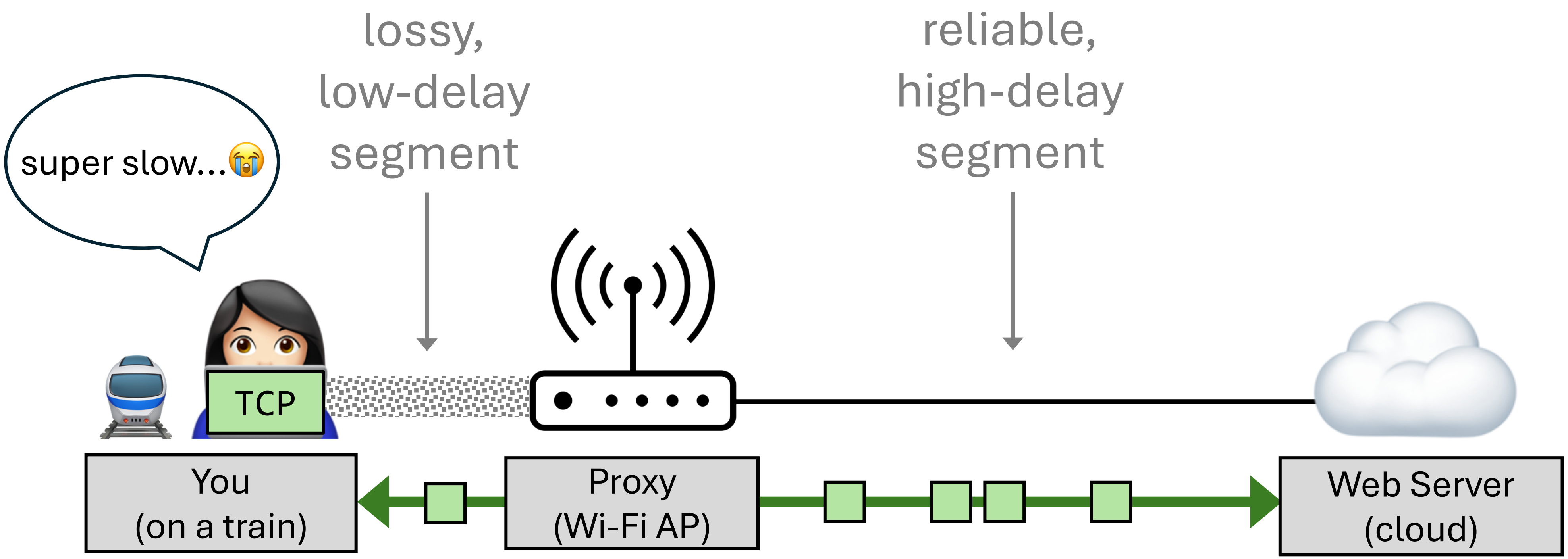
Internet Connection Splitting: What's Old is New Again

Gina Yuan, Thea Rossman, Keith Winstein

Stanford University

July 8, 2025, USENIX ATC, Boston, MA

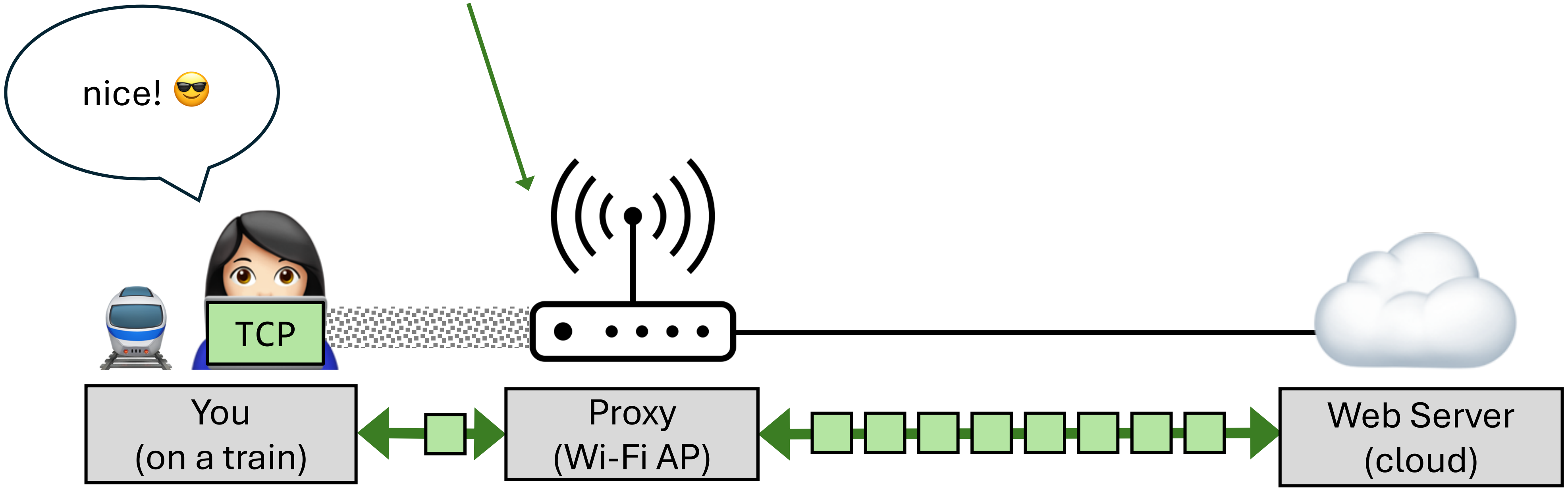
Internet connection-splitting – E2E connection



- Traditional "loss-based" congestion-control algorithms (e.g. CUBIC) treat wireless loss as network congestion in an end-to-end setting

Internet connection-splitting – split connection

connection-splitting TCP PEP
(performance-enhancing proxy)



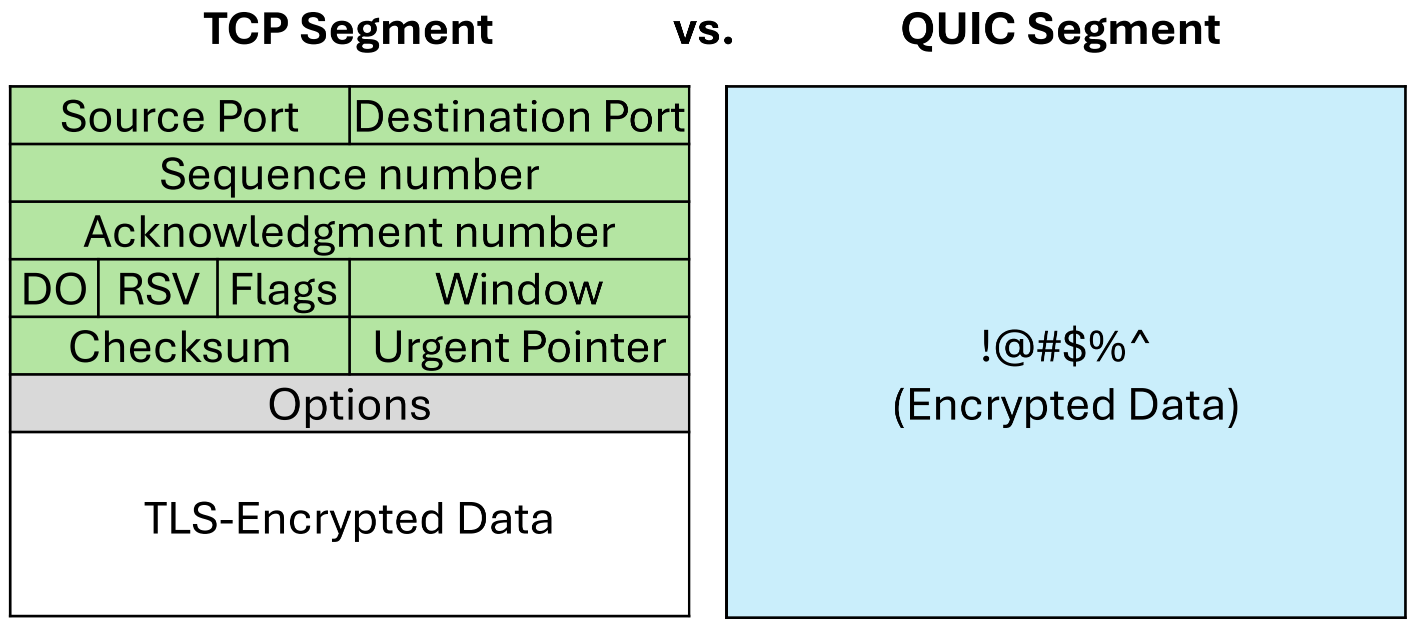
- Customized congestion control (“split CUBIC”) on each path segment
- Widely deployed: 20-40% of paths, especially satellite and wireless[1,2]

[1] Honda et. al. Is it Still Possible to Extend TCP? *IMC*, 2011.
[2] Edeline & Donnet. A Bottom-Up Investigation of the Transport-Layer Ossification. *TMA*, 2019.

A lot has changed since these PEPs were first deployed in the 1990s…

QUIC made connection-splitting impossible.

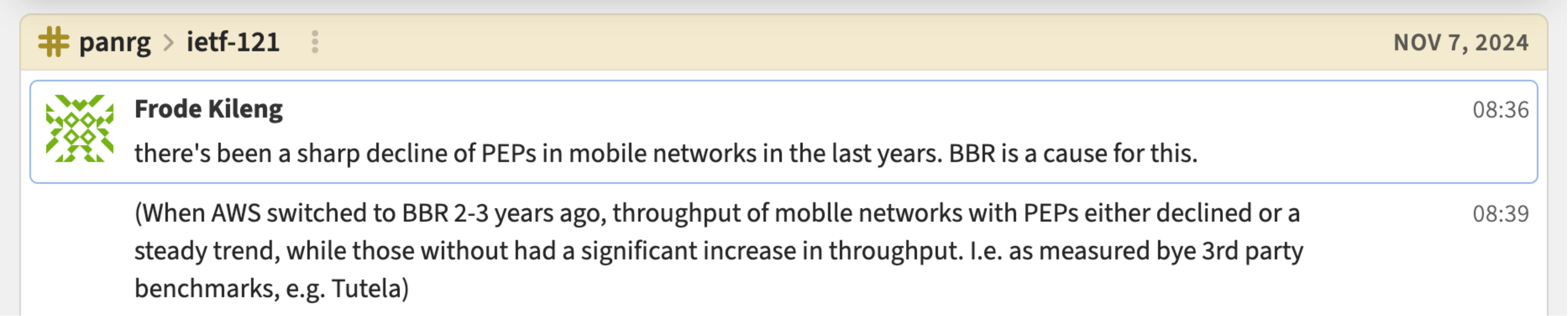
↳ Encrypted transport protocol designed to allow the protocol to evolve over time and to improve performance for HTTPS traffic.



[3] Langley et. al. The QUIC Transport Protocol: Design and Internet-Scale Deployment. *SIGCOMM*, 2017.

BBR made connection-splitting unnecessary.

↳ Congestion-control algorithm that de-emphasizes loss as a congestion signal.


panrg > ietf-121 NOV 7, 2024

Frode Kileng 08:36

there's been a sharp decline of PEPs in mobile networks in the last years. BBR is a cause for this.

(When AWS switched to BBR 2-3 years ago, throughput of moblle networks with PEPs either declined or a steady trend, while those without had a significant increase in throughput. I.e. as measured bye 3rd party benchmarks, e.g. Tutela) 08:39

[4] Cardwell, Cheng, Gunn, Yeganeh, & Van Jacobson. BBR: Congestion-Based Congestion Control. *Communications of the ACM*, 2017.

Is it true? Did BBR and QUIC make connection-splitting obsolete?

Re: Is splitting obsolete? It's complicated!

Finding 1: TCP BBR has benefitted more from splitting over time.

Finding 2: TCP BBRv3 even benefits in some classes of network paths where TCP CUBIC does not.

Finding 3: QUIC implementations of the "same" congestion-control schemes vary significantly.

Measurement Study

- **Plan:** Evaluate various congestion-control schemes in a variety of network settings, both *with* and *without* a connection-splitting PEP.
- **Question:** Does the PEP improve performance with BBR/QUIC?

implementation/protocol/algorithm/version

Linux TCP

×

picoquic

CLOUDFLARE quiche

Google QUICHE

CUBIC,2006

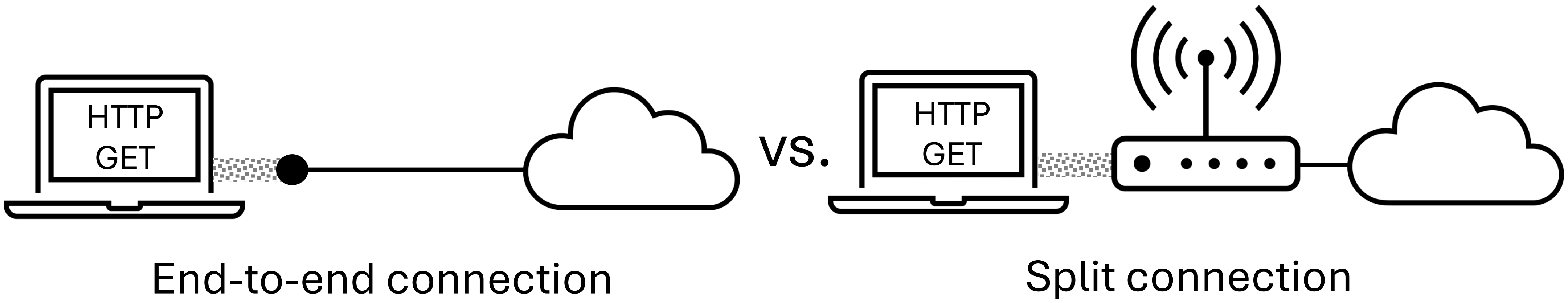
BBRv1,2016

BBRv2,2019

BBRv3,2023

Measurement Methodology

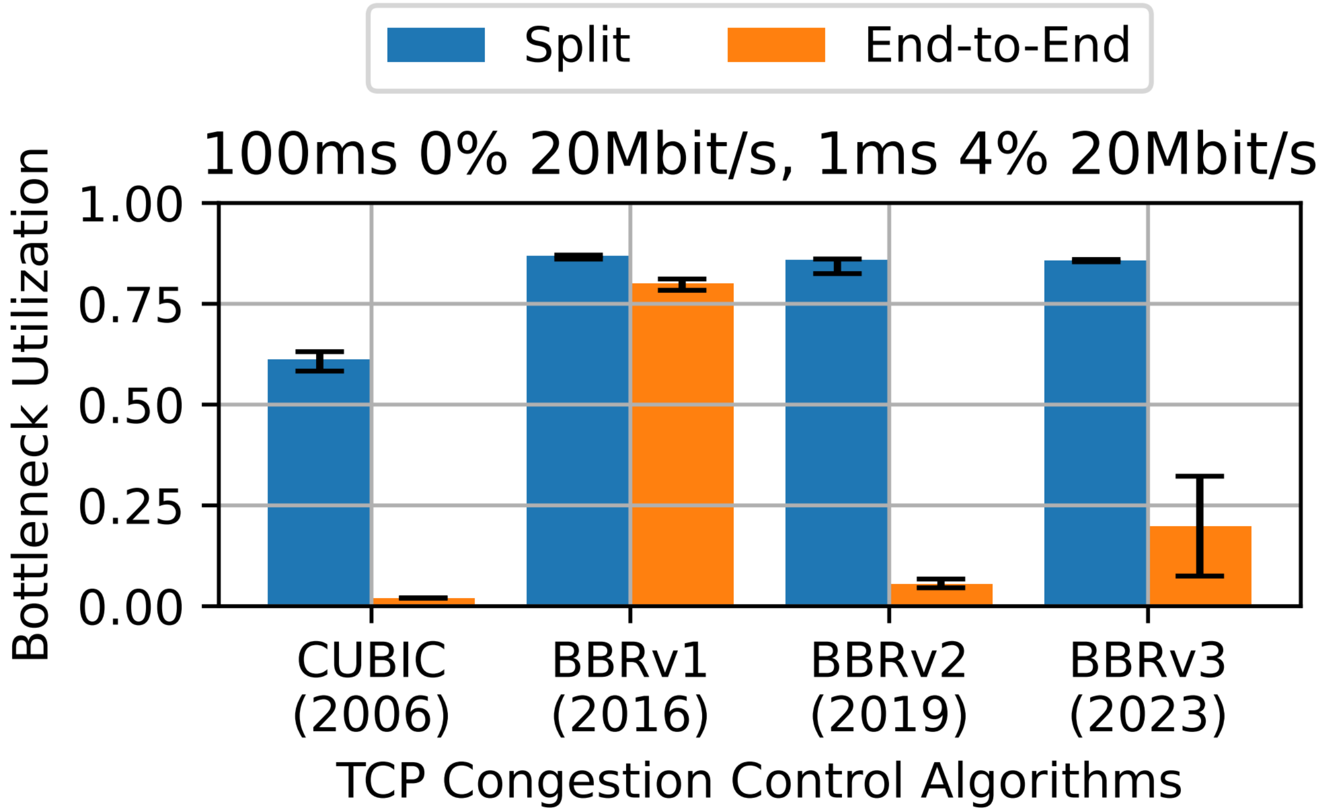
1. Pick a congestion control scheme.
2. Emulate two-segment topology: *bandwidth*, *loss*, *delay*.
3. Compare *end-to-end* vs. *split* throughput of HTTPS download.



Limitations

- 100% emulation study using `tc-netem` (not real-world)
- Metric is long-lived, full-throttle throughput only (not latency)
- Single-flow CCA environment (has fairness implications)

Finding 1: TCP BBR has benefitted more from splitting over time.

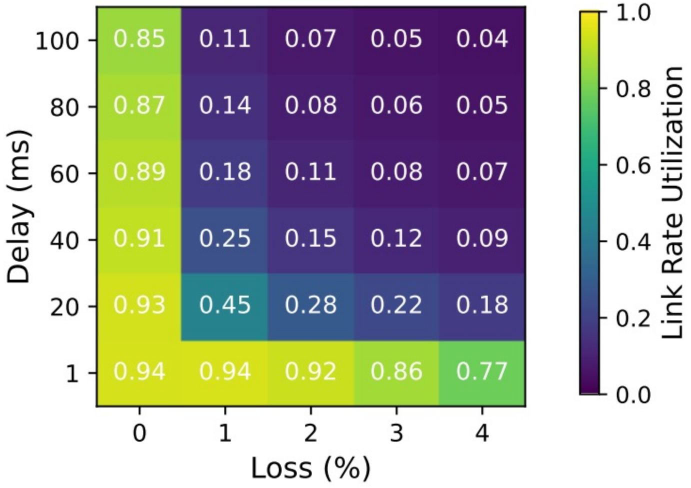


Why? BBRv2+ responded to concerns about TCP friendliness in BBRv1.

Does this generalize to other network settings?

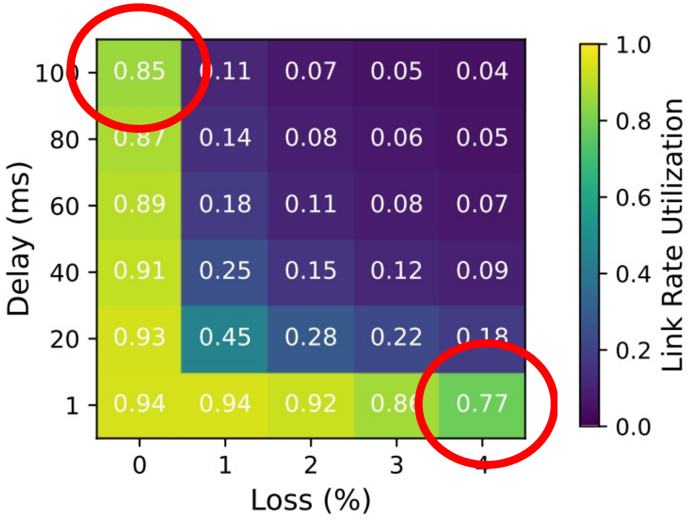
Split Throughput Heuristic: Estimate *split throughput* as the minimum of the measured *end-to-end throughputs* on each segment of the split path.

Cache end-to-end measurements in a heatmap



TCP CUBIC (2006)

Apply the heuristic to estimate *split throughput*

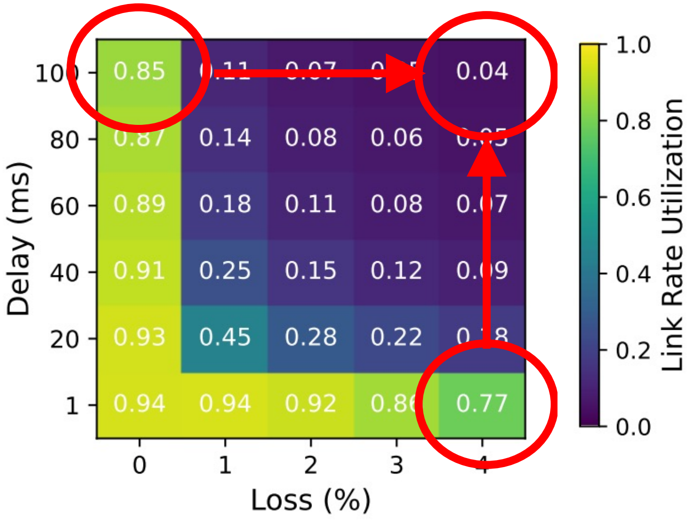


TCP CUBIC (2006)

Split Utilization

min(0.77, 0.85) = 0.77

Estimate *end-to-end throughput* in a split path



TCP CUBIC (2006)

Split Utilization

min(0.77, 0.85) = 0.77

End-to-End Utilization

bw = min(bw1, bw2) = min(10, 10) = 10

delay = delay1 + delay2 = 100 + 1 ≈ 100

loss ≈ loss1 + loss2 = 0 + 4 = 4

0.04

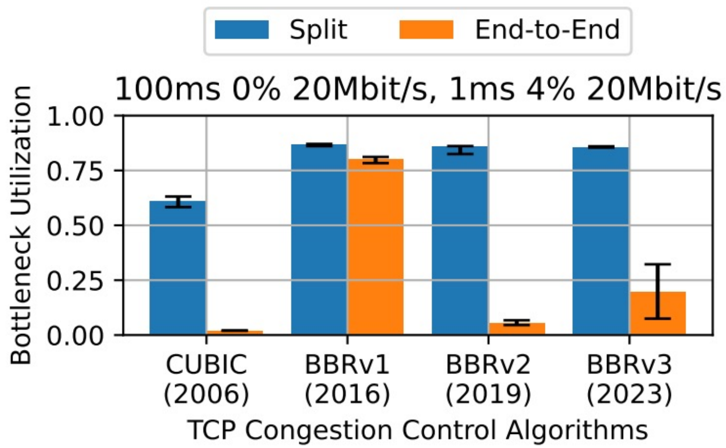
Finding 2: TCP BBRv3 even benefits in some classes of network paths where TCP CUBIC does not.

~~15 · 21 · 25 = 7875 split settings~~

5 · 6 · 5 = 150 end-to-end settings

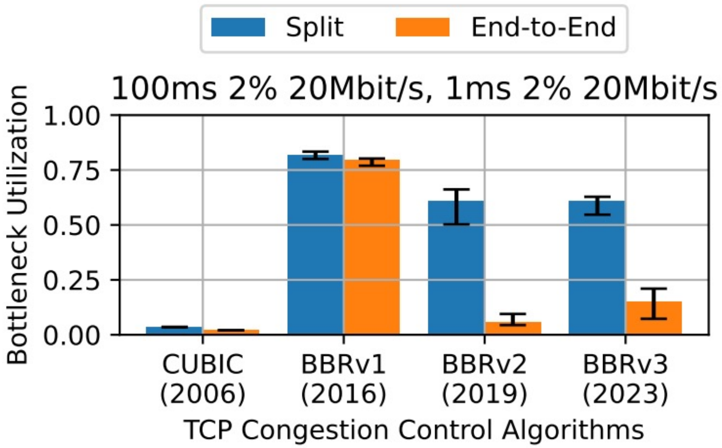
Filter	BBRv1	CUBIC	BBRv3
Initial	7875	7875	7875
Split imprvmnt. $> 3\times$	0	2231	234
Split utilization > 0.5	0	942	188
Asymmetric, last-mile	0	942	38
Asymmetric, lossy	0	0	72
Symmetric, lossy	0	0	78

Finding 2: TCP BBRv3 even benefits in some classes of network paths where TCP CUBIC does not.



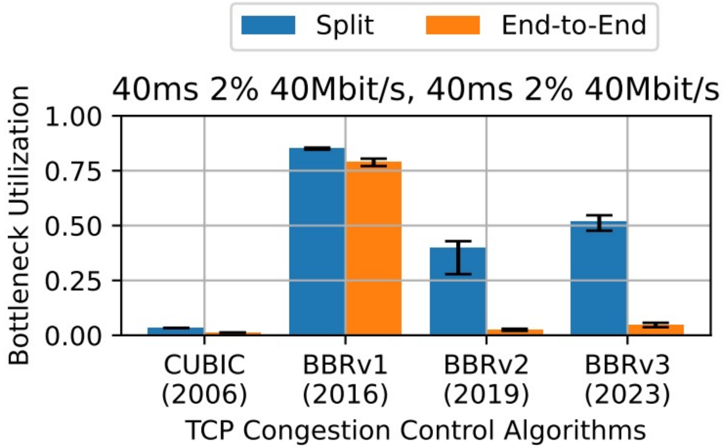
Class I. Asymmetric, last-mile

traditional PEP deployments with wireless link or rate policer



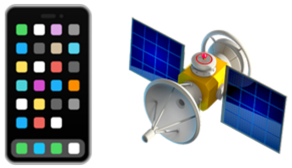
Class II. Asymmetric, lossy

low-resource networks, regions with no IXPs

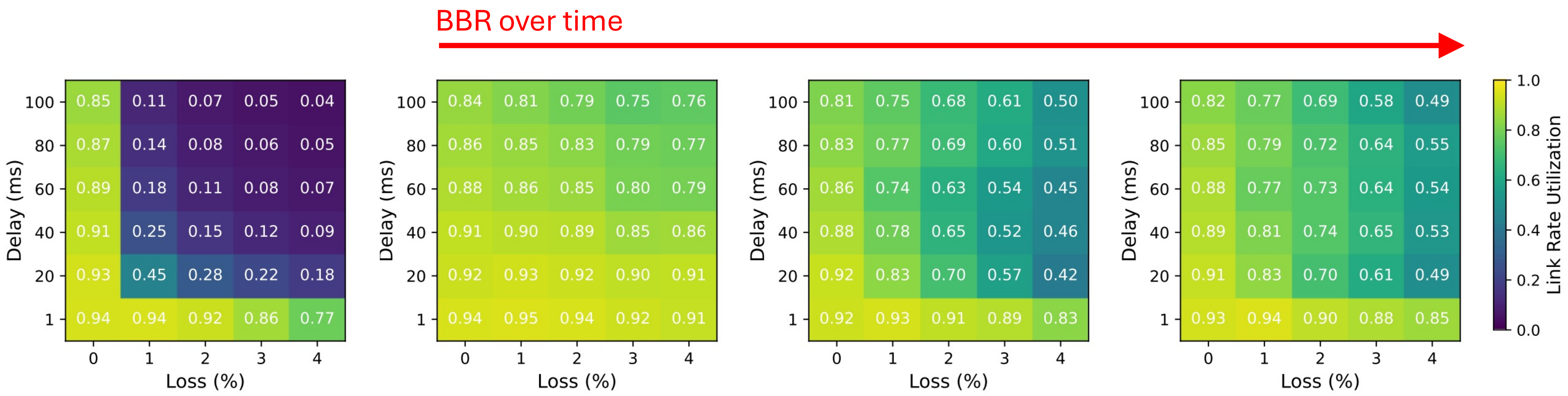


Class III. Symmetric, lossy

wireless ad-hoc and satellites with lossy ”middle-miles”

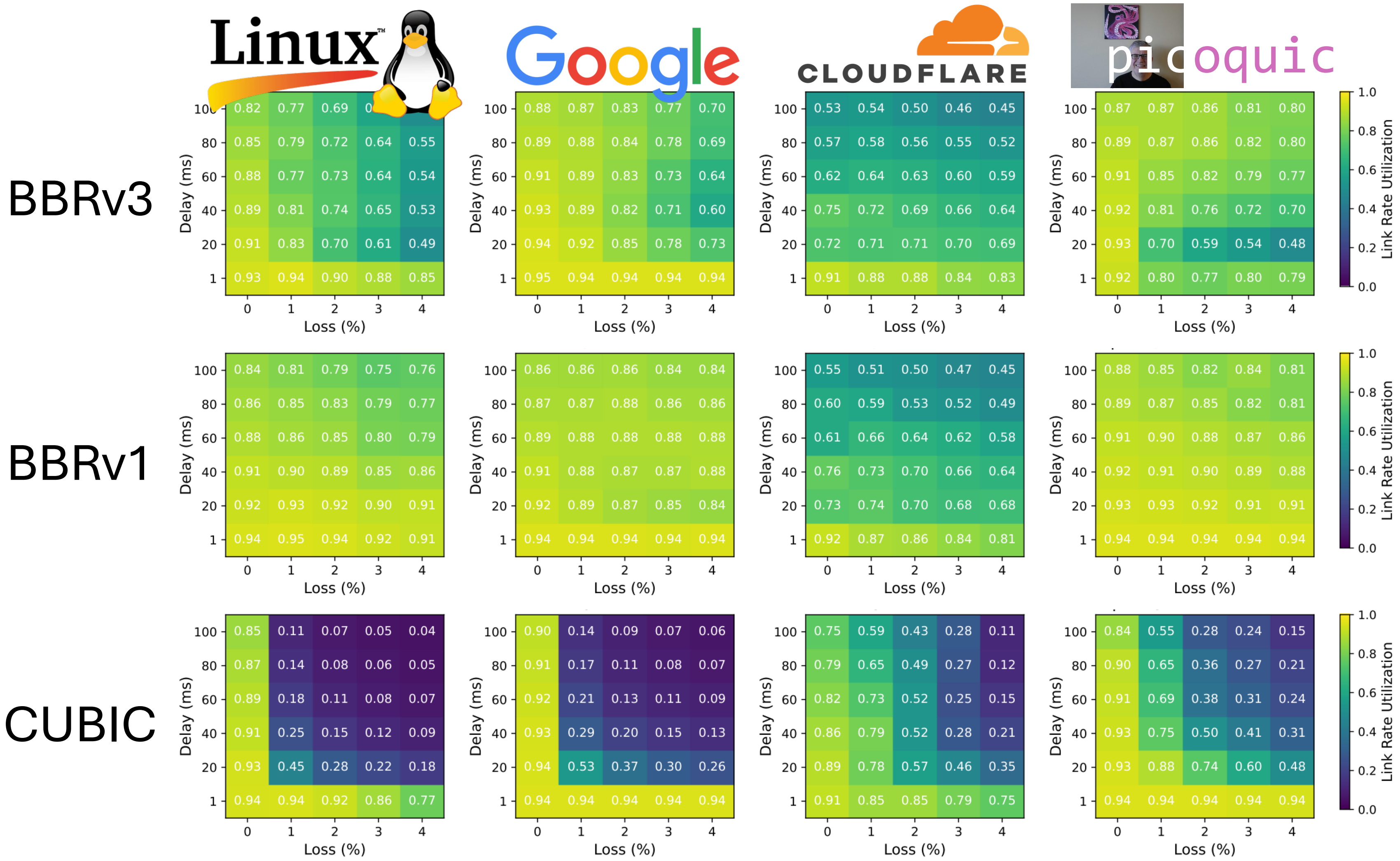


Heatmaps for characterizing end-to-end behavior

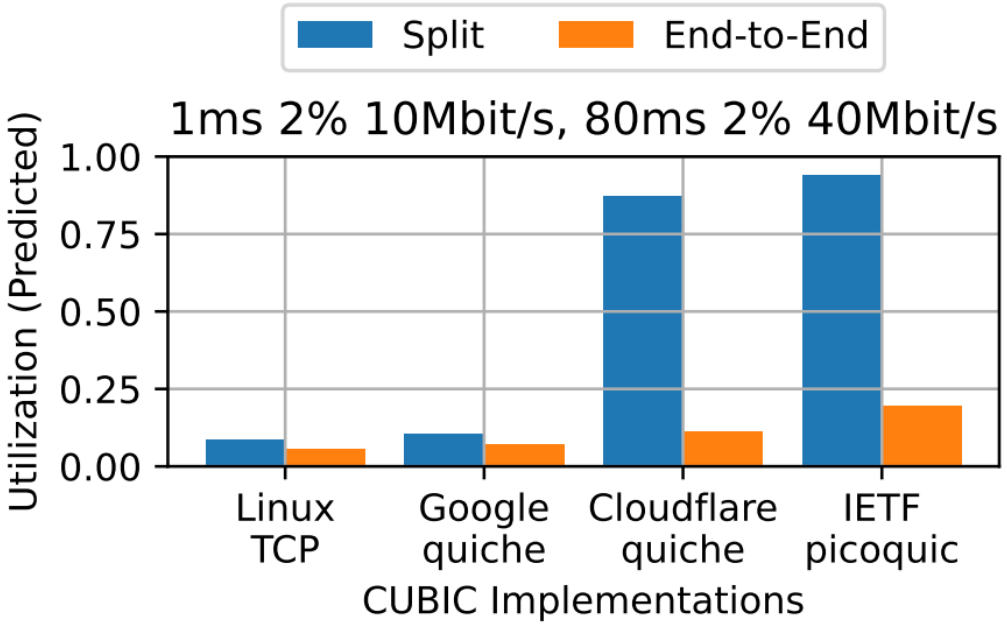


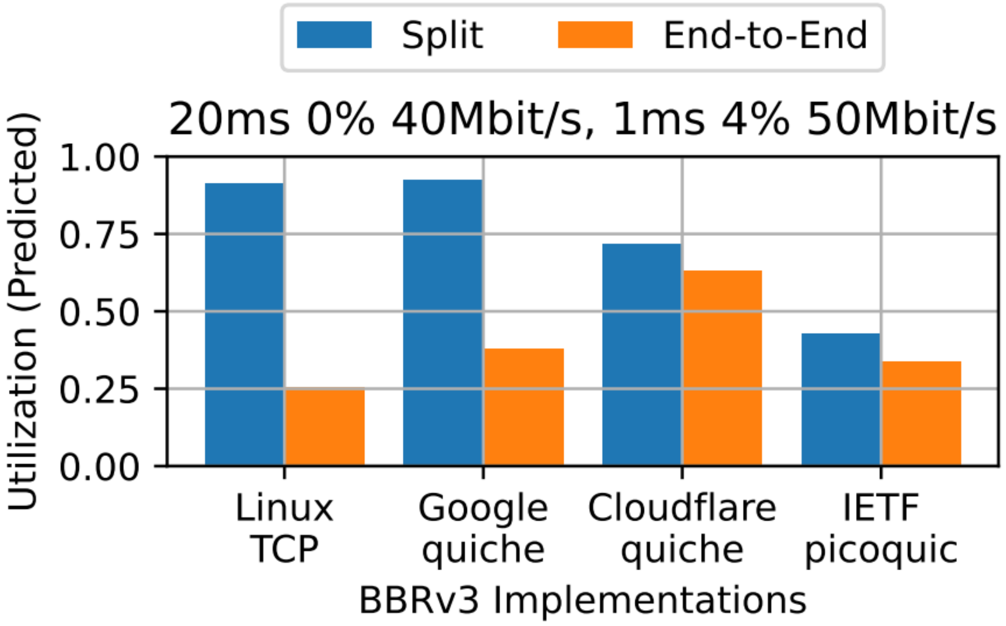
TCP CUBIC (2006) TCP BBRv1 (2016) TCP BBRv2 (2019) TCP BBRv3 (2023)

Finding 3: QUIC implementations of the "same" congestion-control schemes vary significantly, and with Linux TCP's.



Finding 3: QUIC implementations of the "same" congestion-control schemes vary significantly, and with Linux TCP's.





These results have implications for the throughput of the various user-space(!) QUIC implementations in *split* network path scenarios.

Calls to Action

- End-to-end congestion control
  - Refer to CC schemes by implementation/protocol/algorithm/version, e.g. not just “BBR” or even “QUIC BBRv1”, but “Chromium QUIC BBRv1”
  - Standardize what it means to conform to a particular CCA standard, perhaps by creating performance test suites
- Connection-splitting
  - Protocol-agnostic ways to emulate PEPs[5]
  - Real-world studies to validate speculated performance improvements
  - Study other metrics improved by PEPs

[5] Yuan, Sotoudeh, Zhang, Welzl, Mazières, & Winstein. Sidekick: In-Network Assistance for Secure End-to-End Transport Protocols. *NSDI*, 2024.

Summary

1. TCP BBR has benefitted more from splitting over time since "v1" in 2016.
2. TCP BBRv3 even benefits in some settings where TCP CUBIC does not.
3. QUIC implementations of the same CC schemes vary significantly.

Calls to action: Address disparities between different implementations of the same congestion-control algorithms, and develop protocol-agnostic methods to emulate the performance benefits of PEPs.

https://github.com/StanfordSNR/connection-splitting