BBR, the Default Congestion Control Protocol?

—

Paul Bergeron, Sandhya Aneja, Josue Abreu

Motivation & Breakdown

Goal: Evaluate if BBR should replace Cubic as Linux default

- TCP is foundational — **over 90% of internet traffic relies on it**
- **Linux default (Cubic)** chosen in **2007**, networks have changed drastically since
- Rising demands: **cloud traffic, low-latency gaming, high-bandwidth streaming**
- **BBR** introduces a model-driven approach: probes for **bandwidth + RTT**, avoids queue buildup

Research Question

Can BBR (v2/v3) outperform traditional TCP variants (Reno, Cubic) in diverse networking environments, and should it be adopted as the default TCP CCA?

Evaluation Criteria

Criterion	Why It Matters for Default Protocol
Throughput	Bulk transfers, streaming performance
Latency	Gaming, video conferencing responsiveness
Jitter	Stability for interactive apps
Fairness	Coexistence with Cubic, Reno, QUIC flows
Robustness	Handles loss, mobility, dynamic bandwidth
Deployment Practicality	Must work **without upgrading routers/switches**

Types of Congestion Control

BBR disrupts traditional TCP assumptions — it no longer treats loss as congestion.

- **Loss-Based** → Reno, Cubic (react after dropping packets)
- **Delay-Based** → Vegas, TIMELY (react when delays rise)
- **Network-Assisted (ECN/AQM)** → DCTCP, DCTCP-FQ (switch marks packets)
- **Hybrid/Model-Based** → **BBR**, Swift (predicts optimal sending rate)

Experiment Setup

Experiment 1 – Marist Campus Network

- 1–10 Gbps network
- 300MB file downloads
- Compared: BBR, Cubic, Reno, Vegas

Experiment 2 – Home WiFi (100 Mbps)

- Top 500 websites, focusing on 78 Google domains
- Measured throughput + RTT patterns
- Google assumed to be using BBR

Experiment One

Marist Campus Network Experiment (Ethernet)

- **BBR finished transfers in 25 RTTs on average**
- Cubic/Reno: ~28 RTTs **on average → still competitive**
- Vegas lagged at ~50 RTTs **on average**
- Shows **BBR aggressively fills pipe faster** than loss-based protocols

Experiment Two

Home WiFi Experiment

- Google domains show **two tuning profiles**:
  - **Content-heavy services (fonts, blog, insights)** → high throughput + low RTT
  - **Interactive services (mail, maps)** → very low RTT but capped throughput
- Indicates **Google adapts BBR by service type**, not just blindly maximizing bandwidth

Network Environment Analysis

"A protocol cannot become default if it only works on the public internet — it must also perform in specialized high-speed or unstable networks."

Data Center

BBR not ideal as default in pure data center traffic

- Data centers need **microsecond-level latency**
- **DCTCP-based algorithms outperforms BBR** under burst loads due to **ECN feedback**
- BBRv3 improves but **still slower reaction time** vs. Acknowledgement based CC(ACC), Swift
- BBR's **competitive throughput** compared to **DCTCP-based algorithms is well known**

Ethernet

BBR is not deployable as default in Ethernet/InfiniBand clusters yet

- **Lossless Ethernet (RoCE/DCQCN environments)** requires **zero-packet loss**
- ACC achieve **50 microsecond convergence vs. BBR's millisecond response**
- BBR **not compatible with PFC**/priority pause frames without modification

Wireless Networks

BBR only suitable in static WiFi, not mobility-heavy 5G/4G

- Wireless introduces **high RTT variance + non-congestion packet loss**
- **BBR v1/v2 caused buffer overflow in WiFi**
- **BBRv3 is better, but still outperformed by ABC, QCC, Cupid** in mobile environments

Satellite Networks

BBR would cause stalls on satellite networks if made default

- GEO/LEO introduces **500ms RTT + BER**
- **OnlineAEPC & mobility-aware QUIC variants outperform BBR in high-BER links**
- BBR lacks **predictive error-handling** and **multipath awareness**

Comparison Between Protocols

Protocol	Strength	Weakness
BBR	Highest throughput	High jitter / latency under load
Cubic	Stable + conservative	Slower in high-BDP paths
Reno	Balanced / predictable	Lower peak speed
Vegas	Lowest latency	Very low throughput
QCC / ABC	Wireless optimization	Not deployable on wired internet
DCTCP-FQ /ACC	Datacenter low latency	Needs ECN-enabled switches

Comparison With Cubic

BBR vs Cubic

- Cubic = **currently default**
- Pros: **Fair, predictable, backward-compatible**
- BBR = **30% higher throughput** but **higher jitter**
- **If Linux changed default**, Cubic flows could be **starved or see latency spikes**
- Cubic **struggles** in high throughput domains

Comparison With Other Protocols

BBR vs Other Protocols

- **BBR vs. Loss based Protocols:** BBR outpreformes loss based protocols in throughput dependent environments
- **BBR vs. Network Assisted:** Effort has been introduced ECN support into BBR, where BBR maintains competitive throughput and delay, especially in BBRv2 and BBRv3
- **BBR vs. Delay Based:** BBR outperformed delay based protocols like TCP Vegas and BBR Swift continued this improvement, eclipsing protocols like TIMELY

Results

"BBR achieves top throughput performance, but trade-offs appear in every specialized environment."

- Highest throughput: 905 Mbps
- Highest jitter: 4.2 ms
- Lowest latency: Vegas (0.02 ms) but low throughput
- Overall insight: No single protocol dominates all metrics — including BBR

Discussion & Trade Off

"BBR achieves top throughput performance, but trade-offs appear in every specialized environment."

- BBR is great **for** YouTube**, bad** for Zoom
- If you want speed → BBR. If you want stability → Cubic/Reno/Vegas
- A **default Linux algorithm must protect latency-first flows**, not just maximize pipe usage

Wrap Up

BBRv1 should not immediately replace Cubic because of fairness

However: BBR **should be enabled by default for high-bandwidth paths and QUIC flows**

Best path forward: **adaptive default**, not one static protocol

Future Work

Hybrid default model: BBR for >50ms RTT flows, Cubic for <10ms flows

Modify BBR to integrate:

- **PFC Awareness**
- **Wireless airtime estimation like Cupid**
- **BER prediction like OnlineAEPC**

Ask Linux **not to choose one default, but a default strategy**

References

[1] S. Ha, I. Rhee, and L. Xu, “CUBIC: A new TCP-friendly high-speed TCP variant,” *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[2] N. Cardwell, Y. Cheng, S. H. Yeganeh, P. Jha, Y. Seung, K. Yang, I. Swett, V. Vasiliev, B. Wu, L. Hsiao, et al., “BBRv2: A model-based congestion control performance optimization,” in *Proc. IETF 106th Meeting*, 2019, pp. 1–32.

[3] D. Zeynali, E. N. Weyulu, S. Fathalli, B. Chandrasekaran, and A. Feldmann, “BBRv3 in the public Internet: A boon or a bane?” in *Proc. 2024 Applied Networking Research Workshop*, 2024, pp. 97–99.

[4] Y. Chen, L. Wang, J. Wang, S. Liu, K. He, J. Wang, X. Wang, W. Dou, G. Chen, and C. Tian, “Marlin: Enabling high-throughput congestion control testing in large-scale networks,” in *Proc. 20th European Conf. on Computer Systems*, 2025, pp. 460–474.

[5] Y. Zhang, Q. Meng, C. Hu, H. Zhang, S. Wang, and F. Ren, “ACK-driven congestion control for lossless Ethernet,” *IEEE Trans. Netw.*, 2025.

[6] X. Du, J. Li, Y. Shao, W. Wang, S. Hu, J. Zhou, and K. Tan, “Revisiting congestion control for WiFi networks,” in *Proc. 8th Asia-Pacific Workshop on Networking*, 2024, pp. 88–94.

[7] P. Goyal, A. Agarwal, R. Netravali, M. Alizadeh, and H. Balakrishnan, “ABC: A simple explicit congestion controller for wireless networks,” in *17th USENIX Symp. on Networked Systems Design and Implementation (NSDI 20)*, 2020, pp. 353–372.

[8] G. Kumar, N. Dukkipati, K. Jang, H. M. Wassel, X. Wu, B. Montazeri, Y. Wang, K. Springborn, C. Alfeld, M. Ryan, et al., “Swift: Delay is simple and effective for congestion control in the datacenter,” in *Proc. ACM SIGCOMM*, 2020, pp. 514–528.

[9] L. Li, Y. Chen, and Z. Li, “QCC: Driver-queue based congestion control for data uploading in wireless networks,” *IEEE Trans. Mobile Comput.*, 2024.

[10] A. Mishra, W. H. Tiu, and B. Leong, “Are we heading towards a BBR-dominant Internet?” in *Proc. 22nd ACM Internet Measurement Conf.*, 2022, pp. 538–550.

[11] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, “Data Center TCP (DCTCP),” in *Proc. ACM SIGCOMM 2010 Conf.*, 2010, pp. 63–74.

[12] H. Wang, X. Zhang, A. Yang, and B. Sheng, “DCTCP-FQ: Enhancing fairness and convergence time in data center congestion control,” in *Proc. 2025 Int. Conf. on Computing, Networking and Communications (ICNC)*, 2025.

[13] Y. Zhang, Y. Fan, Y. Xiao, H. Chen, P. Xie, L. Liu, and H. Ma, “BBQ: Dynamic-buffer-driven automatic ECN tuning in datacenter,” in *2024 IEEE/ACM 32nd Int. Symp. on Quality of Service (IWQoS)*, 2024, pp. 1–6.

[14] J. Fang and J. Zhang, “Adaptive congestion control strategies for LEO satellite networks,” in *2024 IEEE 24th Int. Conf. on Communication Technology (ICCT)*, 2024, pp. 1055–1060.

[15] W. Yang, L. Cai, S. Shu, and J. Pan, “Mobility-aware congestion control for multipath QUIC in integrated terrestrial-satellite networks,” *IEEE Trans. Mobile Comput.*, 2024.