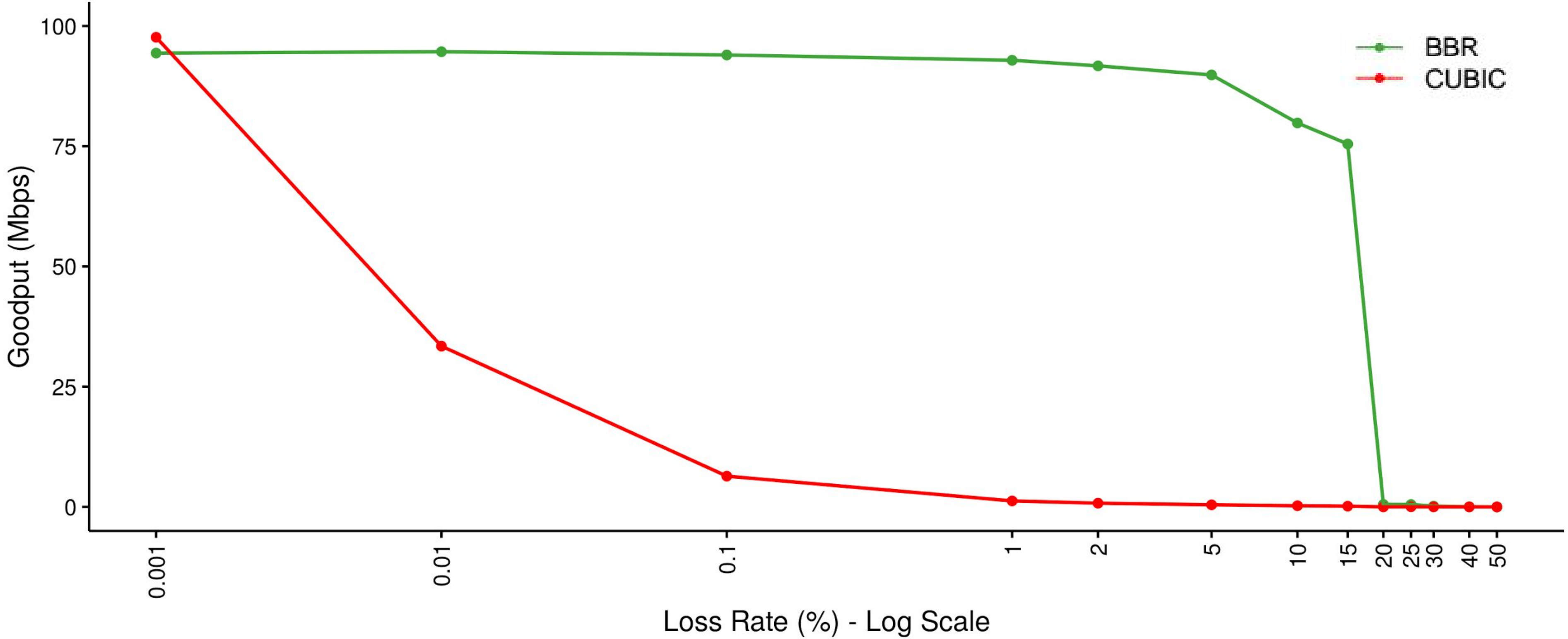
- Every ACK measures RTT and delivery rate
  - Update min_RTT (10 second sliding window)
  - Update max_rate (10 round-trip sliding window)
- Paced sending rate defaults to the maximum observed receiving rate
  - Secondary control limits cwnd to 2*min_RTT*max_rate
- Mostly send at the the default rate
  - Periodically dither rate up to assure that the max_rate is valid
  - Periodically dither down to assure that min_RTT is valid

Dither sending rate to find max BW, min RTT



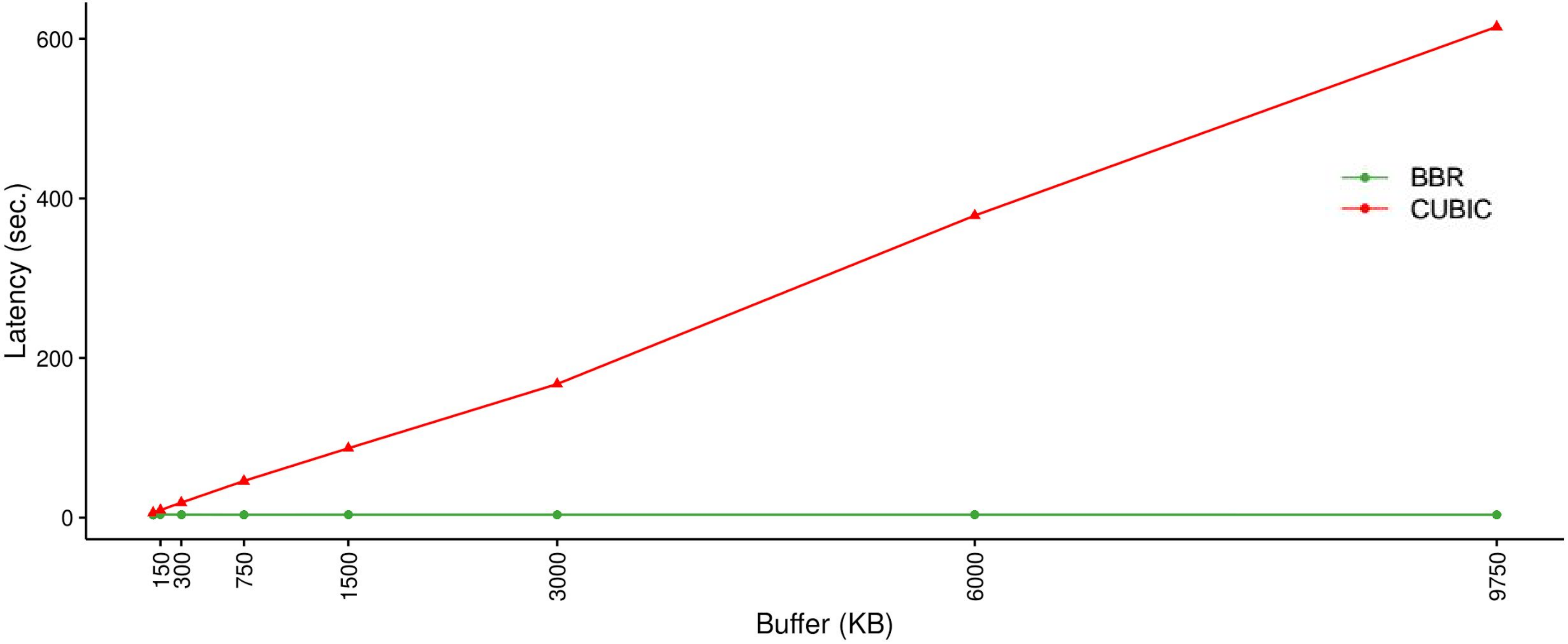
Performance results...

Fully use bandwidth, despite high loss



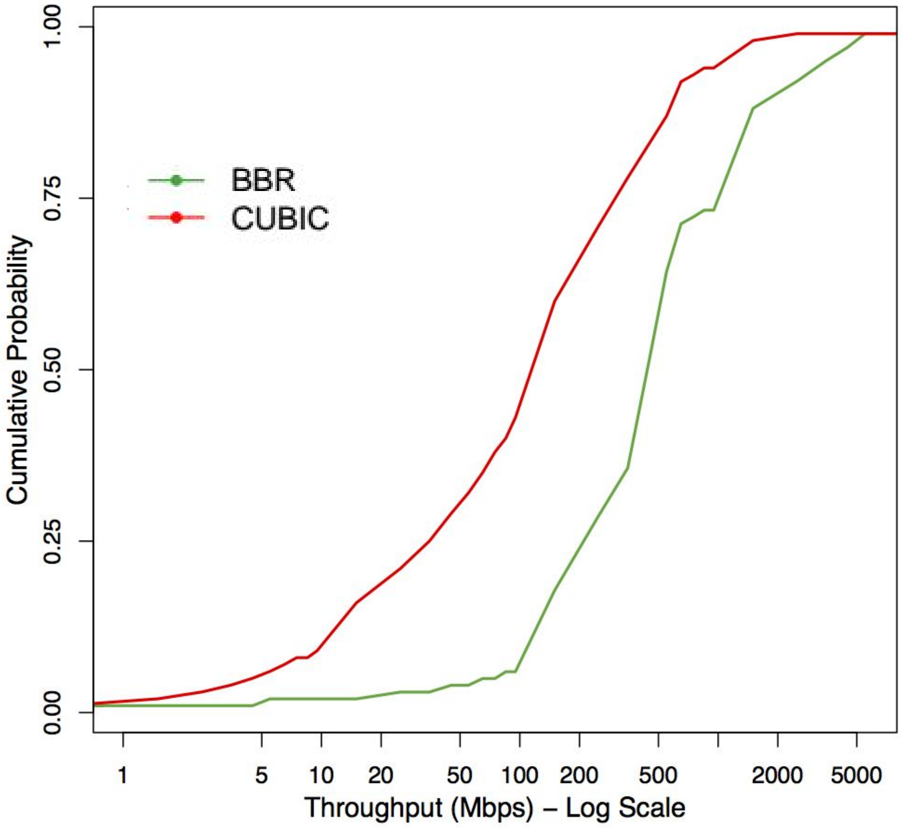
BBR vs CUBIC: synthetic bulk TCP test with 1 flow, bottleneck_bw 100Mbps, RTT 100ms

Low queue delay, despite bloated buffers



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

BBR is 2-20x faster on Google WAN



- BBR used for all TCP on Google B4
- Most BBR flows so far rwin-limited
  - max RWIN here was 8MB (tcp_rmem[2])
  - 10 Gbps x 100ms = 125MB BDP
- after lifting rwin limit:
  - BBR 133x faster than CUBIC

Break from the Past…

Reverted Jacobson 88 principles

- Self clock (packet conservation)
  - Data transmission timing was determined by returning ACKs
  - BBR is paced (timer driven)
- Window controlled (packet conservation)
  - cwnd bounds the data in flight
  - cwnd (mostly) changes slowly
  - BBR uses the window only as a secondary control
- Congestion was "signaled" by losses
  - Losses generally caused by queue overflow
  - Losses cause multiplicative cwnd reductions
  - BBR uses the observed delivery rate directly

Other reverted assumptions

- These are derived from or implied by Jacobson 88
- Reno Macroscopic Model [Mathis 97]
  - performance goes as the square root of loss rate
  - Required background loss rate goes as the square of the data rate
  - ACM "Tests of Time" in 2009
- TCP Friendly Rate control
  - Require all protocols to have TCP like response to loss (and congestion)
  - "TCP Friendly" became dogma of the IETF
  - In today's Internet TFRC is suffering from scaling limitations

New Property: lossless short queues

- Mental model for single flows
  - Estimate when the queue will become empty and send more data just in time
  - Average queuing delays on the order of a few milliseconds
- Multiple BBR flows cause queues with fixed upper bounds
  - Queues are needed signal shared capacity
  - Lossless if the queue buffer space is large enough

New Property: less sensitive to RTT

- BBR can fill Long Fast Networks, even at very long RTTs
  - Single stream 9+ Gb/s on continental scale paths
- Reduced incentive to prefer near bulk data
  - (Near transactions are always better)
- May change the economics of CDNs vs serving via transit
  - Very hard to see how this might play out

New Property: multiple flows hurt perf

- Single stream BBR generally outperforms multiple parallel BBR streams
  - Single stream can fill bottlenecks at phenomenal scales
- Some of this effect is likely due to implementation issues
  - Will get better as BBR matures
- My intuition: some of this effect is likely to be fundamental
  - The single stream case is easy to model, understand and has a well understood perf bound
    - 2 state vars in sender match 2 path properties, while queue length converges to near zero
    - Performance bound is due to required link idle to assure that min_RTT is accurate
  - Multi-stream is much more complicated
    - "Fairness" partially signalled through delay at a standing queue
    - Performance bound can't be higher than single stream case
      - Link idle is still **required** to assure that min_RTT is accurate

New Property: Less sensitive to loss

- Likely to change optimal load levels
- Maintain high performance even with loss
  - Can choose to deliberately under buffer (cheaper switches)
  - No adverse performance impact from statistical loss at short queues
- Sharing with other TCPs can be an issue

New property: interleaved traffic

- (Also applies to CUBIC with fq_pacing)
- Traffic burst size primarily determined by TSO tuning
  - Loosely one burst per mS
- Better traffic mixing
  - "Packet trains" get diced and interleaved between flows
- No opportunities for phase effects (no traffic self synchronization)
- Reduces required queue space (traffic is smoother)

Research Opportunities…

Research needed: more eyeballs on BBR

- BBR rate and cwnd selection are really heuristics
  - We have only begun
  - How can it be improved?
- CUBIC reflects 20+ year of Reno evolution
  - Even so TCP implementations continue to evolve
  - BBR will cause a major reset of our knowledge and experience base
- Fairness under wide ranges of conditions
  - Know advantage over CUBIC with undersized network queue space
  - Know disadvantage behind CUBIC with oversized network queue space

Join the fun: bbr-dev on googlegroups

Collateral Research Questions:

- How might changing baseline queue occupancy change:
  - QoS semantics
    - Less backlog means QoS marking has less effect on the traffic
    - Some applications only need QoS to overcome queue maximizing protocols
      - BBR might reduce the need for QoS
  - ECN semantics
    - What does ECN mean to a protocol that is not driven by drops?
  - How should BBR and AQM interact?
  - LEDBAT, TCP-LP and other delay based less than best effort algorithms
    - Depended on TCP over filling queues

Scheduling vs AQM (WiFi, etc)

- In a pre-BBR world, AQM is paramount, link scheduling is secondary
- In a post BBR world, link scheduling is probably more important than AQM

Adaptive link encoding

- LTE encoding is optimized on the basis of backlog
  - Built in assumption that Reno/CUBIC will create large queues
  - Early version of BBR did not present enough backlog to trigger encoding changes
    - Had to exaggerate the rate probing to cause enough backlog
    - i.e. BBR is tuned to be CUBIC like in this environment
- Could we do better co-engineering LTE and BBR?
  - What is the ideal backlog under various conditions and why?

  (My favorite question!)

General conclusion about research

- BBR changes core assumptions about TCP behavior
- Nearly 3 decades of research has these assumptions (implicitly) baked in
  - What opportunities might we have overlooked?
  - What good ideas don't work in the presence of queue maximizing protocols
    - May have been falsely discarded

General claim: BBR might change many past research results

BBR: status

Fully deployed for all TCP on Google WAN backbones (2x - 100x higher bw than CUBIC)

Replacing CUBIC with BBR on google.com and YouTube

Gradually-expanding global-scale experiment: lower latency, better QoE results

Upstream now in Linux v4.9 (**patch**)

ACM Queue paper with more details: **"BBR: Congestion-based Congestion Control"**

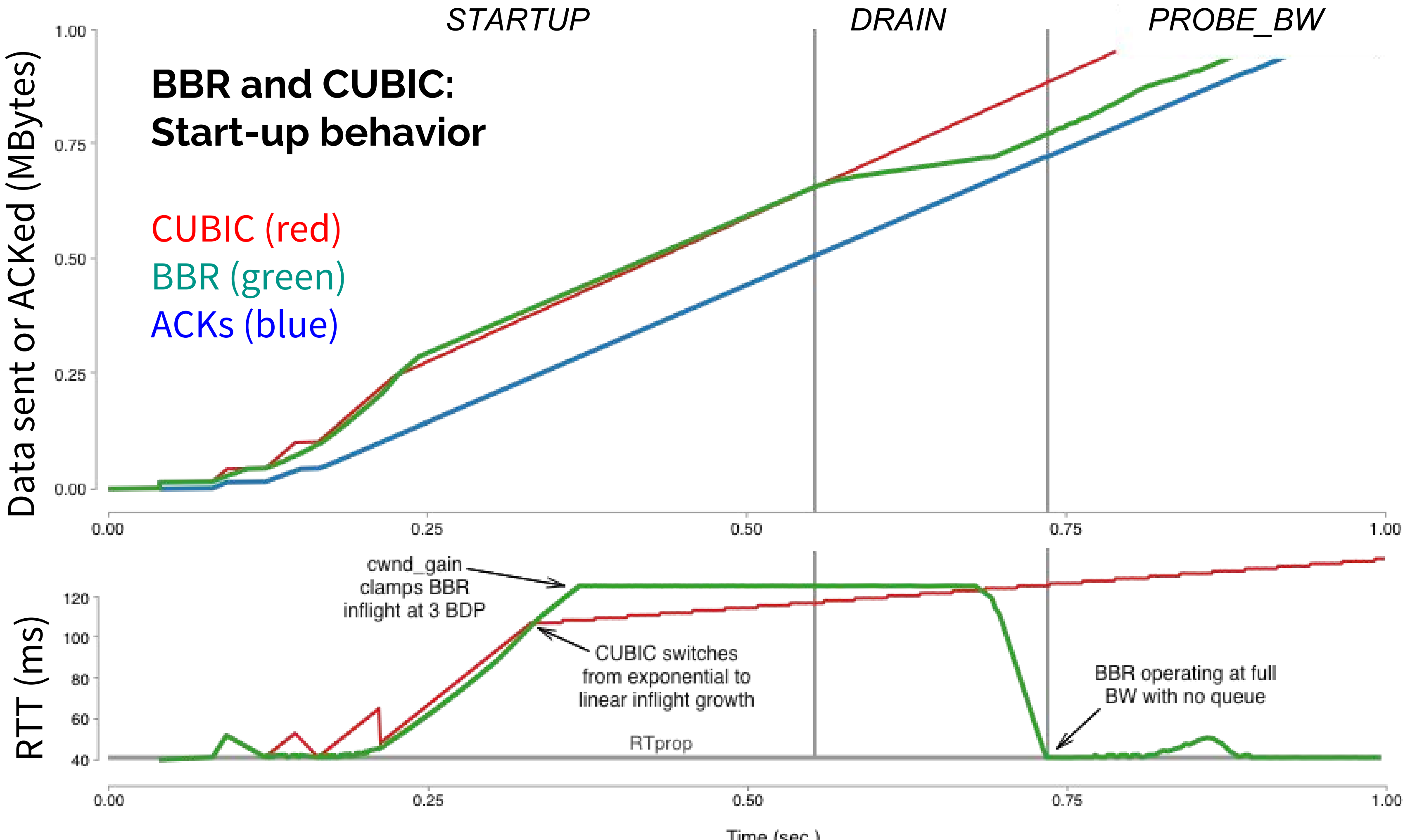
Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, Van Jacobson

Ongoing development: join the **public bbr-dev mailing list** to share and discuss

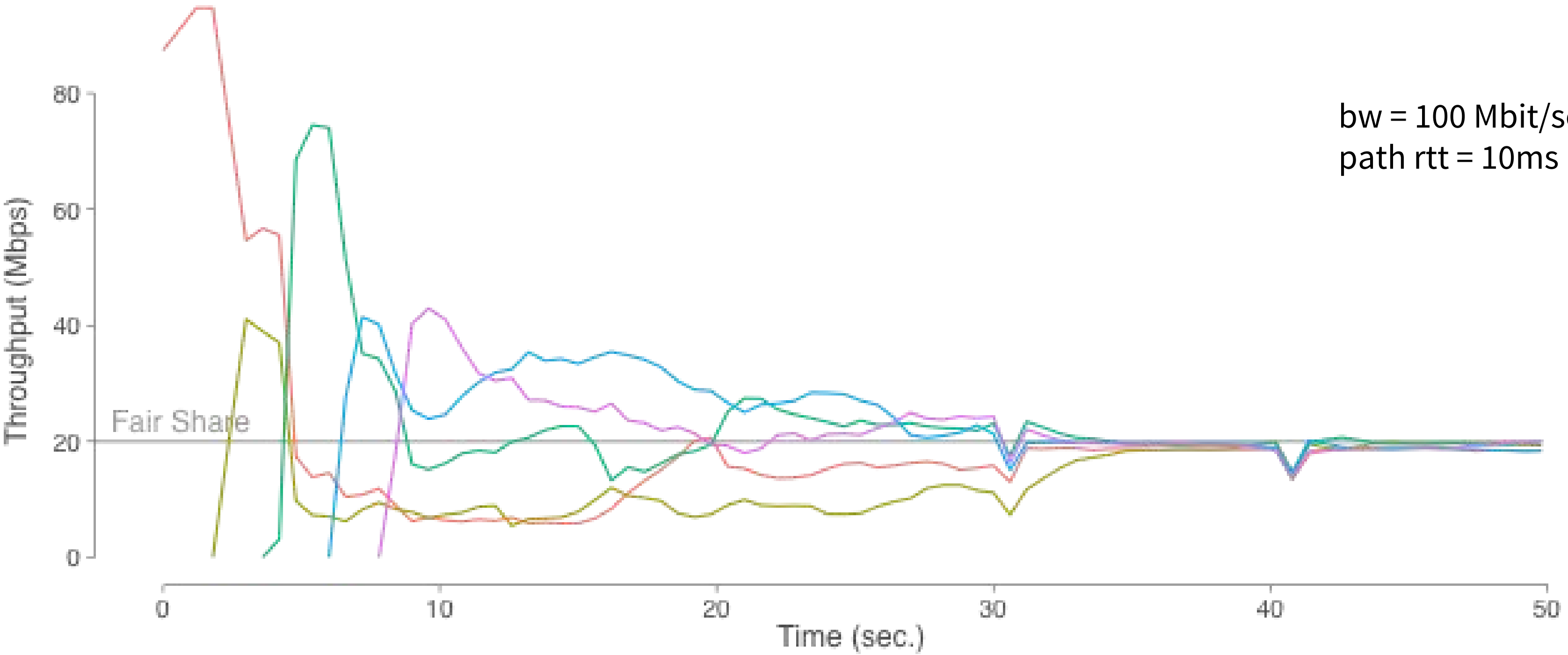
Backup slides...

BBR and CUBIC: Start-up behavior

CUBIC (red)
BBR (green)
ACKs (blue)



BBR convergence dynamics



bw = 100 Mbit/sec
path rtt = 10ms

Converge by sync'd PROBE_RTT + randomized cycling phases in PROBE_BW

- Queue (RTT) reduction is observed by every (active) flow
- Elephants yield more (multiplicative decrease) to let mice grow